# H.264卓越的视频编码设置

*品质不是行为，而是习惯。*

—亚里斯多德

杰森·罗伯特·凯里·帕特森（Jason Robert Carey Patterson），2012年4月

目录

## 目录



本文档详细描述了用于高质量H.264视频编码的一组分辨率，比特率和设置，以及这些选择背后的原因。视频编码是一个权衡的游戏，这些设置代表了一个很好的平衡，并且很难加以改善。

阅读本文档时，假定您熟悉基本的视频编码术语和技术，并且了解具体细节也可能会有所帮助。

**更新**：强烈建议您阅读本文并了解各种设置和折衷方法，但是许多人只想下载设置。考虑到这一点，您可以下载VideoEncoderSettings-201904.zip，其中包含本文设置的格式，该格式适合于我们当前使用的使用x264导入HandBrake以及2012年我们在本文中使用的带有x264Encoder的Compressor。最初是写的。

## 分辨率和比特率

我们提供**7种不同标准宽屏分辨率的**视频文件**...**

- **240P**（424x240，0.10百万像素）
- **360p**（640x360，0.23兆像素）
- **432p**（768x432，0.33兆像素）
- **480p**（848x480，0.41兆像素，“ SD”或“ NTSC宽屏”）
- **576p**（1024x576，0.59像素，“ PAL宽屏”）
- **720p**（1280x720，0.92兆像素，“高清”）
- **1080p**（1920x1080，2.07兆像素，“全高清”）

在如此广泛的分辨率下进行编码是基于以下假设：将使用网络视频嵌入机制，该机制能够检测观看者的Internet连接速度并根据该链接速度以及屏幕大小和播放来选择合适的视频文件浏览器的功能-从而为每个不同的查看器提供他/她可以使用的最佳分辨率和比特率。当然，也可以使用更简单的视频嵌入，直到基本的HTML5<video>标签指向视频的单个版本，但是以各种分辨率和比特率进行编码的整个目的是为了提供一种智能嵌入机制，以合理地使用许多不同版本，从而向那些观看者提供高质量的高清视频，足够快的Internet链接，而逐渐落后于观看速度越来越慢的观看者。

对于每种分辨率，我们使用的**比特率是**普通Internet链接速度（见下文）**的最低64%合理降低**（80%的80%），该速度**仍可实现“非常好”的视觉质量**，并且没有主要的可见压缩伪像。就像保存静止图像以供网站使用一样，我们将质量放在首位，并且仅在不引起任何明显降级的情况下尽可能地进行压缩（希望如此）。如果这意味着给定的分辨率要比其他一些网站使用更高的比特率，那就可以了-网络可以解决这一问题，带宽问题每天都越来越少。将比特率推得太低，并冒着提供模糊，不专业的视频的风险，这是不明智的，而YouTube等其他网站通常会这样做。

对于大多数分辨率，我们**还提供以更高比特率编码的更高质量（HQ）版本**，以使用户拥有足够快速的Internet链接。从正常的“非常好”质量版本到HQ版本的视觉差异通常很小，例如在快速运动过程中模糊较少，在黑暗场景中出现条纹的风险较小，以及在困难淡入淡出时结晶的风险较小。尽管如此，我们还是可以利用用户的链接速度来减少压缩伪像，从而提高质量，前提是用户的链接速度不够快，无法达到下一个更高的分辨率，这在总体清晰度和清晰度方面将是一个重大进步。

最后，我们**还**使用类似于蓝光的很高比特率，以全高清1080p分辨率**对“超级比特”的最终质量版本进行编码**。选择的比特率是20 Mbps，这是H.264 4.0级别允许的最大峰值比特率的安全80%。超级位版本应该几乎无损，与原始主版本几乎没有区别-众所周知，它是“透明”编码。它既是最佳的本地播放（非网络）版本，又是将来进行转码的长期母带（例如：YouTube上传，刻录到DVD / Blu-ray，使用将来的编解码器进行重新编码等）。

音频以AAC格式以44.1 kHz进行编码，以64 kbps的单声道或以128 kbps的立体声进行编码，这两种音频的质量都非常出色，实际上与原始音频没有区别。较低的分辨率全部使用**相同的音频设置，以允许它们**在自适应流传输场景中**在它们之间进行清晰的切换**，而不会听到任何“爆裂声”。在128 kbps的版本是一个立体声相当于相同的设置。从单声道切换到具有相同设置的立体声应该几乎是无缝的，并且在大多数情况下应该只发生一次，因为如果用户的链路速度超过5 Mbps，则用户不太可能需要切换回立体声（以较低的速度（例如3或4 Mbps）进行转换会带来更多问题-链接速度越快，越有可能远远超过我们的需求）。超级比特的终极质量版本使用最大可能的320 kbps AAC比特率，因为它充当长期的主设备，并且如果我们需要从中进行重新编码，我们希望质量损失最小（尽管256 kbps足够了，对于以后的重新编码目的，被认为实际上是无损的）。

选择的确切比特率是...

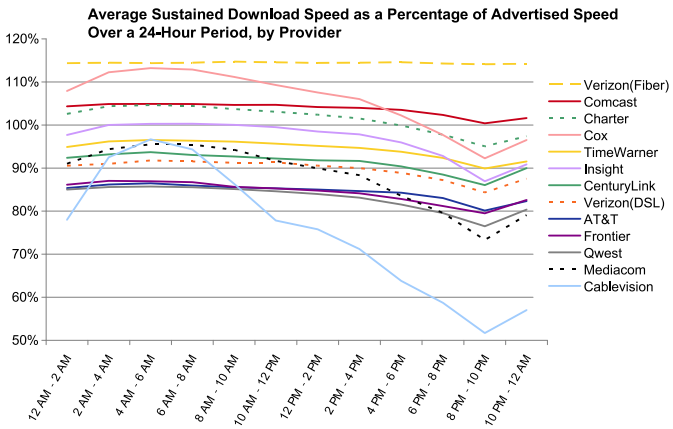
| *名称* | *解析度* | *连结（Mbps）* | *比特率（Mbps）* | *影片（kbps）* | *音讯（kbps）* |
|---|---|---|---|---|---|
| 240p | 424x240 | 1.0 | 0.64 | 576 | 64 |
| 360p | 640x360 | 1.5 | 0.96 | 896 | 64 |
| 432p | 768x432 | 1.8 | 1.15 | 1088 | 64 |
| 480p | 848x480 | 2.0 | 1.28 | 1216 | 64 |
| 480p总部 | 848x480 | 2.5 | 1.60 | 1536 | 64 |
| 576p | 1024x576 | 3.0 | 1.92 | 1856年 | 64 |
| 576p总部 | 1024x576 | 3.5 | 2.24 | 2176 | 64 |
| 720p | 1280x720 | 4.0 | 2.56 | 2496 | 64 |
| 720p HQ | 1280x720 | 5.0 | 3.20 | 3072 | 128 |
| 1080p | 1920x1080 | 8.0 | 5.12 | 4992 | 128 |
| 1080p高画质 | 1920x1080 | 12.0 | 7.68 | 7552 | 128 |
| 1080p超级比特 | 1920x1080 | 不适用 | 20.32 | 20000 | 320 |

...以及较旧的非宽屏内容（只有75%的像素）...

| *名称* | *解析度* | *连结 (Mbps)* | *比特率 (Mbps)* | *影片 (kbps)* | *音讯 (kbps)* |
|---|---|---|---|---|---|
| 240p | 320x240 | 1.0 | 0.64 | 576 | 64 |
| 360p | 480x360 | 1.2 | 0.77 | 704 | 64 |
| 480p | 640x480 | 1.5 | 0.96 | 896 | 64 |
| 480p总部 | 640x480 | 2.0 | 1.28 | 1216 | 64 |
| 576p | 768x576 | 2.3 | 1.47 | 1408 | 64 |
| 576p总部 | 768x576 | 2.5 | 1.60 | 1536 | 64 |
| 720p | 960x720 | 3.0 | 1.92 | 1856年 | 64 |
| 720p HQ | 960x720 | 4.0 | 2.56 | 2432 | 128 |
| 1080p | 1440x1080 | 6.0 | 3.84 | 3712 | 128 |
| 1080p高画质 | 1440x1080 | 9.0 | 5.76 | 5632 | 128 |
| 1080p超级比特 | 1440x1080 | 不适用 | 20.32 | 20000 | 320 |

每个比特率均已选择为普通Internet链接速度（1、1.5、2、2.5、3、4、5 Mbps等）的64%（80%的80%），以便完全但安全地使用常见的连接速度。与静止图像不同，在静止图像中，我们希望最小的美观文件能够尽快完成网页的加载，而对于视频，我们只关心其下载速度足以播放。以比我们需要的回放速度更快的速度下载视频对最终用户没有任何影响，但是可以使用更多的链接速度来提高质量，因此，我们实际上是想避免对用户的视频可用带宽调低。**尽可能安全地增加大小/比特率。**另一方面，过分用力和过多使用用户的链接速度可能会导致暂停（用于缓冲），从最终用户的角度来看，这比稍低的质量要糟糕得多。

因此，我们从与最常见的链路速度相匹配的比特率开始，然后为协议开销（在ADSL上可能高达16%）和繁忙时段的争用保留20%的余量。**假设峰值链路性能的80%长期以来一直是一个不错的指南**，而2011年美国FCC的数据证实情况仍然如此（参见图表）。然后，我们**仅使用剩余80%的80%**（因此仅使用原始广告链接速度的64%），将剩下的**20%留作"比特率净空"，**以使视频能够安全地下载*到平均*目标比特率之前，以覆盖视频中难以编码的部分（例如快速运动）中的比特率波动和峰值。再次，经验表明80%是一个不错，安全的选择，但又不是过于保守（有些人认为多达90%是安全的，而Netflix似乎使用了75%的保守性）。

