作者：马健  
邮箱：[stronghorse\_mj@hotmail.com](mailto:stronghorse_mj@hotmail.com)发布：2013.02.15  
最后更新：2013.02.19

目录  
一、BMP  
二、GIF、PNG  
三、JPEG（JPG）  
四、JPEG 2000  
五、TIFF  
六、DjVu  
七、PDF  
八、小结

CEP、CV、UV中都支持多种图像格式，因此经常有人问我相同的问题：不同的图像格式究竟有什么不同？保存图像的时候究竟应该选择哪种图像格式？

本文希望能够对以上问题给出浅显的回答，当然是否已经浅到能让您理解的程度，就要看造化了。

**一、BMP**

BMP是微软提出的一种图像格式，全称是bitmap，所以也翻译成“位图”。BMP具有下列特点：

1. 面向屏幕显示，所以格式本身与显卡技术的发展相关。比如说在常见的图像格式中，只有BMP支持16位色，因为有整整一代显卡是16位色的。
2. 虽然新一代Windows API已经可以直接显示JPG、PNG等格式的图像，但早期GDI函数只能显示BMP图像，因此一般图像显示、图像编辑软件都会把其他格式的图像文件在内存中解码成BMP，并对BMP中的像素点阵信息进行编辑、修改 、显示，存盘时再存成需要的格式。

在图像压缩方面，虽然按照微软的规定，BMP本身支持无损的RLE（Run Length Encode，行程编码）压缩算法，但除了在Windows 2000源代码中见到几个采用这种压缩算法的BMP文件外，我见到的其他所有BMP文件都是非压缩的，大家一般也都以为BMP是非压缩图像格式，并因此认为BMP文件长度都很大。

之所以很少有人对BMP采用RLE压缩，我猜测可能的原因包括：

1. RLE本身并不是一种很有效率的压缩方法，在最坏情况下压缩后的数据长度比原始数据长度还要长得多。
2. 大家都在偷懒，不压缩当然比压缩更省事。而且非压缩的BMP处理起来更方便（基本上就是一个二维数组），显示更快。

在不压缩的情况下，BMP文件长度为：  
　　　　　BMP文件长度 = BMP文件头长度 + 调色板字节数 + 像素点阵信息长度　　　（1）

在式1中，BMP文件头长度是固定的，调色板只有在图像是调色板图像时才有，16位色、24位色、32位色是没有调色板的。调色板图像允许是单色、16色、256色，而调色板中每个色彩占用4字节，因此即使在最大256色情况下，调色板占用的字节数也只有1024字节=1KB，对文件长度影响不大。

式1中的像素点阵信息长度可以用下式进行估算（结果单位为字节）：  
　　　　　像素点阵信息长度≈图像的像素高度×图像的像素宽度×每像素占用的字节数　　（2）

注意式2中用的是“约等于”，不是“等于”，因为一般BMP文件在存储时，需要把式2中  
　　　　　图像的像素宽度×每像素占用的字节数  
部分（扫描线宽度）圆整成4的倍数，而式2忽略了这种圆整，因此算出来的字节数可能会比实际占用字节数略少。不过这种差异不是很大，所以为了便于计算，就做了简化。真正的计算公式应该是：  
　像素点阵信息长度=像素高度×((像素宽度×每像素占用的bit数+31)÷32×4)　（3）

从式2、3可以看出，BMP文件中真正占大头的是像素点阵信息，而在不采用压缩的情况下，BMP的像素点阵信息与图像的像素高度、像素宽度、每像素占用的字节数相关，与图像内容无关。而要减小BMP文件的长度，可以从以下方面入手：

1. 缩图，即减小图像的像素高度、像素宽度。
2. 减色，即减小每像素占用的字节数。如24位彩图每像素占用24位即3字节，减色成256色后每像素只占用1字节，减成16色后每像素占1/2字节，减色成单色后每像素占用1/8字节，这中间的差异还是非常可观的。

当然天下没有白吃的午餐，以上两种方法都会造成图像质量的变化，而且这种变化不可逆，因此以后想后悔是不行的。不过只要操作得当，其中巨大的收益也值得一干。如在国外地下电子书扫描界大名鼎鼎的操作教程《The Scan and Share tutorial》中，就主张先用300 DPI进行灰度扫描以提高扫描效率（普通家用扫描仪超过300 DPI后扫得很慢），再用软件放大一倍达到600 DPI，然后经过裁边、纠斜、统一尺寸、居中等处理后，减色成单色图像，再转换成PDF或DjVu。现在你知道为什么国外扫描的书籍看起来为什么这么清晰了吧？600 DPI啊！

