

## 第2章 图像素材

### 2.1 图像的基本概念

#### ● 常见图像格式

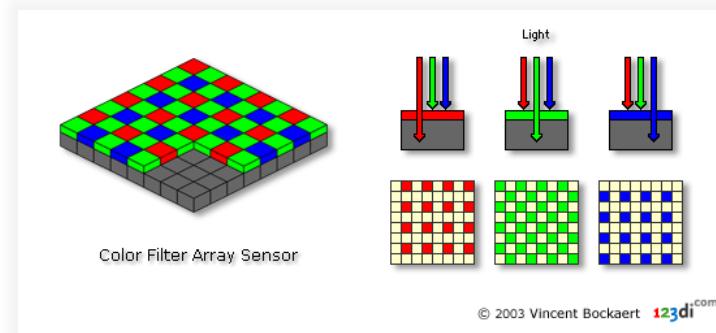
分类	简写	全称	特征
普通类型	GIF	图形交换格式	以超文本标志语言方式显示索引彩色图像
	JPEG	联合图像专家组	用有损压缩方式去除冗余的图像和彩色数据
	PNG	便携式网络图形	无损压缩的位图片形格式
	TIFF	标签图像文件格式	支持很多色彩系统，而且独立于操作系统
	TGA	Tagged Graphics	通道效果与方向性
原始数据	RAW	原始图像	保存信息的损失降到最低
	DNG	数字负片	用于数码相机生成的原始数据文件的公共存档格式
平台特殊格式	BMP	位图	Windows操作系统中的标准图像文件格式
	PICT		PICT是Mac上常见的数据文件格式之一
	PPM	可移植像素图格式	一种linux图片格式
矢量格式数据	WMF	图元文件	是微软公司定义的一种Windows平台下的图形文件格式
more...			

#### ● CCD传感器

CCD (Charge-Coupled Devices, 电荷耦合组件)，具有光电转换、信息存储、延时和将电信号按顺序传送等功能，且集成度高、功耗低，广泛应用于科学、教育、医学、商业、工业、军事和消费领域。

CCD主要是由一个类似马赛克的网格、聚光镜片以及垫于最底下的电子线路矩阵所组成。和传统底片相比，CCD更接近于人眼对视觉的工作方式。

CCD的结构就象一排排输送带上并排放满了小桶，光线就象雨滴撒入各个小桶，每个小桶就是一个像素。按下快门拍照的过程，就是按一定的顺序测量一下某一短暂的时间间隔中，小桶中落进了多少“光滴”，并记在文件中。

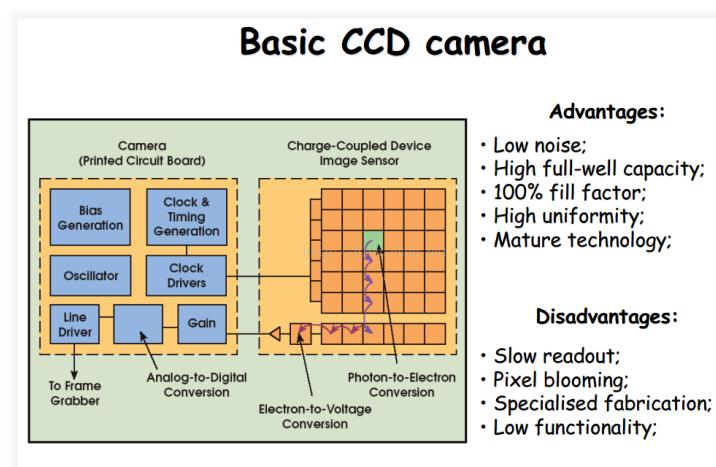


每个感光元件对应图像传感器中的一个像素点。

由于感光元件智能感应光的强度，无法捕获色彩信息，因此彩色CCD图像传感器必须在感光元件上方覆盖彩色滤光片。最常用的做法是覆盖RGB红绿蓝三色滤光片，以 1:2:1 的构成由四个像素点构成一个彩色像素，即红绿、蓝绿间隔排列（如下图），采取这种比例的原因是人眼对绿色较为敏感。

当光照射到CCD硅片上时，在栅极附近的半导体内缠上电子-空穴对，其多数载流子被栅极电压排开，少数载流子则被电场在势阱中形成信号电荷。光注入电荷  $Q_{IP}$  为  $Q_{IP} = \eta q \Delta n_{eo} A T_c$ 。

其中， $\eta$ 为材料的量子效率， $q$ 为电子电荷量， $\Delta n_{eo}$ 为入射光的光子流速率， $A$ 为光敏源面积， $T_c$ 为光注入时间。



- 感光二极管
- 并行信号寄存器：用于暂时储存感光后产生的电荷
- 串行信号寄存器：用于暂时储存并行寄存器的模拟信号并将电荷转移放大
- 信号放大器：用于放大微弱电信号
- 模数转换器：将放大的电信号转换成数字信号

一般的CCD每原色的光度用8位来记录，即其小桶上的刻度有8格，也有的是10位甚至12位，在记录色彩时，尤其是在光线比较暗时，可以更精确。

早期的CCD是隔行扫描的，同一时刻，每两行小桶，只有一行被测量，这样可以提高快门速度，但图像精度大为降低。随着技术的进步，人们已能让CCD记录在几十分之一秒，甚至几千分之一秒的时间里，落进各个“小桶”的“光滴”的量，所以，新的CCD一般都是逐行扫描的。

## ● 数字图像的表示

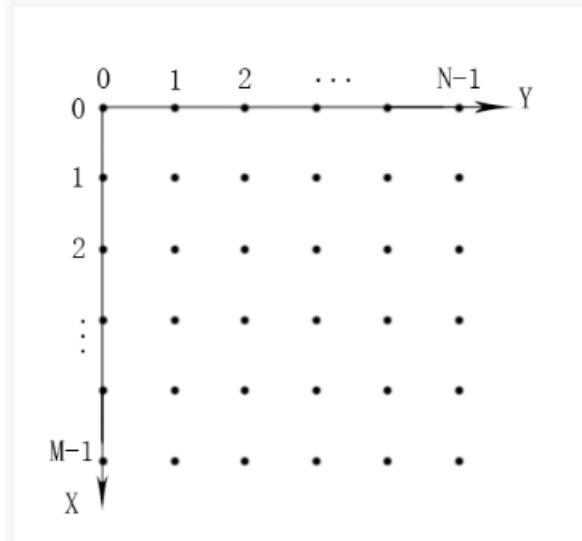
一幅图像可以被定义为一个二维函数  $f(x,y)$ ，其中  $(x,y)$  是空间（平面）坐标，在任何坐标  $(x,y)$  处的幅度  $f$  被定义为图像在这一位置的亮度。图像在  $x$  和  $y$  坐标以及在幅度变化上是连续的。要将这样的一幅图像转换成数字形式，要求对坐标和幅度进行数字化。将坐标值数字化称为取样，将幅值数字化称为量化。因此，当  $x$ 、 $y$  分量及幅值  $f$  都是有限且离散的量时，我们称图像为数字图像。

### — 坐标约定

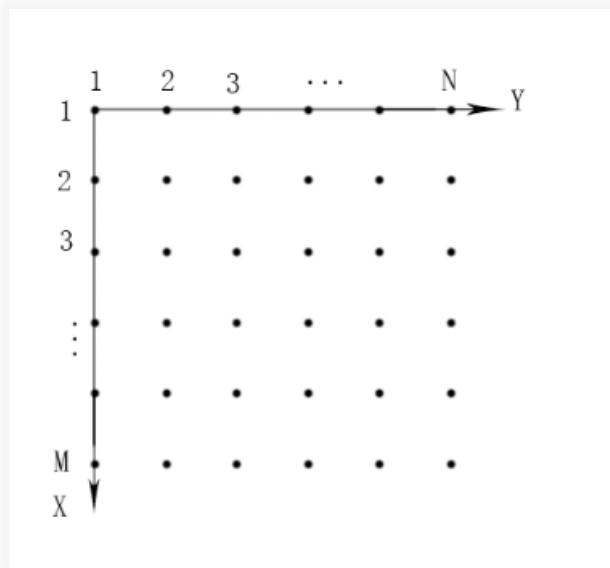
取样和量化的结果都是实数矩阵。假设对一幅图像  $f(x,y)$  采样后，可得到一幅  $M$  行、 $N$  列的图像，我们称这幅图像大小是  $M \times N$ 。相应的值是离散的。为了符号清晰和方便可见，这些离散的坐标都取整数。

这里介绍两种方法表示数字图像。

一种是将图像的原点定义为  $(x,y) = (0,0)$ 。图像中沿着第 1 行的下一坐标点  $(x,y) = (0,1)$ 。符号  $(0,1)$  用来表示沿着第 1 行的第 2 个取样。下图显示这一坐标约定。 $x$  是从 0 到  $M-1$  的整数,  $y$  是从 0 到  $N-1$  的整数。



另外一种是坐标原点为  $(x,y) = (1,1)$ , 如在 matlab 的图像处理工具箱中。这种约定如下图所示:



## - 图像的矩阵表示

根据第一张图的坐标系统, 我们可以得到数字图像的下列表示:

$$f(x,y) = \begin{bmatrix} f(0,0) & f(0,1) & \dots & f(0, N-1) \\ f(1,0) & f(1,1) & \dots & f(1, N-1) \\ \dots \\ f(M-1,0) & f(M-1,1) & \dots & f(M-1, N-1) \end{bmatrix}$$

等式右边是定义的一幅数字图像。阵列中每个元素都被称为图像元素、图画元素或像素。

将数字图像表示成 MATLAB 矩阵:

$$f = \begin{bmatrix} f(1,1) & f(1,2) & \dots & f(1,N) \\ f(2,1) & f(2,2) & \dots & f(2,N) \\ \dots & \dots & \dots & \dots \\ f(M,1) & f(M,2) & \dots & f(M,N) \end{bmatrix}$$

## ● 图像分辨率

图像分辨率指图像中存储的信息量，是每英寸图像内有多少个像素点，分辨率的单位为 **PPI (Pixels Per Inch, 像素每英寸)**，包含的数据越多，图形文件的长度就越大，也能表现更丰富的细节。它和图像的宽、高尺寸一起决定了图像文件的大小及图像质量。

### — 常见分辨率

名称	规格	长宽比	应用
VGA	640 × 480	4: 3	广泛用于公共场所的屏幕显示
CCIR601 DV PAL	720 × 576	4: 3	PAL DV 和 PAL DVD
CCIR601 PAL full	768 × 576	4: 3	平方采样网格比的PAL
SVGA	800 × 600	4: 3	主流投影仪分辨率
XGA	1024 × 768	4: 3	广泛用于笔记本的屏幕分辨率
QVGA	1280 × 960	4: 3	主要用于手机等便携设备屏幕
UXGA	1600 × 1200	4: 3	常用于15英寸屏幕的笔记本
1080 HDTV	1920 × 1080	4: 3	高分辨率数字电视格式
2K	2048 × 1080	1.9: 1	数字影院放映及显示设备
4K	4096 × 2160	1.9: 1	数字影院放映及显示设备

