



rain0993

rain0993.blog.chinaunix.net

linux driver

首页 | 博文目录 | 关于我



rain0993

博客访问： 177422
博文数量： 29
博客积分： 1933
博客等级： 上尉
技术积分： 877
用户组： 普通用户
注册时间： 2011-01-04 23:34

加关注 短消息
论坛 加好友

文章分类

- 全部博文 (29)
- lcd (12)
 - camera (12)
 - android (0)
 - linux驱动 (1)
 - 未分配的博文 (4)

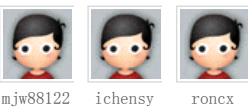
文章存档

- 2013年 (3)
2011年 (26)

我的朋友



最近访客



Camera 图像处理原理分析 2011-06-13 20:03:41

分类:

1 前言

做为拍照手机的核心模块之一，camera sensor效果的调整，涉及到众多的参数，如果对基本的光学原理及sensor软/硬件对图像处理的原理能有深入的理解和把握的话，对我们的工作将会起到事半功倍的效果。否则，缺乏了理论的指导，只能是凭感觉和经验去碰，往往无法准确的把握问题的关键，不能掌握sensor调试的核心技术，无法根本的解决问题。

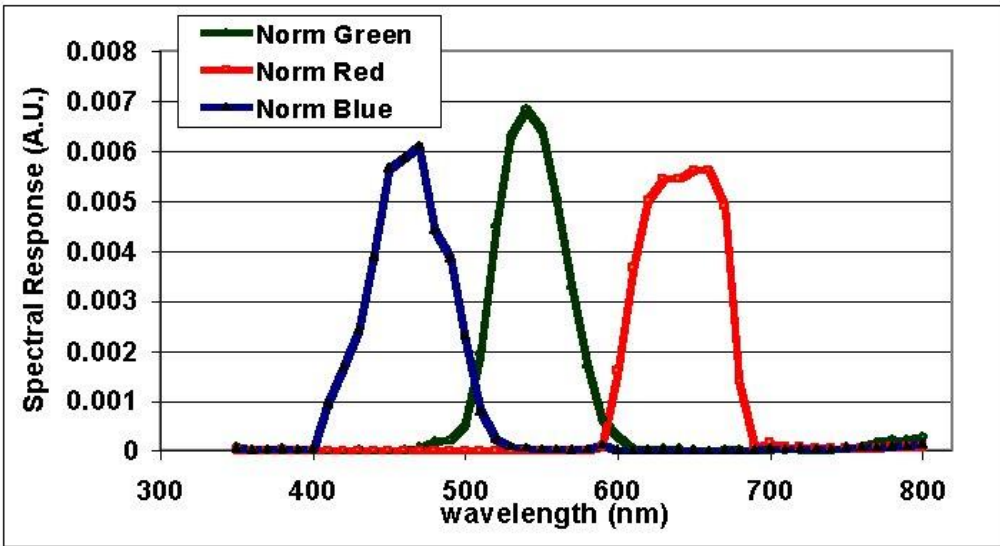
所以，这里笔者结合自己出于对摄影的爱好所学习的一些图像处理相关的原理，试图通过分析一些与Sensor图像处理相关的因素，和大家分享一下自己的一些理解，共同探讨，共同学习进步。

2 色彩感应及校正


2.1 原理


人眼对色彩的识别，是基于人眼对光线存在三种不同的感应单元，不同的感应单元对不同波段的光有不同的响应曲线的原理，通过大脑的合成得到色彩的感知。一般来说，我们可以通俗的用RGB三基色的概念来理解颜色的分解和合成。


理论上，如果人眼和sensor对光谱的色光的响应，在光谱上的体现如下的话，基本上对三色光的响应，相互之间不会发生影响，没有所谓的交叉效应。





但是，实际情况并没有如此理想，下图表示了人眼的三色感应系统对光谱的响应情况。可见RGB的响应并不是完全独立的。



sevenove


Contikil


ian70518


tbtzw


rensong0


wangcong

微信关注



IT168企业级官微
微信号: IT168qiye



系统架构师大会
微信号: SACC2013

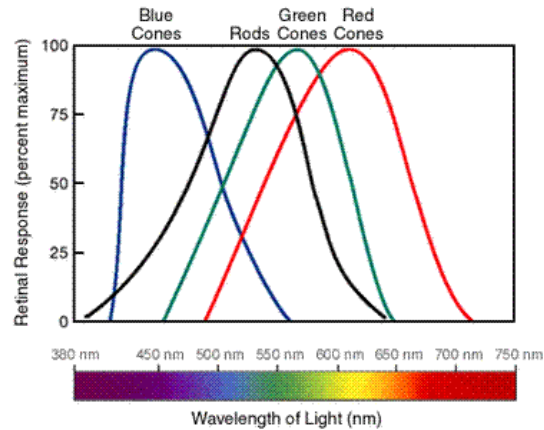
订阅

推荐博文

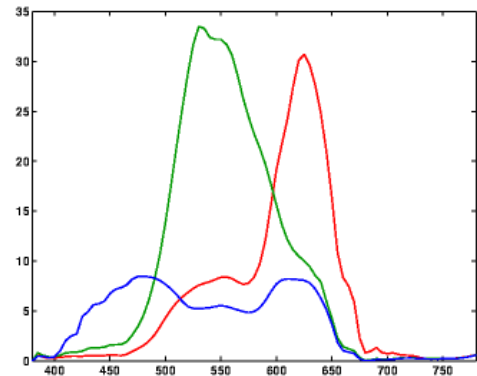
- 理解HTTP幂等性
- MySQL数据导出与导入
- iproute2 策略路由与流量控制...
- Python性能分析指南
- HTPC+NAS+ROUTER(wifi)的实现...

