

THE EXPERT'S VOICE® IN DIGITAL VIDEO PRODUCTION

Digital Video Concepts, Methods, and Metrics

Quality, Compression, Performance, and Power Trade-off Analysis

THE OPTIMIZATION GUIDEBOOK
FOR MULTIMEDIA AND
TELECOMMUNICATIONS PROFESSIONALS

Shahriar Akramullah

Apress
open

Table of Contents

前言

前言	1.1
----	-----

数字视频简介

数字视频简介	2.1
相关概念	2.2
视频压缩	2.3
权衡分析	2.4
新型视频应用	2.5
总结	2.6

视频压缩技术

数字视频压缩技术	3.1
网络限制和压缩	3.2
人类视觉系统	3.3
HVS模型	3.3.1
HVS的应用	3.3.2
压缩技术概述	3.4
数据结构和概念	3.4.1
色度亚采样	3.4.2
降低冗余	3.4.3
熵编码	3.4.4
压缩技术: 成本-收益分析	3.5
变换编码技术	3.5.1
预测编码技术	3.5.2

其他编码技术	3.5.3
率失真理论	3.5.4
总结	3.6

视频编码标准

视频编码标准	4.1
视频编码的国际标准概述	4.2
JPEG	4.2.1
H.261	4.2.2
MPEG-1	4.2.3
MPEG-2	4.2.4
H.263	4.2.5
MPEG-4 (Part 2)	4.2.6
AVC	4.2.7
HEVC	4.2.8
视频质量的国际标准	4.2.9
其他工业标准概述	4.3
VC-1	4.3.1
VP8	4.3.2
VP9	4.3.3
总结	4.4

视频质量度量

视频质量指标	5.1
压缩损失, 伪像, 视觉质量	5.2
压缩损失: 量化噪声	5.2.1
常见的伪影	5.2.2
影响视觉质量的因素	5.2.3
视频质量的评估方法和指标	5.3
主观视频质量评估	5.3.1

客观视频质量评估和指标	5.3.2
基于误差灵敏度的方法	5.3.2.1
峰值信噪比	5.3.2.2
基于结构相似性的方法	5.3.2.3
基于信息保真度的方法	5.3.2.4
时空方法	5.3.2.5
基于显著性的方法	5.3.2.6
网络感知方法	5.3.2.7
基于噪声的质量指标	5.3.2.8
客观编码效率指标	5.3.2.9
基于ITU-T标准的客观的质量度量方法	5.3.2.10
视频质量测量	5.4
主观测量	5.4.1
客观测量及其应用	5.4.2
调参	5.5
影响视频质量的参数	5.5.1
参数之间的权衡	5.5.2
总结	5.6

视频编码性能

视频编码性能	6.1
CPU速度和限制	6.2
提升性能的动机	6.3
对性能的考虑	6.4
资源利用率最大化	6.4.1
专用资源	6.4.2
调整视频参数	6.4.3
决定编码速度的因素	6.4.3.1
系统配置	6.4.3.1.1
工作负载的性质	6.4.3.1.2
编码工具和参数	6.4.3.1.3

独立数据单元	6.4.3.1.3.1
GOP结构	6.4.3.1.3.2
码率控制	6.4.3.1.3.3
多帧参考	6.4.3.1.3.4
率失真的拉格朗日优化	6.4.3.1.3.5
隔行扫描的帧/场模式	6.4.3.1.3.6
自适应去块滤波器	6.4.3.1.3.7
视频复杂度和格式	6.4.3.1.4
基于GPU加速的优化	6.4.3.1.5
性能优化方法	6.5
算法优化	6.5.1
快速算法	6.5.1.1
快速变换算法	6.5.1.1.1
快速帧内预测算法	6.5.1.1.2
快速运动估计算法	6.5.1.1.3
快速模式决策算法	6.5.1.1.4
快速熵编码算法	6.5.1.1.5
并行化方法	6.5.1.2
数据分区	6.5.1.2.1
任务并行化	6.5.1.2.2
流水线技术	6.5.1.2.3
数据并行化	6.5.1.2.4
指令并行化	6.5.1.2.5
多线程技术	6.5.1.2.6
向量化技术	6.5.1.2.7
编译器和代码优化	6.5.2
编译器优化	6.5.2.1
代码优化	6.5.2.2
超频	6.5.3
性能瓶颈	6.5.4
性能度量和调整	6.6
性能思考	6.6.1

性能指标	6.6.2
性能分析工具	6.6.3
总结	6.7

视频应用的耗电量

视频应用的耗电量	7.1
功耗及其限制	7.2
媒体应用的工作负载	7.3
媒体应用用途	7.3.1
面向电量设计	7.4
电源管理的思考	7.5
ACPI和电源管理	7.5.1
操作系统电源管理	7.5.2
Linux电源管理	7.5.2.1
Windows电源管理	7.5.2.2
处理器电源管理	7.5.3
Voltage-Frequency曲线	7.5.4
电源优化	7.6
架构级别优化	7.6.1
算法级别优化	7.6.2
系统整体级别优化	7.6.3
应用级别优化	7.6.4
电源度量	7.7
度量方法论	7.7.1
电源度量的思考	7.7.2
测量电源的工具	7.8
DC电源测量系统	7.8.1
电源测量的软件工具	7.8.2
总结	7.9

低功耗平台上的视频应用的功耗

低功耗平台上的视频应用的功耗	8.1
低功耗设备的重要事项	8.2
低功耗平台上典型的媒体应用	8.3
视频播放	8.3.1
视频录制	8.3.2
视频分发	8.3.3
视频电话（会议）	8.3.4
低功耗系统的状态	8.4
简单ACPI模型的缺点	8.4.1
待机状态	8.4.2
低功耗状态的组合	8.4.3
低功耗平台的电源管理	8.5
电源管理的专用硬件	8.5.1
显示器电源管理	8.5.2
低功耗平台的思考	8.6
软件设计	8.6.1
体系结构的思考	8.6.2
低功耗平台的电量优化	8.7
快速执行然后关闭	8.7.1
Activity调度	8.7.2
减少唤醒次数	8.7.3
突发模式	8.7.4
完善CPU和GPU的并行化	8.7.5
显存带宽优化	8.7.6
显示功耗优化	8.7.7
存储功耗优化	8.7.8
低功耗的度量	8.8
电源的处理器信号	8.8.1
媒体应用的功耗指标	8.8.2
总结	8.9

性能，电量以及质量的权衡

性能, 电量以及质量的权衡	9.1
权衡分析的思考	9.2
权衡分析的类型	9.2.1
参数调整的效果	9.2.2
优化策略	9.2.3
权衡性能和功耗	9.3
Case Study	9.3.1
权衡性能和质量	9.4
Case Study I	9.4.1
Case Study II	9.4.2
权衡功耗和质量	9.5
Case Study	9.5.1
总结	9.6

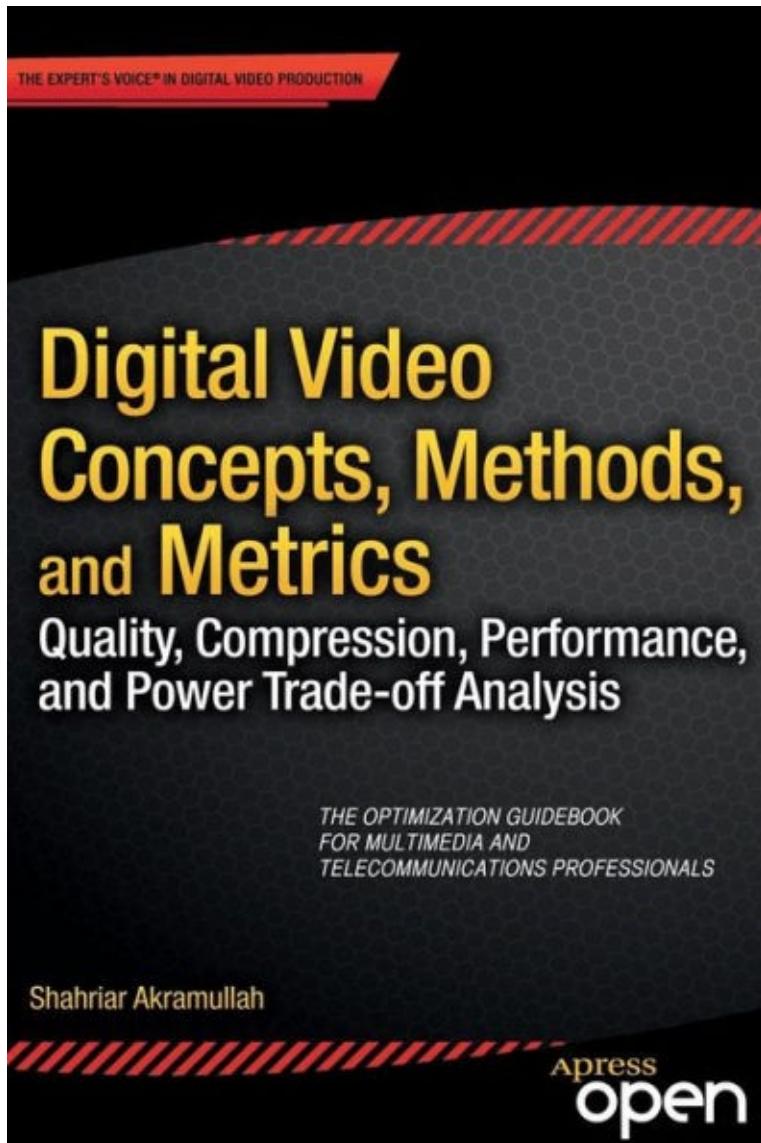
结语

结语	10.1
重点和结论	10.2
对未来的思考	10.3

数字视频概念，方法和测量指标：质量，压缩，性能和电量等的权衡分析（中文版）

该书是《Digital Video Concepts, Methods, and Metrics: Quality, Compression, Performance, and Power Trade-off Analysis》一书的中文版，仅供学习交流之用。希望对想学习和了解数字视频相关技术的同学，有所帮助。

在翻译的过程中，尽可能的保留了原书的内容，并且对原书中的个别内容进行了修正和补充。



项目主页：https://github.com/wangwei1237/digital_video_concepts.git

在线预览：https://wangwei1237.gitbook.io/digital_video_concepts

本项目采用gitbook方式编写，可以利用gitbook将该项目打包成静态网页格式或者PDF 格式，具体可以参见：[gitbook教程](#)。

译者信息：

- 王伟 (17哥) , [github @wangwei1237](#)
- 木子小兀, [github @muzixiaowu](#)
- 刘一卓, [github @GerenLiu](#)
- 孙晓洁, [github@webxiaojie](#)
- 吴玉, [github @wy-xyz](#)
- 杨光, [github @yangguang10](#)

数字视频简介

过去几十年，移动设备有了很多多媒体功能。比如，前向和后向摄像头在手机中是很常规的功能。更进一步的，设备供应商们在提供更高的捕捉，处理和展示高清视频的领域展开了激烈的竞争，也在寻找这个市场的差异化。多媒体的应用需要快速的捕捉能力，但是这些能力是以上升的电量消耗为代价的。任何续航方面的进步都一定程度的解决多媒体拍摄的问题，因此移动设备的续航时间成了一个关键因素。因此，通过各种方法降低系统功耗会是未来的决胜之路，许多因素会被考虑和权衡。