## ● 图像类型

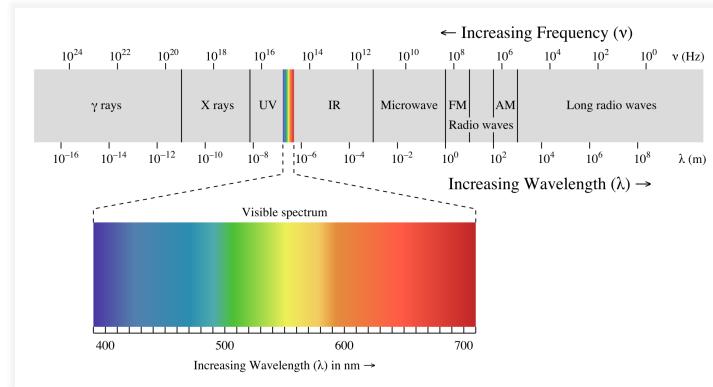
- 二值图像**: 图像上的每一个像素只有两种可能的取值：布尔值0和（其中0表示黑色，1表示白色），所以计算机中二值图像的数据类型通常为1个二进制位。
- 灰度图像**: 灰度图像矩阵元素的取值范围通常为 [0, 255]。因此其数据类型一般为8位无符号整数的 (int8)，这就是人们经常提到的 256 灰度图像。“0”表示纯黑色，“255”表示纯白色，中间的数字从小到大表示由黑到白的过渡色。
- 索引图像**: 为了节省表示图像RGB信息的空间用调色板存储彩色信息 (RGB值)，数据区只存储当前象素的色彩在调色板中的位置，这样就省了很多字节。调色板是颜色的索引，多用于8位图像，即可表示256种颜色。如下图，图像仅仅使用了16种颜色，右图是该图像的调色板：



- **真彩色RGB图像:** 最常见的彩色图像为24位RGB模式, 即每个像素由红 (R) 、绿 (G) 、蓝 (B) 三个分量表示, 分量介于 [0, 255]。可以表示  $2^{24} = 16777216$  种颜色。

## 2.2 颜色空间

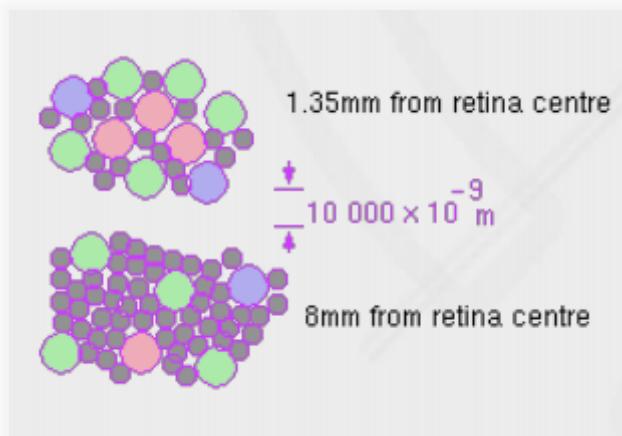
- **可见光谱**



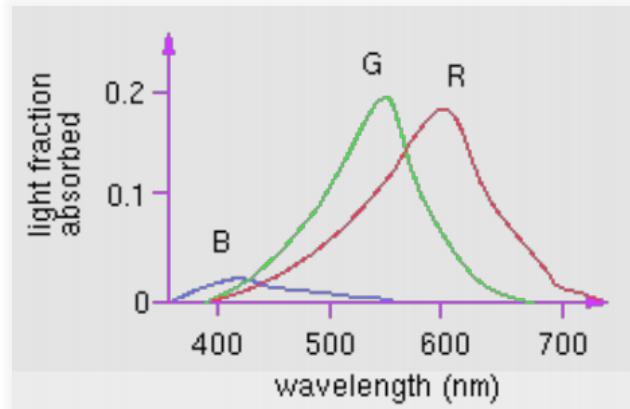
- **人眼结构**

两种视觉细胞: 视杆细胞 (rods) 和视锥细胞 (cones), Rods 数量多于 cones, 所以人眼对光线敏感度 > 对颜色敏感度。

- rods: ~100,000,000, 对光线敏感;
- cones: ~6,000,000—7,000,000, 在强光下工作, 对颜色敏感。



其中有三种视锥细胞, 缺失一种都会导致色盲。



## ● 颜色空间

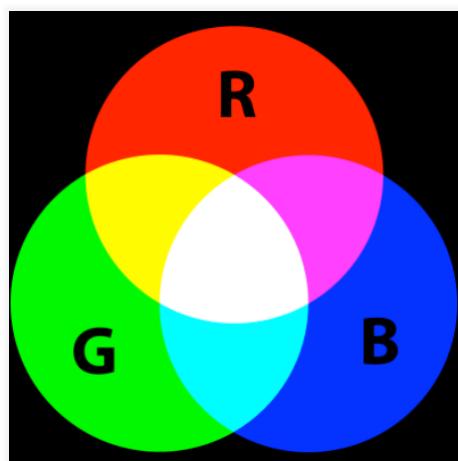
- 设备依赖 (device dependent) 的颜色空间: RGB, CMY, HSV
- 无设备依赖 (device independent) 的颜色空间: CIE XYZ, CIE Lab, CIE YUV

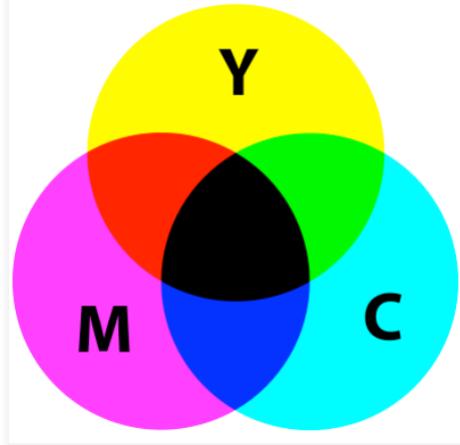
### - RGB & CMYK

RGB是一种依赖于设备的颜色空间：不同设备对特定RGB值的检测和重现都不一样，因为颜色物质（荧光剂或者染料）和它们对红、绿和蓝的单独响应水平随着制造商的不同而不同，甚至是同样的设备不同的时间也不同。

CMYK是彩色印刷时采用的一种套色模式，利用色料的三原色混色原理，加上黑色油墨，共计四种混合叠加，形成所谓“全彩印刷”。

RGB为加色模型，CMYK为减色模型。





### CMYK 转 RGB

$$x = \frac{X}{X + Y + Z}$$

$$y = \frac{Y}{X + Y + Z}$$

$$z = \frac{Z}{X + Y + Z} = 1 - x - y$$

### - CIE XYZ

因为人类眼睛有响应不同波长范围的三种类型的颜色传感器，所有可视颜色的完整绘图是三维的。但是颜色的概念可以分为两部分：明度和色度。例如，白色是明亮的颜色，而灰色被认为是不太亮的白色。换句话说，白色和灰色的色度是一样的，而明度不同。

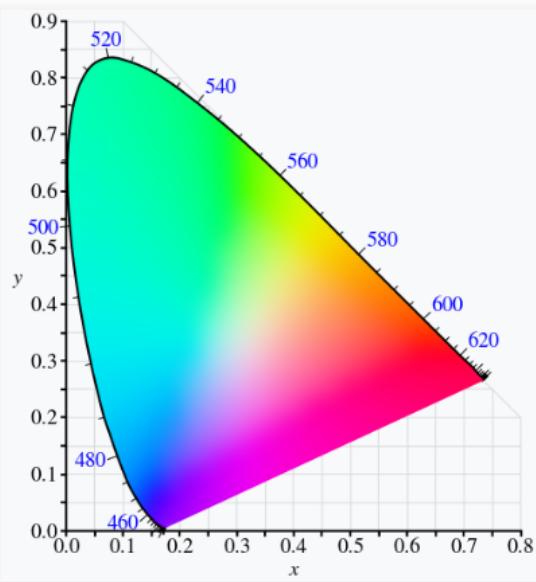
CIE XYZ色彩空间故意设计得Y参数是颜色的明度或亮度的测量。颜色的色度接着通过两个导出参数x和y来指定，它们是所有三个三色刺激值X、Y和Z的函数所规范化的三个值中的两个：

$$x = \frac{X}{X + Y + Z}$$

$$y = \frac{Y}{X + Y + Z}$$

$$z = \frac{Z}{X + Y + Z} = 1 - x - y$$

下面的图象展示了相对色度图。外侧曲线边界是光谱轨迹，波长用纳米标记。注意这个色度图是指定人类眼睛如何体验给定频谱的光的工具。它不能指定物体的颜色（或印刷墨水），因为在观察物体的时候看到的色度还依赖于光源。数学上，x和y是投影坐标，色度图的颜色占据了实投影平面的一个区域。