热词专题

- lua编译(linux)



下图则表示了某Kodak相机光谱的响应。可见其与人眼的响应曲线有较大的区别。



2.2 对sensor的色彩感应的校正

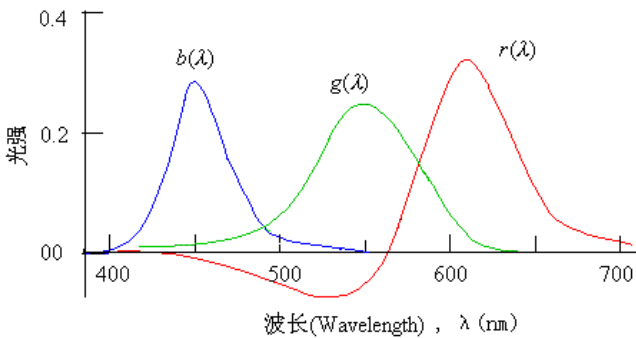
既然我们已经看到sensor对光谱的响应，在RGB各分量上与人眼对光谱的响应通常是有偏差的，当然就需要对其进行校正。不光是在交叉效应上，同样对色彩各分量的响应强度也需要校正。通常的做法是通过一个色彩校正矩阵对颜色进行一次校正。

色彩校正的运算通常是由sensor模块集成或后端的ISP完成，软件通过修改相关寄存器得到正确的校正结果。值得注意的一点是，由于RGB -> YUV的转换也是通过一个3*3的变换矩阵来实现的，所以有时候这两个矩阵在ISP处理的过程中会合并在一起，通过一次矩阵运算操作完成色彩的校正和颜色空间的转换。

3 颜色空间

3.1 分类

实际上颜色的描述是非常复杂的，比如RGB三基色加光系统就不能涵盖所有可能的颜色，出于各种色彩表达，以及色彩变换和软硬件应用的需求，存在各种各样的颜色模型及色彩空间的表达方式。这些颜色模型，根据不同的划分标准，可以按不同的原则划分为不同的类别。



匹配任意可见光所需的三原色光比例曲线

对于sensor来说，我们经常接触到的色彩空间的概念，主要是RGB，YUV这两种（实际上，这两种体系包含了许多种不同的颜色表达方式和模型，如sRGB, Adobe RGB, YUV422, YUV420 ...），RGB如前所述就是按三基色加光系统的原理来描述颜色，而YUV则是按照亮度、色差的原理来描述颜色。

3.1.1 RGB <-> YUV的转换

不比其它颜色空间的转换有一个标准的转换公式，因为YUV在很大程度上是与硬件相关的，所以RGB与YUV的转换公式通常会多个版本，略有不同。

常见的公式如下：

$$Y=0.30R+0.59G+0.11B$$
$$U=0.493(B-Y) = -0.15R-0.29G+0.44B$$
$$V=0.877(R-Y) = 0.62R-0.52G-0.10B$$

但是这样获得的YUV值存在着负值以及取值范围上下限之差不为255等等问题，不利于计算机处理，所以根据不同的理解和需求，通常在软件处理中会用到各种不同的变形的公式，这里就不列举了。

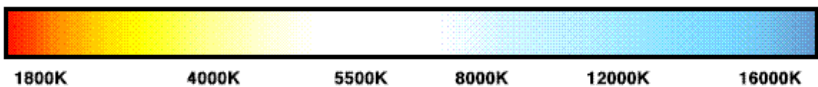
体现在Sensor上，我们也会发现有些Sensor可以设置YUV的输出取值范围。原因就在于此。

从公式中，我们关键要理解的一点是，UV 信号实际上就是蓝色差信号和红色差信号，进而言之，实际上一定程度上间接的代表了蓝色和红色的强度，理解这一点对于我们理解各种颜色变换处理的过程会有很大的帮助。

1.1 白平衡

1.1.1 色温

色温的定义：将黑体从绝对零度开始加温，温度每升高一度称为1开氏度(用字母K来表示)，当温度升高到一定程度时候，黑体便辐射出可见光，其光谱成份以及给人的感觉也会随着温度的不断升高发生相应的变化。于是，就把黑体辐射一定色光的温度定为发射相同色光光源的色温。



常见光源色温：

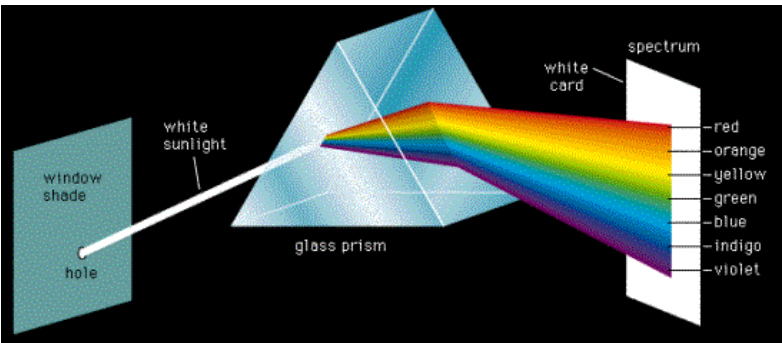
光源	色温 (K)
钨丝灯（白炽灯）	2500-3200k
碳棒灯	4000-5500k
荧光灯（日光灯，节能灯）	4500-6500k
氙灯	5600 k
炭精灯	5500~6500k
日光平均	5400k
有云天气下的日光	6500-7000k
阴天日光	12000-18000k