诚然，在当前的移动设备中高质量的图片和视频是很重要的多媒体组件。同时，在资源有限的环境中，特别需要高效的性能和优化的功耗系统。在过去几十年，涌现出很多工具和技术去解决既能尽可能的显示高清的视觉质量又能解决数字视频功耗和性能方面的问题。迄今为止，这些方面的复杂交互还未能得到有效的解决。

在这本书里，我们会学习数字视频相关的概念，方法和度量方案。另外，为了在视觉质量，性能和功耗之间取得更好的平衡，我们还调研了不同的调优参数方案。书的开始我们会介绍包括视觉数据压缩，噪声，质量，性能和功耗在内的一些数字视频的关键概念。接着我们讨论了视频压缩要考虑的因素，展示了一些视频编码用法和要求。我们接着研究了折中方案，权衡了他们好的用法，挑战和机遇和预期输出。最后，对于新应用这只是一个入门，这本书接下来的章节会建立在这些基础的主题之上。

相关概念

本节讨论了本书中讨论的一些关键概念，这些概念适用于压缩数字视频中的视觉质量，尤其是在当代移动平台上呈现的视觉质量。

数字视频

术语*video*通常指摄像机镜头捕捉到的视觉信息，这些视觉信息表现为时间区间内的图像序列。

视频摄像机源于20世纪30年代早期的电视行业，那个时候机电摄像机已经使用了数十年。之后，阴极射线管（*CRT*）技术的发明使得摄像机进入全电子版本的时代。20世纪80年代，固态传感器——特别是CMOS主动像素传感器（*CMOS active pixel sensors*）取代了模拟管技术，这使得数字视频得以应用。

早期的摄像机根据定义好的扫描规则将模拟视频信号转换为一维时变信号。这些信号使用模拟调幅（*analog amplitude modulation*）传输，并使用录像机或光学技术的模拟激光盘存储在模拟录像带。模拟信号不适合压缩，它们经常转换为数字格式，以便在数字领域进行压缩和处理。

目前，全数字工作流程（从数字视频信号的捕获到消费）的使用已经变得很普遍，主要是由于数字视频的如下特点：

1. 数字视频的记录、存储、恢复、传输、接收、处理更简单方便，它几乎没有错误，因此数字视频可以看作是当今计算系统的另一种数据类型。
2. 与模拟视频信号不同，数字视频信号具备压缩、解压缩的能力。较之于未压缩的格式，压缩格式的视频更容易存储和传输。
3. 随着廉价的集成电路、高速通信网络、快速访问密集存储介质、计算设备的高级架构和高效视频压缩技术的出现，按照不同平台（从移动设备到网络服务器和工作站）所需要的数据速率处理数字视频已经成为现实。

对数字视频的高度关注，特别是在移动计算平台上的应用，对人类活动产生了重大影响。对于未来而言，这种情况肯定会继续存在，并会扩展到整个信息技术领域。

视频数据压缩

视频数字信号需要大量数据来表示。对于视频数据的存储以及传输而言，某种类型的数据压缩是必要的，以便可以用于很多应用。数据压缩可以是无损的，因此在解压缩后可以检索相同的数据。它也可能是有损的，因此在解压缩之后仅恢复原始信号的近似值。幸运的是，视频数据的特征可以容忍一定量的损失，并且人类的视觉系统对这样的损失毫无感知。然而，考虑到系统的限制，所有视频信号处理方法和技术都尽可能的实现最佳的视觉质量。

视频数据压缩通常涉及编码视频数据，一般而言，在传输、存储的时候会采用编码的形式，当需要向观看者呈现某种解压缩的版本时，则对其进行解码。

因此，如果无特殊说明，压缩/解压缩（*compression/decompression*）和编码/解码（*encoding/decoding*）通常表示的是一个意思。一些专业视频应用程序可能使用编码形式的未压缩视频，但这种情况相对较少。

编解码器由编码器和解码器组成。视频编码器比视频解码器复杂得多。编码器通常需要做更多的信号处理操作，因此，设计高效的视频编码器至关重要。虽然视频编码标准规定了解码器比特流（*bitstream*）¹ 的语法和语义，但编码器设计大多是开放的。

第2章详细讨论了视频数据压缩，而重要的数据压缩算法和标准则安排在第3章。

降噪

尽管对数字视频进行压缩和处理是必需的，但是这种处理可能引入非预期的效果，这通常被称为失真或噪声。这些失真或噪声经常称为伪影（*visual artifacts*）²。由于噪声会影响用户接收信号的保真度，相当于会影响终端用户可感知的视觉质量，因此视频信号处理需要将噪声最小化。模拟信号和数字信号的视频压缩均需要考虑信号降噪。

在数字视频中，我们会遇到许多不同类型的噪声。这些噪声包括：

- 来自传感器和视频捕获设备的噪声
- 来自压缩过程的噪声
- 来自有损信道上的传输的噪声

第4章将详细讨论各种类型的噪声。

视觉质量

相对于原始信号而言，经过处理后的视频信号可能会存在感知上的差异（数据压缩阶段会产生信息丢失），而视觉质量则用来度量这种感知差异。视觉质量基本上是用户体验质量（*QoE*³）的度量。理想情况下，在编码系统中最小的信号损失才能实现最高的视觉质

量。

确定视觉质量对于分析和制定决策非常重要。视频质量的结果可以用于系统需求的说明文档，对视频服务和应用进行比较和排序，与其它的视频测量之间进行权衡等。

值得注意的是，数字视频因为压缩技术的存在，其伪影与模拟系统中的伪影完全不同。视频中失真的数量和可见性取决于该视频的内容。因此，伪影的测量和评估以及由此产生的视觉质量与传统的模拟质量评估和控制机制有很大不同。

鉴于数字视频伪影的性质，视觉质量评估和可靠排名的最佳方法是主观观察实验。但是，主观方法非常复杂、麻烦、耗时并且昂贵。另外，主观评估并不适合自动化环境。

另一种评估方法是使用简单的误差测量，例如均方误差 (*MSE*) 或峰值信噪比 (*PSNR*)。因为PSNR对输出信号与输入信号进行比较，因此不一定代表感知的视觉质量，因此，严格地说，PSNR仅是信号保真度的度量，而不是视觉质量的度量。但是，PSNR是业界和学术界使用的最流行的视觉质量评估指标。更详细的信息，请参阅第4章。

性能

视频编码性能一般指视频编码的速度：速度越高，性能越好。在此上下文中，性能优化指的是实现快速视频编码。

通常，计算任务的性能取决于处理器的能力，也就是CPU和GPU的频率达到极限值的处理能力。此外，系统的性能优化还需要考虑：内存、辅助高速缓存、磁盘输入和输出 (I/O) 的容量和速度，高速缓存命中率，任务调度等因素。

视频数据和视频编码任务尤其适用于并行计算，并行计算是提高处理速度的好方法。在完成任务的时间内，保持CPU/GPU处于尽可能的忙碌状态，从而保证资源利用率的最大化，也是一种最佳实践。另外，视频编码的性能优化还有很多其它技术，例如编码参数的调整。更详细的性能优化技术将在第5章中详细讨论。

功耗

如今，移动设备已经成为计算，通信，生产力，导航，娱乐和教育的平台。此外，可植入人体，捕获体内图像或视频并渲染到大脑、或使用生物识别密钥安全地传输到外部监视器的设备可能在将来成为现实。如何为这些设备提供电力将成为一个有趣的问题。简而言之，这一领域需要创新的飞跃。但是，就在我们等待电源技术突破的同时，很多外部可穿戴设备已经开始应用。

电源管理和优化是所有这些现有以及新设备和平台的主要关注点，其目标是延长电池的使用时长。然而，无论是本质上还是因为特殊需要，许多应用程序特别耗电，例如即时二进制翻译。

因此，主要问题是功率或等效的功耗。功率优化旨在降低功耗，从而延长电池寿命。高速视频编码和处理对功率优化提出了进一步的挑战。因此，我们需要了解电源管理和优化的注意事项、方法和工具，第6章和第7章将对此进行具体介绍。

1. bitstream: A sequence of bits that forms the representation of coded pictures and associated data forming one or more coded video sequences. Bitstream is a collective term used to refer either to a NAL unit stream or a byte stream. Rec. ITU-T H.264 (2007)/Cor.1 (01/2009), page 5 ↪

2. 伪影是指原本被扫描物体并不存在而在图像上却出现的各种形态的影像。[百度百科](#) ↪

3. Quality of Experience(QoE) is the degree of delight or annoyance of the user of an application or service. It results from the fulfillment of his or her expectations with respect to the utility and / or enjoyment of the application or service in the light of the user's personality and current state. *Qualinet White Paper on Definitions of Quality of Experience (2012). European Network on Quality of Experience in Multimedia Systems and Services (COST Action IC 1003), Version 1.2, March 2013* ↪

视频压缩

数字视频在处理、存储和传输方面的主要缺点是表示视频信号所需的大量数据。对相机电压变化的简单扫描和二进制编码将产生每秒数十亿比特的数据量。如果不对数据进行压缩，将导致昂贵的存储、传输成本。

典型的高清视频（每个图像三个颜色平面，每个平面为 1920×1080 像素的分辨率，每像素8比特，每秒30帧图像）需要大约每秒15亿比特的数据速率。如果在5Mbps带宽的传输信道上传输，则需要300:1的数据压缩比。显然，有损技术可以满足这种压缩比，但是所带来的影响就是后续的视频重建会在视觉质量上有所损失。

但是，视频压缩技术旨在以指定的数据速率下提供最佳的视觉质量。需要根据视频应用的具体要求、可用的信道带宽、存储容量以及视频特性等因素使用不同的数据速率。旧式公共交换电话网络可以使用33.6Kbps的数据速率，典型的高清电视转播系统（HDTV）则需要月20Mbps的数据速率。

广泛应用

很多视频应用以在线方式捕获、处理、传输、显示视频信号。对于这些应用而言，视频信号的实时处理和通信是必须的条件。这些应用程序使用端到端的实时工作流程，例如视频聊天，视频会议，流媒体，直播，远程无线显示，远程医疗诊断和外科手术等。

另外一种应用则以离线方式记录视频。在这些中，视频信号被记录到存储设备，以进行存档、分析或进一步处理。录制视频的主存储介质从模拟录像带转移到数字DV，Betacam磁带，光盘，硬盘或闪存。除存档外，存储的视频还用于电视、电影制作，监视、监控以及安全和调查领域的离线处理和分析。

这些用途将从尽可能快地视频信号处理中收益，因此，需要提高视频压缩和解压缩的处理速度。

需求冲突

视频压缩在现代移动平台上的需求冲突对人们提出了挑战，人群涵盖了从系统架构师到视频应用的终端用户。压缩数据易于处理，但通常在压缩时会导致视觉质量损失。好的视频编码解决方案必须在保证质量损失很小的条件下制作视频。

另外，某些视频应用受益于高速视频编码。这通常意味着更高的计算要求，进而会导致更高功耗。但是，移动设备通常受资源限制，并且电池寿命通常是其最大问题。某些视频应用可能需要牺牲视觉质量以节约能源。