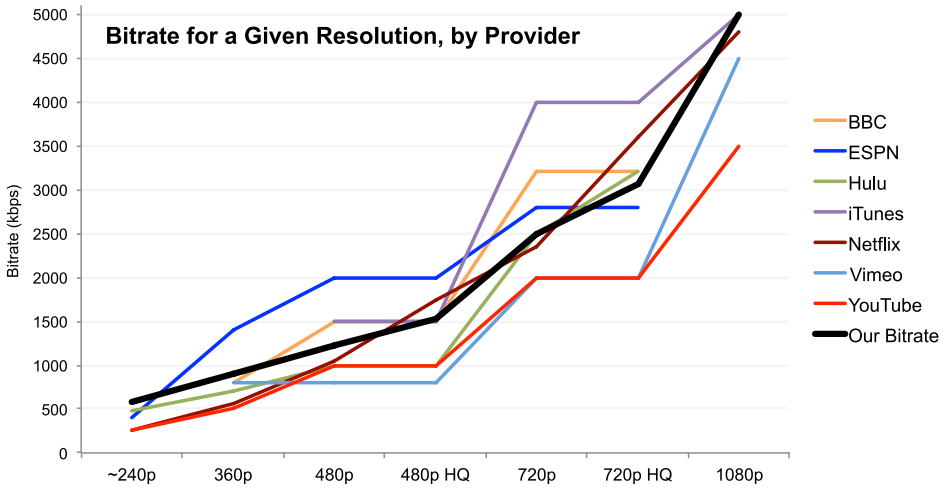



Internet链接速度继续迅速提高，因此，尽管我们选择的比特率高于其他一些视频网站，但出于质量考虑，它们仍然是相当合理的。根据Akamai的数据从2010年的平均真实世界的下载速度（*后*协议开销）已经是8 + Mbps的，在日本，韩国和香港，4.6 Mbps的在美国和加拿大，地方约4 Mbps的在西欧，2.9 Mbps的澳大利亚和2.6 Mbps的俄罗斯。尽管3G手机网络变化很大，但平均也达到2 Mbps。因此，**普通美国人*已经可以***无需等待即可观看我们的视频**的720p高清版本**，以及澳大利亚或俄罗斯的普通480p版本。当然，此类统计中的平均值会因高速而偏斜，因为它是指数曲线，但即使如此，现代国家/地区中约有三分之一的Internet连接的实际下载速度超过5 Mbps，这足以满足720p HQ版本以及70%的版本都超过2 Mbps，因此可以肯定地查看480p版本而无需等待。即使在宽带速度更加不均匀且平均水平落后于大多数现代国家的澳大利亚，2011年的政府统计数据也显示89%的用户无需等待即可观看360p版本（1.5+ Mbps）链接速度），并且45%的用户可以立即查看完整的1080p版本（链接速度为8+ Mbps）。

将我们选择的视频比特率与主要的在线视频提供商进行比较非常有用。所有人都使用H.264作为首选的视频编解码器。下面的比特率已经过iTunes，YouTube和Vimeo的亲自验证，是Hulu，BBC和ESPN的官方声明，并取自Netflix的用户界面...

| 提供者 | 视频比特率（kbps） | | | | |
|---|---|---|---|---|---|
| | ~240p | 360p | 480p | 720p | 1080p |
| 英国广播公司 | | 800 | 1500 | 3200 | |
| ESPN | ~400 | 1400 | 2000 | 2800 | |
| 葫芦 | ~480 | 700 | 1000 | 2500<br>3200 | |
| 的iTunes | | | 1500 | 4000 | 5000 |
| 奈飞 | ~250 | 560 | 1050<br>1750 | 2350<br>3600 | 4800 |
| Vimeo | | 800 | | 2000 | 4500 |
| YouTube的 | 250 | 500 | 1000 | 2000 | 3500 |
| **我们的比特率** | **576** | **896** | **1216<br>1536** | **2496<br>3072** | **4992<br>7552** |

或者，更图形化地...




从分辨率来看...

- 在**240P我们用比别人更高的比特率**，以保证专业水准的质量可接受的最低水平，而不是可笑的看着模糊/块状，一拉的YouTube。即使以这样的比特率，整体的模糊和细节的缺乏也是一个问题，但是严重的压缩伪像通常仅在高速运动和淡入淡出的时候才注意到，这是一个确定的斗争。除此之外，内容看上去非常不错，只是分辨率很小，分辨率很低。
- 在**360p时，我们的比特率几乎是YouTube和Netflix的两倍，比Hulu高28%**，大约等于（高于12%）以质量为导向的BBC和Vimeo，再次提供了专业的质量水平。淡入度仍然很成问题，但其余时间质量实际上还不错。在此分辨率及下一个分辨率下，ESPN使用了非常高的比特率，应将其忽略（请参见下文）。
- 在**480p时，我们正处于中间位置**，介于YouTube / Netflix / Hulu（低端）和BBC / iTunes（高端）之间。在这一点上，质量达到了我们要求的"非常好"的标准，几乎没有可见的压缩伪影，也没有主要的伪影。尽管比特率约为四分之一，但它将通过标准清晰度的数字电视！
- 在**480p总部，我们使用与BBC和iTunes相同的比特率**，仅比Netflix低一点（12%），Netflix是唯一提供第二个更高质量480p版本的提供商。比特率提高26%有助于减少任何残留的可见压缩伪像，因此视频质量甚至更高，只有最困难的内容（例如高细节衰减）会引起明显的毛刺。
- 在**576p处，没有一家主要的提供商提供可比较的版本**，因此为简单起见，已将其排除在图表之外。在质量方面，与480p相同的注释在此处和以下均适用，在较低比特率下质量非常好，在较高比特率下质量更好。与其他提供商不同，我们选择包括576p版本，因为3-4 Mbps的范围是相当普通的连接速度，并且我们可以用额外的带宽多提供45%的像素，而不是简单地将其丢弃。
- 在**720p时，我们再次处于中间位置**，介于低端的YouTube / Vimeo和高端的BBC之间，与Netflix和Hulu大致相等。
- 在**720p HQ时，我们使用的比特率与BBC和Hulu大致相同**，比Netflix低一点（15%），远低于iTunes的疯狂比特率（请参阅下文）。
- 在**1080p时，我们远远高于YouTube，略高于Vimeo，并且与Netflix和iTunes相等**，因为没有必要尝试降低–没有太多链接超过7 Mbps但不是8 Mbps，因此降低到4352不会有太大的改善，除非降低质量。

- 在**1080p HQ上，目前没有一家主要供应商提供可比的版本**，因此出于简单性的考虑，它已经被排除在图表之外，尽管利基供应商也提供类似的高比特率，并且在海盗世界中常见于高质量的1080p编码。1080p HQ的比特率增加了50%，而其他HQ版本的比特率增加了17-26%，因为这是我们提供的最终版本，我们希望确保其具有出色的质量。

看链接速度...

- 对于非常慢的链接，我们以640 kbps的速度提供了一个相当不错的240p小版本
- 对于1.5+ Mbps的链接，我们提供了一个很好的360p版本
- 对于1.8+ Mbps链路，我们提供了非常好的432p版本
- 对于2+ Mbps的链接，我们提供了非常好的480p版本，甚至在2.5+ Mbps时也更好
- 对于3+ Mbps链路，我们提供了非常好的576p版本，甚至在3.5+ Mbps时甚至更好
- 对于4+ Mbps链路，我们提供了非常好的720p版本，甚至在5+ Mbps时甚至更好
- 对于8 Mbps以上的链接，我们提供了非常好的1080p版本，并且在12 Mbps以上的情况下表现出色

如图所示，确实有3个提供商阵营。首先，有些**提供商的比特率似乎太低：YouTube和Vimeo**，以及较低分辨率的Netflix和Hulu。他们并没有那么在意质量，而是在确保质量的播放而不需付出一切代价，即使质量很差。由于我们的目标是具有非常好的视觉质量，而没有主要的可见压缩伪像，因此，在所有分辨率下，我们选择的比特率均显着高于YouTube和Vimeo。在较低的分辨率下，出于质量的考虑，我们也使用比Netflix和Hulu更高的比特率，尽管它们在较高的分辨率下相同。有趣的是，Netflix和Hulu是仅有的几个以某些分辨率（480p和720p）提供多种比特率的设备，以充分利用用户的Internet链接速度来提供更高的质量。

其次，有些**提供商或多或少地同意我们选择的比特率：分辨率较高的Netflix和Hulu，以及BBC**。我们与Netflix在向480p和720p的适当过渡点方面达成了近乎完美的协议，并且与BBC就高质量比特率达成了近乎完美的协议。

最后，有些**提供商的比特率似乎过高：Apple的iTunes电影和电视节目，以及ESPN**，尤其是在较低分辨率下。两种情况都有技术说明。iTunes对此有点偷偷摸摸，但实际上并没有提供真正的848x480p作为选项，而是使用在水平方向变形地缩放到宽屏的640x480p 。Apple这样做是为了简化用户操作–只有一个“ SD”可在Apple的所有设备上使用，甚至是旧版iPhone 1和更早的视频iPod。像这样降低分辨率意味着只能提供75%的像素，这当然意味着Apple的480p比特率实际上只能覆盖75%的像素，或者换句话说，它有效地将每个像素的比特率提高了33%。iTunes还使用比我们720p更高的比特率，再次提高了约30%。令人高兴的是，在720p时，iTunes并没有像在480p时那样牺牲分辨率，而是使用高达4 Mbps的720p视频来解决带宽问题！在这两种情况下，这主要是因为Apple需要使用高比特率来补偿标准QuickTime H.264编码器的质量差，该质量始终在编解码器比较中排在最后，即使一眼看上去也更糟-我们在1216年的480p编码千比特每秒在1500 x 1.33 = 2000 kbps时比它们更好，每像素比特率高64%！ESPN的比特率在所有分辨率下都高于我们的分辨率，但最小的版本除外（对于那些疯狂的体育迷，ESPN会牺牲*很多*质量以确保其播放时不惜一切代价），但是比特率过高在较低的主流分辨率下尤其明显。这是因为ESPN 仍然使用旧的RTMP自适应流传输（您认为那已经死了，不是吗？），迫使他们使用不太灵活的*恒定比特率*编码（喜欢！）。