随着色温的升高，光源的颜色由暖色向冷色过渡，光源中的能量分布也由红光端向蓝光端偏移。

值得注意的是，实际光源的光谱分布各不相同，而色温只是代表了能量的偏重程度，并不反映具体的光谱分布，所以即使相同色温的光源，也可能引起不同的色彩反应。

人眼及大脑对色温有一定的生理和心理的自适应性，所以看到的颜色受色温偏移的影响较小，而camera的sensor没有这种能力，所以拍出来的照片不经过白平衡处理的话，和人眼看到的颜色会有较大的偏差（虽然人眼看到的和白光下真实的色彩也有偏差）。

太阳光色温随天气和时间变化的原因，与不同频率光的折射率有关：



波长长的光线，折射率小，透射能力强，波长短的光线，折射率大，容易被散射，折射率低，这也就是为什么交通灯用红色，防雾灯通常是黄色，天空为什么是蓝色的等等现象的原因。

知道了这一点，太阳光色温变化的规律和原因也就可以理解和分析了，留给大家自己思考。

1.1.1 色温变化时的色彩校正

所以从理论上可以看出，随着色温的升高，要对色温进行校正，否则，物体在这样的光线条件下所表现出来的颜色就会偏离其正常的颜色，因此需要降低 sensor对红色的增益，增加sensor对蓝光的增益。同时在调整参数时一定要考虑到整体亮度的要保持大致的不变，即以YUV来衡量时，Y值要 基本保持不

变，理论上认为可以参考RGB->YUV变换公式中，RGB三分量对Y值的贡献，从而确定RGAIN和BGAIN的变化的比例关系。但实际情况比这还要复杂一些，要考虑到不同sensor对R, B的感光的交叉影响和非线性，所以最佳值可能和理论值会有一些偏差。

1.1.2 自动白平衡原理

1.1.2.1 原理

自动白平衡是基于假设场景的色彩的平均值落在一个特定的范围内，如果测量得到结果偏离该范围，则调整对应参数，校正直到其均值落入指定范围。该处理过程可能基于YUV空间，也可能基于RGB空间来进行。对于Sensor来说，通常的处理方式是通过校正R/B增益，使得UV值落在一个指定的范围内。从而实现自动白平衡。

1.1.2.2 特殊情况的处理

在自动白平衡中，容易遇到的问题是，如果拍摄的场景，排除光线色温的影响，其本身颜色就是偏离平均颜色值的，比如大面积的偏向某种颜色的图案如：草地，红旗，蓝天等等，这时候，强制白平衡将其平均颜色调整到灰色附近，图像颜色就会严重失真。

因此，通常的做法是：在处理自动白平衡时，除了做为目标结果的预期颜色范围外，另外再设置一对源图像的颜色范围阈值，如果未经处理的图像其颜色均值超出了该阈值的话，根本就不对其做自动白平衡处理。由此保证了上述特殊情况的正确处理。

可见，这两对阈值的确定对于自动白平衡的效果起着关键性的作用。

1.1.3 某平台的例子

英文代码	中文界面	色温	色温
RGAIN, GGAIN, BGAIN			
cloud	阴天	7500k	0x1D4C, 0x00CD, 0x0085, 0x0080
daylight	日光	6500k	0x1964, 0x00A3, 0x0080, 0x0088
INCANDESCENCE	白炽光	5000k	0x1388, 0x00A5, 0x0080, 0x0088
FLUORESCENT	日光灯	4400k	0x1130, 0x0098, 0x0080, 0x00A8
TUNGSTEN	钨丝灯	2800k	0x0AF0, 0x0080, 0x0081, 0x00A4

可以看到随着色温的升高，其变化规律基本符合上节中的理论分析。不过这里多数参数与理论值都有一些偏差，其中日光灯的色温参数设置与理论值有较大的偏差，实际效果也证明该日光灯的参数设置使得在家用日光灯环境下拍摄得到的照片颜色偏蓝。修改其参数后实拍效果明显改善。（再查一些资料可以看到通常会有两种荧光灯色温 4000 和 5000K，目前我所接触到的应该是5000K居多）

1.1.4 调试和验证

具体参数的调整，应该在灯箱环境下，使用各种已知色温的标准光源对标准色卡拍摄，在Pc机上由取色工具测量得到其与标准色板的RGB分量上的色彩偏差，相应的调整各分量增益的比例关系。为了更精确的得到结果，曝光量增益的设置在此之前应该相对准确的校正过