因此，必须平衡这些相互冲突的需求和目的。正如我们将在后面的章节中看到的那样，可以通过调整、平衡视频编码参数以获得期望的结果。

硬件vs软件

视频压缩系统可以使用专用的专用集成电路（ASIC），现场可编程门阵列（FPGA），基于GPU的硬件加速以及纯粹的基于CPU的软件来实现。

ASIC是为特定用途定制的，通常经过专门的优化以执行特定任务，它们不能用于执行规定任务以外的其它任务。虽然ASIC速度快、对错误具有鲁棒性、提供稳定的性能，但它们不灵活、实现单一算法、不可编程、不能易于修改，因此ASIC面临着很快就会过时的问题。现代ASIC通常包含整个32-bit处理器，类似ROM、RAM、EEPROM、Flash的存储单元和其它模块。类似的ASIC常被称为SoC(片上系统)。

FPGA由可编程逻辑模块和可编程互连组成。它们比ASIC灵活得多，相同的FPGA可用于许多不同的应用。典型用途包括从标准零件构建原型。对于较小的设计或较低的产量，FPGA可能比ASIC设计更具成本效益。但是，FPGA通常不会针对性能进行优化，并且性能问题通常不会随着问题规模的扩大而扩展。

基于GPU的硬件加速通常提供中间立场。在这些解决方案中，有一组可编程执行单元，一些性能和功耗优化的固定功能硬件单元。复杂的算法可以利用可编程执行单元的并行处理优势，同时固定功能单元提供快速处理能力。还可以基于某些反馈信息更新参数，从而使固定功能单元具备重复应用的能力。因此，这些解决方案具有灵活性和可扩展性，同时还针对性能和功耗进行了优化。可用参数的调整可确保在给定比特率下的高视觉质量。

因为运行在通用处理器上，基于CPU的软件实现最灵活。因此，相同的方案通常可移植到各种平台。虽然基于软件的实现存在很多性能增强方法，但它们通常无法达到所需的性能水平，并且手动调整各种参数和低级别代码的维护变成了令人生畏的任务。但是，在软件实现中很容易调整各种编码参数。因此，通过调整各种参数，软件实现可以在给定的压缩下提供最佳的视觉质量。

权衡分析

权衡分析是研究不同替代方案的成本/效益，以确定收益超过成本的地方。在视频编码中，权衡分析需要在考虑应用要求、平台约束和视频复杂性的前提下分析调整各种编码参数的效果、性能结果、功率节省和对视觉质量的影响。

视频编码参数的调整会影响性能和视觉质量，因此良好的视频编码解决方案可以平衡性能优化和视觉质量。在第8章中，有一个研究性能和质量之间权衡的案例。

虽然希望实现高速编码，但在不同限制条件的平台上，这有时也是很难实现的一件事。特别是，节能通常是现代计算平台的首要任务。因此，在第8章中，还有一个研究性能和功率优化之间的权衡的典型案例。

基准和标准

目前用于视频编码解决方案排名的基准并没有考虑视频的所有方面。另外，工业标准中也不存在面向权衡分析的方法以及度量的行业基准。标准之间的不同导致用户常常猜测：对于特定的视频应用而言，哪些视频编码参数会产生令人满意的结果。本书通过解释所涉及的概念、方法和指标，帮助读者了解视频编码参数对视频度量的影响。

挑战和机遇

数字视频技术领域的新的挑战和机遇成为推动权衡分析的因素。

- 压缩格式的数字视频的需求正在不断增长。随着更高分辨率，更大色彩位深，更高动态范围以及更高质量的视频的诉求，相关的计算复杂性正在产生滚雪球效应。这些发展对视频编码系统的算法和体系结构提出了挑战，需要对其进行优化和调整，以获得比标准算法和体系结构更高的压缩以及更高的视频质量。
- 目前，多种国际视频编码标准可用于处理各种视频应用。有的标准是从以前的标准发展而来，新的标准使用新的编码特性和工具，从而对旧的标准进行调整，其目的就在于获得更好的压缩效率。
- 尤其是在移动环境中，低功耗计算设备越来越多地成为视频应用的选择平台。但是，它们在系统功能方面仍然存在限制，这种情况会给优化工作带来挑战。尽管如此，仍然需要对诸如保持电池寿命等目标进行权衡。
- 一些视频应用受益于更快的处理速度。有效利用资源，资源专业化和调整视频参数有助于在不影响视觉质量的前提下实现更快的处理速度。

- 在任何给定平台上获得最佳视觉质量需要倍加小心地控制编码参数以及在许多替代方案中做出明智地选择。但是，这样的工具还不存在。
- 调整视频编码参数会影响各种视频度量，此时需要通过这种参数调整来进行需求的折衷。为了能够平衡一个测量中的增益与另一个测量中的损耗，需要了解编码参数、参数之间如何相互影响以及如何影响不同的视频测量。目前，没有统一的方法来分析这些可用的权衡方案是否合适，有必要对该主题进行系统深入的研究。
- 权衡分析可以揭示视频编码解决方案的优点和缺点，并可以对不同的解决方案进行排序。

权衡分析的结果

权衡分析在许多现实视频编码场景和应用中都很有用。此类分析可以显示某个编码特性的价值，因此在特定的应用程序需求以及系统的限制下，可以很容易地决定是否要增加或者删除该编码特性。权衡分析有助于评估视频编码器的优势和劣势，有助于调整参数以实现编码器的优化，有助于基于权衡比较两种编码解决方案，还有助于基于一组标准对多种编码解决方案进行排名。在各种限制和应用需求的情况下，权衡分析也能帮助用户决定是否要启用某些可选的编码特性。而且，用户可以利用权衡分析的结果来做出明智的产品选择。

新型视频应用

计算机的性能已经提高到不再仅用于科学和商业目的水平。我们拥有庞大的计算能力，可实现前所未有的用途和应用。我们正在使用视觉、声音、触碰、手势等彻底改变人机交互的方式。许多新的应用程序已经可用或正在出现在我们的移动设备上，例如感知计算，3-D图像和视频捕获，手势、人脸识别，基于VR的教育和娱乐。

这些应用程序出现在各种设备中，并且可能包含合成视频和自然视频。由于平台能力的快速变化以及这些新兴应用程序的创新性，尤其是从优化的角度来看，很难设置策略来解决此类应用程序的视频组件。但是，通过了解各种视频测量的基本概念，方法和指标，我们将能够将它们应用于未来的应用程序中。

总结

本章讨论了与数字视频，压缩，噪声，质量，性能和功耗相关的一些关键概念。它介绍了各种视频编码注意事项，包括用法，要求以及硬件编码和软件编码在实现上的不同方面。还讨论了权衡分析以及视频领域未来面临的挑战和机遇。本章为后续章节中的讨论奠定了基础。

数字视频压缩技术

数字视频在通信，信息消费，娱乐和教育等领域发挥着重要作用，并对日常生活中的经济、社会、文化有很大影响。在21世纪的第一个十年中，视频作为现代生活的信息媒介的支配地位已经确立——从数字电视到Skype，从DVD到蓝光，从YouTube到Netflix。由于表示数字视频需要大量数据，有必要压缩视频数据以用于实际的传输、通信、存储和流媒体应用。

在本章中，我们首先简要讨论数字网络的限制以及数字视频传输所需的压缩程度。这会为进一步讨论压缩奠定基础。接下来讨论人类视觉系统（HVS）。然后解释数字视频压缩中常用的术语、数据结构和相关概念。

我们会讨论构成压缩方法核心的各种冗余消除技术和熵编码技术。接下来介绍各种压缩技术的概述及其各自的优点和局限性。本章还将简单介绍率失真曲线，该方法作为压缩效率的度量并且可以用来比较两种编码解决方案。最后，会讨论影响和表征压缩算法的各种因素。

网络限制和压缩

在综合业务数字网络（ISDN）出现之前，通常可用的网络为普通老式电话业务（POTS）。POTS基于模拟信号传输来传送语音呼叫。无处不在的电话网络意味着传真、调制解调器等通信服务会倾向于使用这些可用的模拟网络。ISDN可以使用数字网络传输语音信号和视频信号，但宽带ISDN（B-ISDN）标准化的延迟使得基于分组的局域网（如以太网）更加流行。现在，许多网络协议支持在有线或无线技术上传输图像或视频。表2-1列举了不同网络协议的码率和带宽能力。

表2-1各种网络协议支持的带宽

网络协议	带宽
POTS	2.4 kbps (ITU* V.27) 14.4 kbps (V.17) 28.8 kbps (V.34) 33.6 kbps (V.34bis)
DS 0	64kbps
ISDN	64kbps (Basic Rate Interface) 144kbps (Narrow band ISDN)
DS 1	1.5–2Mbps (Primary Rate Interface)
局域网	10Mbps
宽带ISDN	100-200Mbps
千兆以太网	1Gbps

The ITU V-series international standards specify the recommendations for vocabulary and related subjects for radiocommunication.

在20世纪90年代，由于数字视频对数据速率的要求，利用POTS或ISDN传输原始数字视频数据非常困难；即便可以传输，成本也非常昂贵。ITU-R 601格式1的原始数据速率为165 Mbps，这超出了当时网络的能力。为了部分解决数据速率问题，CCITT2的第15专家组（SGXV）将公共图像格式（CIF）定义为与图像速率无关的公共图像参数值。CIF指定了许多图像速率，例如24Hz, 25Hz, 30Hz, 50Hz和60 Hz。在30Hz时，分辨率为352×288的图像需要的数据速率降至约37Mbps，这符合DS 0网络的带宽要求，并且可能让视频数据传输变为现实。

随着计算能力的提高，多年来视频编码和处理操作变得越来越简单。这推动了对更高视频分辨率和数据速率的需求，从而可以获取更高的视觉质量。ITU-R建议书BT.601, BT.709, BT.2020分别定义了越来越高分辨率（从普通视频到高清视频再到超高清视频）

的视频格式。多年来，这些建议也在不断发展。例如，针对高清电视（HDTV）的BT.709就在早期规范（例如BT. 709-3）的第一部分定义了传统模拟高清电视的相关参数。但是，如今这些参数已经不再使用，因此BT. 709又在规范的第2部分增加了基于方形像素（square pixel）CIF的HDTV系统参数。在最新的BT. 709-6的规范中，已经没有模拟视频信号的内容了。

网络的不断发展，为当今行业的需求提供了支持。此外，压缩方法和技术也变得更加精细。

人类视觉系统

人类视觉系统 (*HVS, human visual system*) 是由大脑管理的、人类神经系统的一部分。神经系统和大脑通过大约1000亿个神经细胞进行电化学通讯，这些神经细胞称为神经元¹。神经元会产生脉冲或抑制现有脉冲，从而产生多种现象：从马赫带 (*Mach bands*)²，到视觉频率响应的带通特性，再到眼睛的边缘检测机制。研究及其复杂的神经系统并不是一件难事，因为神经系统中只有两种信号：一种是长距离 (*long distances*) 信号，另一种是短距离 (*short distances*) 信号。无论这些信号所携带的信息是视觉信息、还是听觉信息、亦或是其他信息，对于神经元而言，这些信号都是相同的。