在不考虑后续图像压缩的情况下，图像尺寸放大一倍，再从24位色减色成单色（1位色），则虽然图像放大造成图像像素面积是原来的4倍，但每像素占用的字节数只有原来的1/24，因此总的文件长度约为原来的1/6。如果扫描是按256级灰度扫描的，扫完后图像尺寸放大一倍，再从256级灰度减色成单色（1位色），则虽然图像放大造成图像像素面积是原来的4倍，但每像素占用的字节数只有原来的1/8，因此总的文件长度约为原来的1/2。

《The Scan and Share tutorial》中鼓吹先放大再减色，其实是钻了一个心理上和软件上的空子：当图像像素尺寸超过屏幕显示区像素尺寸（说白了就是一屏显示不下整个图像）时，人们通常会选择缩图显示，而对于有良知的常规软件来说，单色图像缩图后通常会变成256级灰度，用中间的灰度点来掩饰缩图造成的锯齿等缺陷。因此 虽然放大、减色后文字边缘看起来可能没有原始扫描图像清晰、圆润，总觉得有点毛刺，但实际缩图显示的时候，效果并不比原来差，但文件长度却可以大为减小。

CX的大图版PDG，其实也是缩图后增加灰度级别的一个例子：大图版都是从清晰版PDG缩图来的，但是黑白二色的清晰版缩图后存为16级灰度或256级灰度，在中间灰度级别的过渡下看起来效果还不错。快速版PDG同样是从清晰版缩图得到，但为了使用DjVu压缩文字层而强制将缩小后的图像仍然保持黑白二色，没有中间过渡，看起来就差了很多。

BMP格式是微软自己定义的，在Windows API中也提供了常规操作接口，因此图像的获取、保存、调入都很简单，很多“数字图像处理”相关教材也都提供了相关例子代码，有兴趣的搜一下就有了。

**二、GIF、PNG**

GIF全称Graphics Interchange Format，译做“图形交换格式”，发音[dʒif]；PNG全称Portable Network Graphics，译做“可移植网络图形”，发音[ping]。

GIF、PNG都是无损压缩图像格式，只不过GIF用的是LZW压缩算法，PNG压缩采用的是与winzip类似的压缩算法。换句话说，PNG基本上相当于zip压缩后的BMP。所以每次看到有人怀疑BMP转PNG会不会出现图像质量损失时，我的回答都是一贯的：你自己找一个文件，zip、unzip一百次，然后看文件内容有没有变。

LZW与zip（更专业的说法是Flate）算法谁的压缩率更高？在《PDF Reference sixth edition》第71页中是这样说的：  
Usually, both Flate and LZW encodings compress their input substantially. However, in the worst case (in which no pair of adjacent characters appears twice), Flate encoding expands its input by no more than 11 bytes or a factor of 1.003 (whichever is larger), plus the effects of algorithm tags added by PNG predictors. For LZW encoding, the best case (all zeros) provides a compression approaching 1365:1 for long files, but the worst-case expansion is at least a factor of 1.125, which can increase to nearly 1.5 in some implementations, plus the effects of PNG tags as with Flate encoding.

看不懂？那就换一个通俗的说法：两种压缩算法的压缩率各有千秋，很多时候看人品和运气。所以存储图像的时候究竟该选GIF还是PNG，从压缩算法上是很难选择的，主要还是看用途。

GIF格式出现得比PNG早，目前广为流行的是GIF89a标准，符合该版本规定的GIF文件的特色是：

* 最多支持256色调色板图像，不支持24位、32位图像。现在有些软件支持24位、32位的GIF，但与GIF89a标准不兼容，因此基本上属于小众软件。
* 支持动画。在常见图像格式中GIF是唯一支持动画的，但动画中的每帧图像均受上一条限制，因此显示一些人工制作的动画没有问题，但把一些色彩丰富的视频转换成GIF动画后，由于色彩强制减成256色，会造成图像质量下降，出现“等高线”等现象。
* 支持背景透明，即在调色板中指定一种颜色作为透明色。在绘制GIF图像时，需要绘制的像素颜色如果是透明色，需要进行透明处理。

很显然，GIF的“透明”具有下列特点：

* 只支持“透明”，不支持“半透明”，即一个像素要么是透明，要么是不透明，不能是“透明度60%”。
* 透明色是GIF文件结构中的一个数据字段，因此有没有透明色，对GIF文件长度无影响。我曾经在某电子书论坛看到某人激动地宣布他的“伟大发现”：在把BMP转成GIF并指定透明色后，他惊奇地发现文件长度变小了，他猜测是因为他“把透明色偷走了”，即他以为GIF不需要存储透明部分。看过本文后我想大家都知道问题出在哪里：24位色BMP减色成256色，再采用LZW压缩，文件不变小才怪，与透明不透明的则没有什么相干。