**更新：**提供商有时会根据经验和用户反馈来更改其提供的比特率。例如，在撰写本文的几年后，Netflix将其1080p 4800kbps产品分为两个普通/ HQ变体，分别为4300kbps和5800kbps，并同时将其720p HQ产品从3600kbps降至3000kbps。此后的几年，Netflix开始对流行内容进行特定于内容的编码，以在2D动画和简单的“会说话的内容”等情况下节省带宽。但是，上面推荐的比特率仍然是一个很好的选择。

## *编码器软件-x264*

视频编码器软件的选择对最终质量有非常大的影响，可能比除比特率以外的任何其他选择都要多。**我们使用并推荐出色的开源x264编码器**，它本质上是当今H.264视频编码的黄金标准，并且已经使用了几年。自2006年以来，x264一直每年都赢得年度MSU MPEG-4 AVC / H.264视频编解码器比较比赛，以及许多其他编解码器比较和评论。它最接近的竞争对手通常是用于Adobe Media Encoder和Microsoft Expression Encoder等应用程序的MainConcept H.264编码器。尽管MainConcept还是一个非常好的编码器，但是x264在任何给定的目标比特率下，无论是主观（IMHO）还是*客观地*通过比较中使用的任何度量标准（PSNR，SSIM等）进行测量，都能可靠地产生更好的质量。

在我们特定的情况下，我们使用x264Encoder QuickTime编解码器，它以插件组件的形式提供x264，该插件组件集成到Mac OS X QuickTime视频库中，从而允许从大多数Mac应用程序轻松使用，包括我们选择的批处理编码工具，压缩机。我们还可以通过其他非基于QuickTime的编码工具（例如HandBrake）使用x264 ，但是它们可能无法完全访问所有x264的设置，并且它们可能也不支持其他QuickTime编解码器，例如Flip4Mac WMV。我们还将其用于整体编码系统。使用x264Encoder的Compressor可以很好地满足我们的要求，尽管由于Compressor和QuickTime 7（基于它）都是32位的，因此性能确实损失了百分之几。

x264编码的高质量主要归因于...

- **主动运动估计搜索**，使用大量初始候选预测变量，然后进行复杂的，不均匀的多六边形搜索（以提早退出速度），然后再进行细分，从而有助于在图像中找到尽可能多的时间和空间冗余使用全速率失真优化的像素优化，以考虑每个选择的实际最终成本与收益

- **出色的比特率控制/分配**，使用宏块级分析（" MB树"）通过来自未来帧的实际运动矢量跟踪每个宏块的引用程度，从而使编码器仅降低每个帧区域的质量它们正在快速变化（未来将不再引用），而不是像大多数编码器那样降低整个帧的质量-本质上是传统的比特率控制，但适用于每个16x16宏块的级别，而不是整个帧的级别-在存在移动的前景对象的情况下帮助保持清晰，稳定的背景
- **B帧的智能，自适应，可变使用**，而不是像大多数编码器那样仅使用固定模式（如IBBPBBPBBPBB），通过插入更昂贵但图像质量更高的I和P帧来更好地利用可用比特率它们最适合用作参考框架，这在任何时候都很好，但是在淡入淡出时尤其重要（要压缩得最困难的事情之一）
- 自适应量化，它改变每帧中每个宏块的量化，以避免在包含精细/边缘细节（例如平静的水，天空和动画）的平坦区域中出现模糊/阻塞
- **完整的速率失真优化，**用于运动矢量细化，宏块划分（细分每个宏块，平衡其他运动矢量的成本与剩下的要编码的较简单残差图像的收益之间的平衡）以及最终量化（关键的有损步骤！），它会选择成本最佳的运动矢量，宏块划分和量化，基于成本与收益的关系，使用从选择到最终熵编码的每个可能选择的真实，实际成本，以及所测得的图像质量收益通过RDO指标（请参见下文）
- **一种"心理-视觉"速率失真优化指标**，它通过不强调模糊的"低误差但低能耗"的选择来尝试更好地匹配*感知的*视觉质量，而不是使用绝对差之和（SAD）等更简单的指标，峰值信噪比（PSNR）或图像的结构相似度（SSIM），所有这些趋向于趋向于低的数字像素差异，但趋于模糊

x264速度也很快，其中SIMD向量指令用于大多数原始操作，以及良好的多线程功能，可在多核和多处理器系统的第二次编码过程中实现接近线性的加速（视频编码自然是高度并行的问题，当然，这使得并行化变得非常容易）。对于软件编码器而言，x264速度非常快，并且如果使用快速设置，则与专用编码硬件相比实际上具有竞争力，同时可以获得更高的质量。我们当然用*得多*慢设置，以达到尽可能高的质量，但我们仍然欣赏X264的速度。

其他编码器：即使您确实使用了其他编码器（例如MainConcept），您仍然应该发现本文档很有用，因为大多数编码器基于基础H.264格式本身和一般视频编码的性质提供相似的设置。一个字的警告，但- **你*不*使用苹果的标准H.264编码，自带内置的QuickTime的一个**，因为它不是非常好，始终是最后的编码器比较，特别是在低比特率。

## *编码策略和通行证（--pass）*

在确定分辨率，比特率和编码器软件之后，下一个最重要的设置是整体编码策略。本质上有4种可能性...

- **恒定的质量** –使用固定的量化器，并允许最终的文件大小/比特率任意波动（例如，以某种质量保存JPEG图像时，不同图像的文件大小将有所不同，具体取决于它们的复杂程度）
- **恒定比特率** –使每个帧使用相同数量的比特，并让最终产生的质量任意波动（就像将JPEG图像全部保存到相同的文件大小一样，某些图像的质量取决于其他图像的复杂性，质量会比其他图像差）
- **单遍可变比特率** –从一帧到下一帧都改变质量和比特率，为较硬的帧提供更多的比特，以便在目标平均比特率下优化整体质量，但不了解视频的整体内容用于几帧的小超前窗口
- **2通道可变比特率** –从一帧到下一帧都改变质量和比特率，为较硬的帧提供更多的比特，以在目标平均比特率下获得最高的总体质量，并提前了解视频的完整内容以分配最多的比特适当地在整个视频中使用，可能会受到本地峰值比特率的限制

恒定质量，固定量化器的策略很简单，但对于Internet视频却没有太大的兴趣，因为比特率变化太大，最终文件的平均比特率是不可预测的。毕竟，我们最终的目标是特定的Internet链接速度。当然，恒定*比特率*方法可以解决此问题，并且可以使其正常工作，但是通常必须使用过高的比特率才能保证视频中难以编码的部分的质量（如ESPN）例如上面的示例），因为无论帧是容易编码还是难以编码，每个帧都会获得相等的比特分配。通常，这意味着将比特率设置得足够高以应付最坏的情况，这意味着99%的时间比特率都*太高*且浪费。

单遍，可变比特率编码速度很快，并且产生的质量相当好，但是**单遍意味着编码器必须*猜测*未来可能会发生什么**，并"以防万一"，并留出合理的余地，因为编码器并不灵敏且不知道其余视频的内容。它不会知道一个部分是特别容易的场景，而另一部分是很难编码的场景，因此相对于较难的场景，易于编码的场景倾向于获得太多的位，而那些较难的场景会做得更好。

借助**2遍编码，编码器在将单个位写入**输出文件**之前对视频进行了完整的遍历**，以精确地了解要最有效地使用这些位的位置（哪些场景是困难的场景等）。除非您处于减少编码时间确实很重要的高音量情况下，或者您无法知道即将发生什么的实时情况下（除非您有足够的时间等待），否则**两次编码可变比特率编码的编码时间会更长绝对值得，**并且可以显着提高质量。

***全部：*** 2遍可变比特率编码，以达到目标比特率的最佳质量，并尽可能明智地使用每个比特。

## *H.264配置文件（--profile）*

该H.264轮廓是一个兼容性问题。它定义了播放器所需的功能，即播放器必须支持完整H.264格式的哪些功能才能播放文件。自然，编码器无法使用播放器无法保证具有的任何功能，因此使用较高配置文件会使用更多功能，以在相同的目标比特率下提供更好的质量，但会阻止某些年龄较大或较低端的播放器播放档案共...