了解HVS的工作方式非常重要，因为：

- 了解HVS可以解释用户对其观看的内容的准确度的感知。
- 了解HVS有助于从诸如亮度 (*luminance*) 和空间频率 (*spatial frequencies*) 的视觉信号的物理量角度了解视觉信号的构成，还有助于制定信号保真度的度量。
- 了解HVS有助于通过各种属性来表示感知到的信息——例如明度 (*brightness*)，颜色，对比度，运动，边缘和形状等。并且还有助于确定HVS对这些属性的敏感性。
- 了解HVS有助于利用HVS的明显缺陷给用户以最真实的视觉感知。一个比较好的例子是彩色电视：HVS对颜色信息的丢失较不敏感，因此通过色度亚采样减小彩色电视的传输带宽则变得较为容易。

HVS的主要组成部分包括：眼睛，通向大脑的视觉通道，称为视觉皮层 (*visual cortex*) 的部分大脑。眼睛捕获光并将其转换为神经系统可以理解的信号，然后这些信号沿着视觉通道传输并处理。

因此，眼睛是视觉信号的传感器。眼睛是一个光学系统，外界的图像投射到位于眼睛后部的视网膜 (*retina*) 上。进入视网膜的光透过多层神经元，最后对光线敏感的感光器 (*light-sensitive photoreceptors*)。感光器是可以将入射光能转换为神经信号的特殊神经元。

视网膜有两种类型的感光器：视杆细胞 (*rods*) 和视锥细胞 (*cones*)。视杆细胞在低光度（暗视觉）下也较为敏感，但是无法区分光的颜色，仅能感知光的亮度，并且多分布在中央凹 (*foveal*)³的边缘。视杆细胞还负责周边视觉 (*peripheral vision*)⁴，并有助于运动检测和形状检测。来自多个视杆细胞的信号会聚到单个神经元，从而使得周边视觉的灵敏度很高，但周边视觉的分辨率却很低。另一方面，视锥细胞具有色彩感知功能，其在强光（明视觉）下最为敏感，并且具有带通响应特性。视锥细胞分为三种类型，分别对长波 (L)，中波 (M) 和短波 (S) 比较敏感。视锥细胞构成颜色感知的基础。视锥细胞大多

集中在视网膜的中央区域，该区域也称为中央凹。视锥细胞负责中央视觉，并且在黑暗中相对较弱。来自每个视锥细胞的信号会经过多个神经元的编码，从而导致其具备较高的分辨率，但是其灵敏度较低。

视网膜上大约有650万个视锥细胞，1亿个视杆细胞，视杆细胞的数量比视锥细胞高出一个数量级。这使得HVS对运动信息和结构信息更为敏感，而对颜色信息的丢失则不敏感。另外，HVS的运动敏感性要高于纹理敏感性。例如，与运动的动物相比，HVS对于伪装不动的动物会更难以感知。最后，HVS的纹理敏感性高于视差 (*disparity*)⁵ 敏感性。例如，不需要太精确的分辨率就可以感知3D深度⁶。

视觉信号通过视神经从视网膜传递到大脑的各个处理中心，因此即使视网膜可以完美地检测到光，这种检测能力也可能无法得到充分利用，或者不为大脑所感知。视觉皮层位于枕叶的距状裂周围，负责高级视觉的相关处理。

除了初级视觉皮层（占HVS的最大部分）外，视觉信号还会达到其他的大约20个皮层区域，但目前对这20多个其他皮层的功能的了解并不多。视觉皮层中的不同细胞具有不同的专长，并且它们对不同的刺激（例如特定的颜色，图案的方向，频率，速度等）具有不同的敏感性。

初级视觉皮层的简单细胞以可预测的方式响应特定的空间频率，方向和相位，并用作定向带通滤波器。复杂细胞是初级视觉皮层中最常见的细胞，也具有方向选择性。与简单细胞不同，复杂细胞可以在其感受范围中的任何位置对正确定向的刺激做出反应。某些复杂细胞是方向选择的，而某些细胞则对大小，角落，曲率或线条的突然中断敏感。

HVS能够适应大范围的光强度或亮度，从而使我们能够区分任何光线水平下相对于周围亮度的亮度变化。物体的实际亮度不依赖于其周围物体的亮度。但是，所感知的物体的亮度却会受到周围环境亮度的影响。因此，具有相同亮度的两个对象在不同的环境中可能具有不同的感知亮度。对比度用来度量这种相对亮度变化。等量的亮度对数的增量被视为等量的对比度的变化。HVS可以检测到1%的对比度变化。

¹. 神经元概述 ↵

². 马赫带指人们在明暗变化的边界，常常在亮区看到一条更亮的光带，而在暗区看到一条更暗的线条。这就是马赫带现象，马赫带不是由于刺激能量的分布，而是由于神经网络对视觉信息进行加工的结果。 ↵

³. 视网膜中感光器密度最大的部分称之为中央凹。 ↵

⁴. 人有两种视觉，中央视觉和周边视觉。中央视觉用来直视事物观察细节，而周边视觉则展现视野中的其他区域，也就是人眼能看到的周边区域。人可以用眼睛的余光观察事物，人们对场景的认知来自周边视觉。有驾车经验的小伙伴应该会有这样的感受，在开车的时候眼睛是看着行驶方向的远处，这里的远处其实是中央视觉，对前方的路况会观察的比较仔细，而周边的环境在视野中其实是模糊的，我们的眼睛是时刻

在采集道路信息的，只要周边环境有异动或危险情况，我们就会通过眼睛把路况传递给大脑，大脑就会指挥脑袋转向有特殊路况的方向，同时眼睛的注意力也会移动到特殊路况上。整个过程其实是周边视觉转变成中央视觉的一个过程。人对场景的认知来自周边视觉，周边视觉决定中央视觉观察的位置。 ↵

⁵. 双目立体视觉融合两只眼睛获得的图像并观察它们之间的差别，使我们可以获得明显的深度感，建立特征间的对应关系，将同一空间物理点在不同图像中的映像点对应起来，这个差别，我们称作视差(Disparity)图像。 ↵

⁶. 那么提到视差图，就有深度图，深度图像也叫距离影像，是指将从图像采集器到场景中各点的距离（深度）值作为像素值的图像。获取方法有：激光雷达深度成像法、计算机立体视觉成像、坐标测量机法、莫尔条纹法、结构光法。 ↵

HVS模型

视觉感知使用了人类大脑中80%以上的神经元，由此可知，视觉感知的过程非常复杂。尽管已经在该领域做了大量的研究工作，但仍未对其全过程得到很好的理解。HVS模型通常用于简化需要可视化/感知的复杂的生物过程。由于HVS由非线性空间频率信道组成，因此可以使用非线性模型对其进行建模。为了简化分析，一种方法是将线性模型作为第一近似模型，该模型会忽略HVS的非线性因素。然后，对该近似模型进行完善和扩展以包括HVS的非线性因素。

第一近似模型

该模型认为HVS是线性的，各向同性的（*isotropic*），并且是时空不变的。线性是指，如果从物体发出的光的强度增加，则HVS的响应应成比例地增加。各向同性意味着方向不变。尽管在实践中，HVS是各向异性的，并且它对旋转的对比光栅（*rotated contrast grating*）的响应取决于光栅的频率以及方向角度，但简化模型忽略了这种非线性因素。时空不变性很难修改，因为HVS并不是均匀的。但是，空间不变性假设在光轴和中央凹区域附近部分成立。时间响应是复杂的，在简单模型中通常不会考虑。

在第一近似模型中，对比度灵敏度为用空间频率的函数关系表示HVS的光学传递函数（OTF, *optical transfer function*）。OTF的大小称为调制传递函数（MTF, *modulation transfer function*），如图2-1所示。

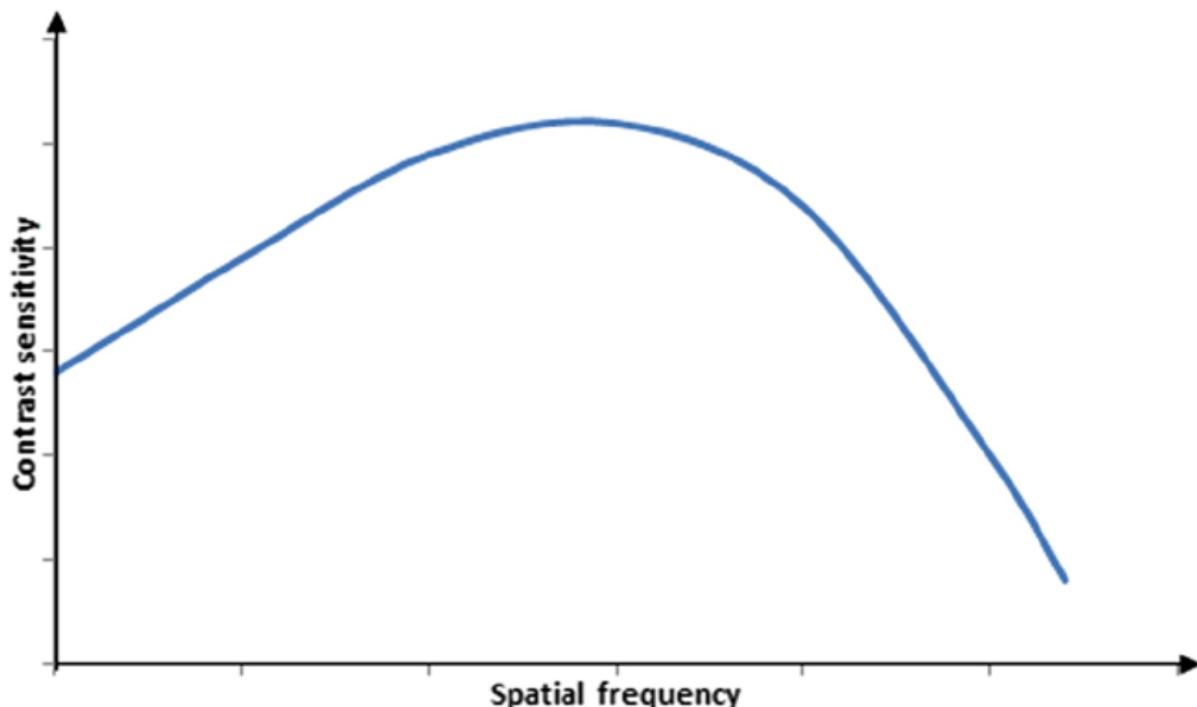


图2-1. 典型的MTF图

图2-1中的曲线表示各种空间频率对应的可见性阈值，从图中可以看出，该曲线呈倒U形，并且其大小随观看距离和视角而变化。曲线的形状表明，HVS对中频最敏感，而对高频则不敏感，从而表现出带通（band-pass）特性。

因此，可以用带通滤波器表示MTF。将低通滤波器和高通滤波器组合在一起，可以更准确地对MTF建模。低通滤波器对应于眼睛的光学器件。即使对于没有视力障碍的人，眼睛的镜片也不是完美的。这种缺陷导致球面像差（*spherical aberration*）¹，球面像差表现为图像在焦平面上的模糊现象。可以将这种模糊建模为二维低通滤波器。瞳孔直径为2mm~9mm。该孔径也可以建模为低通滤波器，其最高截止频率对应于2mm，并且频率随着瞳孔直径的增大而减小。

另一方面，高通滤波器解决了以下现象。给定位置的视网膜后神经信号（*post-retinal neural signal*）可能会受到一些横向定位感光器的抑制。这种抑制被称为横向抑制，并且会导致马赫带效应。对于马赫带效应，在光强度平滑斜坡（*ramp*）的过渡区域附近会出现可见光带。这是从恒定亮度的一个区域到另一个区域的高频变化，并通过滤波器的高通部分进行建模。