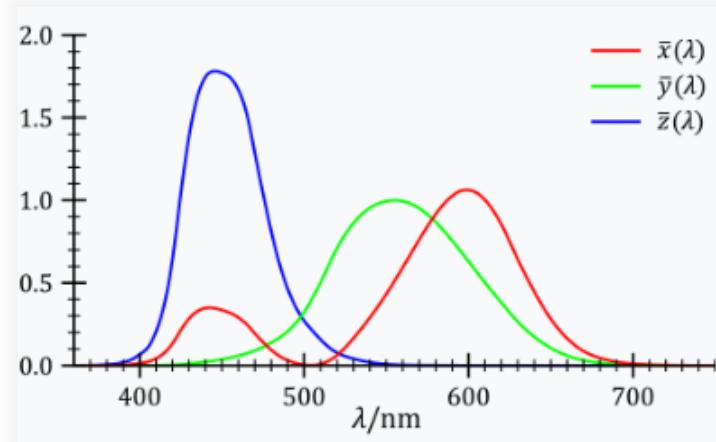
在使用CIE RGB颜色匹配函数开发了人类视觉的RGB模型之后，特殊委员会的成员希望开发出与CIE RGB色彩空间有关的另一个色彩空间。它假定 Grassmann 定律成立，这个新空间通过线性变换而有关于CIE RGB空间。新空间将以三个新颜色匹配函数来定义。带有 $I(\lambda)$ 的颜色的对应的XYZ 值为给出为：

$$X = \int_0^{\infty} I(\lambda) \bar{x}(\lambda) d\lambda$$

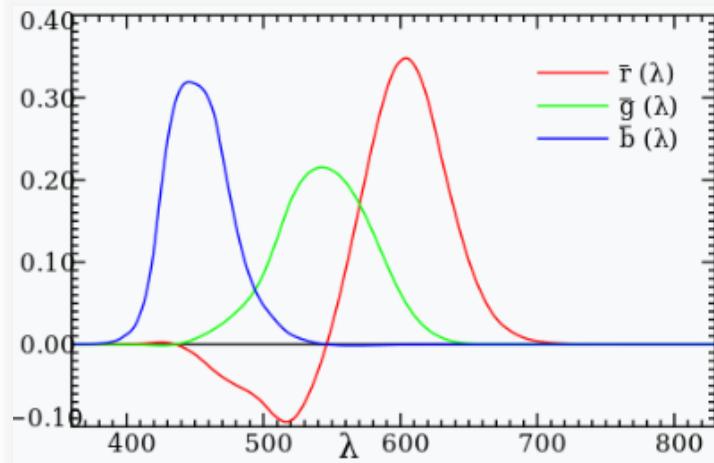
$$Y = \int_0^{\infty} I(\lambda) \bar{y}(\lambda) d\lambda$$

$$Z = \int_0^{\infty} I(\lambda) \bar{z}(\lambda) d\lambda$$

在380 nm到780 nm之间的（间隔5 nm）CIE 1931标准色度观察者XYZ函数：



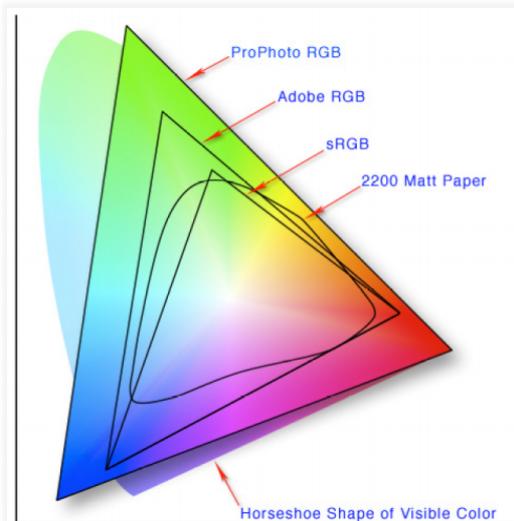
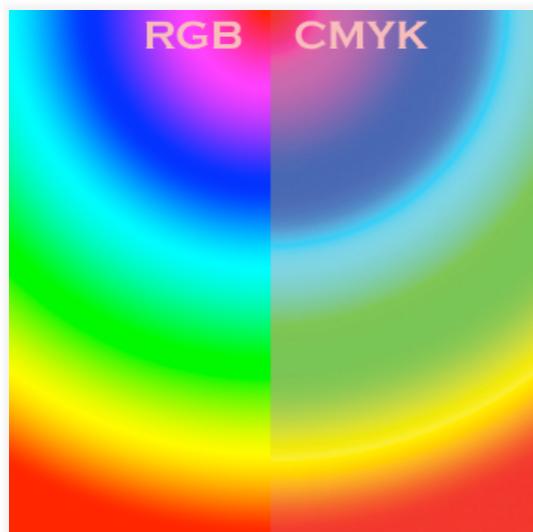
CIE 1931 RGB颜色匹配函数。颜色匹配函数是匹配水平刻度标示的波长的单色测试颜色所需要的原色数量：



从CIE RGB空间到XYZ空间的线性变换。CIE特殊委员会确定了标准变换如下：

$$\begin{bmatrix} X \\ Y \\ Z \end{bmatrix} = \frac{1}{b_{21}} \begin{bmatrix} b_{11} & b_{12} & b_{13} \\ b_{21} & b_{22} & b_{23} \\ b_{31} & b_{32} & b_{33} \end{bmatrix} \begin{bmatrix} R \\ G \\ B \end{bmatrix} = \frac{1}{0.17697} \begin{bmatrix} 0.49 & 0.31 & 0.20 \\ 0.17697 & 0.81240 & 0.01063 \\ 0.00 & 0.01 & 0.99 \end{bmatrix} \begin{bmatrix} R \\ G \\ B \end{bmatrix}$$

#### - RGB :: CMYK :: XYZ



#### - YUV

YUV是被欧洲电视系统所采用的一种颜色编码方法（属于PAL），是PAL和SECAM模拟采用的。在现代彩色电视系统中，通常采用三管彩色摄影机或彩色摄影机进行取像，然后把取得的彩色图像信号经分色、分别放大校正后得到RGB，再经过电路得到亮度信号Y和两个B-Y（即U）、R-Y（即V），最后发送端将亮度和色差三个信号分别进行编码，用同一信道发送出去。这种色彩的表示方法就是所谓的YUV色彩空间表示。采用YUV色彩空间的重要性是它的亮度信号Y和色度信号U、V是分离的。

YUV的发明是由于彩色电视和黑白电视的过渡时期，黑白电视只有Y频道，也就是灰阶值，而彩色电视规格是以YUV/YIQ的格式处理彩色电视图像，把UV视作表示彩度的C，如果忽略掉C讯号，剩下的Y就是黑白电视讯号。

RGB要求人眼对于色彩的感应，YUV则重视视觉对于亮度的敏感程度，Y表示亮度，UV表示彩度。

### YUV 与 RGB 的转换

U 和 V 元件可以被表示成原始的 R, G, 和 B (R, G, B 为 γ 预校正后的)：

$$\begin{aligned} Y &= 0.299 * R + 0.587 * G + 0.114 * B \\ U &= -0.169 * R - 0.331 * G + 0.5 * B + 128 \\ V &= 0.5 * R - 0.419 * G - 0.081 * B + 128 \end{aligned}$$

$$\begin{bmatrix} Y' \\ U \\ V \end{bmatrix} = \begin{bmatrix} 0.299 & 0.587 & 0.114 \\ -0.14713 & -0.28886 & 0.436 \\ 0.615 & -0.51499 & -0.10001 \end{bmatrix} \begin{bmatrix} R \\ G \\ B \end{bmatrix}$$

## 2.3 图像压缩方法与标准

图像压缩方法可以分为有损压缩和无损压缩，不管采用哪种压缩方法，两者的本质内容都是一样的，其基本原理都是在不影响文件的基本使用的前提下，通过某种特殊的编码方式，只保留原数据中一些“关键点”，去掉数据中的重复的、冗余的信息，从而达到压缩的目的。下面将简要介绍有损压缩和无损压缩以及一些常用的算法。

### • 无损压缩

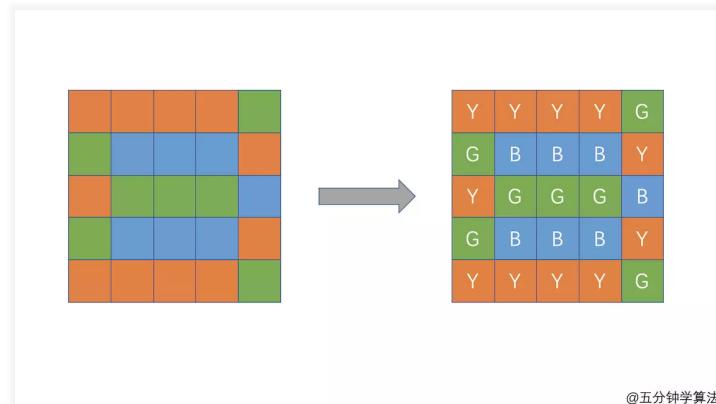
无损压缩格式是利用数据的统计冗余进行压缩的，经过无损压缩的内容可完全恢复原始数据而不引起任何失真，但压缩率是受到数据统计冗余度的理论限制，一般为2:1到5:1。这类方法广泛用于文本数据、程序和特殊应用场合的图像数据（如指纹图像、医学图像等）的压缩。即指使用压缩后的数据进行重构（或称还原、解压缩），重构后的数据与原来的数据完全相同；无损压缩用于要求重构的信号与原始信号完全一致的场合。

基于一些信息的理论，无损压缩有一些通用的编码方式，如RLC编码（游程编码）、VLC编码（可变长编码）、LZW编码（串表压缩算法）、字典编码和算术编码等。

#### - RLC编码

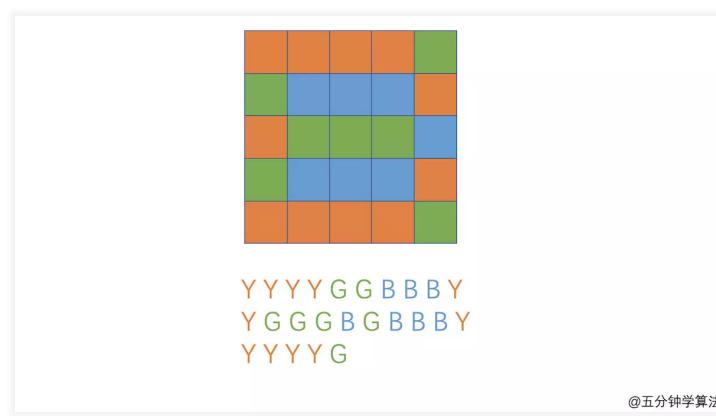
RLC编码适合的场景是数据本身具有大量连续重复出现的内容，因为它使用重复字节和重复的次数来简单描述来代替重复的字节。比如数据中有大量的0出现，RLC编码无需记录所有的0，而是用两位（count, value）即可表示这一堆连续的0。