- **基线配置文件是常规H.264格式的子集，该格式**丢弃了CPU占用量最大的功能，主要是B帧，加权预测和最终熵编码步骤的CABAC算术编码。删除这些功能使H.264视频可以由性能很差的处理器播放，但是为此牺牲了很多质量。Baseline是非常老的手持设备（如早期的视频iPod和原始的iPhone 1 / 3G）支持的唯一配置文件。

- **主要配置文件是原始的H.264格式**，并且是某些流行的较旧的Apple设备（如iPhone 4，iPad 1和Apple TV 2）*正式*支持的配置文件，尽管这些设备内部A4处理器中的硬件解码器实际上支持高级配置文件。（见下文）。为什么苹果公司没有说明规格是个谜，但是规格却是*错误*的-iTunes只要它们的分辨率和比特率在一定范围内，iTunes就会很高兴地将高端视频同步到那些设备（甚至是iPhone 3GS）。限制（请参阅下面的H.264级别）。iTunes在其720p分辨率视频中采用了高调配置， 就像YouTube一样，并且在这些设备上都可以正常播放。实际上，**几乎没有任何设备实际上*仅*支持主配置文件，它始终是基准配置文件或高配置文件**，因为高配置文件是在最初的H.264标准发布仅18个月后（即2004年）推出的，远比H.264真正"流行" "以及许多为此设计的硬件解码器的*方式*。

- 高配置文件包含一些附加功能，这些功能是在原始H.264标准发布后不久根据其实际经验而添加的。您可以**将H.264 1.1版视为高端配置。**它最初是更深奥的保真度范围扩展的一部分，其中包括利基配置文件（请参见下文），但是高配置文件则更加有用，因为它添加了一项关键功能–可以为DCT自适应使用8x8块大小的选项，而不只是4x4。这样可以在不显着影响播放复杂性或CPU负载的情况下，对压缩效率进行适度但引人注目的改进，从而在给定的比特率下提高质量。高是Blu-ray，iPhone 4S，iPad 2支持的配置文件和 Apple TV 3（自然是更高版本的机型），以及所有主流台式机/笔记本电脑：QuickTime 7.2（2007年7月），Flash 9.3（2007年12月），Windows Media Player 12（2009年7月为Windows 7的一部分），或VLC，MPlayer等的任何非古代版本（libavcodec于2006年左右）。

- 还有一些利基配置文件，例如*high10*（每个颜色分量提供10位）和*high444*（提供完整的4：4：4色度采样）。这些文件并未广泛支持播放，但旨在用于"高保真"用途，例如母版的长期存档存储。特别是，high444通过消除量化而具有可选的无损模式，尽管所产生的比特率非常大，与ProRes相当。使用这些配置文件进行归档是多余的，浪费的，考虑到缺乏通用的播放支持，这可能是不明智的，这就是为什么它们未得到广泛使用的原因。

x264的默认设置为高配置文件，但x264Encoder QuickTime插件的iPod预设使用基线配置文件。HandBrake的iPhone 4， iPad 1和Apple TV 2预设也使用高调。

***240p / 360p：***基准配置文件，目的是为了安全地在任何设备上播放，因为这些版本是最终的后备版本，可能会通过网络传递给知道什么的设备，包括原始的iPhone 1 / 3G。

***432p +：***高配置（默认），即使iPhone 4， iPad 1和Apple TV 2尚未正式支持它，因为它可以正常工作并产生更好的质量（如果iTunes，YouTube和HandBrake可以做到这一点，我们可以！）。

兼容性：我们假设所使用的网页视频嵌入机制足够智能，不会嵌入分辨率大于屏幕分辨率的视频（某些特殊例外，例如iPhone 1 / 3G在480x320屏幕上支持360p视频）。因此，上述设置意味着**我们假设屏幕分辨率大于640x360的任何设备也都具有H.264高配置文件的能力**，但是任何低于此**分辨率的设备**可能仅支持基线配置文件。这似乎是一个相当安全的选择，尽管可能会有*极少数*的旧版Android，Windows或Blackberry屏幕分辨率高于640x360但仅支持基线配置文件的手机/平板电脑（是否真的存在这种情况？）。今天，不应仅靠基线配置文件支持来设计新设备，但是如果确实发生过此问题的任何常见情况，则有可能可以在网页视频嵌入机制中检测到它们，并且在那种情况下仅使用360p版本。

质量：对于播放器支持高配置但Internet链接很慢的情况（例如连接速度慢的常规笔记本电脑），使用高配置的第二个240p / 360p版本可以实现更好的视频质量。不幸的是，以相同的格式，相同的分辨率*和*格式支持不同版本的视频。在大多数网页视频嵌入机制中，无法基于其他一些辅助条件使用相同的比特率。特定程度的设备和功能决策可能需要某种方式来指示每个视频文件变体所需的功能（基本上是子格式），也许通过特殊的文件命名方案，以及某种设备检测机制和功能JavaScript方面的功能数据库，它们都很脆弱，并且随着功能和设备数量的增加而变得笨拙。我们必须在某处画线。幸运的是，超过70%的用户拥有足够快的Internet链接，至少可以达到480p版本，而在480p版本中，我们确实使用高配置。

## *H.264隐含配置文件（各种）*

对于240p / 360p，仅使用基线配置文件意味着禁用以下默认启用功能，为了完整起见，我们也明确将其关闭...

- **B帧**（在QuickTime对话框中对帧重新排序）–允许 B帧（双向帧）。

- **加权预测**（--weightp＆-- weightb）–允许使用不同的权重引用不同的参考帧，然后将它们组合在一起，这对于淡入淡出特别有用。

- **8x8 DCT块**（--8x8dct）–允许将8x8块用于DCT转换，而不仅仅是4x4块。

- **CABAC熵编码**（--cabac）– 在编码流水线的最终无损压缩步骤中使用空间效率更高但占用大量CPU的算术编码，而不是更简单的可变长度编码，以使整体压缩提高10-20%。

兼容性：少数播放器无法正确处理加权预测，包括非常老版本的Flash，CoreAVC，某些旧LG中的联发科硬件解码器，Phillips和Oppo Blu-ray播放器以及Sony PlayStation Portable（PSP）。幸运的是，截至2010年年中，几乎所有这些错误都已得到解决，除了PSP 出于其他原因无论如何我们都不支持，因此我们可以安全地使用加权预测，这对于良好的淡入淡出至关重要。

## H.264级别（-级别）

该H.264水平是与速度和分辨率的兼容性问题。H.264配置文件（以上）定义了播放器必须支持的视频压缩*功能*，而H.264级别定义了播放器可以处理的峰值比特率，以及最大分辨率和在此期间存储在内存中的最大参考帧数。播放（请参阅下文）。x264的默认设置是根据峰值比特率，分辨率，参考帧数和其他设置自动设置输出文件的级别。有**没有理由改变这个设置，但它是一个好主意，对所用的水平。**

***全部：***自动（默认），对于240p，该级别为2.1，对于360p / 432p为3.0，对于480p / 576p / 720p为3.1，对于1080p为4.0。

## 峰值比特率（--vbv-maxrate和--vbv-bufsize）

限制峰值比特率是必不可少的事情，这是为了在视频难以编码的部分（例如快速运动）期间限制目标比特率的尖峰和超限的严重性。像H.264这样的视频编解码器自然会在视频过程中改变比特率，以将更多比特分配给最需要它们的帧，这对于实现高质量至关重要。但是，在网络播放场景中，无论是自适应流式播放还是播放时更常规的渐进式下载，我们都不能让比特率波动太大，因为目标峰值的严重尖峰或持续超频可能会导致播放暂停（缓冲）），而且暂停时间*很多* 从最终用户的角度来看，它比在高动态场景中的模糊还差。

这是一个权衡- **过于严格峰值比特率限制将降低质量*很多***通过移除可变比特率的最明智地使用位的能力（比特率的变化是一个*很好的*东西），但**过于慷慨的限制将使暂停用于缓冲更容易**，尤其是对于每个分辨率的目标链接速度范围内的较慢连接，因为即使我们拥有20%的比特率净空，播放器也无法下载到足够大的峰值，也就是说（即使：我们知道用户的Internet链接可以下载，至少是目标平均比特率的1.25倍，请参见前面有关分辨率和比特率的部分）。

x264的默认设置是根本不限制峰值比特率，除了x264Encoder QuickTime插件的iPod预设，它通过256k缓冲区将其限制为10 Mbps，这是视频iPod和iPhone 1 / 3G中硬件解码器的限制（即：H.264级别3.0，请参见上面的H.264级别）。大多数人和编码工具都建议峰值比特率是目标平均比特率的两倍，有的只有1.5倍，有的没有极限。

***240p / 360p：***使用256k缓冲区将目标比特率加倍（工作至1.8 / 1.14秒），以支持原始iPhone 1 / 3G和类似设备。

***432p-1080p：***使用1.5秒的缓冲区（1.5倍的峰值比特率）将目标比特率提高一倍，这是慷慨的，假设我们20%的比特率裕量将涵盖大多数峰值和一般波动，因此仅需要真正有问题的持续超限被编码器削波（质量下降）。

***1080p超级位：*** 25 Mbps，以保持在H.264 4.0级内（达到已经很高的目标比特率的1.25倍），并具有30兆位的缓冲区，这是蓝光播放器允许的最小大小，因此对于大多数1080p硬件解码器（工作时间为1.2秒）。

暂停风险：432p-1080p的设置相当宽泛，对于编码器级别的大多数内容，可能几乎不会导致尖峰削波，仅留下20%的比特率净空。如果有的话，我们稍微倾向于更好的整体质量，但有可能会暂停在每个链接速度范围内最慢的链接上缓冲的风险。如果用户跳入视频中间并进入暂时使用目标比特率两倍的高动态场景，则播放器将开始播放，然后突然暂停并必须等待缓冲。将目标比特率设置为1.5倍，并使用1秒钟的缓冲区将是一种更安全，更保守的设置，尽管即使那样也不能完全避免"跳入高动态场景"的风险，这几乎是不可避免的，

## 量化曲线压缩（--qcomp）

该**量化器*将*视频（和图像）在压缩期间的量化的主有损步骤临界值**，并且该值是在一个块逐块和帧接一帧为基础来控制质量变化。较低的量化器为每个块消除了频率系数的DCT矩阵中较少的次要值，从而保留了更多原始信号的频率分布，这意味着输出更接近原始图像，并留下了更多要写入文件的系数。在非常低的量化器（不到10个）的情况下，输出看起来几乎与输入相同，因为几乎没有频率信号的一部分被遗漏（尽管并非*完全如此*） 即使量化器为1，也是如此，因为仍在舍入到整数）。