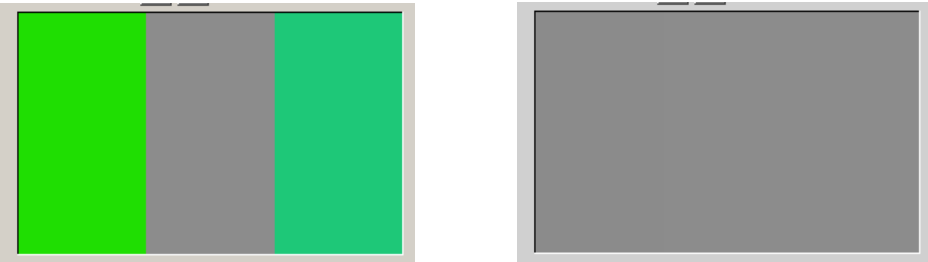
颜色相关特效处理

1.1 grayscale（灰阶）

灰阶图的效果就是将彩色图片转换为黑白图片。

1.2 理论

理论上，在YUV空间，将UV分量丢弃，只保留Y分量，这样就可以得到黑白图像，这也是彩色电式机信号能兼容黑白电视机的原理。如下图理论上Y值一样的颜色（右边是用acdsee转成灰度图的效果），在grayscale模式下看应该是一样的颜色。



算法上的操作，理论上应该把UV值改成灰色对应数值就可以了。不过根据软件算法和硬件结构的不同，具体代码也会有不同。

1.3 以某平台为例

核心的两行代码如下：

```
SET_HUE_U_GAIN(0);
```



```
SET_HUE_V_GAIN(0);
```

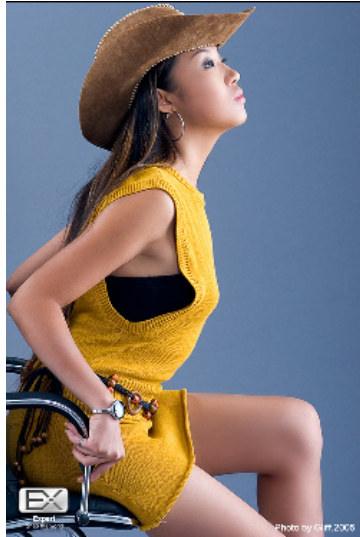
这里设置UV_GAIN为0，如果UV_offset设置为128的话，最终得到的UV就是128，这就和理论是相符合的。

1.4 sepia / sepiagreen / sepiablue

所谓的复古（绿，蓝）就是在灰阶的基础上，对UV值额外再做了一个offset，将灰度图转换成某种颜色的梯度图。理论上为了获得蓝色效果，应该增加蓝色差信号，减小红色差信号。即增大U，减小V。

以sepiablue效果为例，这里的字节的MSB表示符号位：所以88为88，158为-30。

```
SET_HUE_U_GAIN(0);  
SET_HUE_V_GAIN(0);  
SET_HUE_U_OFFSET(88);  
SET_HUE_V_OFFSET(158);
```



1.5 negative

所谓负片效果，就是将图像的颜色反转，看起来就像是在看胶片底片时的效果。这从理论上也很容易理解和处理，就是在RGB空间，取其补色，具体的操作就是用255分别减去RGB得到新的RGB值。通常会在ISP中实现该功能。



2 小结

理解了原理，要做出其它颜色变换方面的效果就很容易了。

基本上，在颜色校正和处理方面，需要考虑的相关参数大致包括：

自动WB上下限，自动白平衡时的目标范围，RGB gain, UV gain, UV offset, color correction. 有些还会有saturation 和 hue相关的设置。

从sensor或ISP硬件处理的流程上说，通常方向是先做RGB gain，再做color correction, 最后做YUV空间的处理。所以调整效果的时候，为了减少参数之间的相互影响，基本上也可以按这个顺序来调整参数。

1.1 亮度感应及曝光

1.1.1 感光宽容度

从最明亮到最黑暗，假设人眼能够看到一定的范围，那么胶片（或CCD等电子感光器件）所能表现的远比人眼看到的范围小的多，而这个有限的范围就是感光宽容度。

人眼的感光宽容度比胶片要高很多，而胶片的感光宽容度要比数码相机的ccd高出很多！了解这个概念之后，我们就不难了解，为什么在逆光的条件下，人眼能看清背光的建筑物以及耀眼的天空云彩。而一旦

拍摄出来，要么就是云彩颜色绚烂而建筑物变成了黑糊糊的剪影，要么就是建筑物色彩细节清楚而原本美丽的云彩却成了白色的一片

再看人眼的结构，有瞳孔可以控制通光量，有杆状感光细胞和椎状感光细胞以适应不同的光强，可见即使人眼有着很高的感光宽容度，依然有亮度调节系统，以适应光强变化。

那么对于camera sensor来说，正确的曝光就更为重要了！

1.1.2 自动曝光和18%灰

对于sensor来说，又是如何来判断曝光是否正确呢？很标准的做法就是在YUV空间计算当前图像的Y值的均值。调节各种曝光参数设定（自动或手动），使得该均值落在一个目标值附近的时候，就认为得到了正确的曝光。

那么如何确定这个Y的均值，以及如何调整参数使得sensor能够将当前图像的亮度调整到这个范围呢？这就涉及到一个概念 18%灰，一般认为室内室外的景物，在通常的情况下，其平均的反光系数大约为18%，而色彩均值，如前所述，可以认为是一种中灰的色调。这样，可以通过对 反光率为18%的灰板拍摄，调整曝光参数，使其颜色接近为中等亮度的灰色（Y值为128）。然后，对于通常的景物，就能自动的得到正确的曝光了。



当然这种自动判断曝光参数的AE功能不是万能的，对于反光率偏离通常均值的场景，比如雪景，夜景等，用这种方法就无法得到正确的曝光量了。所以在sensor的软件处理模块中，通常还会提供曝光级别的设定功能，强制改变自动曝光的判断标准。比如改变预期的亮度均值等。