我们再来看一个简单的例子：编码一个在 5 \* 5 方块上使用三种颜色绘制的图像。



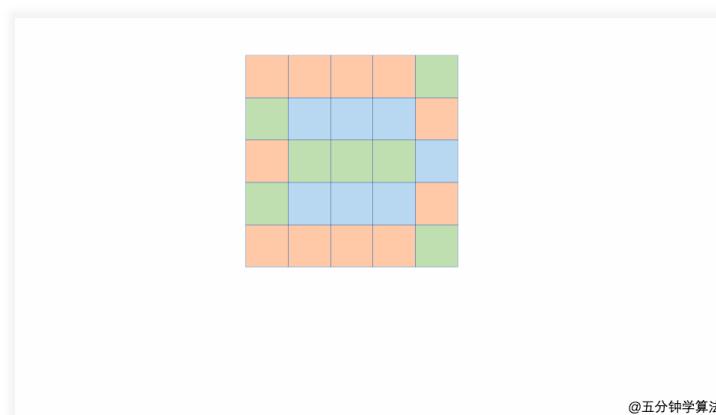
@五分钟学算法

根据方块不同的颜色匹配不同的字母。这里使用Y代表黄色，使用G代表绿色，使用B代表蓝色。那么，根据这样的规则，上图的图形编码就变成了 25 个字母，如下图所示。



@五分钟学算法

接下来，我们可以通过将第一个“YYYY”部分表示为“Y4”，这样就可以将其缩短两个字符。按照这种操作，上图的 25 个字符就能缩短为 20 个字符了。



@五分钟学算法

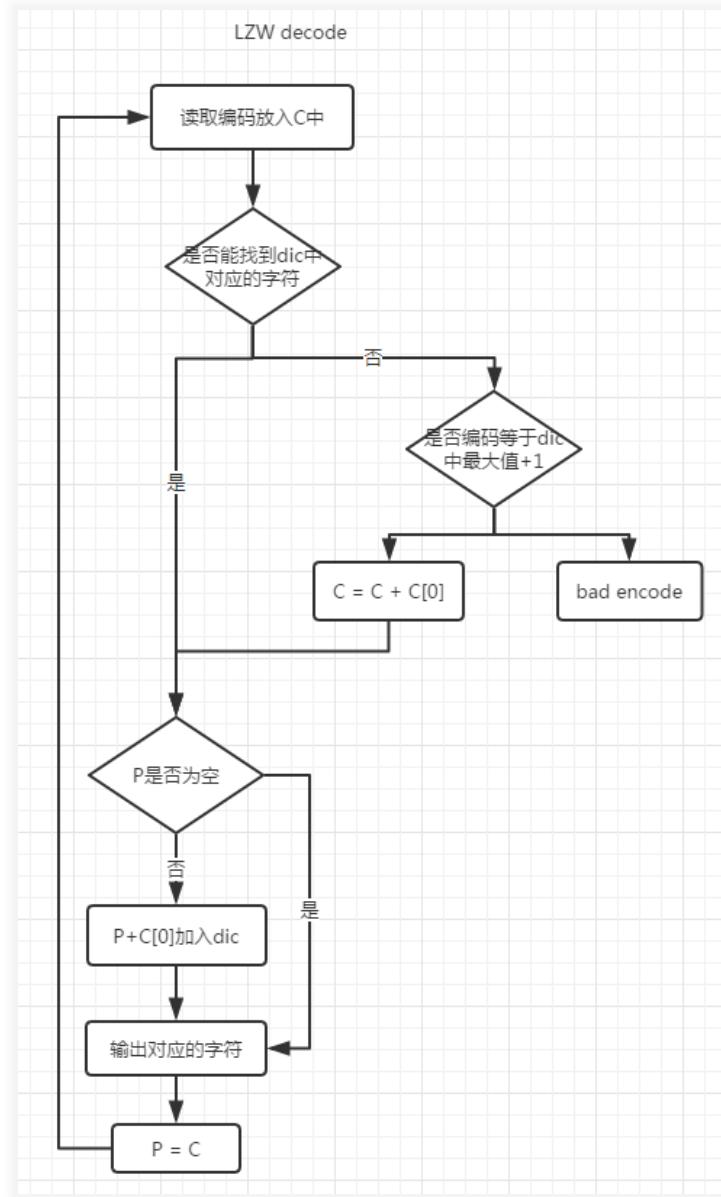
这样，如果我们知道每行有 5 个方块，原始图像就可以从代码中提取出来了，也就是将图片进行解压。

然而，游程编码也不是万能的，它也有它的适用性与局限性。如果对连续性极其差的数据进行游程编码，字符数将不减反增。

RLC编码尽管简单并且对于通常的压缩非常低效，但它有的时候却非常有用，JPEG就使用了这种编码方式。

## - LZW编码

LZW编码首先建立一个字符串表，把每一个第一次出现的字符串放入串表中，并用一个数字来表示，这个数字与此字符串在串表中的位置有关，并将这个数字存入压缩文件中，如果这个字符串再次出现时，即可用表示它的数字来代替，并将这个数字存入文件中。压缩完成后将串表丢弃。下面给出了LZW的具体算法流程：



LZW编码把出现过的字符串映射到记号上，这样就可能用较短的编码来表示长的字符串，从而实现压缩。例如对于字符串“ABABAB”，可以看到子串AB在后面重复出现了，这样我们可以用一个特殊记号表示AB，例如数字2，这样原来的字符串就可以表示为“AB22”。如果我们规定数字0表示A，数字1表示B，最后得到的压缩后的数据就可以表示为“0122”。这样我们就得到了一个字典：

Symbol	String
0	A
1	B
2	AB

有了压缩后的编码0122，结合字典，我们就能解码得到原字符串“ABABAB”。

除此之外，还可以把两种算法结合起来使用，比如gzip使用LZ77算法和Huffman编码进行PNG、PPP、HTTP、SSH的压缩。

## ● 有损压缩

有损压缩利用了人类对图像或声波中的某些频率成分不敏感的特性，在压缩过程中会损失一定的信息，却不会影响人对图像或声波的整体感知；虽然不能完全恢复原始数据，但是所损失的部分对理解原始图像的影响缩小，却换来了大得多的压缩比，即指使用压缩后的数据进行重构，重构后的数据与原来的数据有所不同，但不影响人对原始资料表达的信息造成误解。有损压缩适用于重构信号不一定非要和原始信号完全相同的场合，如图像和声音的压缩，这是因为其中包含的数据往

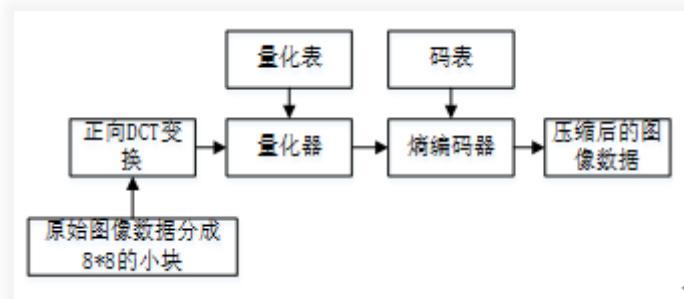
往多于我们的视觉系统和听觉系统所能接收的信息，丢掉一些数据而不至于对声音或者图像所表达的意思产生误解但可大大提高压缩比。

有损编码通常会分为量化和编码两个过程。在量化时，会去掉一些信息比如高频部分。而编码的算法可以是离散余弦变换、离散小波变换、Karhune–Loeve变换等。

## ● 静止图像编码的国际标准

静止图像编码的国际标准有JPEG和JPEG-2000两种。

JPEG标准是一个通用的静止图像压缩标准，可适用于所有连续色调的静止图像压缩和存储。它既可以用于有损压缩，也可以用于无损压缩。JPEG标准的目的在于支持用于大多数连续色调静态图像压缩的各种各样的应用，这些图像可以是任何一个色彩空间，用户可以调制压缩比，并能达到或者接近技术领域中领先的压缩性能，且具有良好的重建质量。这个标准的另一个目标是对普遍实际的应用提供易处理的计算复杂度。在ISO公布的JPEG标准方案中，JPEG包含了两种压缩方式。一种是基于DCT变换的有损压缩编码方式（如下图所示），它包含了基本功能和拓展系统两部分；另一种是基于空间DPCM（差分脉冲编码调制）方法的无损压缩编码方式。



而JPEG-2000标准的主要动机是利用基于小波变换的压缩技术提供一套全新的图像编码系统。在中高码率上能够提供很好的压缩图像质量，但随着码率的降低（例如低于0.25bpp），图像的主观质量下降很快。与原来JPEG的标准相比，除了采用小波变换外，JPEG-2000增加了一批新功能，使得JPEG-2000的应用范围大大扩展，从数码相机、预出版到医用图像。其主要性能和新增功能包括：

1. 高压缩效率，压缩率到0.1bit/pixel时，仍能获得相当不错的重建图像。
2. 图像传输从有损到无损。
3. 分辨率伸缩性，质量（像素精度）伸缩性。
4. 在不解压缩的情况下，随机的访问码流中的数据段。
5. 能够定义兴趣区域（如医用图像中的某区域），对该区域采用高分辨率甚至无损编码。
6. 使用重同步标记以便增加在高误码率信道（如移动通信）中的鲁棒性。
7. 高质量、保真度彩色图像处理（更大的图像尺寸，更多的bit像素）。
8. 使用alpha通道以便满足未来的图形处理和因特网需要。
9. 信息嵌入，嵌入非图像信息，如说明文字、声音等数据信息。
10. 图像加密。

但是JPEG2000并没有完全取代JPEG，相反，目前似乎还是JPEG用得比较广泛。

## 2.4 HDR图像

高动态范围图像（High-Dynamic Range，简称HDR），相比普通的图像，可以提供更多的动态范围和图像细节，根据不同的曝光时间的LDR（Low-Dynamic Range）图像，利用每个曝光时间相对应最佳细节的LDR图像来合成最终HDR图像，能够更好的反映人真实环境中的视觉效果。

现实真正存在的亮度差，即最亮的物体亮度，和最小的物体亮度之比为 $10^8$ ，而人类的眼睛所能看到的范围是 $10^5$ 左右，但是一般的显示器、照相机能表示的只有256种不同的亮度。计算机在表示图象的时候是用8bit(256)级或16bit(65536)级来区分图象的亮度的，这区区几百或几万无法再现真实自然的光照情况。

HDR文件是一种特殊图形文件格式，它的每一个像素除了普通的RGB信息，还有该点的**实际亮度信息**。

保存在高动态范围图像中的数据经常是线性的，这就意味着它们表示亮度或者radiance的相对或者绝对值（gamma 1.0）。高动态范围图像每个颜色通道需要比传统图像更多的数据位，这是因为它的线性编码以及需要表示从 $10^{-4}$ 到 $10^8$ （人眼可见亮度范围）甚至是更大范围的数值（如：CMOS图像传感器能提供高达**120分贝**的动态范围）。

在HDR中经常使用16位“half precision”或者32位浮点数表示高动态范围像素。但是，如果使用合适的传递函数进行变换，一些应用中的高动态范围像素可以用**10–12**位表示亮度，8位表示色度，并且不会带来任何可见的量化误差。



HDRI图像示例

## ● HDR的存储编码格式

HDRI (High-Dynamic Range Image) 就是记录采用了HDR技术的图象数据文件。常用的HDRI文件有OpenEXR、Radiance RGBE、FloatTIFF三种格式。