PNG在名字上看起来与GIF颇为相似，双方事实上也真的是很有渊源的：原先大家都是搞GIF的，突然有一天Unisys公司声称GIF中使用的LZW算法侵犯了该公司的专利，并开始对所有支持GIF格式的软件收取专利费用，于是大家就怒了，干脆扔掉GIF，改用没有专利纠纷的zip算法，并结合最新的图像需求，鼓捣出了PNG格式。这段故事在giflib和libpng的相关 文档中都有说明，有兴趣的可以去看看。

PNG的特点是：

* 除调色板图像外，还支持多通道图像，每通道采样数可以是8位、16位等。这段话文绉绉的听不懂？那就简单点吧：PNG支持24位真彩图像，所以网络上的很多高质量素材图都采用PNG，而不是GIF，以免受256色限制。
* 对于调色板图像，支持与GIF一样的透明色。对于多通道图像，支持Alpha通道透明，即可以实现“半透明”效果。当然多出来的Alpha通道也要消耗存储空间。
* 不支持动画。

总之，我认为可以按照如下原则决定选GIF还是PNG：

* 如果需要动画效果，没得选，只能是GIF。
* 如果想存储24位色，同样没得选，只能是PNG。
* 如果想实现半透明，只能选PNG；如果想在调色板基础上实现透明，选PNG或GIF均可。
* 对于调色板图像，如果您真的、真的对文件大小非常在意，到了锱铢必较的程度，唯一的办法就是GIF、PNG各压一遍，然后选文件最小的那个。如果怕麻烦，建议直接选PNG，从统计学的角度说LZW的压缩率不如zip，但在压缩速度上LZW优于zip。

前面说过，PNG相当于对BMP进行zip压缩（GIF是对BMP进行LZW压缩），因此最终PNG的文件长度受下列因素限制：

* 原始BMP文件长度，参见上面的式（1）～（3），即图像的像素高度、像素宽度、每像素占用字节数。
* 像素数据的可压缩性。所有无损压缩算法的压缩率都建立在数据的重复性上，比如说对于字符串aaaaaaabbbbb，可以记为a7b5，就从12个字符压缩成了4个字符，但对于字符串afhrswnm如果采用同样的方式，就达不到“压缩”的效果。听不懂？那就换成简单的说法：在图像的像素高度、像素宽度、每像素占用字节数均相同的情况下，图像越“干净”，则最终PNG图像尺寸越小。不信？请找一张满是文字的PNG，用软件处理成纯白，然后另存，比较一下文件长度就知道了。

所以PNG、GIF文件长度缩水的窍门就是：

* 采用与BMP图像长度缩水相同的窍门，包括缩图、减色等。
* 用图像处理软件进行处理，让图像尽量干净。简单点说：一张白纸的文件长度是最小的，白纸上出现的黑点、灰点、彩色点越多，图像文件长度越大。

上面的步骤最好综合采用，不要有什么遗漏。比如说我就见过某漫画组发行的自扫漫画，辛辛苦苦把图像处理得很干净，甚至手工把图像灰度级别处理到了16级，但是在保存的时候，不知道为什么却是按照256级灰度保存的，结果本该每像素只需要占用1/2字节的，活活搞到占用1字节。

在标准化方面，GIF格式通行的标准规范是《GIF89a Specification》，支持LZW压缩的开源项目是[giflib](http://sourceforge.net/projects/giflib/)， 为避免专利纠纷而去掉LZW支持的开源项目是libungif，不过也有很多人直接用Windows本身提供的接口读取GIF文件，要打专利官司也是微软的事情。PNG的标准规范及开源项目均由[libpng](http://www.libpng.org/)组织维护，不存在政出多门的问题。

**三、JPEG（JPG）**

JPEG全称是Joint Photographic Experts Group，译做“联合照片专家组”，其中“联合”的含义是指国际电信电报咨询委员会（Consultative Committee of the International Telephone and  
Telegraph, CCITT）和国际标准化组织（International Standardization Organization, ISO）联合组成的一个图像专家小组。这个专家组定义了JPEG国际标准，即ISO 10918，按照这个标准实现的图像格式就称为JPEG格式，三位文件扩展名缩写为JPG。所以洋气一点的发音是[dʒəpeg]，土人的发音是“勾批鸡”。