1.1.3 曝光级别设定

在多数数码相机和拍照手机上都可以看到曝光级别设定的功能，如前所述，这种设定实际上是在自动曝光的基础上给用户一定的曝光控制能力，强制改变camera sensor的曝光判断标准，获得用户想要的效果。

通常的做法就是改变Y值均值的预期值，使得sensor在自动曝光时以新的Y预期值为目标，自动调整Exptime 和 AG。

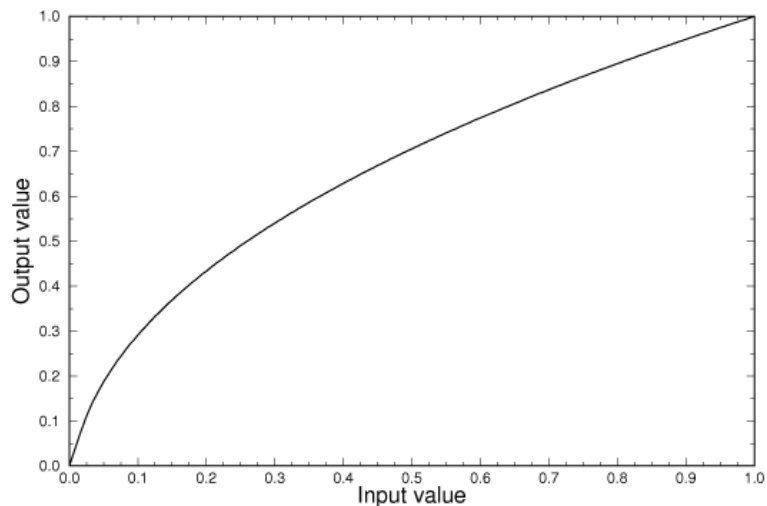
1.1.4 gamma校正

曝光的均值正确了，不代表整体图像的亮度分布就和人眼所看到的保持一致了。

事实上，人眼对亮度的响应并不是一个线性的比例关系，而各种涉及到光电转换的设备的输入输出特性曲线一般也是非线性的，且表现为幂函数的形式： $y=x^n$ ，所以整个图像系统的传递函数是一个幂函数。 $g = g_1 \times g_2 \times \dots \times g_n$

对于sensor来说，其响应倒是接近为线性关系，所以为了在各种设备上正确输出符合人眼对亮度的响应的图像，就需要进行校正。

幂函数的指数的倒数就是通常所说的gamma值。



归一化的gamma曲线

校正的函数可以表示为 $y = x^{1/\gamma}$ ，通常对于Window的输出显示系统，gamma值为2.2，而对于苹果的输出显示系统和打印系统来说，gamma值为1.8。

实际上，sensor在做gamma校正的时候，通常也一并作了从raw格式的10bit的数据到8bit数据的转换，所以这时候的公式可以表示为 $y = x^{1/\gamma}$ 对比度

对比度的调整在一定程度上说，其实也就是对gamma曲线的调整，增大对比度就是提高Gamma值。对于图像处理来说，也有在硬件gamma校正后，单独由软件再进行一次类似的幂函数变换来调整对比度。

1.1.6 曝光参数的调整

曝光强度的调整，可以通过改变曝光时间，也可以通过改变亮度增益AG来实现。

曝光时间受到帧频的限制，比如摄像时要求15帧每秒的话，这时候曝光时间最长就不能超过1/15s，可能还有别的条件限制，实际的曝光时间还要短，在光线弱的情况下，单独调整曝光时间就无法满足帧频的需要了。

这时候还可以调整增益AG，来控制曝光的增益，降低曝光时间。但是，这样做的缺点是以牺牲图像质量为代价的，AG的增强，伴随的必然是信噪比的降低，图像噪声的增强。

所以，以图像质量为优先考虑的时候，曝光参数的调节通常是优先考虑调节曝光时间，其次在考虑曝光增益。当然曝光时间也不能过长以免由于抖动造成图像的模糊，而在拍摄运动场景时，对曝光时间的要求就更高了

1.1 抗噪处理

AG 的增大，不可避免的带来噪点的增多，此外，如果光线较暗，曝光时间过长，也会增加噪点的数目（从数码相机上看，主要是因为长时间曝光，感光元件温度升高，电流噪声造成感光元件噪点的增多），而感光元件本身的缺陷也是噪点甚至坏点的来源之一。因此，通常sensor集成或后端的ISP都带有降噪功能的相关设置。

1.1.1 启动时机

根据噪点形成的原因，主要是AG或Exptime超过一定值后需要启动降噪功能，因此通常需要确定这两个参数的阈值，过小和过大都不好。

从下面的降噪处理的办法将会看到，降噪势附带的带来图像质量的下降，所以过早启动降噪功能，在不必要的情况下做降噪处理不但增加处理器或ISP的负担，还有可能适得其反。而过迟启动降噪功能，则在原本需要它的时候，起不到相应的作用。

1.1.2 判定原则和处理方式

那么如何判定一个点是否是噪点呢？我们从人是如何识别噪点的开始讨论，对于人眼来说，判定一个点是噪点，无外乎就是这一点的亮度或颜色与边上大部分的点差异过大。从噪点产生的机制来说，颜色的异常应该是总是伴随着亮度的异常，而且对亮度异常的处理工作量比颜色异常要小，所以通常sensor ISP的判定原则是一个点的亮度与周围点的亮度的差值大于一个阈值的时候，就认为该点是一个噪点。