HDR：对应的编码方式主要是 RGBE / XYZE：

- TIFF：对应的编码方式主要是 RGB / LogLuv24 / LogLuv32
- EXR：对应的编码方式主要是 HalfRGB

HDR对以往的运算方法做了以下二方面的重大改进：

- **使用16bit、32bit的数据来提高像素数据的精度。**继续使用8bit的数据来记录像素的特征不能满足HDR数据所需要的高精度运算的要求，在这种情况下，我们考虑使用16bit、32bit的数据记录来提高像素数据的精度。使用了更多的数据来保存像素特征之后，无论是像素的对比度还是像素可以体现的色彩数目都有了巨大的提高。
- **图象数据采用浮点数据。HDR真正的巨大变革来自于浮点数据的引入。**我们可以采用浮点方式来处理和存放亮度数据，抛弃不准确的整数数据；同时计算机在引入浮点数据来存储像素的各个参数并且在运算的全过程都使用浮点数据，这样就可以有效的提高据的精确度。

### - HDR格式

编码方式为RGBE，即通过ldr的rgb值，配合一个该组值对应的exponent来还原最初的HDR颜色信息。在一个32bits的RGBE中，通常是R8G8B8E8的占位方式。假设场景中的原始scene-referred的颜色为RsGsBs，那么将其向RGBE的映射操作如下式所示：

$$E = \lceil \log_2(\max(R_s, G_s, B_s)) + 128 \rceil R = \lfloor \frac{256R_s}{2^{E-128}} \rfloor G = \lfloor \frac{256G_s}{2^{E-128}} \rfloor B = \lfloor \frac{256B_s}{2^{E-128}} \rfloor$$

对于由RsGsBs到RGB的还原对上述计算进行逆操作即可

XYZE与RGBE类似，只不过其存储与编码的是XYZ空间中的颜色值，而不是RGB空间。

由于XYZ空间中不需要存储负值（RGB与XYZ颜色空间的区别），因而使用上述only positive的方法来存储的话使用XYZE会比RGBE得到更大的色域，因为不会存储无效值。

### - TIFF格式

基本的TIFF存储格式比较简单，直接使用3个32bits的RGB通道来存储原始的信息，因而永远是scene-referred的。但是这种方式太过于简单，占用空间太大并且无法进行有效的数据压缩（压缩算法对于3个float的数值压缩比不高），因而就有了基于其的改进方法，LogLuv24、LogLuv32。

LogLuv的方法是将一个颜色值转化为luminace (Y) 及其对应的u、v值来存储，即Luv空间。Luv空间是RGB与XYZ之后的另外一个颜色空间，其与XYZ、RGB均可以进行相互转换，其空间内的颜色值具有的特点是可以直接通过Luv之间的欧氏距离而得到两个颜色之间的差异程度，因而在图像处理与识别中很有用。注意，这里的原始scene-referred空间为XYZ，因而在存储与解析时就涉及到XZY-Luv之间的转换。

- **LogLuv24：一个颜色值用24bits进行存储，其中的bit分割为：L10、C14。**

其中的 L10 即中将Luminace值（即XYZ空间中的Y值）转化为一个log函数映射的10bits浮点数进行存储；C14 为

uv中的u值通过14bits进行保存的一个浮点数。这里由于只存储了uv中的一个值，因而另外v值需要通过一个uv查询表获得，这样就需要在LogLuv24的存储与解码时均需要使用这样的一个uv对应表来协助完成。

$$L_{10} = \lfloor 64(\log_2 Y_s + 12) \rfloor Y_s = 2^{\frac{L_{10}}{64}-12}$$

- **LogLuv32**: 一个颜色值通过32bits进行存储，其中的bit分配为：s1、L15、u8、v8。

其中的 s1 为符号位，用来标识Luminance值的正负情况（当然，由XYZ空间转换过来的一定为正）； L15 为15个bit位的L值； u8、v8 分别为表示uv值的两个byte，这里就不再需要查uv表了。

$$L_{15} = \lfloor 256(\log_2 Y_s + 64) \rfloor Y_s = 2^{\frac{L_{15}}{256}-64}$$

## — EXR格式

Exr格式是由 ILM (Industrial Light & Magic, 工业光魔) 给出的HDR存储格式，现基本上已经成为影视产业中HDR应用的一个标准格式。关于exr格式的HDR编解码，现在有开源的的OpenExr可以用，其中还包含了对exr格式的压缩等操作，各种使用exr的引擎或renderer (比如unity、blender、pbrt) 中使用的都是它。

常见的OpenEXR文件是 **FP16** (16bit Float Point, 也被称为half Float Point) 数据图像文件，每个通道的数据类型是 FP16，一共四个通道64bpp，每个通道1个bit位用来标志“指数”，5个bit用来存放指数的值，10个bit存放色度坐标的尾数，其动态范围从  $6.14 \times 10^{-5}$  到  $6.41 \times 10^4$ 。

在OpenEXR的算法里面共使用16bit来表示光照数据。虽然看起来和使用16bit亮度通道运算位数相同，但是OpenEXR巧妙的采用了1个bit位用来标志“指数”，5个bit用来存放指数的值，10个bit存放色度坐标  $(u, v)$  的尾数。这样就轻易的解决了浮点数值由于位数少而精度不高的问题，大大地拓宽的在FP16下的动态范围。

根据实际的计算结果：在正规化的情况下OpenEXR可以提供和人眼基本相同的动态范围，最暗到最亮是  $0.00006103515625 \times (6.14 \times 10^{-5})$  到  $65504 \times (6.41 \times 10^4)$ ，动态范围是9.03；非正规化条件下，OpenEXR可以提供从最暗到最亮的数值从  $0.000000059604644775390625 \times (5.96 \times 10^{-8})$  到  $65504 \times (6.41 \times 10^4)$ ，化为动态范围表示就是12。

## ● HDR应用场景

室内拍摄时，画面中包括室较暗的场景和室外十分明亮的景物HDR可以使室内场景和室外景物在张照片中都得到很好地展现。总之，只要是那些画面相对稳定而又存在较大明暗反差的场景，都以用HDR的手法来拍摄并通过HDR后期处理来合成。

### — 光摄影

由于风光摄影常面对大光比画面，很适合HDR技术。

### — 花草等静态摄影

这类题材于色彩变化和光影丰富，同时又相静止，利于前期拍摄。

### — 日落日出和夜景

此类题材画面中天空和面景物明暗反差很大，数码相机的动态范围满足不了如此大的动态范围，是HDR最适合的应用题材。尤其是拍夜景，HDR可以很好地展现一般夜景无法记录下的天空微弱光线与地面源所产生的对比效果。

## ● 色调映射

在查看高动态范围图像时会遇到一个问题，CRT、LCD、打印机以及其它图像显示方法只能显示有限动态范围的图像，如果简单的将真实世界的整个亮度域线性压缩到照片所能表现的亮度域内，则会在明暗两端同时丢失很多细节，这显然不是所希望的效果。因此需要通过色调映射将高动态范围图像“转换”成可以查看的图像，根据场景的当前亮度，**将HDR映射到LDR上**，并保证图像细节不丢失，不失真。

整个Tone Mapping的过程就是首先要根据当前的场景推算出场景的平均亮度，根据这个平均亮度选取一个合适的亮度域，再将整个场景映射到这个亮度域得到正确的结果。其中最重要的几个参数：

**Middle grey:** 整个场景的平均灰度，关系到场景所应处在亮度域。

**Key:** 场景的Key将决定整个场景的亮度倾向，倾向偏亮亦或是偏暗。

首先我们需要做的是计算出整个场景的平均亮度，有很多种计算平均亮度的方法，目前常用的的是使用log-average亮度来作为场景的平均亮度，通过下面的公式可以计算得到：

$$\bar{L}_\omega = \frac{1}{N} e^{\sum_{x,y} \log(\delta + L_\omega(x,y))}$$

其中 $L_\omega(x,y)$ 是像素点 $x,y$ 的亮度， $N$ 是场景内的像素数， $\delta$ 是一个很小的数用来应对像素点纯黑的情况

$$L(x,y) = \frac{a}{\bar{L}_\omega} L_\omega(x,y)$$

上面的公式用来映射亮度域， $a$ 即是前面所讲的Key值，用来控制场景的亮度倾向，一般来说，会使用几个特定的值，0.18是一个适中的Key，0.36或者0.72相对偏亮，0.09甚至0.045则是偏暗。完成映射的场景为了满足计算机能显示的范围还要将亮度范围再映射到[0,1]区间，可以通过下面的公式简单的得到[0,1]区间的亮度。

$$L_d(x,y) = \frac{L(x,y)}{1 + L(x,y)}$$

不过这样得到的结果并不总是令人满意的，所以一般扩展为如下面的公式，公式中的参数 $L_{white}$ 用来控制场景中的曝光，凡是亮度超过 $L_{white}$ 的像素都会被置为纯白。如果 $L_{white}$ 的值非常大，则这个参数在公式中将不起任何作用，如果非常小则场景将变为几乎全白。 $L_d$ 即为我们所要的映射后的 $x,y$ 像素点的亮度值。

$$L_d(x,y) = \frac{L(x,y)(1 + \frac{L(x,y)}{L_{white}^2})}{1 + L(x,y)}$$

## • 计算摄像学

计算摄像学 (Computational Photography) 是一门将计算机视觉、数字信号处理、图形学等深度交叉的新兴学科。旨在结合计算、数字传感器、光学系统和智能光照等技术，从成像机理上来改进传统相机，并将硬件设计与软件计算能力有机结合，突破经典成像模型和数字相机的局限性，增强或者扩展传统数字相机的数据采集能力，全方位地捕捉真实世界的场景信息。



计算摄像学的各个维度

笼统来说，拓展了传统数码摄影中的某个或多个因素的维度来成像的方法，就是计算摄影学。总结来看，成像包含空间、时间、视角和深度、以及光谱等多个维度。

HDR就是计算摄像学中的一种办法，拓展的是传统成像中的光圈大小。

## 2.5 主流图像格式分析

### • GIF

## - 历史

- 1987年6月15日，gif在CompuServe公司诞生，取名87A，改进了具有256色的黑白图像传输以及使用图形控制扩展（GCE），通过定时延迟实现了动画效果。
- 起初，gif要解决的问题是：如何使计算机显示图像同时节省内存？gif的设计初衷是用于在计算机之间交换图像，这也是它得名为“图形交换形式”的原因，甚至在线的第一张彩色图片的格式就是gif格式的图片。
- 1995年，Unisys Corp.公司对它拥有的专利，LZW算法，也就是gif实现压缩的核心算法，收取专利费。这一举措导致gif图片一度险些被淘汰（特别是开发人员们绕过LZW算法研究出了PNG格式图片时）。
- 2004年，专利的收费在全球都暂停了，但是gif在静态图片的市场已经被淘汰，但是凭借其革命性的压缩算法，它依旧垄断了动态图像的市场直至今日。