对JPEG最广泛的误解之一就是以为这个格式是包打天下的万能格式，不管什么内容的图像都存储为JPEG。其实这个误解反过来想一下就不难明白：如果JPEG真的如此万能，其他格式早就死光光了，根本就不可能存活到现在。

在ISO 10918-1第一节“Scope（范围）”中对JPEG的适用范围有明确界定：This CCITT  Recommendation | International Standard is applicable to continuous-tone — grayseale or colour — digital still image data. 即JPEG针对的是“连续色调（continuous-tone）”的灰度或彩色图像，除此之外的图像并非JPEG的本意。通俗点说，“连续色调”的图像指色彩或灰度级别比较丰富、过渡自然的图像，如摄影作品或写实作品。而所含颜色 数很少、具有大块颜色相近的区域或亮度差异十分明显的图片，如白纸黑字扫描后的图像，或人工绘制、填充大块纯色后的图像，显然不属于连续色调的图像，这些图像保存成TIFF、GIF或PNG要比JPEG更适当。

这种“JPEG万能”的误解对普通人来说没什么，但对专业人士来说就意味深长了。几年前我曾经听人说过一个传闻：某档案电子化项目招标前，在与各厂家 进行了几轮技术交流后，甲方突然提出要求所有扫描图像全部存储为JPEG格式，于是众厂家哗然，纷纷猜测有人被搞定了。据说后来投标过程中果然怪事连连、主客颠倒。告诉我这个传闻的人对我解释这种猜测的原因是：

* 甲方的业务严重依赖于网上调阅扫描生成的电子档案，而且存量档案数目巨大，不论从存储、传输还是OCR等再利用的目的出发，都应该仔细衡量，选择最合适的文件格式，才不会对甲方当前及未来的业务造成负面影响。
* 该项目在行业内并非第一次吃螃蟹，在此之前同行业已经有多个成功案例，经过多次论证后原始扫描图像采取的都是JPEG + TIFF的方式，即彩色图像用JPEG，黑白图像用TIFF。后期再根据需要将原始扫描格式转换成PDF或其他格式。这次一反常态，对其中原因又语焉不详， 自然会被怀疑受到某些厂家的导向。
* 有实力的扫描外包商使用专业高速扫描仪，能够自动判别扫的是黑白页面还是彩色页面，对黑白页面进行内部处理后出来的黑白图像，要比扫成灰度后再用软件处理成黑白效果更好（《The Scan and Share tutorial》针对家用扫描仪，不在此例），效率也更高。而不那么有实力的扫描外包商使用不那么专业的扫描仪，为了使图像看起来更“干净”，通常会增加扫描时的亮度， 结果导致黑色变成了灰色，只能存为JPEG文件。CX有一段时间扫描的清晰版PDG看起来白花花一片，文件长度也大了很多，readfree上就有人在猜CX是 否换了一个成本低廉的扫描外包商。

当然传闻毕竟是传闻，我也没去证实。不过对于300 DPI扫描分辨率的黑白图像，JPEG格式与CCITT G4压缩的TIFF格式之间的文件长度差异确实挺大的，这个见过真正清晰版PDG的人都应该清楚。

网上对于JPEG另外一个比较常见的问题是“JPEG究竟是有损还是无损？”。其实在ISO 10918-1中，规定JPEG可以是基于离散余弦变换（discrete cosine transform，DCT）的有损压缩，也可以是不基于DCT的无损压缩。只不过人们用JPEG的目的一般都是追求有损压缩所带来的高压缩比，JPEG的无损压缩反而成了鸡肋，只具有学术意义，实际中反而没人去用了。

有损压缩比无损压缩的压缩比更大，其原因细说起来可以写一篇专业论文，通俗一点说就是：既然允许出现图像损失，有些肉眼分辨不出的图像细节就可以扔掉了，不必像无损压缩那样锱铢必较，自然就可以压得狠一点。

对于有损压缩的JPEG图像，其最终文件长度除了受到图像像素宽度、图像像素高度、每像素占用字节数（灰度1字节，彩色3字节），及图像是否“干净”的影响外，还有一个重要因素是压缩时指定的质量系数。需要指出的是，即使指定质量系数为100%，有损就是有损，无论如何它也变不成无损。 另外这种有损压缩造成的图像质量损失具有累积性，因此如果想对图像进行多次编辑，或者预计以后要对图像进行编辑，就不应该将图像存储为JPEG格式，以免造成不必要的损失。

ISO 10918-1中对JPEG的各种模型描述得比较详细，但对如何用文件存储JPEG压缩后的数据则语焉不详，只说数据要分段存储，并在段头用2字节的标记（marker）说明该段数据是什么意思——感觉和现在的XML也差不多，只不过是二进制的而不是可读字符的。在标准中定义了一些 常用标记，但也留下了活口，如可以由应用软件自行扩展的APPn标记，就允许扩展出APP0～APP16，共16个标记。