处理的方式，通常是对周围的点取均值来替代原先的值，这种做法并不增加信息量，类似于一个模糊算法。

对于高端的数码相机，拥有较强的图像处理芯片，在判定和处理方面是否有更复杂的算法，估计也是有可能的。比如亮度和颜色综合作为标准来判定噪点，采用运算量更大的插值算法做补偿，对于sensor固有的坏点，噪点，采用屏蔽的方式抛弃其数据（Nikon就是这么做的，其它厂商应该也如此）等等。

1.1.3 效果

对于手机sensor来说，这种降噪处理的作用有多大，笔者个人认为应该很有限，毕竟相对数码相机，手机sensor的镜头太小，通光量小，所以其基准AG势必就比相机的增益要大（比如相当于普通家用数码相机ISO800的水平），这样才能获得同样的亮度，所以电流噪声带来的影响也就要大得多。这样一来，即使最佳情况，噪点也会很多，数据本身的波动就很大，这也就造成我们在手机照片上势必会看到的密密麻麻的花点，如果全部做平均，降低了噪点的同时，图像也会变得模糊，所以手机噪点的判断阈值会设得比较高，以免涉及面过大，模糊了整体图像。这样一来一是数据本身就差，二是降噪的标准也降低了，造成总体效果不佳。

1.2 数码变焦

数码变焦可以有两种形式：

其一，是通过插值算法，对图像进行插值运算，将图像的尺寸扩大到所需的规格，这种算法就其效果而言，并不理想，尤其是当使用在手机上的时候，手机上的摄像头本身得到的数据就有较大的噪声，再插值的话，得到的图像几乎没法使用。实际上，即使是数码相机的数码变焦功能也没有太大的实用价值。如果插值算法没有硬件支持，则需要应用层实现。我司某平台的数码变焦用的就是该种办法。

其二，其实是一种伪数码变焦的形式，当摄像头不处在最大分辨率格式的情况下，比如130万像素的sensor使用640*480的规格拍照时，仍旧设置sensor工作在1280*960的分辨率下，而后通过采集中央部分的图像来获取640*480的照片，使得在手机上看来所拍物体尺寸被放大了一倍。也有很多手机采用的是这种数码变焦方式，这种办法几乎不需要额外的算法支持，对图像质量也没有影响，缺点是只有小尺寸情况下可以采用。此外在DV方式下也可以实现所谓的数码变焦放大拍摄功能。（这应该是一个卖点，对DV来说，这种数码变焦还是有实际意义的）

要采用这种变焦模式，驱动需要支持windowing功能，获取所需部分的sensor图像数据。

1.3 频闪抑制功能

1.3.1 何谓频闪

日常使用的普通光源如白炽灯、日光灯、石英灯等都是直接用220/50Hz交流电工作，每秒钟内正负半周各变化50次，因而导致灯光在1秒钟内发生100（50×2）次的闪烁，再加上市电电压的不稳定，灯光忽明忽暗，这样就产生了所谓的“频闪”。

下表显示了几种光源的光强波动情况：

光源类型	工作频率	波动深度
白炽灯	50Hz	5~15%
钠灯或金卤灯	50Hz	80~130%
荧光汞灯（水银灯）	50Hz	65%
普通荧光灯（铁芯）	50Hz	55%
电子荧光灯（含 CFL）	20~35KHz	30~20%
	40~50KHz	15~10%
	80KHz	3%
三相交流荧光灯	50Hz	3~5%

因为人眼对光强变化有一定的迟滞和适应性，所以通常看不出光源的亮度变化。但是依然还是会增加眼睛的疲劳程度。所以市场上才会有所谓的无频闪灯销售。

1.3.2 对频闪的抑制

对于camera sensor来说，没有人眼的迟滞和适应过程，所以对光源亮度的变化是比较敏感的。如果不加抑制，在预览和DV模式下，可能会有明显的图像的明亮变化闪烁的现象发生。

如何解决呢？考虑到频闪的周期性，在一个周期内，光源亮度的累积值，应该是大体一致的，所以，如果控制曝光的时间是频闪周期的整倍数，那么每一帧图像的亮度就大体是一致的了，这样就可以有效地抑制频闪对图像亮度的影响。

所以，在自动曝光的模式下，sensor会根据频闪的频率，调整曝光时间为其周期的整倍数。因为各地的交流电的频率不同，所以有50Hz/60Hz之分。

在具体设置相关Sensor寄存器的时候，要根据电流频率和sensor的时钟频率，分辨率等，计算出频闪周期对应的时钟周期数等。

1 前言

自然界的颜色千变万化，为了给颜色一个量化的衡量标准，就需要建立色彩空间模型来描述各种各样的颜色，由于人对色彩的感知是一个复杂的生理和心理联合作用的过程，所以在不同的应用领域中为了更好地满足各自的需求，就出现了各种各样的色彩空间模型来量化的描述颜色。我们比较常接触到的就包括 RGB / CMYK / YIQ / YUV / HSI等等。

对于数字电子多媒体领域来说，我们经常接触到的色彩空间的概念，主要是RGB，YUV这两种（实际上，这两种体系包含了许多种具体的颜色表达方式和模型，如sRGB, Adobe RGB, YUV422, YUV420 ...），RGB是按三基色加光系统的原理来描述颜色，而YUV则是按照亮度，色差的原理来描述颜色。