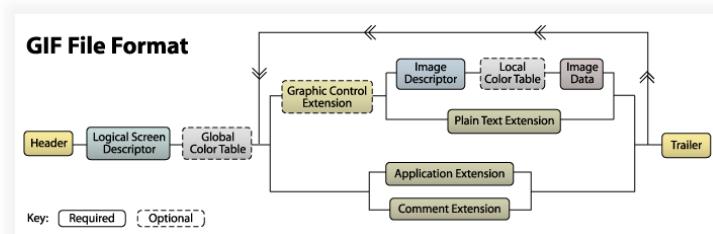


第一张火遍互联网的gif

## - 格式详解

GIF文件由一系列数据块组成。前两个块是固定长度和固定格式。后面的是可变长度，但可以自我描述。它们由一个标识块类型的字节，一个有效载荷长度字节，一个有效载荷组成。

下图显示了所有不同类型的块以及它们在文件中的位置。（中间箭头循环的部分可以重复多次）



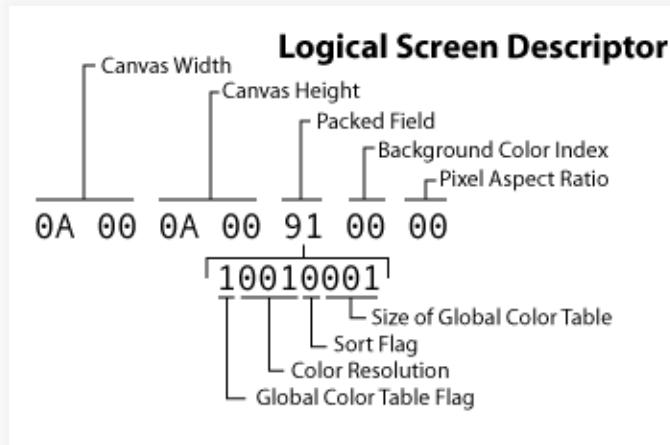
### 1. Header 标头块

前三个字节称为签名。这些值应始终为“GIF”（即 $47 = "G"$ ,  $49 = "I"$ ,  $46 = "F"$ ）。

接下来的三个指定 用于编码图像的规范版本。

### 2. Logical Screen Descriptor 逻辑屏幕描述符

该块告诉解码器此图像将占用多少空间。

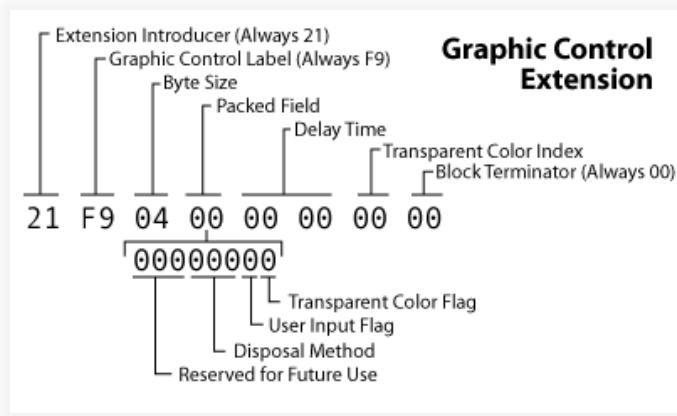


### 3. Global Color Table 全局色表（可选）

实质上是颜色的索引表，一次读取三个字节以获取每种颜色。比如，第一种颜色是#FFFFFF，该值的索引为0。

#### 4. Graphic Control Extension 图形控件扩展

图形控件扩展块用于指定透明度设置和控制动画。



##### • 透明度设置

在颜色表中设置一种颜色，该颜色将转换为“不可见墨水”。然后，在绘制图像时，每当遇到这种特殊颜色时，背景就会被显示出来。

第一步，透明色标志设置为1，即图形控件扩展的打包字节中的最低位。这将告诉解码器我们希望图像具有透明成分。第二步，设置特定颜色作为不可见墨水，这种颜色必须在活动颜色表中，并且可以通过将该值设置为颜色表中颜色的索引，在“透明颜色索引”字节中指定其值。

假设白色的索引值为0，并且想要设定白色为透明颜色，那么透明颜色索引块 (Transparent Color Index)指定值00。

##### • 动画

打包字段(Packed Field) 存储三个值。前三个（最高）位被“保留以备将来使用”，因此这些位保留为零。

接下来的三位指示处置方法，指定当移至下一个图像时当前图像数据会发生什么情况，可以通过用户自定义。值为1，这告诉解码器将图像保留在原处并在其顶部绘制下一个图像，值为2意味着应该将画布恢复为背景色，值为3表示解码器应在绘制当前图像之前将画布还原到其先前状态。

#### 5. Image Descriptor 图像描述符

一个GIF文件可能包含多个图像。每个图像都以一个图像描述符块开头。该块正好是10个字节长。

第一个字节是图像分隔符2C。接下来的8个字节表示下一张图像的位置和大小。

最后一个字节也是一个压缩字段。字节中的第一个（最高有效）位是本地颜色表标志。将此标志设置为1允许您指定后面的图像数据使用与全局色表不同的色表。

第二位是隔行标志。隔行扫描在显示器上的效果是的第一遍会立即出现，首先将图形显示为模糊，然后将其锐化，随后的遍历将填充行。

#### 6. Local Color Table 本地色表

仅对紧随其后的图像数据块有效的色表

#### 7. Image Data 图像数据

LZW编码后的图像数据，LZW的具体编码过程已在之前的部分展示，这里将略过。

该块的第一个字节是LZW最小代码大小。该值用于解码压缩的输出代码。其余字节代表数据子块。数据子块是1–256字节的组。子块中的第一个字节告诉您随后有多少个字节的实际数据。

#### 8. Plain Text Extension 纯文本扩展名

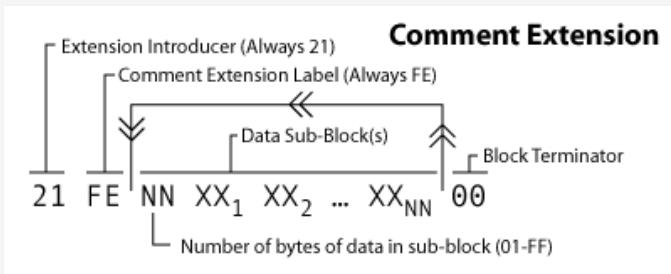
GIF89规范允许您指定要在下一张图像上叠加的文本标题。此功能从未实现。

#### 9. Application Extension 应用扩展

GIF89规范允许将特定于应用程序的信息嵌入到GIF文件本身中。此功能使用不多。

#### 10. Comment Extension 注释扩展

可以将ASCII文本嵌入到GIF文件中，并且有时用于包含图像描述，图像信用或其他人类可读的元数据，例如图像捕获的GPS位置。



第一个字节是扩展介绍者21。下一个字节始终是注释标签FE。数据子块用于注释。最后一个字节 00代表注释到达末尾。

### 11. Trailer 尾部

始终未3B，标记结尾。

## - 特征

- **有限的颜色表**: 单个GIF图像中最多只能引用256种颜色。尽管此有限的调色板可以减小文件大小，并且在屏幕上观看时完全可以接受。
- **颤动**: 用于限制文件大小的另一个技巧是抖动。抖动并不是GIF的真正功能，它只是GIF图像中经常使用的一种技术。通过将较少数量的彩色“点”混合在一起，该技术可用于创建更大色深的幻觉。当能够显示的颜色少于原始图像中显示的颜色时，则使用相邻像素的图案来模拟显示不足的颜色的外观。
- **LZW**: 无损压缩算法。
- **透明度**: 可以在处理显示设备的图像时忽略调色板中一种颜色的指定。
- **隔行扫描**: 通过首先显示图像的低分辨率版本并逐渐显示完整版本，这种机制可以使图像在屏幕上显示更快。
- **动画**: 为文件头添加了一些增强功能，使Netscape等浏览器可以按定时和/或循环顺序显示多个GIF图像。这种机制允许制作较小的粗略动画，这是横幅广告中经常使用的非常流行的功能。
- **解析度**: 大多数GIF图像的分辨率在72到90 dpi之间，非常适合在屏幕上查看，但不足以用于印刷使用。

## ● PNG

## - 历史

- **动机**: 前文提及的Unisys凭借LZW压缩算法的专利对支持gif的软件创作者们收取版税。Thomas Boutell领导的非正式互联网工作组致力于设计gif格式的替代品——更好，更小，可扩展且免费
- 1995年1月4日，PNG的第一个草案出炉。当时称其为“便携式位图格式”。一个三字节的签名，使用块号而不是块名称，最大像素深度为8位，并且没有指定的压缩方法。
- 在这之后PNG的草案不断修改与完善。到1995年2月开始，已经制作了七份草案，PNG格式也得到了解决。
- 1996年，Internet工程任务组（IETF）将0.95版发布为Internet草案，10月中旬，互联网编号分配机构（IANA）正式批准将“image / png”作为正式的互联网媒体类型，将image / gif和image / jpeg合并为Web的非实验图像格式。

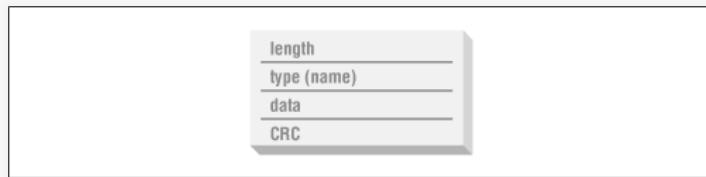


PNG的代表图片

## — 格式详解

PNG图像的基本构建块是块。除了文件中的前8个字节之外，PNG图像仅由块组成。

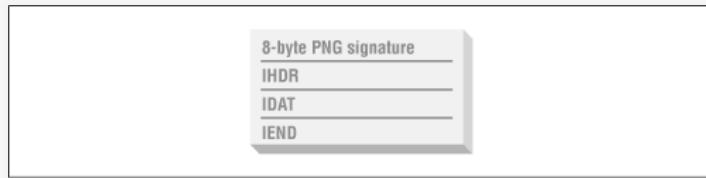
- 前八个字节是PNG文件签名，是PNG文件的重要组成部分，因为无论文件名如何，它们都可以被标识为PNG文件并且可以检测到最常见的文件传输损坏类型。
- 下图为PNG格式文件中的一个块。每个块都具有相同的结构：4字节长度，4字节类型，0到2,147,483,647字节之间的数据和一个4字节的循环冗余校验值（CRC）



长度字段仅指数据字段的长度，而不是块类型或CRC。块类型（名称）通常是助记符，例如IHDR。数据字段将在下文描述。CRC覆盖块类型字段和块数据，并且即使没有块数据也始终存在。