目前比较流行，以至于成为事实上的标准的JPEG文件存储格式是Eric Hamilton提出的JPEG File Interchange Format（JFIF，JPEG文件交换格式），其标志是文件头为FF D8 FF E0，从第6字节开始是四个固定的字符JFIF。事实上JFIF文件的文件头是固定的，早年间国内D版扫描漫画格式ufo格式就是把JFIF的前几个字节加密，然后就作为专有格式开始卖钱了，结果分分钟就被专家看出破绽。

随着数码相机的流行，另外一种JPEG文件存储格式也开始崭露头角——EXIF格式。这种格式其实和JFIF并不矛盾，只不过文件以FF D8 FF E1开头，从第6字节开始是Exif这4个字符。EXIF比JFIF多出来的这个以APP1（FF E1）为标记的数据段，存储了照片的缩略图、照片拍摄参数等，其他数据段都一样。其实理论上说是可以把JFIF（采用APP0标记，即FF E0）和EXIF混合存储在一个文件中，因为双方的数据段标记并不矛盾，但不知道为什么要独立出来。

ISO 10918-1中规定的APPn标记是一个大大的后门，一方面便于应用软件根据自己的需要对所存储的数据进行扩展，如JFIF扩展了APP0，EXIF扩展了APP1，但另外一方面也会引发 一些混乱。这方面最著名的一个例子是Adobe扩展出来的APP14，用于存储CMYK色彩空间的JPEG图像——JFIF格式存储的是单通道的灰度图像，或3通道的彩色图像，而出版界广泛使用的是4通道的CMYK色彩，因此作为出版界龙头老大的Adobe挺身而出，扩展出APP14以支持CMYK色彩空间。不幸的是，Adobe对其技术细节很少透露，目前我见到的大多数JPEG解码软件对CMYK似乎也不太重视，都只用一套转换 公式实现从CMYK到RGB的转换，结果总是有些图像解码后颜色看起来很怪异——APP14段包含的参数可不止一个。我自己也是在大量测试的基础上，才为CEP、CV等选定了两套转换公式，至少解码我手上的CMYK色彩空间的JPEG图像不成问题。所以从兼容性上考虑，建议在Photoshop等软件中输出JPEG文件时，尽量不要选CMYK色彩空间，而是先转换成RGB或灰度后再输出JPEG，否则在第三方软件下解码出来的不一定是 您想要的色彩。

如果想对JPEG数据进行分析，我推荐小众软件cPicture，不仅能够查看JPEG的数据结构，而且可以直接编辑EXIF信息，或无损旋转、裁剪JPEG图像。当然裁剪只有在满足一定条件才是无损的。

在JPEG编码、解码方面，最著名的开源项目是[Independent JPEG Group](http://www.ijg.org/)（IJG，独立JPEG小组）发布的[JPEG LIB](ftp://ftp.simtel.net/pub/simtelnet/msdos/graphics/)，其v6b版本运行稳定、容错力强，而且流传甚广（比更新的v8、v9用得更多），甚至成为检查JPEG文件兼容性的标准工具——能够用IJG v6b解开的JPEG文件，基本上在大多数软件中解码都没有问题，反之则可能有问题。在Pdg2Pic中，我也是利用了v6b超强的容错性来实现“JPG修复”功能——把有问题的JPEG压缩的PDG文件用v6b强行解开，然后重新编码，避免稍微有点问题就造成SSREADER的JPEG解码器崩溃、退出。

另外一个名气比IJG小一点，但速度快得多的是Intel发布的[Intel JPEG Library](http://developer.intel.com/software/products/perflib/ijl/)（IJL），它专门针对Intel处理器进行了彻底的优化，因此在Win-tel环境下速度一流。这个工具一共只有六个函数（ijlInit、ijlFree、ijlRead、ijlWrite、ijlGetLibVersion、ijlErrorStr），而且随机文档中附有完整的VC++实例，因此对于VC程序员来说不用费什么脑子就可以对JPEG应用自如。

IJL有两个主要的版本：1.5版和2.0版。前者免费，后者收费。2.0之后IJL就被Intel Integrated Performance Primitives （IPP）v6.1中新推出的UIC（Unified Image Codec）所取代。