即使只是RGB YUV这两大类色彩空间，所涉及到的知识也是十分丰富复杂的，在下自知不具备足够的相关专业知识，所以本文主要针对工程领域的应用及算法进行讨论。

2 YUV相关色彩空间模型

2.1 YUV 与 YIQ YcrCb

对于YUV模型，实际上很多时候，我们是把它和YIQ / YCrCb模型混为一谈的。

实际上，YUV模型用于PAL制式的电视系统，Y表示亮度，UV并非任何单词的缩写。

YIQ模型与YUV模型类似，用于NTSC制式的电视系统。YIQ颜色空间中的I和Q分量相当于将YUV空间中的UV分量做了一个33度的旋转。

YCbCr颜色空间是由YUV颜色空间派生的一种颜色空间，主要用于数字电视系统中。从RGB到YCbCr的转换中，输入、输出都是8位二进制格式。

三者与RGB的转换方程如下：

RGB → YUV

$$\begin{bmatrix} Y \\ U \\ V \end{bmatrix} = \begin{bmatrix} 0.299 & 0.587 & 0.114 \\ -0.147 & -0.289 & 0.436 \\ 0.615 & -0.515 & -0.100 \end{bmatrix} \begin{bmatrix} R \\ G \\ B \end{bmatrix}$$

实际上也就是：

$$Y=0.30R+0.59G+0.11B, \quad U=0.493(B-Y), \quad V=0.877(R-Y)$$

RGB → YIQ

$$\begin{bmatrix} Y \\ I \\ Q \end{bmatrix} = \begin{bmatrix} 0.299 & 0.587 & 0.114 \\ 0.596 & -0.275 & -0.321 \\ 0.212 & -0.523 & 0.311 \end{bmatrix} \begin{bmatrix} R \\ G \\ B \end{bmatrix}$$

RGB → YCrCb

$$\begin{bmatrix} Y \\ Cr \\ Cb \end{bmatrix} = \begin{bmatrix} 0.299 & 0.578 & 0.114 \\ 0.500 & -0.4187 & -0.0813 \\ -0.1687 & -0.3313 & 0.500 \end{bmatrix} \begin{bmatrix} R \\ G \\ B \end{bmatrix} + \begin{bmatrix} 0 \\ 128 \\ 128 \end{bmatrix}$$

从公式中，我们关键要理解的一点是，UV / CbCr信号实际上就是蓝色差信号和红色差信号，进而言之，实际上一定程度上间接的代表了蓝色和红色的强度，理解这一点对于我们理解各种颜色变换处理的过程会有很大的帮助。

我们在数字电子多媒体领域所谈到的YUV格式，实际上准确的说，是以YcrCb色彩空间模型为基础的具有多种存储格式的一类颜色模型的家族（包括YUV444 / YUV422 / YUV420 / YUV420P等等）。并不是传统意义上用于PAL制模拟电视的YUV模型。这些YUV模型的区别主要在于UV数据的采样方式和存储方式，这里就不详述。

而在Camera Sensor中，最常用的YUV模型是YUV422格式，因为它采用4个字节描述两个像素，能和RGB565模型比较好的兼容。有利于Camera Sensor和Camera controller的软硬件接口设计。

3 YUV2RGB快速算法分析

这里指的YUV实际是YcrCb了（8位）YUV2RGB的转换公式本身是很简单的，但是牵涉到浮点运算，所以，如果要想实现快速算法，算法结构本身没什么好研究的了，主要是采用整型运算或者查表来加快计算速度。

首先可以推导得到转换公式为：

$$R = Y + 1.4075 * (V - 128)$$

$$G = Y - 0.3455 * (U - 128) - 0.7169 * (V - 128)$$

$$B = Y + 1.779 * (U - 128)$$

3.1 整型算法

要用整型运算代替浮点运算，当然是要用移位的办法了，我们可以很容易得到下列算法：

```

u = YUVdata[UPOS] - 128;
v = YUVdata[VPOS] - 128;

rdif = v + ((v * 103) >> 8);
invgdif = ((u * 88) >> 8) + ((v * 183) >> 8);
bdif = u + ((u * 198) >> 8);

r = YUVdata[YPOS] + rdif;
g = YUVdata[YPOS] - invgdif;
b = YUVdata[YPOS] + bdif;

```

为了防止出现溢出，还需要判错计算的结果是否在0-255范围内，做类似下面的判断。

```

if (r>255)
    r=255;

if (r<0)
    r=0;

```

要从RGB24转换成RGB565数据还要做移位和或运算：

```

RGBdata[1] = ( (r & 0xF8) | (g >> 5) );
RGBdata[0] = ( ((g & 0x1C) << 3) | (b >> 3) );

```

3.2 部分查表法

查表法首先可以想到的就是用查表替代上述整型算法中的乘法运算。

```

rdif = fac_1_4075[u];
invgdif = fac_m_0_3455[u] + fac_m_0_7169[v];
bdif = fac_1_779[u];

```

这里一共需要4个1维数组，下标从0开始到255，表格共占用约1K的内存空间。uv可以不需要做减128的操作了。在事先计算对应的数组元素的值的时候计算在内就好了。

对于每个像素，部分查表法用查表替代了2次减法运算和4次乘法运算，4次移位运算。但是，依然需要多次加法运算和6次比较运算和可能存在的赋值操作，相对第一种方法运算速度提高并不明显。