包含非线性因素的改进模型

线性模型的优点是，可以利用傅立叶变换技术进行分析，此时只要知道MTF，就可以确定任何输入激励的系统响应。但是，对于HVS而言，线性模型是不够的，线性模型忽略了系统中的、重要的非线性因素。例如，光线刺激感光器会引起感光器细胞膜存电位差，该电位差会调制神经脉冲的频率。目前已经确定，该脉冲频率是光强度（*light intensity*）的对数函数（韦伯-费希纳定律）。这种对数函数可以近似HVS的非线性。但是，一些实验结果表明，信号在高频的空间频率时会出现非线性失真的现象，但是在低频空间频率却不会发生信号的非线性失真。

如上的结果与对数非线性之后再增加线性独立的频率信道的模型不一致。因此，与HVS最一致的模型是一种简单地将低通滤波器置于对数非线性之前的模型，如图2-2所示。该模型还可以扩展用于颜色的空间视觉，其中在低通滤波器和对数函数之间增加了从光谱能量空间到三刺激空间²的转换，并且低通滤波器被三个独立的滤波器（每个频段一个滤波器）取代。

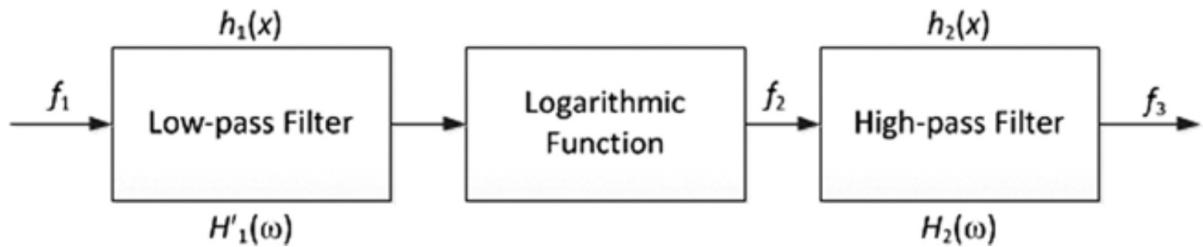


图2-2. 用于HVS的非线性模型

模型的意义

这种——低通、非线性、高通的结构不仅限于空间响应，甚至不限于光谱空间响应。目前还发现，该基本结构对也可用于HVS的时间响应建模。该模型的基本前提是HVS使用低频的空间频率作为特征。作为低通滤波器的结果，快速的离散变化表现为连续变化。这与把离散的时变视频帧显示为连续时间的视频以产生平稳运动的感觉是一致的。

该模型还表明，HVS与可变带宽滤波器类似，并由输入图像的对比度控制。输入图像的对比度增加，系统的带宽则减小。因此，需要限制带宽以最大化信噪比。由于噪声通常包含一些高频的空间频率，因此限制系统传递函数的最小带宽是明智的。但是，在实际的视频信号中，高频细节也非常 important。因此，利用该模型，只能以模糊高频细节为代价来实现噪声滤波，并且需要适当的权衡以获得最佳的系统响应。

模型的应用

在图像识别系统中，关联低频空间频率滤波的图像和视觉初级接受区域的存储原型，其中低通—非线性—高通模型可以充当预处理器。例如，在具有可变对比度信息的复杂场景的识别和分析中，当用户将注意力转移到复杂场景的各个子部分时，基于此模型的自动化系统可以计算该子部分的平均局部对比度并相应地调整过滤器参数。此外，在图像编码和视频编码的情况下，也可以利用该模型作为预处理器，以在仅对相关信息编码之前适当地反映噪声过滤效果。同样，该模型还可以用作带宽减少和高效存储系统的预处理器。

HVS模型的框图如图2-3所示，其中描述了与晶状体，视网膜和视觉皮层有关的部分。

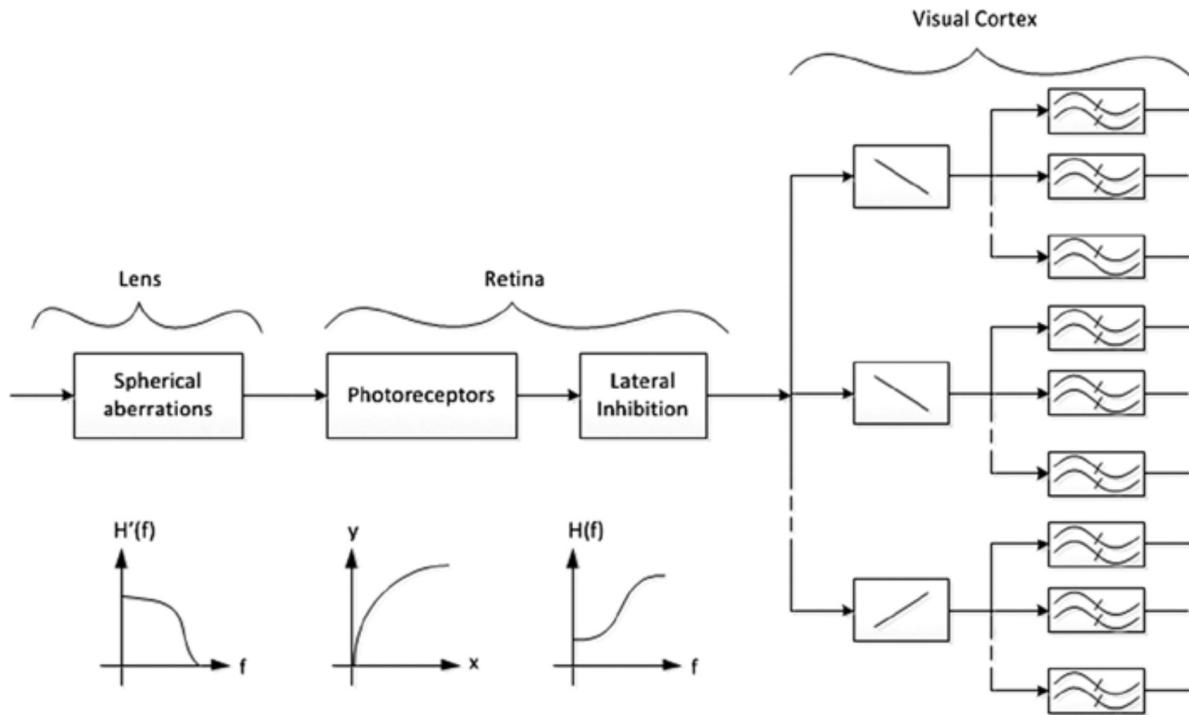


图2-3. HVS的框图

在图2-3中，第一个部分是空间各向同性的低通滤波器，其代表晶状体的球面像差、瞳孔的影响、以及有限数量的感光体对频率的限制。接下来的部分是感光器的非线性特性，用对数曲线表示。在视网膜层级，此非线性变换之后是与横向抑制现象相对应的各向同性的高通滤波器。最后，有一个定向滤波器组，用以代表视觉皮层细胞。框中的线条表示定向滤波器。接下来是另一个由双波表示的滤波器组，用于检测刺激的强度。值得一提的是，由于远离中央凹区域的分辨率会降低，因此整个系统是变化的。

1. 球面像差(spherical aberration)是由于透镜表面是球面而引起的。由光轴上同一物体发出的光线，通过镜头后，在像场空间上不同的点会聚，从而发生了结像位置的移动。 ↪
2. 三刺激值tristimulus values是引起人体视网膜对某种颜色感觉的三种原色的刺激程度之量的表示。 ↪

利用HVS

利用HVS的特性，并调整HVS模型的参数，可以在视觉质量损失和视频数据压缩之间做出权衡。特别地，可以获得如下的收益。

- 通过限制带宽，可以在空间或时间维度上以两倍带宽的频率对视觉信号进行采样，而不会损失视觉质量（满足采样的奈奎斯特理论）。
- 在快速大规模场景变化和物体剧烈运动期间，HVS的灵敏度会降低，从而导致运动掩蔽效应 (*motion masking*)¹。在这种快速变换场景下，强度的时间不连续性会导致可见度阈值的升高。可以利用这个特性实现更有效的压缩，而不会产生明显的伪像。
- 与运动信息相比，纹理信息可以压缩得更多，而视觉质量的损失可以忽略不计。如本章稍后所讨论的，几种有损压缩算法允许对纹理信息进行量化并导致质量损失，同时对运动信息进行无损编码。
- 由于HVS对丢失颜色信息的敏感性较低，因此色度亚采样是一种在不显著影响视觉质量的情况下降低数据速率的可行技术。
- 亮度和对比度信息的压缩可以通过丢弃高频信息来实现。这将损害视觉质量并引入伪像，但是损失量的参数是可控制的。
- HVS对结构变形很敏感。因此，测量这样的变形，尤其是对于诸如图像或视频之类的高度结构化的数据，可以为形变量是否为用户所接受提供评估标准。尽管可接受性是主观的，而不是普遍的，但是结构形变度量可以用作客观评估标准。
- HVS使人们可以将更多注意力放在复杂图像的感兴趣的部分上，而不必在意其他部分。因此，可以对图像的不同部分施加不同的压缩量，从而获得更高的整体压缩率。例如，与背景相比，可以对图像的前景对象使用更多的bits，并且对质量也不会造成实质性影响。

¹. 掩蔽效应 (Masking Effects)，指由于出现多个同一类别（如声音、图像）的刺激，导致被试不能完整接受全部刺激的信息。其中，视觉掩蔽效应包括明度掩蔽效应和模式掩蔽效应，其影响因素主要包括空间域、时间域和色彩域；听觉掩蔽效应则主要包括噪声、人耳、频域、时域和时间掩蔽效应。想象一只在太阳前面飞翔的小鸟，你看到小鸟从左边飞到你和太阳之间，然后小鸟消失，因为太阳光线的亮度太高；当小鸟移出太阳区域，你就又能看到它了。就像在一个安静的环境中，吉他手的手指轻轻滑过琴弦的响声都能听到；但如果同样的响声在一个正在播放摇滚乐曲的环境中，一般人就听不到了。 ↪

压缩技术概述

高清无压缩视频（*HD*）数据流需要大约20亿比特/秒的数据带宽。由于需要大量数据来表示数字视频，视频信号必须易于压缩和解压缩，从而使的视频数据的存储或传输成为现实。数据压缩是指利用数据的统计特性来减少存储或传送数据所需的比特数，这些数据包括数字，文本，音频，语音，图像和视频。幸运的是，由于视频数据在垂直，水平和时间维度存在相关性和冗余性，因此视频数据具有高度可压缩的特性。

变换和预测技术可以有效地利用数据之间可用的相关性，信息编码技术也可以利用视频数据中存在的这些统计特性。数据压缩技术可以是无损的，因此数据解压缩时能够精确的重建原始数据。然而，在视频数据压缩领域，通常会采用有损压缩技术。有损压缩的技术利用了HVS对某些色彩损失和特殊类型的噪声不敏感的特性，从而可以在不影响用户视觉的条件下将不敏感的信息丢弃。

视频的压缩和解压缩过程会利用信息编码理论，并且压缩后的数据是以编码比特流（*coded bit stream*）格式呈现。因此，视频压缩和解压缩通常也被称为视频编码和解码。