甲**高质量视频编码将变化两者的比特率和量化器从一个帧到下一帧。**例如，在涉及快速运动的难以编码的帧期间，有必要使用更多的比特，就像您对可变比特率编码所期望的那样，但是明智的做法是通过增加量化器来降低质量，而不是不必要地进行维护通过在那些困难的帧上扔掉过多的位并浪费掉这些位来获得高品质，因为当这些位本可以用于提高视频其他帧中其他位置的质量时，快速运动将大大掩盖任何改进的质量。另一方面，**让量化器过高会降低质量**，使质量下降可见，并在视频难以编码的部分（例如快速运动和淡入淡出）期间导致明显的"不良"部分。人眼特别注意这些"不良"部分，注意到它们是与视频正常总体质量不同的*故障*。

改变比特率和改变量化器之间的权衡，即在困难的帧上扔更多的比特或降低其质量，是受施加到量化器曲线的"压缩"程度控制的-量化器随时间变化的曲线（在x264中的每个宏块级别，而不是整个帧）。设置从0到1变化。设置为0时，完全不允许比特率变化，从而产生恒定比特率编码，结果是在难以编码的视频片段中质量严重下降。在另一极值1处，允许比特率根据需要疯狂地变化（以峰值比特率为准）（请参见上文）以在难以编码的部分保持高质量，可能浪费这些位，因为视频的难以编码的部分通常是快速运动或淡入淡出，因此从本质上讲是瞬态的，并且各个帧都在变化如此之快以至于人类很难看到高的单个帧质量。默认值0.6应该被认为是一个合理的折衷，但是似乎很少有证据支持该特定值，因为它实际上是从较早的XviD编码器（x264的前身）继承而来的。

一个量化压缩的详细讨论结果表明，假设进行2遍编码和合理的峰值比特率设置，则高于0.6的值会产生更好的结果，而我们自己的测试很容易证实了这一点。对于低比特率编码而言，这种差异尤其明显，默认值0.6经常会产生"看起来不错的静态和慢动作场景，以及非常差的快速移动场景"。似乎普遍同意，对于低比特率编码，高于0.6的值更好-甚至那些主

张默认值保持在0.6的人也承认这一点。当然，对于高比特率编码而言，这并不重要，正是因为它们是高比特率，所以质量已不再是问题了–两者看起来都很好，而且0.6之间可能没有明显的区别和0.9。底线是**可变比特率绝对是视频编码的*魔力***，而**我们要做的最后一件事就是抑制这种魔力。**

***全部：*** 0.9。

## *最小量化（--qpmin）*

从0（真1）默认情况下，提高最低量化*不应该*是必需的，而事实上只会迫使在极少数情况下，框架的某些区域比理想的低质量的，其中一个非常低的量化可以用于非常高细节的块。不幸的是，**某些播放器（例如QuickTime）在播放量化器低于3的文件时会出现错误**，这就是x264Encoder QuickTime插件默认默认为最小量化器4和HandBrake同样默认为3的原因。幸运的是，将最小量化器提高到3基本上对现实世界的质量没有影响，因为任何低于10的量化器实际上都是无损的。

***全部：*** 3，用于QuickTime兼容性。

## *每宏块比特率控制前瞻（--rc-lookahead）*

x264使用复杂的宏块级比特率控制（“ MB-tree”），可跟踪来自未来帧的实际运动矢量对每个宏块的引用程度，从而使编码器仅降低正在变化的每个帧区域的质量快速（以后将不再引用），而不是像大多数编码器那样降低整个帧的质量-本质上是传统的比特率控制，但适用于每个16x16宏块的级别，而不是整个帧的级别。在存在移动的前景对象的情况下，这尤其有助于保持清晰，稳定的背景。

使用**此每宏块的比特率分析增加质量更长的“先行”**，允许更有效的细粒度，每个宏块使用的可用比特率的，**但编码将需要更长的时间，并使用显著多个存储器**，特别是在高分辨率（有毕竟，在一个1920x1080高清晰度帧中有超过8000个16x16宏块）。x264的默认值为40帧，随着距离增加到大约60帧，返回值会递减，超出此范围没有实际收益。

***全部：*** 60。

## *场景变化检测（-场景）*

关键是要检测**何时场景已更改，以及何时该是插入关键帧**（I帧）**的好时机**。x264的场景变化检测设置与直觉相反，变化阈值则相反- **较高的值会*增加*检测到的场景变化数量**。x264的默认值为40，但是奇怪的是x264Encoder QuickTime插件的预设全部使用80，这将导致检测到很多不必要的场景更改。

***全部：*** 40（默认为x264）。

## *最大关键帧间隔（--keyint）*

关键帧（I帧）之间的最大时间间隔对质量有重大影响，这使其成为最重要的调整设置之一，也是最困难的决策之一。编码器当然会尝试在场景更改时使用关键帧（请参见上文），但是对于很多内容，此值很重要，因为许多场景的时间超过5甚至10秒。有**太多的关键帧，严重降低质量**，因为来自先前帧的复用图像区域的效率在每个关键帧完全丧失-编码器要在每个关键帧，以“重新开始”。因此，对于给定的目标比特率，我们想要尽可能少的关键帧来实现最高质量。

另一方面，我们**仍然希望有足够的关键帧以使搜寻和快进行为良好**，因为播放器只能在播放过程中直接跳到“引擎盖下”的关键帧，并且通常只会在快进和快退时显示关键帧。更高的速度（在低速（例如2倍或3倍）下，它们通常可以播放每帧）。因此，跳到时间轴上的任意点会变得越来越迟缓，因为关键帧越少，因为仅为了重构最终目标帧就需要解码更多的中间增量帧，即使先前关键帧和目标帧之间的那些中间帧赢得了实际上没有显示。如果视频是使用自适应流媒体部署的，播放器可能会在播放期间根据可用的网络带宽在不同版本之间动态切换，因此这种切换也只能发生在关键帧（要切换到的流）上，因此我们也不希望关键帧之间的距离太远。

x264的默认值为250帧（以30fps的速度为8.3秒），但是没有充分的理由，x264Encoder QuickTime插件的默认值为60帧（2秒）。x264团队最近在一年一度的MPEG-4 AVC / H.264视频编解码器比较竞赛中使用了500帧（16.7秒）的设置，但这肯定会使事情有些过头了。数字电视通常使用1或2秒，但这故意过短，以使频道切换速度更快，并在发生干扰时快速恢复错误。DVD使用非常短的～0.5秒关键帧间隔，并且使用Blu-ray1秒，因为它们使用非常高的比特率（因此质量不是问题），并且尽管从相对较慢的光盘读取，但它们仍要保证良好的快进行为。对于Internet视频，最常见的建议是10秒。

***全部：*** 12秒（30帧/秒，30帧/秒），足以覆盖绝大多数场景，并且仅产生“自然的”场景更改关键帧，大约可以推动它。在最坏的情况下，快进时每12秒只能看到一帧视频，这几乎是不可能的。假设以10倍的速度快进，这意味着每分钟6秒（5个关键帧），从而每个关键帧在屏幕上显示1.2秒。跳到时间轴上的任意点当然也感觉很慢，但是即使在1080p时*也*不会*太*痛苦，并且随着速度更快的计算机的使用，它会变得更好。

## *最小关键帧间隔（--min-keyint）*

在**H.264格式实际上支持两种不同类型的I帧** -传统的关键帧"IDR" I帧代表合适的重新启动点（IDR代表"瞬时解码器刷新"），和其他较小的，非IDR I帧其中，非IDR I帧之后的帧仍可以引用I帧之前的帧，这意味着非IDR I帧 *不能用作重启点*，但仍被编码为I帧！后一种情况仅对处理极少发生的极端闪光和其他突然的单幅质量变化非常有用。

最小关键帧间隔通过防止将完整的IDR I帧放置得比此数量的帧更近来控制每次出现这两种类型的I帧中的哪一个。x264的默认值是最大关键帧间隔的10%，通常大约为1秒，这是有道理的，因为不需要比这更接近两个重启点。不幸的是，众所周知，**某些播放器在具有非IDR I帧的视频中进行**快进或擦洗时会遇到麻烦，包括一些最新版本的QuickTime和Flash，这就是x264Encoder QuickTime插件默认默认值为1的原因，I框架完整的IDR I框架。

***全部：*** 1（默认为x264Encoder，而不是默认为x264），用于与QuickTime，Flash和其他各种播放器兼容。

## *运动估计搜索模式（--me）*

归根结底，低比特率的高质量视频编码就是要**在先前的帧中找到相似的图像区域，然后重新使用它们。**搜索模式是在运动估计期间用于搜索与每个可能参考帧中的每个宏块最相似的区域，以便为每个宏块选择最佳运动矢量的模式。**这是花费大量编码时间的地方。**更彻底的搜索模式将找到更好的匹配，从而产生更好的运动矢量，从而导致运动补偿后剩下的要编码的残差图像更少，因此在给定的目标比特率下质量更高。当然，更彻底的搜索在编码过程中也需要花费更长的时间。

最简单的搜索模式是简单的4点菱形（左/右/上/下），下一个最简单的是6点六边形，然后是复杂的不均匀多六角形（UMH），最后是成熟的穷举搜索（确实非常慢）。

x264的默认值是简单的六边形，但众所周知，**x264实施不均匀多六边形搜索是其相**对于其他编码器的**最大优势之一**，通常可实现完全穷举搜索的0.5%。面向质量的预设，包括x264Encoder QuickTime插件的"优化"预设，均使用不均匀的多六边形搜索。