长度字段和CRC值的组合已经足以检查PNG文件的基本完整性。

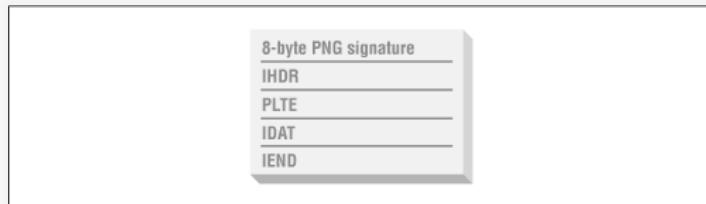
- 两种简单的PNG格式文件



这种文件由PNG签名和只有三种块类型组成：图像头块IHDR、图像数据块IDAT、以及图像结尾部分IEND

- IHDR是PNG图像中的第一个块，包括有关图像类型的所有详细信息：高度和宽度，像素深度，压缩和过滤方法，隔行扫描方法，是否具有alpha（透明）通道，以及它是真彩色，灰度还是彩色映射（调色板）图像。
- IDAT包含图像的所有压缩像素数据。在大多数图像中，压缩数据会分成多个IDAT块以提高鲁棒性。块的CRC位于末尾，遇到大IDAT的流时应用程序可以迫使用户等待整个块到达。
- IEND不包含任何数据，仅表示图像中没有更多块，可以检查PNG文件是否完整以及内部是否一致。

以上类型足以构建带或不带alpha通道的真彩色和灰度PNG文件，但如果要创建属于调色板的图像需要如下结构：



PLTE，调色板块。PLTE仅包含一系列红色，绿色和蓝色值，其中0表示黑色，255表示全强度。

以上被提及的块也被成为关键快，在PNG的发展过程中出现了很多辅助块帮助实现更多功能。

- 像素格式

PNG图像中的像素是数字，可以是调色板中样本数据的索引，也可以是样本数据本身。

一个像素的样本数据由一到四个数字组成的元组组成。无论像素数据表示调色板索引还是显式样本值，这些数字都称为通道，并且图像中的每个数字都以相同的格式编码

通道数取决于图像是灰度图像还是彩色图像以及是否具有alpha通道。

- 索引 = 1 通道，可以是 1,2,4,8 bpc
- 灰度 = 1 通道，可以是 1,2,4,8,16 bpc
- Gray + Alpha = 2 通道，可以是 8 或 16 bpc
- Truecolor (RGB) = 3 通道，可以是 8 或 16 bpc
- RGBA = 4 通道，可以是 8 或 16 bpc

- 压缩

PNG的压缩过程是完全无损的。它分两个阶段完成：预测（也称为过滤），然后进行压缩。

- 过滤（预测）

PNG格式使用称为“过滤”的格式的增量编码。对于像素的每条扫描线，当前像素以与左侧像素，上方像素和左侧上方像素某种关系编码。

过滤采用ABC，然后使用它来预测x的值。我们将x替换为的值是预测值与实际值之间的差。

每一行可以自行选择最佳的滤波方法，从而可以产生最少量的唯一符号。

这些滤波是按通道而不是按像素完成的。这意味着将滤镜应用于扫描线像素的所有红色值，然后分别应用于扫描线的所有蓝色值。

- 压缩

一旦在扫描线上进行了过滤，就将其传递给 LZ77 算法的后代，即 DEFLATE。该算法将 LZ77 编码与霍夫曼编码器结合在一起，先进行 LZ77 编码后进行 huffman 编码。

压缩通过两个步骤完成：用指针匹配和替换重复的字符串，根据使用频率用新的加权符号替换符号。

- LZ77

三个字节以上的重复串才进行编码，否则不进行编码。

Lz77匹配合查找的时候用了哈希表，一个head数组记录最近匹配的位置和prev链表来记录哈希值冲突的之前的匹配位置。

- huffman

对Lz77得到的压缩后结果，需要统计字符生成编码表，根据码表对内容进行编码，具体的压缩大小在于精细分配结构体的位域来实现Huffman编码的压缩效果的。

## - 特征

- alpha通道

不再为每个像素存储三个字节，而是需要四个字节：红色，绿色，蓝色和alpha或RGBA。GIF支持简单的二进制透明度（任何给定的像素都可以是完全透明或完全不透明的）而PNG为一般图像提供了254级的部分可变透明性

- 伽玛和色彩校正

伽玛校正是指能够校正计算机解释颜色值的方式差异的能力。

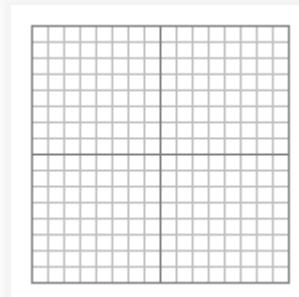
伽玛信息是将单个数字与计算机显示系统相关联的一种方法，目的是表示潜伏在图形卡的数模转换器（RAMDAC）中以及显示器的高压电子枪和显示磷光体中的棘手的物理现象。

Gamma只是总体亮度的第一近似值，但对于休闲用户来说通常就足够了。要求更高的用户将另外希望调整各个红色，绿色和蓝色通道中的差异，即所谓的色度，PNG也支持。

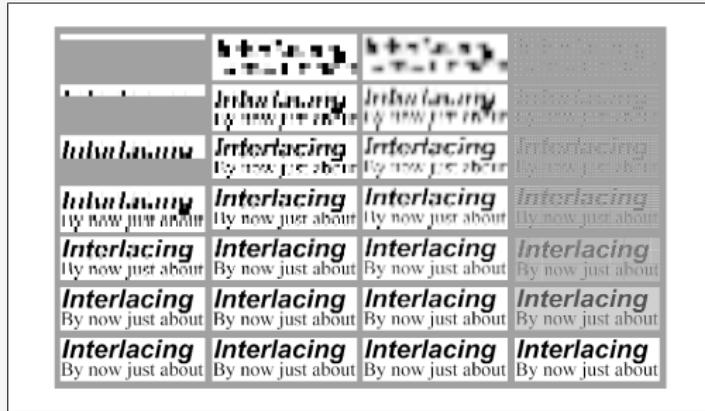
- 隔行和逐行显示

GIF的隔行扫描：首先以非常拉伸的块状外观出现，并逐渐被填充，直到显示全分辨率图像为止。这种方式导致了尽管GIF的第一遍包含图像数据的八分之一，但八倍的因子完全是以垂直分辨率为代价的。在水平方向上，每条线在显示后即达到全分辨率，这意味着第一遍中的每个像素都将拉伸八倍。

PNG的隔行扫描方法是二维的，并且在一半以上的遍历中根本不涉及拉伸，它们的像素始终显示为正方形：



下图将从左到右依次显示GIF，正常PNG，带插值的PNG和具有稀疏显示的PNG的隔行显示的比较。



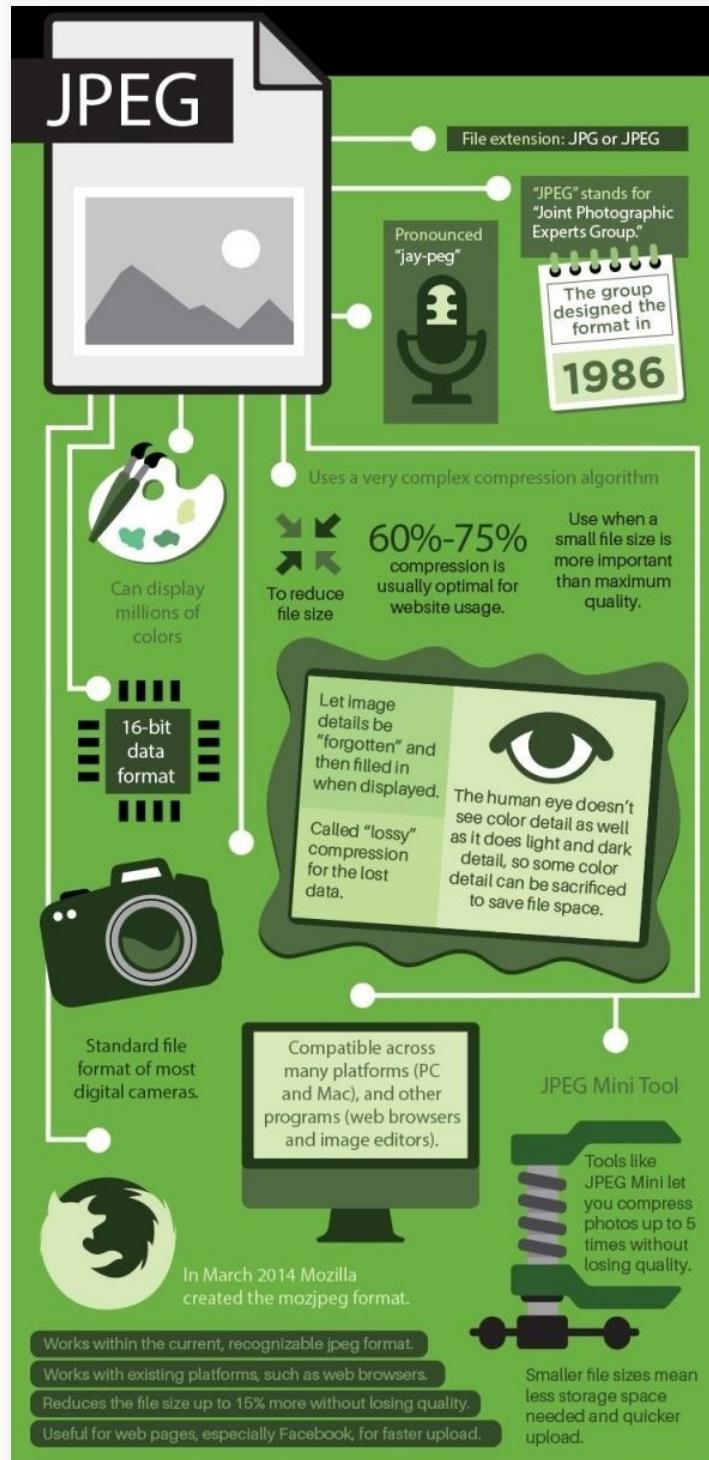
- **无损压缩**

PNG的压缩是最好的压缩方法之一，不会丢失图像数据，也无需支付专利或其他许可费用

- **JPEG**

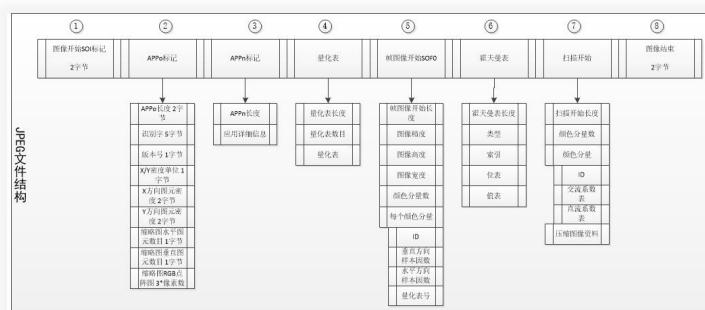
- **历史**

- 1986年，PEG委员会由CCITT和ISO标准组织成立，旨在为图像压缩设定全球标准。这项工作在1991年初已完成技术上的工作，后来被批准为国际标准组织（ISO）。最初，JPEG是针对全色静止帧应用程序的，其平均压缩比达到15: 1。