数据结构和概念

数字视频信号通常被表征为一种计算机数据形式。视频信号传感器通常输出三种颜色信号——红色，绿色和蓝色，这就是经常说的RGB颜色。RGB三种信号分别转换为数字形式并存储为图像元素（像素）阵列。和模拟视频信号不同，RGB信息不需要模拟视频信号中必须存在的消隐¹或同步²脉冲。水平和垂直分布的像素阵列称为图像（image）或位图（bitmap），在视频领域称其为视频帧。时间序列上帧的集合则构成了完整的视频信号。位图的相关参数有5个：内存中的起始地址，每行的像素数，每帧的行数，间距值，每个像素的位数。在以下讨论中，图像（image）和帧（frame）表述的是一个意思。

信号和采样

A/D(*analog to digital*)转换就是讲模拟信号转换为离散信号，A/D转换主要通过采样（sampling）实现。采样就是按照一定时间间隔 Δt 在模拟信号 $x_a(t)$ 上逐点采取其瞬时值。

对于任意的 $T > 0$ ，如果 $x(n) = x_a(nT)$ ，则 $x(n)$ 就是模拟信号 $x_a(t)$ 的采样信号，其中 T 成为采样周期， $2\pi/T$ 为采样频率或采样率。

图2-4展示了信号 $x_a(t)$ 的空域表示及其对应的采样信号 $x(n)$ 。

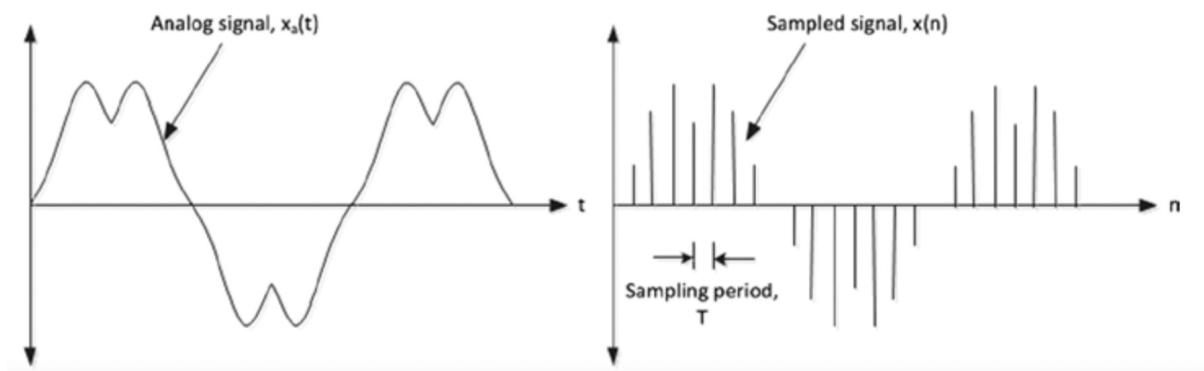


图2-4. 模拟型号的空域表示及其采样

利用傅里叶变换可以将信号的空域表示转换成频域表示，信号的频域表示以 $2\pi/T$ 的均匀间隔描述模拟信号的频率 $X_a(j\Omega)$ ，信号的振幅为 $\frac{1}{T}$ 的倍数。如图2-5所示。

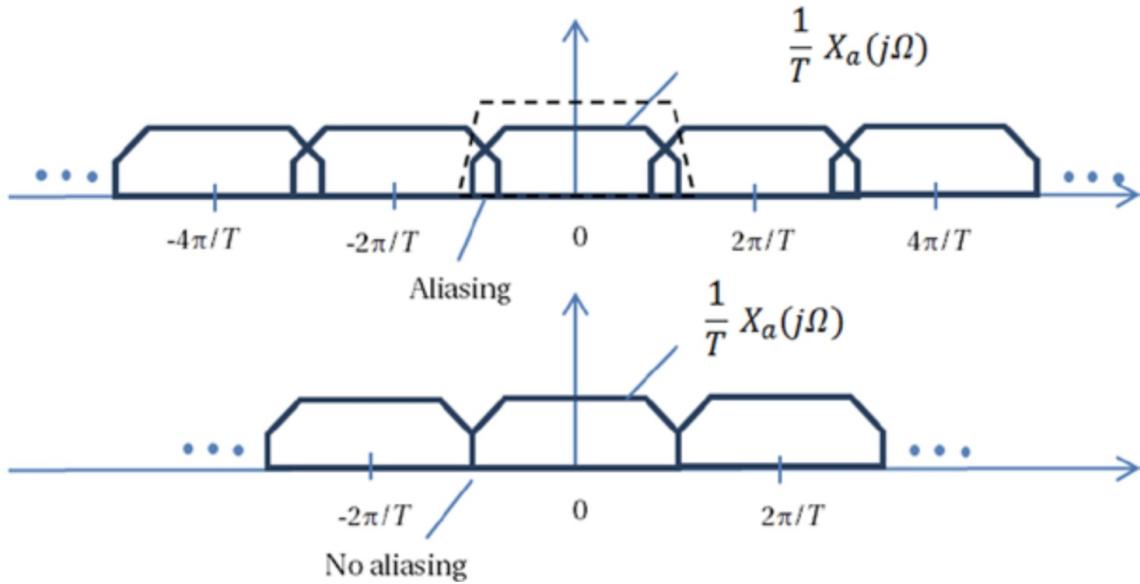


图2-5. 模拟带限信号(analog bandlimited signal)采样的傅里叶变换

对于信号 $X_a(t)$ 的变换版本 $X_a(j\Omega)$ 而言，如果在 $X_a(j\Omega)$ 上存在重叠，则会因为提取的信号中有相邻副本的残余而导致发生混叠³ 现象。如果不存在混叠，就可以利用采样信息 $x(n)$ 对信号 $x_a(t)$ 进行还原。对于频率为 $[-\frac{\pi}{T}, \frac{\pi}{T}]$ 的限带信号 (*band limited signal*)， $\frac{2\pi}{T}$ 或者更高的采样率会避免混叠，此时的采样信号不会丢失有用的信息。1928年，美国电信工程师奈奎斯特首先提出该采样定理，因此称为奈奎斯特采样定理。1933年，苏联工程师科捷利尼科夫首次用公式严格地表述这一定理，因此在苏联文献中称为科捷利尼科夫采样定理。1948年信息论的创始人香农对这一定理加以明确地说明并正式作为定理引用，因此在许多文献中又称为香农采样定理。

奈奎斯特采样定理可以用于一维或多维信号的采样。当然，可以通过使用更少的样本来实现信号的压缩，但是在采样频率小于信号最大频率的两倍的情况下，将会出现令人讨厌的混叠伪影。

常用术语

在数字视频中经常使用的几个术语我们需要了解一下。

幅型比(*aspect ratio*)

几何图形在不同维度上的尺寸的比例，例如，图像的幅型比是指图像的宽度和高度的比例。

DAR(*display aspect ratio*)

计算机显示器的宽高比。DAR的常见值为4:3和16:9，其中16:9的DAR为宽屏显示器的宽高比。

PAR(*pixel aspect ratio*)

像素宽高比。如果把像素想象成一个长方形，PAR即为这个长方形的长与宽的比。当长宽比为1时，这时的像素我们成为方形像素。最常用的PAR为方形像素 (*square pixel*)。实际上还存在其他的PAR，例如12:11，16:11，但是这些很少使用。由于历史原因，PAR在视频行业中的作用非常重要。随着数字显示技术、数字广播技术和数字视频压缩技术的发展，PAR一直是解决视频帧差异的最常用方法。如今，如上的三种技术均主要使用方形像素。

SAR(*storage aspect ratio*)

对图像采集时，横向采集与纵向采集构成的点阵中横向点数与纵向点数的比值。比如VGA图像 $640/480 = 4:3$ ，D-1 PAL图像 $720/576 = 5:4$ 。 $SAR * PAR = DAR$

基色 (*primary colors*)

因为其它颜色可以从基色的线性组合获取，例如RGB颜色模型的基色为红色、绿色和蓝色，CMYK模型的基色为青色、品红色、黄色和黑色。色彩模型中的基色是颜色空间的基本组成部分。色彩空间中的所有颜色的全集称为色域 (*color gamut*)。标准RGB (sRGB) 是计算机最常用的色彩空间。国际电信联盟 (ITU) 已经给出了用于标清 (SD)，高清 (HD) 和超高清 (UHD) 电视的色彩基色的建议。如上的建议分别包含在ITU-R BT.601、ITU-R BT.709和ITU-R BT.2020定义的数字演播室标准中。sRGB使用ITU-R BT.709建议中定义的基色。

亮度 (*Luma*)

图像的明亮 (*brightness*) 程度，通常用于表示图像的黑白信息。虽然色彩学中的照度 (*luminance*) 和视频工程中的亮度 (*luma*) ⁴有一些细微的差别，但是在讨论视频的时候，亮度和照度表示的是一个概念。实际上，照度 (*luminance*) 由正比于光功率的线性RGB的加权组合构成，该加权和不是计算而来，而是单位面积内光能量的表示。而亮度 (*luma*) 是非线性传递函数单独应用到每个线状光的RGB分量后 (RGB分量转换为 $R'G'B'$ ，'用来表示非线性变换)，再经过非线性伽马函数 ($y = x^\gamma, \gamma = 0.45$) 校正 $R'G'B'$ 分量后的加权和构成。需要伽玛函数来补偿感知视觉的特性，以便在感知上将噪声均匀地分布在从黑色到白色的色调范围内，同时使用更多位来表示对人眼更敏感的颜色信

息。没有学过色彩学的电视工程师经常误将亮度 (Y' , luma) 说成照度 (lumance) , 并表示为 Y , 这导致了很多的混乱。虽然在视频中, 我们可以模糊这两者的细微差别, 但是在用的时候务必需要清楚其差别。

色度 (*chroma*)

亮度通常都会和色度 (*chroma*) 一起描述。色度主要用来表示光的色彩信息。由于人的实力对颜色信息的感知能力比亮度感知能力要弱很多, 因此对于色度信息而言, 经常会采用亚采样的方式。色度亚采样会在不丢失敏感的色度信息的前提下以较低分辨率处理和存储色度信息。在分量视频 (*component video*) 中, 传递色彩信息的3个分量都是分别传送的。在传送的过程中, 并不直接发送 $R'G'B'$ 信息, 取而代之的是传送三个衍生的分量——亮度信息 (Y') 和两个色差信息: $B' - Y'$ 和 $R' - Y'$ 。⁵ 在模拟视频中, 分别用 U 和 V 来表示如上的色差信息。而在数字视频中, 我们分别用 C_B 和 C_R 来表示如上的色差信息。实际上, U 和 V 仅仅用在模拟视频中, 但是这中表述也经常会被误用在数字视频中。

色品 (*chromaticity*)

色品用来描述色彩信号的特性, 而色度 (*chroma*) 用来描述信号之间的色差信息。一般而言区分色品和色度的概念是一件很容易的事情。实践中, 我们仅仅利用色品信息 (不考虑照度信息) 来客观度量色彩信息的质量。色品的特征在于色相 (*hue*) 和饱和度 (*saturation*) 。色相用于说明颜色是“红色”, “绿色”等。色相以单色调的色轮中的度数来度量。颜色信号的饱和度则是它与灰色的差异程度。图2-6展示了ITU-R BT.709和ITU-R BT.2020建议书中定义的色品图。从图中可以看出, 因为不同标准的差异性, 利用ITU-R BT.2020色彩表示的视频信号不能直接在BT.709色彩的设备上显示。为了如实的重建真实的色彩, 需要对不同标准的基色进行适当的转换。