***全部：*** 凹凸不平的多六角形。

## *运动估计搜索范围（-范围）*

如上所述，最后，低比特率的高质量视频编码都是关于寻找相似性区域并重新使用它们的。**搜索区域越大，编码器就越有可能找到良好的匹配**，从而留下较少复杂的残差图像进行编码，从而在给定的目标比特率下产生更好的质量，但是**较大的搜索也将花费更长的时间**。自然，因为相同的相机或物体移动在高分辨率情况下覆盖了更多像素，所以高清材料从更大的搜索范围中受益的多于更低的分辨率。x264的默认值为16像素，但是请注意，对于不均匀的多六边形搜索，搜索模式会更改并在几个不同的级别上进行迭代，因此该范围并不是*字面上的* 精确为16像素。

搜索范围的增加会严重降低收益，在大多数情况下，由于看不到增益，编码时间大大增加，这是因为距离较远的区域不太可能非常相似。另一方面，较大的搜索范围对高动态场景最为有用，这些场景恰好是最难编码的场景，在这种场景中质量最有可能出现明显下降，并且最需要找到任何可用的相似性紧迫。因此，**较大的搜索范围虽然通常无济于事，但在快速运动的这几个关键时刻至关重要，这**对编码**的整体感知质量**产生**了巨大影响**（"您做简单的事情不是很好，您对困难的工作做得如何"）。

***240p-720p：*** 32，这对于720p而言具有出色的覆盖率，并且对于较低的分辨率来说是过大的杀伤力，但是由于其较低的比特率，我们需要在较低的分辨率下可以找到的每一个相似之处，并且无论如何在较低的分辨率下编码都相对较快。

***1080p：*** 48，以说明在1.5x1.5倍大帧内的相同相对运动。

## *子像素运动矢量细化（--subme）*

运动矢量细化控制编码器应花费多少时间和精力进行H.264四分之一像素运动矢量的宏块划分决策和最终运动矢量细化。甲**可能的最终运动向量的更全面的评估会找到更好的匹配**，从而产生更好的运动矢量，从而导致在给定的目标比特率左到运动补偿后编码，因此更高质量的较不复杂的残留图像。自然，在编码过程中**更全面的评估也将花费更长的时间**。最终，就质量与编码时间的权衡而言，此设置和前两个设置（搜索模式和搜索范围）是"橡胶与道路相遇"的地方。**这是我们为更高质量的编码付出的代价。**

设置1-5不使用速率失真优化，而是用于快速，低质量的编码情况，例如实时视频会议。设置6启用I和P帧的速率失真优化，设置 7添加B帧的 RDO ，设置 8启用I和P帧的 RDO优化，设置 9添加B帧的 RDO精细，设置10使所有帧的四分之一像素RDO细化（需要完整的RDO，请参阅下文）。x264的默认值为7。

***全部：*** 10。

## *预测的运动矢量（直接）*

H.264格式允许编码器使用**预测的运动矢量，而不是实际对每个矢量进行显式编码**，从而节省了一些位，从而略微提高了质量。该预测可以是空间的（根据相邻块进行的预测）或时间的（根据先前的帧进行的预测）。x264的默认设置是空

间设置，但它也提供了一种自动模式，该模式为每个帧选择最佳选择（需要2遍编码）。

***全部：*** 自动。

## *参考帧数（--ref）*

与以前的编解码器不同，H.264支持多个参考帧，因此**每个P或B帧中的每个宏块都可以使用其运动矢量引用不同的参考帧**，从而可以找到更好的匹配项，从而减少了复杂的残差图像。编码，从而在给定的目标比特率下获得更好的质量。在某些特殊情况下（例如，框架中突然的较大但临时的变化），仅允许使用2个参考框架会产生很大的质量改进。这从根本上解决了在处理名人新闻事件等内容时困扰诸如MPEG-2之类的早期编解码器的"照相机闪光灯"问题，因为有问题的闪存帧的内容不必对未来的帧产生负面影响，现在可以将其指向框架*之前* 闪光灯代替。

**将参考帧的数量增加到超过2可能会找到更好的匹配，但是自然会遭受** 3或4帧后**返回值的严重降低**，因为时间上相隔较远的帧可能越来越不同，因此对于寻找相似之处。参考帧数量的增加也极大地增加了编码时间，因为运动估计搜索（视频编码的最慢部分）必须在所有可能的参考帧上进行，以找到最佳匹配。

硬件解码器对播放期间允许的最大参考帧数进行了限制，以其支持的H.264级别表示（请参阅前面的部分）。尽管参考帧数的限制在名义上与"视频内存大小"有关，但实际上，根据内存系统在30 MHz时的预期性能，对H.264级别的任何限制实际上更多地是解码速度的一般指标。给定数量的参考帧的工作量。参考帧越多，内存位置（重用）越低，活动工作集就越大，这意味着更多的缓存未命中以及等待主内存所花费的时间更多，这大大降低了性能。硬件的内存延迟和带宽在这里起着重要的作用，常规CPU的L2 / L3高速缓存的大小和速度，或SoC视频解码器模块中内部暂存RAM的大小也是如此。*远远*超出所需）。

> 在其正常H.264级别4.0 / 4.1的1080p处，最大参考帧数为4帧。YUV 4：2：0格式的1080p帧占用1920x1080x1.5字节或不到3 MB，因此4个参考帧占用12 MB，外加3 MB用于当前正在解码的帧，这意味着4.0级播放器将15 MB用于这些帧从概念上来说很适合16 MB，但远远超过了当前大多数主流处理器的片上内存（当前CPU的L2 / L3缓存范围为2至8 MB）。类似地，正常水平为3.1的720p最多具有5个参考帧，这对于播放器来说仅需不到8 MB的空间，以及我们用于360p的3.0级，对于720x480 NTSC最多有6个参考帧，对于720x576 PAL最多有5个参考帧，据推测，这大约不到4 MB。

不幸的是，一些流行的硬件解码器*无法*流畅地播放使用解码器声称的最大参考帧数的视频（例如：旧版本Android上的NVIDIA Tegra 2处理器），如果播放数量超过该数量，则会导致回放过程中没有回放或严重卡顿参考帧的数量太高。这通常是由于缓存大小不足和/或缺乏适当的预取功能来覆盖主内存延迟所致。出于类似的原因，一些软件播放器还需要大量参考框架。

x264的默认值为3个参考帧，这也是YouTube使用的参考帧。iTunes为1080p使用4，但对于720p仅使用2，大概是为了与较慢的较慢计算机兼容（Apple出于相同的原因，也避免了720p的CABAC，并使用高比特率进行补偿）。到了极点，iTunes仅使用1个参考帧进行480p，这很荒谬，因为对于任何现代计算机而言，播放480p H.264基准视频应该相对容易。也许苹果只是在超保守，以防将来出现问题，他们希望他们的后备SD视频在要求的性能方面尽可能不高要求，但即使不使用2个参考帧也会使事情变得过分！

***全部：*** 4，应该发现尽可能多的有用相似之处，不会浪费可笑的编码时间，符合目标H.264级别，对于几乎所有已知的硬件解码器都是安全的（假设Android 3.1+（在Tegra 2处理器上），并且不应过分强调软件播放器。

兼容性：Sony PlayStation Portable（PSP）是一款重要设备，无法使用我们选择的4个参考帧播放H.264视频。鉴于其480x272的屏幕分辨率，再加上PSP的各种视频播放限制，PSP可能播放的视频的唯一版本将是非宽屏内容的最低质量的320x240版本，而对于-更常见的宽屏内容。因此，我们根本不支持将PSP用作视频的目标。

性能：测试表明，将参考帧的数量减少到仅2个并不能显着提高众所周知的速度较慢的软件播放器（例如Windows上的QuickTime 7）的播放速度。两者之间的差异几乎无法测量，最多只能达到百分之几，也许以21fps而不是20fps的速度播放压力较大的720p HQ HQ视频（其他软件播放器在同一系统上可以达到全部30fps，所有硬件解码器也一样）。考虑到视觉质量的损失，实际中硬件加速播放的广泛使用以及长期的考虑，"坏"播放器的播放速度小幅增长是不值得的，因为这并不能真正解决问题–播放效果仍然无法令人满意，大多数人不会

## *B帧作为参考帧（--b-金字塔）*

在**H.264格式允许B帧被用作用于其它帧的参照帧**（为什么不是吗？），其**有时也被称为B帧金字塔**因为一个的"金字塔"B帧的形式，以较低的B帧指向较高的B帧，然后指向一个或多个P帧，最后指向I帧（关键帧）。

不幸的是，**某些播放器不能正确地将B帧作为参考帧处理**，从而在播放过程中造成视觉损坏。这通常是因为在播放过程中进行计算密集的去块步骤（请参阅后面的部分）比在B帧（所有帧中的大多数）上执行的"不那么充分"执行，以节省时间，有时甚至完全跳过了。结果，未正确解码的B帧不完全包含编码器期望的像素（并且H.264格式正式要求它们！），从而导致任何后续帧的视觉效果均不正确引用B帧，通常丑陋的涂抹/撕裂。

可悲的是，蓝光规格*不*要求B帧，以支持作为参考帧，即使是完整的H.264规范的一部分，作为结果，一些蓝光硬件解码器不支持。一些软件播放器还提供了减少或关闭B帧去块化的选项，以提高性能。因此，我们不能依靠可靠地实现H.264标准的那部分，而必须避免使用B帧作为参考帧。幸运的是，由于没有使用任何B帧作为参考帧，因此造成的质量损失非常小，与类似的I / P帧一样 几乎总是可用。