## — 格式详解

JPEG的文件结构整体如下图所示：



PEG文件大体上可以分成两个部分：标记码和压缩数据。如图所示，第一行为标记码，下方为压缩数据。

- SOI, Start of Image, 图像开始, 固定值0xFFD8
- APP0, Application, 应用程序保留标记0, 固定值0xFFE0

信息	长度	值
数据长度	2字节	①~⑨个字段的总长度
标识符	5字节	固定值0x4A46494600, 即字符串“JFIF”
版本号	2字节	一般是0x0102, 表示JFIF的版本号1.2
X和Y的密度单位	1字节	0: 无单位; 1: 点数/英寸; 2: 点数/厘米
X方向像素密度	2字节	取值范围未知
Y方向像素密度	2字节	取值范围未知
缩略图水平像素数目	1字节	取值范围未知
缩略图垂直像素数目	1字节	取值范围未知
缩略图RGB位图	长度可能是3的倍数	缩略图RGB位图数据

- APPn, Application, 应用程序保留标记n, 其中n=1~15 (任选), 固定值0xFFE1~0xFFFF

信息	长度	值
数据长度	2字节	①~②个字段的总长度
详细信息	数据长度-2字节	内容不定

- DQT, Define Quantization Table, 定义量化表, 固定值0xFFDB

信息	长度	值
数据长度	2字节	字段①和多个字段②的总长度
量化表	数据长度-2字节	

- SOF0, Start of Frame, 帧图像开始, 固定值0xFFC0

信息	长度	值
数据长度	2字节	①~⑥六个字段的总长度
精度	1字节	每个数据样本的位数, 通常是8位
图像高度	2字节	图像高度 (单位: 像素)
图像宽度	2字节	图像宽度 (单位: 像素)
颜色分量数	1字节	1: 灰度图; 3: YCrCb或YIQ; 4: CMYK
颜色分量信息	颜色分量数×3字节 (通常为9字节)	

- DHT, Define Huffman Table, 定义哈夫曼表, 固定值0xFFC4

信息	长度	值
数据长度	2字节	字段①和多个字段②的总长度
哈夫曼表	数据长度-2字节	

- DRI, Define Restart Interval, 定义差分编码累计复位的间隔, 固定值0xFFDD

信息	长度	值
数据长度	2字节	固定值0x0004, ①~②两个字段的总长度
MCU块的单元中的重新开始间隔	2字节	设其值为n, 则表示每n个MCU块就有一个RSTn标记。第一个标记是RST0, 第二个是RST1等, RST7后再从RST0重复。

- SOS, Start of Scan, 扫描开始, 固定值0xFFDA

信息	长度	值
数据长度	2字节	①~④两个字段的总长度
颜色分量数	1字节	应该和SOF中的字段⑤的值相同, 即1: 灰度图; 3: YCrCb或YIQ; 4: CMYK
<b>颜色分量信息</b>		
颜色分量ID	1字节	
直流/交流系数符号	1字节	高4位: 直流分量使用的哈夫曼树编号; 低4位: 交流分量使用的哈夫曼树编号
<b>压缩图像数据</b>		
谱选择开始	1字节	固定值0x00
谱选择结束	1字节	固定值0x3F
谱选择	1字节	在基本JPEG中总为00

- EOI, End of Image, 图像结束, 固定值0xFFD9

## — 特征

- 高度可控的压缩度。用户独立选择比率质量/文件大小
- 小文件, 有损压缩
- 格式兼容, 并且可以在所有计算机, 平板电脑和移动设备上的任何浏览器
- 适用于具有许多颜色和对比度过渡的全彩色逼真图像
- 图像质量高, 压缩程度小

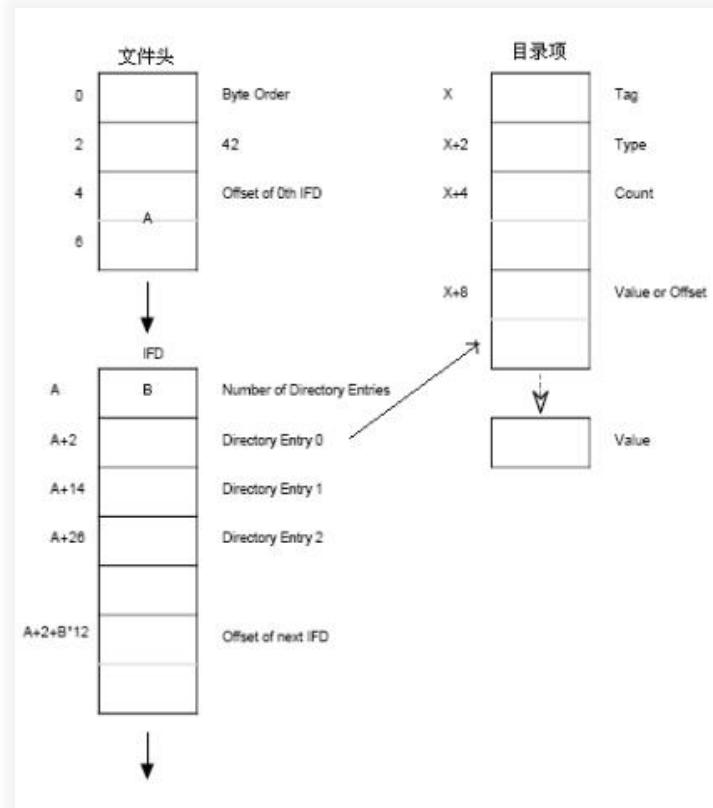
## • TIFF

### — 历史

- TIFF文件格式是出于以下考虑而开发的: 一种与文档扫描仪结合使用的标准格式。
- 1986年, TIFF规范由Aldus Corporation发布, 作为存储由扫描仪和桌面出版应用程序创建的黑白图像的标准方法。
- 1987年4月发布了TIFF的第一个广泛使用的版本4.0, 增加了对未压缩的RGB彩色图像的支持
- 1988年8月发布了TIFF Revision5.0, 存储调色板彩色图像并支持LZW压缩算法。
- Aldus被Adobe收购, 因此Adobe现在拥有版权, 但他们尚未发布TIFF的任何新版本。

### — 格式详解

TIFF图像是靠指针连接来组织数据的, 一般来说由四部分组成: 文件头、文件目录、目录内容、图像数据。具体结构如下:



- 具体内容详细说明

```

头项:
0000 4D4D-II表示intel的小序字节
0002 42 002A-版本号
0004 1st IFD offset 0000000A表示IFD有10项

IFD:
000A 图象宽度 0100 0004 00000001 00000300
0016 图象高度 0101 0004 00000001 00000200
0022 每像素的位 0102 0003 00000003 00000082 --在s2的位置用三个WORD来表示
0023 压缩方式 0103 0003 00000001 00000001 --1表示不压缩
003A 光度滴定 0106 0003 00000001 00000002 --对于RGB的是2
0046 带的偏移 0111 0004 00000001 00000088 --相对于整个文件,
                                                 也就是在s8的位置是带的数据,
                                                 而整个文件组成一带
0052 像素颜色成分 0115 0003 00000001 00000003 -对于RGB图形, 这个值是3,
                                                 表示一个像素有三个颜色成分
005e 每带行数 0116 0004 00000001 00000200 --这里表示一带有512行,
                                                 也是一个图象在一带中表示
006a 每带的字节数 0117 0004 00000001 00120000 --1179648字节=768*512*3
0076 像素的颜色分量如何存储 011C 0003 00000001 00000001 -1表示按照
                                                 RGBRGB...的顺序来存储

上面4个字节表示不了的数据
0082 每像素的位 08 00 08 00 08 00 --对应上面的像素位
Image Data:
00000088 带1中的数据
End of example

```

## 特征

- 色彩空间

- 艺术线条 (纯黑白)
- 灰阶
- 伪彩色, 从1位到8位 (在Photoshop中也称为调色板颜色或索引颜色)
- RGB
- YCbCr
- CMYK
- CIELab

- 压缩

支持大量无损压缩算法, 如 PackBits, LZW, CCIT。

- 兼容性

大多数TIFF文件格式都是未压缩的，旨在独立于任何特定的硬件或软件。可以调整TIFF图像的大小，增加每英寸的点数（DPI），而不会降低质量。

可以在计算机之间以及从一个应用程序到另一个应用程序之间传输它们，而不会损失质量或任何兼容性问题。

几乎所有图像编辑应用程序和许多操作系统都支持TIFF，因此，它通常是归档数字照片的首选格式。

- **文件大小**

单个TIFF文件可能会占用100兆字节（MB）或更多的存储空间，这是同等JPEG文件的很多倍。

## • 四种主流格式的用法比较

	PNG	GIF	JPEG	TIFF
编辑，调色板图像，快速保存	✓	✓		✓
编辑，真彩色图像，快速保存	✓			✓
”最终”编辑，最佳压缩	✓			
编辑，最大的编辑器可移植性	✓	✓		✓
Web，真彩色图像，不透明			✓	
Web，调色板图像，不透明	✓	✓		
Web，具有”打开/关闭”透明度的图像	✓	✓		
Web，部分透明的图像	✓			
Web，跨平台颜色一致性	✓			
Web，动画		✓		
Web，最大的浏览器可移植性	✓	✓	✓	
Web，尽可能少的图像	✓		✓	

## 2.6 小结

### • 图像存储的重要概念

- 矩阵：数字图像的表示
- 像素：数字图像的基本元素
- 分辨率：图像中存储的信息量
- 图像类型
  - 二值图像
  - 灰度图像
  - 索引图像
  - 真彩色RGB图像
- 色彩空间
  - 设备依赖：RGB, CMY, HSV
  - 无设备依赖：CIE XYZ, CIE Lab, CIE YUV
- 常见图像格式
  - 普通类型：GIF, JPEG, PNG, TIFF, TGA
  - 原始数据：RAW, DNG
  - 平台特殊格式：BMP, PICT, PPM
  - 矢量格式：WMF
- HDR图像
  - 编码格式：HDR、TIFF、EXR

- 应用场景：光摄影、静物摄影、日出日落、夜景
- 图像压缩
  - 无损压缩：RLC编码、LZW编码、VLC编码、字典编码、算术编码
  - 有损压缩：DCT变换、DWT变换、Karhune–Loeve变换

## • 图像格式转换工具

- ACDSEE
- imagemagik (Linux)
- XnView