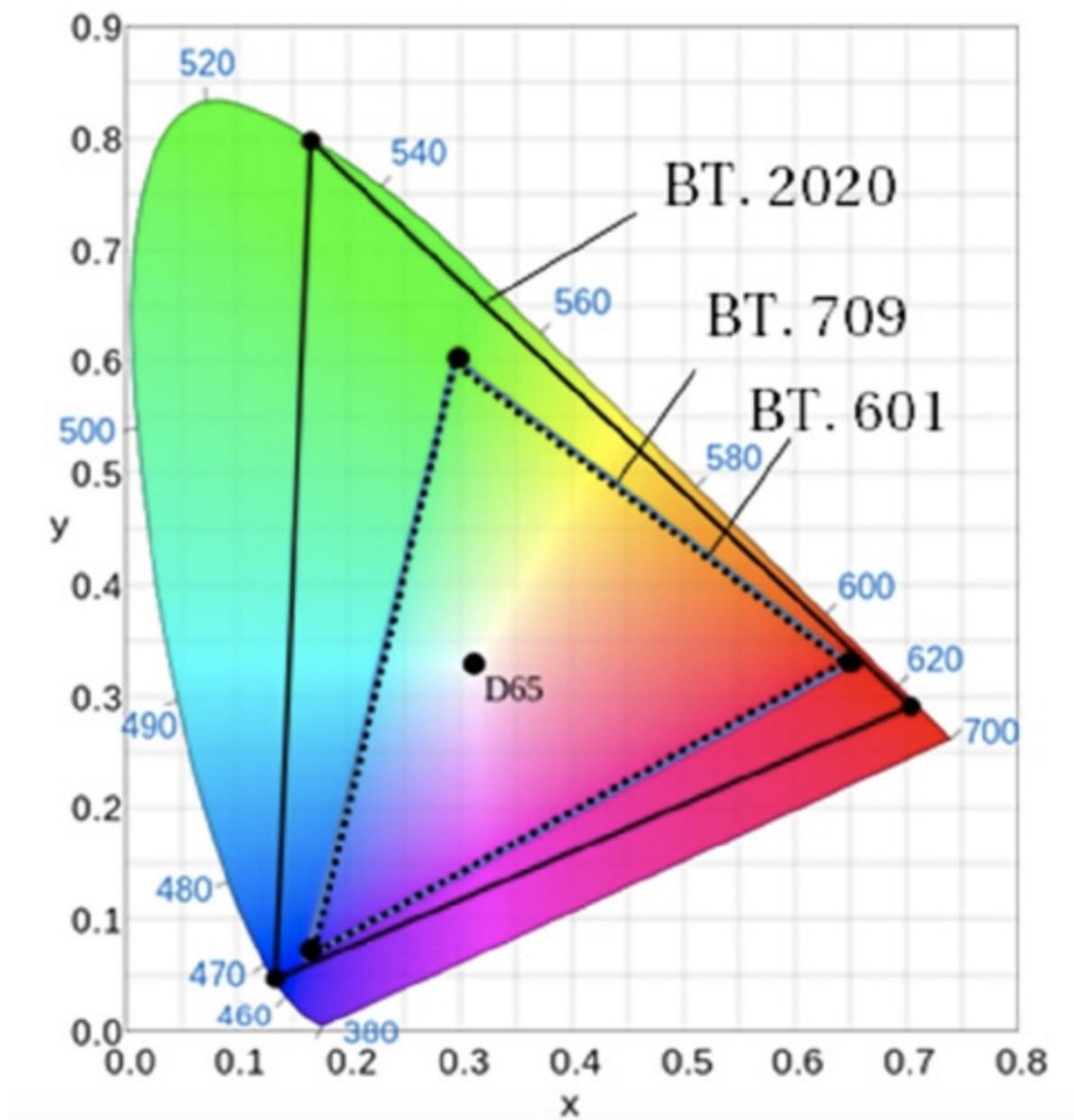


图2-6. ITU-R BT.601, BT.709, BT.2020 定义的色品坐标及其基色坐标, D65代表白色

可以用如下的公式将 $R'G'B'$ 样本转换为 $Y'C_B C_R$ 样本。

$$Y' = K_r R' + K_g G' + K_b B'$$

$$C_B = \frac{B' - Y'}{2(1 - K_b)}$$

$$C_R = \frac{R' - Y'}{2(1 - K_r)}$$

表2-2中列举了ITU-R的不同建议中定义的 K_r, K_g, K_b 。

表2-2. Constants of R'G'B' Coefficients to Form Luma and Chroma Components

标准	K_r	K_g	K_b
ITU-R BT.2020	0.2627	0.6780	0.0593
ITU-R BT.709	0.2126	0.7152	0.0722
ITU-R BT.601	0.2990	0.5870	0.1140

值得注意的是，如上的ITU-R建议还定义了用于表示从黑色到白色之间的可见范围的位深 (*bit depths*)。具体可以参见ITU-R的相关建议：ITU-R BT.2020⁶，ITU-R BT.709⁷，ITU-R BT.601⁸。

¹. 消隐：电视系统中，扫描正程期间传送图像信号，逆程期间不传送图像信号。电子束逆程扫描在荧光屏上出现回扫线，将对正程的图像造成干扰，影响图像的清晰度。因此电视机在行、场扫描逆程期间需要使电子束截止，以消除行、场逆程回扫线，即实现消隐。方法是在电视台让同步机发出消隐信号使接收机显像管在行、场逆程扫描期间关断电子束。 ↪

². 同步：在电视系统中，为了使电视机重现的图像与摄像机拍摄的图像完全一致，要求接收端与发送端的电子束扫描必须同步。所谓同步是指收、发端扫描的频率(快慢)和扫描的相位(起始位置)完全相同。如果收、发端扫描不同步，则重现的图像会变形或不稳定，严重时图像混乱不能正常收看。为保证收、发端行场扫描同步，电视台同步机发出行、场同步信号，使电视接收机正确地重现图像。 ↪

³. 混叠是指取样信号被还原成连续信号时产生彼此交叠而失真的现象。当混叠发生时，原始信号无法从取样信号还原。而混叠可能发生在时域上，称做时间混叠，或是发生在频域上，被称作空间混叠。 ↪

⁴. Digital Video and HD:Algorithms and Interfaces, Chapter 10, Constant luminance. ↪

⁵. Digital Video and HD:Algorithms and Interfaces, Chapter 12, Introduction to luma and chroma, color difference coding, p123. ↪

⁶. <https://www.itu.int/rec/R-REC-BT.2020/en>, Parameter values for ultra-high definition television systems for production and international programme exchange. ↪

⁷. <https://www.itu.int/rec/R-REC-BT.709/en>, Parameter values for the HDTV standards for production and international programme exchange. ↪

⁸. <https://www.itu.int/rec/R-REC-BT.601/en>, Studio encoding parameters of digital television for standard 4:3 and wide screen 16:9 aspect ratios. ↪

色度亚采样

如前所述，HVS对颜色信息的敏感度要弱于亮度信息。利用这一特点，技术人员开发了在不会显著降低视觉质量的情况下减少色度信息的方法。色度亚采样（*chroma subsampling*）是一种常见的降低数据带宽的技术，该技术在模拟视频和数字视频编码方案中都有应用。除了视频，色度亚采样还可以用于流行的单图像（*single image*）编码算法中。这些图像编码算法由ISO和ITU-T的联合图像专家组（*JPEG, Joint Photographic Experts Group*）定义。利用颜色信息的高相关性和HVS的特性，色度亚采样降低了整体的数据带宽。例如，对于矩形图像而言，水平方向上2:1的色度亚采样算法会将整体数据带宽降低1/3，但是在正常观看距离下，数据带框的降低几乎没有导致可察觉的视觉质量损失，从而实现数据带宽的节省。

4:4:4 to 4:2:0

通常，在 $R'G'B'$ 颜色空间中捕获图像，然后使用图2-2的转换公式将图像转换到 $Y'UV$ 颜色空间（对于数字视频而言，会转换到 $Y'C_B C_R$ 空间。简单起见， $Y'UV$ 和 $Y'C_B C_R$ 表示一个概念）。转换之后得到的 $Y'UV$ 图像是全分辨率（*full resolution*）图像，全分辨率图像分别具有4:4:4的 Y' ， U 和 V 分量的采样比。这意味着对于 Y' （亮度）的每四个样本，存在四个 U 样本和四个 V 样本存在于图像中。这些比率通常定义为 4×2 的样本区域，在比率4:a:b中，基于 4×2 样本区域的顶行和底行中的色度样本的数量来确定a和b。因此，4:4:4图像具有完全水平和垂直色度分辨率，4:2:2图像具有一半水平分辨率和完整垂直分辨率，4:2:0图像在水平和垂直方向都有一半的分辨率。4:2:0不同于4:1:1，因为在4:1:1中，一个样本存在于 4×2 区域的每一行中，而在4:2:0中，两个样本存在于顶行，但在底行没有。图2-7展示了常见的色度格式。

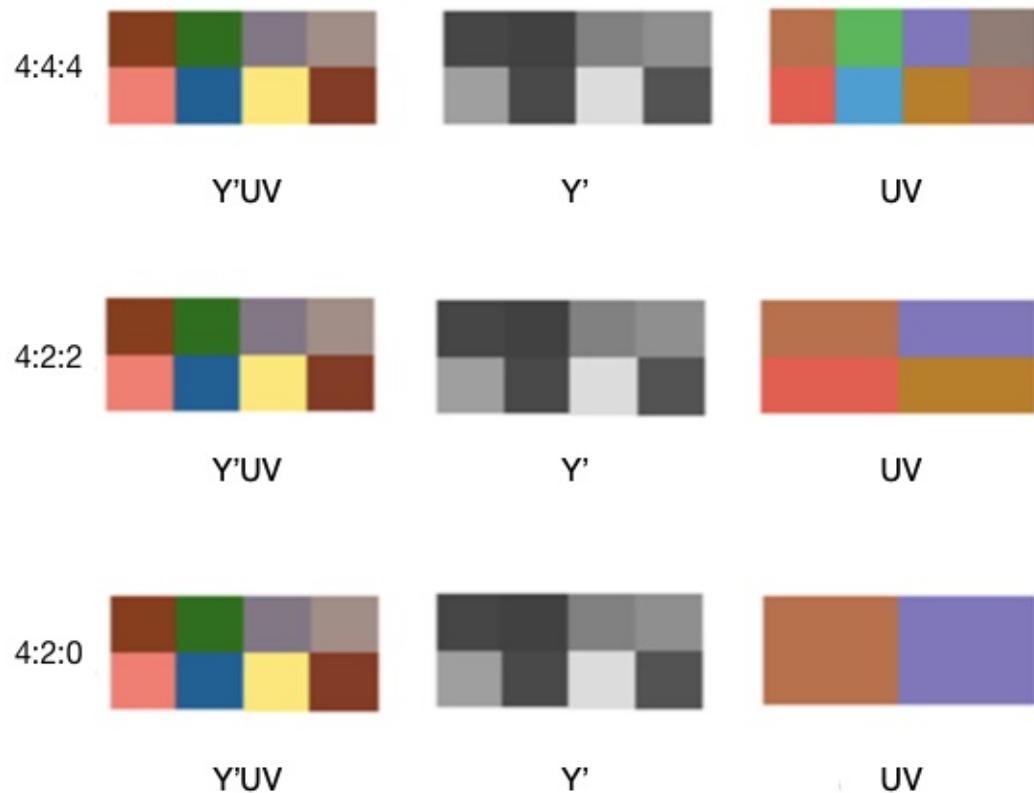


图2-7. 4:a:b采样示例

亚采样也称为下采样 (*downsampling*) 或采样率压缩 (*sampling rate compression*)。如果输入信号以某种方式不受带宽限制，则亚采样会导致混叠现象和信息丢失，从而导致不可逆操作。为了避免混叠，在大多数应用中，在进行亚采样之前通常会使用低通滤波器确保信号具有带宽限制。大多数国际标准都会使用4:2:0的图像格式，因为这种格式利用了颜色分量之间的高度相关性为可接受的感知质量提供了足够的颜色分辨率。通常将相机捕获的 $R'G'B'$ 图像转换为 $Y'UV$ 4:2:0格式以进行压缩和处理。为了将4:4:4图像转换到4:2:0图像，通常采用两步法：首先，通过水平滤波和亚采样将4:4:4图像转换为4:2:2图像；然后，通过垂直滤波和亚采样将得到的图像转换为4:2:0格式。亚采样的过滤器如图2-8所示。