***全部：*** 关闭，以与不对B帧应用适当去块化的不良播放器兼容。

## *P帧早期跳过检测（--fast-pskip）*

**如果变化非常小，则** P帧跳过检测**允许编码器在处理的早期阶段跳过对P帧中宏块的考虑。**这样平均可以将编码速度提高约20%，在几乎所有使用理智的比特率的情况下，几乎没有质量损失。引用一位x264程序员的话：“ no-fast-pskip实际上什么也不做。这是一个安慰剂选项，如果您认为它有重要意义，那么您的视线可能会愚弄您，您应该停止对自己不喜欢的设置进行过多修改不明白。” 因此，x264的默认值为P帧跳过检测功能已打开，尽管x264Encoder QuickTime插件默认为关闭状态，这可能是由于一个古老的错误所致，即P帧跳过检测过于激进，并且不能很好地处理蓝天区域，该错误已经很长时间了自修复以来。

***240p-1080p：*** 开（默认）。

***1080p超级比特：*** 关闭，以确保我们捕获每一个最后的微小细节，因为该版本可作为将来重新编码的长期母带。

## *B帧的自适应数量（--b-adapt）*

x264编码器支持使用**自适应数量的B帧，而不是**像IBBPBBPBBPBB这样**的固定模式**，并且此设置控制该自适应决策。x264的默认设置是快速，简单的算法（1），但它还支持速度较慢，质量更高的“最佳”算法（2）。慢速算法是面向质量的预设（包括x264Encoder QuickTime插件的“优化”预设）的默认设置。奇怪的是，x264Encoder QuickTime插件的iPod预设将其设置为1，即使它们使用基线配置文件，因此也不使用B帧。

***240p / 360p：*** 0 （基线配置文件禁用了B帧）。

***432p +：*** 2（缓慢，“最佳”）。

## *最大B帧数（--bframes）*

x264编码器将自适应地决定何时使用B帧以及使用多少B帧（请参见上文），直至达到给定的限制。**允许更长的连续B帧序列对质量有好处，**因为B帧在压缩方面是最有效的帧类型，但是考虑到大量B帧会大大减慢编码速度，并且由于编码器很少选择而导致收益递减实际使用4或5以上，而1-3则更为常见。

较长的B帧序列还会冒着不断增长的错误传播的风险，导致视频质量逐渐下降，然后在视频切换回更高质量的P或I帧时出现轻微可见的“脉冲”。这类似于较早的DivX / XviD编解码器（H.264的前身）在以非常长的P / B帧序列编码的中等运动内容期间出现的“关键帧泵送”拖尾脉冲问题 尽管由于关键帧间隔通常至少是几秒钟，而不是几分之一秒，因此DivX / XviD问题要严重得多，但它是由关键帧间隔过长引起的，从而留出了更多的时间进行错误积累涂片，使得跟随“脉冲”的质量跃升更为明显。

请注意，设置为2时，所有帧中的约67%将为B帧，如果 3 帧上升到最大75%，则4上升到最大80%，5上升到最大83%。编码器实际上经常使用5个连续的B帧，这不太可能。换句话说，B帧类型在所有情况下（IBBBPBBBPBBBBPBBBBPBBBPBBB）都在所有其他帧类型中占主导地位，并且此设置仅相对于百分比略微增加了它。

x264的默认值是最多3个B帧。YouTube仅使用2个B帧，大概是为了加快编码速度。Jan Ozer建议使用2或3（为什么2？只是编码时间？）。大多数x264程序员在定位最高质量时都建议使用更高的设置，有些甚至高达8甚至16，尽管这样做确实会使编码速度大大降低，但并没有增加实际效果– B帧的使用增加不到1%不足以对质量产生明显影响。**众所周知，某些硬件解码器会出现较长的B帧序列**，例如某些较早的ATI / NVIDIA GPU和许多早期Android手机和平板电脑中使用的NVIDIA Tegra 2处理器。

***240p / 360p：*** 0 （基线配置文件禁用了B帧，在x264Encoder中显示为1）。

***432p-1080p：*** 3（默认值），用于与较旧的ATI / NVIDIA GPU和基于Tegra 2处理器的Android设备以及可能具有类似错误的其他硬件解码器兼容（3 B帧是相当“标准”的设置，并且很多人都在使用它，因此它几乎不可能破坏任何东西）。

***1080p超级比特：*** 5，大约与实际使用的一样多。

## *速率失真优化（-网格）*

速率失真优化（aka：trellis）是一种缓慢而有效的蛮力优化技术，它使用视频质量指标针对每个可能的选择详尽地测量了原始母版的失真和以位为单位的实际成本，一直处理到最终的熵编码，**本质上考虑了每种可能性，并根据**所使用**的RDO度量，根据成本与收益进行了最佳选择。**默认情况下，x264使用非常好的“心理视觉” RDO指标（请参阅下文），但也可以指示x264 使用更简单的指标，例如PSNR或SSIM。

速率失真优化最初旨在优化量化（--trellis = 1，x264的默认值），但也可以在编码管道的较早版本中使用，从运动矢量精炼开始，包括宏块划分和块类型决策（--trellis = 2）。以前，x264的基线配置文件不支持速率失真优化，因为它仅与CABAC最终熵编码一起使用，但是在现代版本的x264中不再如此。

尽管RDO确实在诸如宏块分区和量化之类的可能性中选择了最佳选择，但是该选择仅在*本地*意义上是最佳的，仅适用于那个宏块。它没有考虑整体的，更全局的情况，包括该帧或将来帧中的其他块可能在何处以及如何引用该宏块。因此，将术语"最佳"用于RDO是常见的但不准确的。RDO度量标准也不是与人类感知的完美匹配，但是即使使用了RDO度量标准，即使使用了无限范围的穷举运动矢量搜索，RDO仍不能保证对当前帧的真正最佳编码。整个视频。尽管如此，RDO确实提供了很好的选择，在大多数情况下可能接近最佳选择。

***全部：*** 2（用于运动矢量细化，宏块划分和量化）。

## *心理视觉优化（--psy-rd）*

与其他编码器相比，x264的主要优势之一是它使用心理视觉优化来提高*主观*质量。心理视觉优化**试图更好地匹配人类视觉系统*对*图像的*感知和解释*。**从理论上讲，我们的大脑处于这样一种状态，即它不只是希望图像看起来与原始图像相似，还希望图像感觉自己具有相似的复杂度–否则，即使看起来也"更糟"从技术上讲，如果在技术上更接近原始版本的话。

换句话说，**我们人类宁愿在图像上看到一个稍微失真但同样精细的区域，而不是一个不失真但稍微模糊的区域。**因此，此优化通过更改速率失真度量以在量化，宏块划分等过程中取消强调模糊的"低误差但低能耗"的选择而起作用，而不是使用峰值信噪比（PSNR或图像的结构相似性（SSIM），它们倾向于趋向于较少的实际数字像素差异，但趋于过多的模糊。

可以将心理视觉优化效果的强度从默认值1.0进行调整，或者增加到1.5左右以更趋向于更细微的细节，纹理，胶片颗粒和最终的噪声伪影，或者减小到0.5左右以趋于于向更进一步的倾斜。平滑，平坦，最终输出更加模糊。还有一个心理视觉格网设置，由于仍被认为是实验性的，因此目前已被禁用（默认值为0.0），但将来启用时也应默认为1.0，对于基于颗粒胶片的内容和用于清洁计算机图形。

***全部：*** 1.0 / 1.0（可能是未来的默认设置）。

## *基于DCT的抽取（--dct-抽取）*

基于DCT的抽取允许编码器基于简单的DCT阈值测试跳过它认为不必要的编码块，因为要编码的残差非常小，因此可能在视觉上无法察觉。这样**就避免了对块执行慢速失真优化的需要，从而加快了编码速度，该过程可能不会获得任何分配给它们的位**，因此在大多数情况下几乎不会造成质量损失。它还趋于稳定图像区域的微小变化，如果这些微小变化不是"真实的"，则在视觉上是一件好事。

自然，让速率失真优化通过考虑所有块，让速率失真优化决定应该更积极地跳过或量化哪些块，这样质量会更好，因为RDO会在不依赖于任意值的情况下产生"最佳"选择（即使非常好）。）阈值测试，但是基于DCT的抽取通常可以大大提高编码速度，而不会造成可见的质量损失，尤其是在背景稳定的情况下，例如"会说话的头"镜头。

不幸的是，基于DCT的抽取**有时会导致**在缓慢的淡入淡出或非常黑暗的场景中**跳过细微的变化**，即使在高比特率下也会导致条纹或步进。胶片颗粒或相机传感器的噪声通常足以满足基于DCT的抽取的要求，因为分辨率和比特率足够高，足以捕获该细节级别，但颗粒不可见或**非常干净的**黑暗场景对于基于DCT的抽取，**像计算机图形这样的资源可能是一个问题**。x264的默认设置是启用基于DCT的抽取。

***全部：*** 关闭，以在诸如计算机图形之类的干净资源情况下保持良好的缓慢衰减。

## *P帧中的4x4分区（--partitions p4x4）*

花时间考虑使用较小的4x4宏块分区大小（4x4、4x8和8x4）的可能性在I帧中很有用，但在P 帧中却没有多大益处（并且在B帧中根本不支持）x264）。它**极大地减慢编码一个微不足道的增益质量最高分辨率正常**，虽然它**可以在低分辨率略有提高质量，如果比特率足够高**，以利用它来捕捉更多的细节。要引用的X264开发商之一："它（原文如此）几乎没有用，除了在相对较高的比特率下，即使如此，它在低分辨率下也特别有用，即使那样也不是那么好。" x264的默认设置是不考虑P帧中的 4x4分区。

***240p / 360p：*** 继续，因为我们需要在非常低的分辨率/比特率下可以获得的一切，尤其是考虑到我们在这些分辨率下只能使用H.264基线配置文件，而较小的块尺寸在低分辨率下可能会有所帮助，再加上在低分辨率下的编码时间成本仍然是合理的。

***432p +：*** 关闭（默认）。

## *解块（--deblock）*

与早期的编解码器不同，H.264希望播放器在播放期间执行某种形式的解块操作，以使宏块的边缘模糊化，从而避免最终图像中出现可见块（人眼比虚假边缘更不易察觉）。事实上，**虽然一些优秀的球员更早的编解码器可以*任意*申请解封，H.264实际上*需要*它，采取的是事实的优点，以提高整体压缩**使用显著*解封*帧作为未来P-和参考帧B帧。因此，H.264有时被称为具有"环路内"解块功能，因为**它是正式播放过程中的必不可少的部分**，而不是可选的额外功能。

解块是H.264播放中计算量较大的部分之一，虽然为了节省时间而不能跳过或"少做**一些**"以节省时间，但**某些播放器在B帧上执行的解块较少（所有帧中的大多数）），这就是为什么这些播放器不支持将B帧用作**其他帧的参考帧的原因（请参见前面的部分）。

可以在编码期间设置解块平滑的强度和阈值...