免费的IJL v1.5是闭源软件，但因其比IJG更快的速度而流传甚广，只是在我看来还存在下列问题：

* 内存漏洞。这个用IJL的DLL版看不出，要用Static Lib版才能看出来：每打开一个文件，VC会报告72字节（3通道JPG）或24字节（单通道JPG）的漏洞。网上有文章说明如何用内联汇编打补丁来解决这个问题。
* 不支持EXIF中指定的旋转参数。这个问题不大，可以在解码后自己旋转，但需要自己再去解码EXIF。
* 手册上说不支持CMYK色彩空间，但自带的JPGView例子支持某些CMYK的JPEG，另外一些解码后色彩不正确。这个也可以修正，同样需要自己进行处理。
* 容错性太差。某些数据异常的文件用IJG能解码，用IJL则不行。有源代码的话很容易修正，没有源代码就难了。

IJL v2.0提供部分源代码。但从我试用的情况看，除修正了内存漏洞外，其他问题依旧存在，包括部分JPEG文件用IJG能解码而IJL不能解码的问题。不过毕竟有一些源代码，所以可以自己动手修正。这句话换一个角度理解就是：如果没有源代码，建议就不要用IJL了，不论是v1.5还是v2.0，免得因为某些文件不能解码而被人说。当然如果只是自己用 也就无所谓了。

从速度上来说，在目前的主流CPU上IJL v2.0要比v1.5快，毕竟可以采用最新的处理器指令。

**四、JPEG 2000**

JPEG 2000是JPEG的升级版，标准编号ISO/IEC 15444。除了追求压缩比、图像质量外，JPEG 2000还提出了一些花狸狐哨的新东东，如“感兴趣区域（regfon一of-interest coding)”特性，即可以任意指定图像上你感兴趣区域的压缩质量，还可以选择指定的部份先解压缩。

在技术方面，JPEG 2000放弃了JPEG所采用的以离散余弦转换(discrete cosine transform，DCT）为主的区块编码方式，而改用以小波变换（wavelet transform）为主的多解析编码方式。这后面的数学基础实在太深奥了，有兴趣的还是去看专著吧。

在ISO/IEC 15444中，专家组同样没有指定JPEG 2000的具体实现，而是鼓励大家互相竞争。目前比较有名的开源项目包括Michael D. Adams的[JasPer](http://www.ece.uvic.ca/~mdadams/jasper)、[openjpeg](http://www.openjpeg.org/)组织的LibOpenJPEG等，其中前者似乎名气更大、用的如更多一点。但在我看来，这两个加起来在速度、兼容性方面也比不上[kakadu software](http://www.kakadusoftware.com/)的kakadu。

我个人对JPEG 2000最感兴趣的一点是：可以定量指定压缩后的文件长度，kakadu和Jasper都支持这个特性。DjVuToy在转换3层DjVu成PDF时，对IW44压缩的前景层、背景层就是 转换成JPEG 2000压缩，并且按照原始DjVu中IW44数据流长度计算压缩后的JPEG 2000长度，以保证转换前后文件长度差异不大。DjVu的IW44压缩也是基于小波变换的，对于常规连续色调图像在压缩比相同情况下与JPEG 2000效果差不多，但在低码率情况下似乎质量比JPEG 2000更好一点。

**五、TIFF**

TIFF全称是Tag lmage File Format，译作“标记图像文件格式”，发音[tif]。其标准最早由Aldus公司提出，但现在的v6版标准的发行者已经是Adobe了。Tag的意思与JPEG标准里的marker差不多，即TIFF文件也是分段存储的，每段开头用tag标记，与现在的XML差不多，只不过是二进制的。

与前面的几种常见图像格式相比，我个人认为TIFF的特点是：

* 支持多页打包存储，即一个TIFF文件可以包含多页，甚至是整本书。这个和PDF有点像。
* 对色彩的支持相当丰富，一般人根本绕不清。这可能与参与TIFF标准制定的厂家多为扫描仪、打印机厂家有关。反过来说，TIFF在扫描、传真、打印、出版领域比较流行。
* TIFF标准比较开放，自定义空间比较大，所以TIFF根本就是一个大空筐，各家都在不停地往里面塞自己的东西，在某些情况下会出现兼容性问题。可能就是因为这个原因，虽然TIFF文件出现得比较早，但从NetScape到IE到Opera到Chrome，没有一款HTML浏览器支持TIFF。

在压缩算法方面，TIFF标准规定的基线（base line）是不压缩、CCITT G3压缩（用于黑白图像）、RLE压缩（用于灰度、彩色图像）。“不压缩”就不说了，RLE也不是什么有效率的压缩算法，所以在一般人的印象中，TIFF文件都有一个“文件比较庞大”的坏名声。