3.3 完全查表法

那么是否可以由YUV直接查表得到对应的RGB值呢？乍一看似乎不太可能，以最复杂的G的运算为例，因为G与YUV三者都相关，所以类似 $G=YUV2G[Y][U][V]$ 这样的算法，一个三维下标尺寸都为256的数组就需要占用2的24次方约16兆空间，绝对是没法接受的。所以目前多数都是采用部分查表法。

但是，如果我们仔细分析就可以发现，对于G我们实际上完全没有必要采用三维数组，因为Y只与UV运算的结果相关，与UV的个体无关，所以我们可以采用二次查表的方法将G的运算简化为对两个二维数组的查表操作，如下：

```
G = yig2g_table[ y ][ uv2ig_table[ u ][ v ] ];
```

而RB本身就只和YU或YV相关，所以这样我们一共需要4个8*8的二维表格，需要占用4乘2的16次方共256K内存。基本可以接受。但是对于手机这样的嵌入式运用来说，还是略有些大了。

进一步分析，我们可以看到，因为在手机等嵌入式运用上我们最终是要把数据转换成RGB565格式送到LCD屏上显示的，所以，对于RGB三分量来说，我们根本不需要8bit这么高的精度，为了简单和运算的统一起见，对每个分量我们其实只需要高6bit的数据就足够了，所以我们可以进一步把表格改为4个6*6的二维表格，这样一共只需要占用16K内存！在计算表格元素值的时候还可以把最终的溢出判断也事先做完。最后的算法如下：

```

y = (YUVdata[Y1POS] >> 2);
u = (YUVdata[UPOS] >> 2);
v = (YUVdata[VPOS] >> 2);

r = yv2r_table[ y ][ v ];
g = yig2g_table[ y ][ uv2ig_table[ u ][ v ] ];
b = yu2b_table[ y ][ u ];

RGBdata[1] = ( (r & 0xF8) | (g >> 5) );
RGBdata[0] = ( ((g & 0x1C) << 3) | (b >> 3) );

```

这样相对部分查表法，我们增加了3次移位运算，而进一步减少了4次加法运算和6次比较赋值操作。在计算表格元素数值的时候，要考虑舍入和偏移等因数使得计算的中间结果满足数组下标非负的要求，需要一定的技巧：采用完全查表法，相对于第一种算法，最终运算速度可以有比较明显的提高，具体性能能提高多少，要看所在平台的CPU运算速度和内存存取速度的相对比例。内存存取速度越快，用查表法带来的性能改善越明显。在我的PC上测试的结果性能大约能提高35%。而在某ARM平台上测试只提高了约15%。

3.4 进一步的思考

实际上，上述算法：

```
RGBdata[1] =( ( r & 0xF8)  | ( g >> 5) );
RGBdata[0] =( ((g & 0x1C) << 3) | ( b >> 3) );
```

中的 (r & 0xF8) 和 (b >> 3) 等运算也完全可以在表格中事先计算出来。另外，YU / YV的取值实际上不可能覆盖满6*6的范围，中间有些点是永远取不到的无输入，RB的运算也可以考虑用5*5的表格。这些都可能进一步提高运算的速度，减小表格的尺寸。另外，在嵌入式运用中，如果可能尽量将表格放在高速内存如SRAM中应该比放在SDRAM中更加能发挥查表法的优势。

4 RGB2YUV ?

目前觉得这个是无法将3维表格的查表运算简化为2维表格的查表运算了。只能用部分查表法替代其中的乘法运算。另外，多数情况下，我们需要的还是YUV2RGB的转换，因为从Sensor得到的数据通常会用YUV数据，此外JPG和MPEG实际上也是基于YUV格式编码的，所以要显示解码后的数据需要的也是YUV2RGB的运算 8) 运气运气。

转载<http://blog.csdn.net/colorant/>

阅读 (23625) | 评论 (1) | 转发 (6) |

上一篇: android编译全过程
下一篇: 基于V4L2的视频驱动开发

0

相关热门文章

A sample .exrc file for vi e...	linux dhcp peizhi roc
IBM System p5 服务器 HACMP ...	关于Unix文件的软链接
游标的特征	求教这个命令什么意思，我是新...
busybox的httpd使用CGI脚本(Bu...	sed -e "/grep/d" 是什么意思...
DB2 9 应用开发（733 考试）认...	谁能够帮我解决LINUX 2.6 10...

给主人留下些什么吧！~~



firepowers

2014-08-19 11:49:29

camera讨论群: 330777216. 欢迎camera达人，驱动调试工程师，效果turning工程师，CAMERA的APK技术牛人入内。纯碎技术，谢绝广告。集聚高通、MTK因特尔等平台各路人马，为打造中国camera第一联盟而不断努力。

[回复](#) | [举报](#)

[评论热议](#)

请登录后评论。

[登录](#) [注册](#)

[关于我们](#) | [关于IT168](#) | [联系方式](#) | [广告合作](#) | [法律声明](#) | [免费注册](#)

Copyright 2001-2010 ChinaUnix.net All Rights Reserved 北京皓辰网域网络信息技术有限公司. 版权所有

感谢所有关心和支持过ChinaUnix的朋友们

京ICP证041476号 京ICP证060528号