- **强度**（也称为Alpha）控制沿块边缘**的平滑量，**对于大多数情况，默认值为0。最高约+2的正值可以使图像更平滑，最终使图像柔化/模糊，而负值约低到-3则可进行较少的平滑，并保留更多的清晰度/细节，这会带来一些可见的块状伪影。低于-3的任何内容都不应真正考虑。
- **阈值**（也称为beta）确定要激活**平滑块所必须达到的平坦程度**，0是一个很好的默认值，很少需要更改。实际的阈值计算非常复杂– 引用x264开发人员的话："有一个基于[qp + 2 * alpha]的阈值，另一个基于[qp + 2 * beta]的阈值。纹理/渐变/必须通过两个阈值在应用任何过滤之前。" 最多只能将β阈值进行很小的更改，但是将-1稍作更改可以提高总体清晰度，但有一些可见的遮挡的危险。

***240p-1080p：*** 0：0（默认）。

***1080p超级***比特**：** -2：-1，因为我们的比特率非常高，因此可以保持尽可能高的清晰度，因此不会出现可见的阻塞伪像。

性能：减少去块的数量确实提高了非常慢的软件播放器（例如Windows上的QuickTime 7）的播放速度，但即使beta阈值降至-2，差异也仅适中，只有几个百分点。考虑到阻塞工件的视觉丑陋性，在实践中广泛使用硬件加速的播放以及长期考虑，在"不良"软件播放器上仅获得很小的播放速度是不值得的。

## *色彩空间标记*

仅使用简单的数字值（例如（175，0，0））指定的颜色实际上有点模棱两可。当然，这意味着175 / 255ths最大的红色的，但*什么*最红？它的实际外观将取决于屏幕的亮度，伽玛曲线（亮度响应曲线），色彩饱和度，色域（整个色彩范围），色温（温暖/凉爽），白点设置等。因此，我们被迫使用颜色空间的**概念，**该**概念保存在称为颜色配置文件的文件中**，该标签应用于标记所有面向照片的图像和视频，以便它们可以正确显示，并且在不同屏幕上的外观大致相同。

色彩空间标记的主要例外是网站上使用的图像，您应该在其中将图像转换为网络的标准sRGB色彩空间，而*不要*嵌入颜色配置文件，以确保在不支持色彩空间的Web浏览器中保持一致的颜色，以及减少文件大小（嵌入颜色配置文件会额外增加2-10k，对于小型网络图片而言，这是很大的开销）。对于网络，您可以基于以下假设：用户屏幕的色彩空间约为sRGB，且伽玛值为2.2，或该像素的某种合理平衡的超集。您不能假设Web浏览器支持色彩空间，因为大多数浏览器都不支持。

视频文件的色彩空间和/或gamma标记在输出的QuickTime文件的标头中插入了*colr*（现代）或*gama*（旧的，已弃用的）标记，并**在实际的H.264比特流本身中插入了*VUI*参数**，以指示文件的色彩空间。然后，可感知色彩空间的现代视频播放器应用程序可以将文件的颜色和亮度级别准确地映射到回放屏幕上，以便显示视频，从而使视频看起来与您预期的大致相同，并且在不同的画面上相同。

现在，仅Gamma标记已过时，并且H.264格式的VUI参数不支持此标记，并且在转换为建议的.mp4容器文件格式时，所有QuickTime级别的Gamma标记都丢失了，因此，如今仅涉及现代色彩空间标记。

现代的视频文件色彩空间标记使用**3个数字，每个数字都**指向**已知广播标准表的索引：** RGB颜色基色定义了"红色"，"绿色"和"蓝色"的确切含义；传递函数（伽玛的现代等效形式）；和转换矩阵。它们一起形成"非恒定亮度编码"或NCLC。

不幸的是，许多当前的视频播放器应用程序不支持色彩空间标记，包括Flash，Windows Media Player（大多数配置，有时被GPU驱动程序覆盖），VLC和MPlayer，所有这些都只是通过数值不变或使用它们自己的颜色来发送色彩内部，通常是固定的，通常是"唯一的"颜色/伽玛校正。

具有多种播放器应用了大量的实验，在各种不同的操作系统，手机，平板电脑和独立的电视屏幕上后，经过大量的和大量的和大量的阅读的主题，只有3周色彩空间的Tagging这确实物...

- **没有标记**（默认）不会声明任何有关视频文件色彩空间的内容，而是将其置于视频播放器应用程序的幻想之下，这不是一个好主意，尽管无论如何对于许多播放器来说都是如此，因为许多人会忽略色彩空间标记。
- **SMPTE-C色彩空间（NCLC 6-1-6）**是**旧的NTSC标准清晰度电视色彩空间**的当前版本，也称为"建议601"。它的官方白点为D65（6500K），灰度系数为2.2，尽管大多数家用NTSC CRT显像管通常都较暗，实践中的灰度系数为2.3-2.5（甚至是好的）。因此，许多不支持色域的较早的播放器应用程序都将假定视频已编码，期望伽玛值约为2.3-2.5，并且如果屏幕不是那么暗（或者播放器仅假定屏幕的伽玛值为2.2）。播放器将使视频变暗以（不正确地）进行补偿-因此，没有色彩空间意识的播放器将显示的视频常常比应显示的暗一些。

在视频播放器应用的广泛缺失的色彩空间感知功能，是关于"QuickTime对伽马虫"，这是具有讽刺意味，因为QuickTime播放器的的为数不多的球员之一许多文章的主要原因*不*尊重色彩空间和伽玛标签。"错误"实际上更多是缺乏可配置性，因为QuickTime H.264 *编码器*使用SMPTE-C颜色空间（NCLC 6-1-6）标记所有SD尺寸的编码，而使用HDTV颜色空间标记所有HD尺寸的编码。（NCLC 1-1-1），其中QuickTime 在导出过程中根据需要执行颜色*转换*，因此它们最终*在支持色域的播放器中*看起来一样，但在忽略色彩空间标记的播放器中有所不同！QuickTime H.264编码器应

- **HDTV色彩空间（NCLC 1-1-1）是现代高清内容和屏幕**的色彩空间标准，也称为"建议709"。像SMPTE-C一样，它的官方白点为D65（6500K），并且伽玛值总体约为2.2，尽管它在黑色附近包含一个小的线性部分。然而，这一次，实际上2.2的伽玛实际上反映*在*了良好的LCD /等离子屏幕上，从而实现了更好的阴影细节。HDTV还定义了稍有不同的，更好的红色/绿色/蓝色原色，从而允许更多的饱和色。HDTV色彩空间具有标准sRGB色彩空间形式的非常详细的同级，它使用与HDTV相同的红色/绿色/蓝色原色和白点，并且具有近似相同的2.2近似伽玛（尽管在精确的伽玛曲线上存在一些细微但不明显的差异）。

提供一个选项来显式设置输出文件的色彩空间（如x264Encoder那样），但遗憾的是，它不可以-标签本身可以在事后由其他工具更改，但是颜色已经可以已转换为新的色彩空间。尽管如此，QuickTime Player和QuickTime插件实际上通过在播放期间遵守标记来做"正确的事情"。只是世界其他地方在做错事，包括Apple的其他一些应用程序。

请务必注意，**SMPTE-C和HDTV色彩空间略有不同**，并且实际上相差很多，可以在并排查看或在它们之间翻转时清晰可见（您可以阅读这些区别的简短摘要，或者更详细地介绍）。 。但是，可以安全地假定**大多数现代电视屏幕都希望尽可能地与HDTV / sRGB色彩空间匹配**，并且假定较早的SMPTE-C内容可以映射到新的色彩空间–在最坏的情况下它看起来会有点褪色（那个关于NTSC的笑话又意味着"不再有相同的颜色"？）。

现代计算机屏幕的标准化程度不高，但其技术与LCD HDTV相同。所有主要操作系统建议的灰度系数为2.2，尽管实际值会根据屏幕的实际颜色配置文件而略有不同。**计算机屏幕**的红色/绿色/蓝色原色和**色域变化相当大**，并且色域通常比HDTV差一些，至少是好色域，这是由于将具有良好背光源的好屏幕嵌入薄型屏幕的限制，轻便的笔记本电脑外壳。同样，可以放心地假设**大多数计算机屏幕都尝试或多或少地匹配HDTV / sRGB色彩空间。** 大多数手机和平板电脑还尝试遵循大约2.2的伽玛值和与HDTV / sRGB相似的原色，尽管现实世界中的设备之间再次存在很大差异。

***全部：*** HDTV（NCLC 1-1-1），请记住，某些播放器和屏幕会将视频显示得暗一些。