除基线部分外，TIFF标准还定义了扩展部分，除专门针对黑白图像的CCITT G4压缩外，LZW、JPEG也被扩展进来了。但是否支持扩展部分由各厂家自行决定，所以如果想保持最大兼容性，应该尽量少用扩展部分。但TIFF基线规定的算法实在是太没有效率，所以CEP在编码TIFF时，使用了扩展部分的CCITT G4（用于黑白图像）和LZW算法（用于灰度、彩色图像）。

TIFF的开源项目是[libtiff](http://www.libtiff.org/)，该项目升级频繁，在标准扩展方面更是急先锋，不仅扩展到支持PNG所用的zip压缩算法，连JBIG（相当于DjVu的JB2压缩）、JPEG 2000都扩展进来了，但这些扩展的压缩算法也只有对应版本或更高版本的libtiff才能解开，谈不上啥兼容性。

TIFF标准中有定义，但实际使用中有歧义的一种压缩算法是JPEG压缩。在TIFF标准中，认为JPEG中的各种表结构与像素数据是可以分离的，而且多页可以共用相同的表结构，因此一个完整的JFIF文件转换成TIFF后，会被分解成好几个部分，解码的时候需要把各个部分组装起来后再解码。这对软件开放人员来说无疑是一种挑战，在我看到的情况中，各扫描仪厂家生成的JPEG压缩的TIFF文件基本上只能被自家的软件解开，当年我在某项目中为了兼容不同厂家生成的JPEG压缩TIFF文件费了九牛二虎之力，最后还是决定灰度、彩色图像直接存成JPEG文件算了，不再强求统一 存储成TIFF格式。libtiff对TIFF标准中的这种规定也嗤之以鼻，称之为“Old JPEG”，并在自己的代码中扩展了新的JPEG支持，规定各表结构不再分开存储，而是可以将整个JPEG文件嵌入TIFF中。

在TiffToy中，“文件合并”部分在合并JPEG进TIFF时，就用了libtiff的招数，将整个JPEG文件嵌入TIFF，生成后的TIFF文件在基于libtiff的软件下打开应该没有问题，至少我自己的CV、CEP、UV打开都没有问题。“文件拆分”部分在导出“Old JPEG”的图像时，使用libtiff的tiff2pdf中的源代码，将JPEG文件的各部分拼合成一个整体后再写入JPEG文件，因此可以实现无损转换。

另外正是因为我知道libtiff“包罗万象”的雄心与现状，我在开发FreePic2Pdf时，才会决心以它提供的tiff2pdf为基础。事实也证明，至少在TIFF转PDF方面，很少有比它更强 、更专业的。

我不是搞出版的，所以TIFF中花狸狐哨的色彩支持对我来说没用，但TIFF中专门针对黑白图像的CCITT G4压缩对我来说很有用：我喜欢把书籍扫描图像放大后处理成单色图像，存为CCITT G4压缩的TIFF文件。

CCITT G4是一种无损压缩算法，只能用于压缩黑白二值图像，不能用于灰度或彩色图像。该算法最早用于传真领域，现在则扩展到出版等领域。传真领域通常传递的都是“白纸黑字”，所以对于以白底为主，中间有黑色文字、线条的图像，CCITT G4有很好的压缩比，但如果超出这个范围，则可能会越压越大。我就听一个朋友说过，某人开发传真软件，结果一次测试的时候软件突然崩溃，打电话过去一问，原来对方把一条毛巾放进传真机里传了过来。coolman在测试山寨版PDG编码器时，也曾把一张人物 + 风光的彩色照片抖动处理成半调黑白图像（用黑点的疏密程度表示灰度），结果发现压缩成CCITT G4的PDG后比原始的BMP文件要长得多。

总之一句话：CCITT G4作为一种无损图像压缩算法，画面越干净，压出来的文件长度越小。

**六、DjVu**

对于DjVu格式，我想我在《DjVu转PDF》、《别了，DjVu！》、《DjVu、PDF中的隐藏文本》中已经说得够多了，没有必要再浪费口水，只想提几点忠告：

* 如果原始图像的分辨率不足300 DPI，就别想着转成DjVu了。如果非要转，就不要到处去问为什么自己转出来的图像质量这么差 ，看起来这么模糊。
* 在没有真正搞清楚1层、2层、3层DjVu的区别及适用范围，彻底理解DjVu的各项压缩参数的含义前，还是别去碰DjVu吧，以免误人误己。 如果非要碰，建议保存好原始扫描图像，给未来的自己一个后悔的机会。
* 如果看到DjVu格式的中文电子书中有错别字，或者笔画缺胳膊少腿的，不应该首先就怀疑原书的作者或扫描者有问题，而应该怀疑制作DjVu的家伙有问题，是否在扫描分辨率不足的情况下强行使用 了大压缩比的有损压缩。典型的例子就是某些快速版的PDG。

**七、PDF**

如果我直接说PDF是一种图像文件格式，正常情况下是会被人笑掉大牙的。但考虑到现在经常有人把图像转成PDF，另外一些人则天天想着从PDF再转回图像，所以在这里还是谈一下PDF中的图像吧。

在《PDF Reference sixth edition》第3.3节中规定了10种Filter，其中ASCIIHexDecode Filter、ASCII85Decode Filter不会有人用于图像，Crypt Filter是用于加密的，剩下的7种与常见图像格式的关系为：

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 过滤器名称 | 对应压缩算法通称 | 对应图像格式 | 压缩类型 | 说明 |
| LZWDecode | LZW | GIF、TIFF | 无损 | 通常用于索引色（调色板）图像 |
| FlateDecode | ZIP | PNG、TIFF | 无损 | 索引色图像或真彩图像 |
| RunLengthDecode | RLE | BMP、TIFF | 无损 | 通常用于单色图像 |
| CCITTFaxDecode | G3/G4 | TIFF | 无损 | 专用于单色图像 |
| JBIG2Decode | JBIG2 | JBG、DjVu | 无损 | 专用于单色图像 |
| DCTDecode | JPEG | JPG、TIFF | 有损 | 用于256级灰度、24位真彩连续色调图像 |
| JPXDecode | JPEG2000 | JP2、J2K、JPC | 有损/无损 | JPEG的升级标准 |

所以在从图像转PDF时，常见的图像格式都可以无损转换成PDF。判断无损还是有损转换的方法其实很简单：如果转换前后的文件长度基本不变，基本上就可以认为是无损的，否则是 有损的。当然这里面可能出现的例外就是非压缩的BMP、TIFF在转换成LZW或ZIP压缩后，长度会大幅下降；或者其他格式的单色图像转换成CCITT G4，也可能导致文件长度的变化比较明显。

而在从PDF中提取图像时，如果图像的色彩空间不需要进行什么特殊的转换，也能做到无损转换。但如果PDF中的色彩空间与图像文件格式的色彩空间对不上，那就是另外一回事了。判断一个软件是无损提取还是有损提取，可以依据：

* 提取前后的文件长度。如果长度基本差不多，就可能是无损，否则就可能是有损。
* 是否只能把所有图像都转成一种格式。如果所有图像都只能转成一种格式，或者提取图像前要求用户选择提取后的图像保存的格式，则不用问了，软件是懒人写的。如果软件本身能够根据PDF中图像数据的特点，自动选择最合适的导出格式，自然就不会再问用户要导出成什么格式。

正是因为从PDF无损提取图片比较麻烦，所以一般不主张扫描后直接生成PDF文件（某些扫描仪自带的软件可以直接生成PDF），而是建议先扫描成图像，再根据需要转PDF，同时注意保存原始图像文件，以满足其他需要。

**八、小结**

前面费了那么多口水，还是总结一下，对图像格式的选择给点建议吧：

* 对于白底黑字的扫描图像，建议灰度扫描（除非您有非常专业的扫描仪），扫描分辨率不低于300 DPI，有条件的话再放大处理成600 DPI，然后减色成单色，存为CCITT G4压缩的TIFF文件，或无损压缩的DjVu文件。
* 如果想玩动画，在常见图像格式中只能选GIF。
* 如果想玩透明，在常见图像格式中只能选GIF、PNG，不过GIF只支持基于调色板的背景色透明，PNG还支持Alpha通道透明。 对基于调色板的GIF、PNG，透明与否不会改变文件长度，增加Alpha通道则会增加PNG文件的长度。
* 如果是图像处理的中间格式，建议选择BMP或PNG。
* 如果图像中有大片纯色，建议选择GIF或PNG，同时进行减色处理。
* 如果是连续色调图像，如自然风光、人物等照片，或彩色、灰度扫描图像，建议选择JPEG。
* 如果想定量控制生成的文件大小，可以考虑JPEG 2000。
* 对于JPEG格式，建议不要选择CMYK色彩空间。
* 对于TIFF格式，建议不要选择除CCITT G3/G4、LZW之外的压缩算法。

对于无损压缩的GIF、PNG、采用CCITT G3/G4或LZW压缩算法的TIFF等，在图像的像素尺寸不变情况下：

* 画面越“干净”，最终文件长度越小。
* “减色”也能有效减小最终文件长度，如从24位彩色减成256级灰度，或从256级灰度减成16级灰度。

（完）