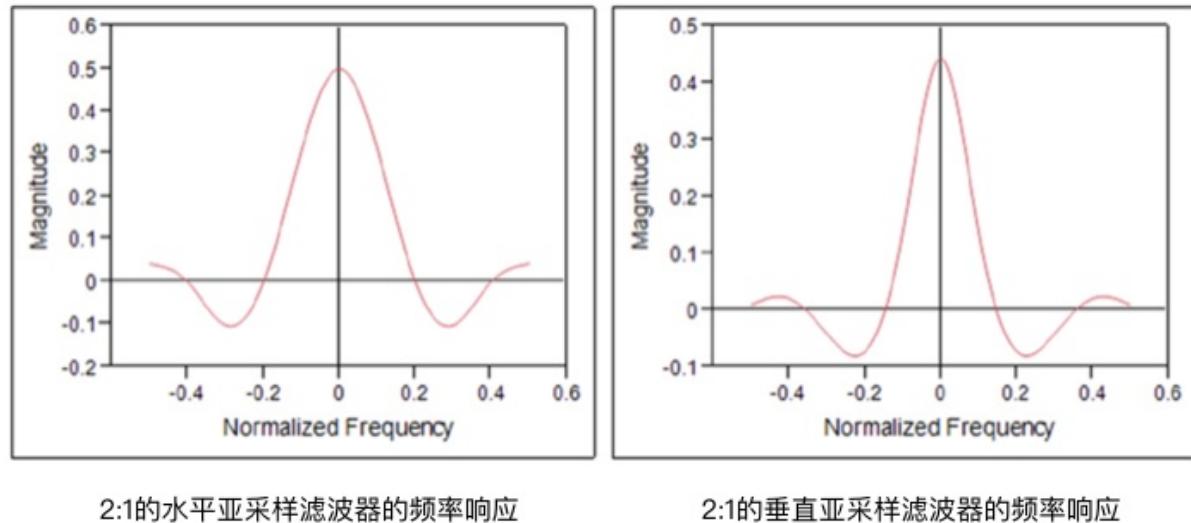


图2-8. 用于2:1亚采样的FIR(Finite Impulse Response)滤波器

表2-5给出了图2-8有限脉冲响应 (FIR) 滤波器的滤波器系数。在该示例中，当水平滤波器具有零相位差时，垂直滤波器具有0.5采样间隔的相移。

表2-5. 有限脉冲响应 (FIR) 滤波器的滤波器系数

水平	0.0430	0.000	-0.1016	0.0000	0.3105	0.5000	0.3105	0.0000
垂直	0.0098	0.0215	-0.0410	-0.0723	0.1367	0.4453	0.1367	-0.0723
归一化	-0.5	-0.4	-0.3	-0.2	-0.1	0	0.1	0.2

降低冗余

数字视频信号在相邻像素和相邻帧之间包含许多相似和相关的信息，这使得通过删除或减少冗余来实现压缩成为一种理想的压缩方案。我们已经讨论了色度亚采样以及这种亚采样并没有带来视觉上的差异的事实。从这个意义上讲，色度的全分辨率是一种冗余信息。通过对色度信息进行亚采样，可以降低数据速率，即实现数据压缩。除此之外，在数字视频信号中还存在其它形式的冗余信息，接下来会对这些冗余信息进行简要的介绍。

空间冗余

数字化操作终结了使用大量的bits来表示图像或视频帧的情况。然而，由于视频中冗余信息的存在，可能会使用更少的bits来表示帧的信息。冗余定义为：1减去表示图像所需的最小bits与表示它的实际bits的比值。冗余的范围大致在46%和74%之间¹。对于具有大量空间细节的图像（例如树叶场景）冗余通常为46%。对于细节信息较少的图像（例如面部图片）冗余通常为74%。压缩技术旨在通过移除或减少可用的冗余信息来减少表示帧所需的bits。

空间冗余（也称为图像内相关，intra-picture correlation）是图像或视频帧内部的相邻像素值之间存在的相关性。视频帧中的相邻像素彼此之间通常非常相似，尤其是当帧被分解成亮度和色度分量时，这种相似性会更加明显。帧可以被划分成较小的像素块以利用这种像素相关性。在像素块内部，像素之间的相关性是很高的。换句话说，在帧的某一小区域内，空间维度的变化率通常较低。这意味着，在视频帧的频域表示中，大部分能量通常集中在低频区域，而高频的边缘信息相对较少。图2-9展示了视频帧中存在的空间冗余。

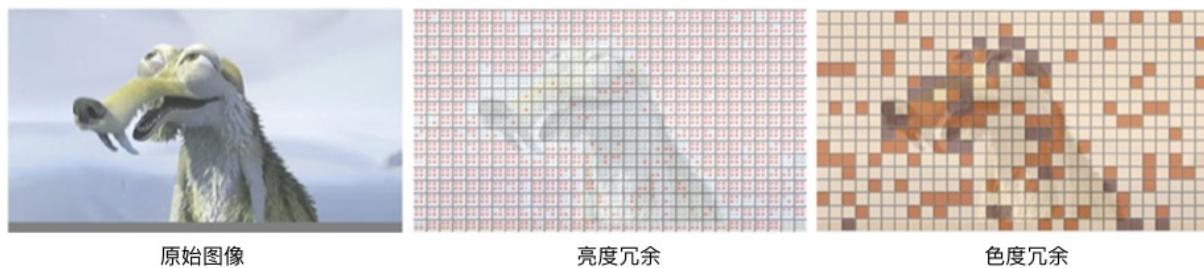


图2-9. 视频帧中的空间冗余

帧中存在的冗余取决于很多参数，例如，采样率，量化级别的数量以及源或传感器噪声都会影响压缩的实现程度。更高的采样率，低量化级别和低噪声意味着更高的像素之间的相关性和更高的可利用的空间冗余度。

时间冗余

时间冗余（也称为图像间相关性，inter-picture correlation）取决于视频中不同图像或帧之间的相关性。数字视频中存在大量的时间冗余。为了人类能够感知到平滑、连续的运动，视频经常以超过15fps的帧速率显示，这也要求相邻帧之间需要具备一定的相似性。图2-10展示了视频的时间冗余。需要注意的是，降低的帧率会带来一定的数据压缩，但这也会导致可感知的视频闪烁。

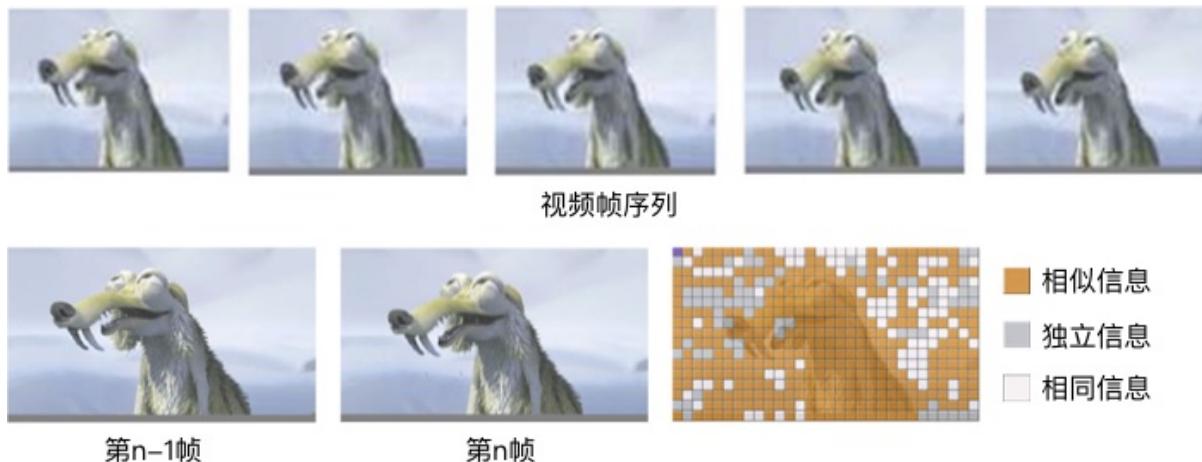


图2-10. 视频的时间冗余信息

因此，帧可以用相邻的参考帧以及和参考帧之间的差异信息表示。在传输系统的接收端会重建独立帧，所以不需要传输先前的参考帧。利用来自已经接受到的参考帧的信息，仅仅使用帧之间的差异信息就足以完美的重建帧。由于时间冗余，帧之间的差异信号通常非常小。只编码差异信号并将其发送到接收端，接收器则可以利用差异信号和已经接受到的参考帧获得视频帧，从而实现非常高的压缩量。图2-11显示了如何利用时间冗余的示例。



图2-11. 利用时间冗余重建帧

通常使用运动补偿（motion-compensated）来表示信号差异，以最小化其中的信息量，从而可以获得更高的压缩。图2-12展示了使用运动补偿来减少信息的示例。



图2-12. 利用运动补偿降低信息量

预测和重建实际上是无损的，预测算法越好，差异信号中剩余的信息越少，这样就能导致更高的压缩。因此，每一代新的国际视频编码标准都试图改进前一代的预测算法。

统计冗余

在信息理论中，描述信源的数据是信息和数据冗余之和。数据压缩的目标是减少或消除不必要的冗余。如之前所述，视频信号的特征在于具有各种类型的冗余，包括空间冗余和时间冗余。除此之外，在视频信号的数字表示中还包含统计冗余，也就是说，通常在传输之前可以消除额外的比特。

例如，二进制图像（传真图像或视频帧）可以被视为0和1的字符串，0表示白色像素，1表示黑色像素。这些字符串，其中相同的bit会现在一系列或连续的数据中，因此可以使用行程代码进行表示。行程代码是指每个1（或0）构成的字符串的地址，后跟该字符串的长度。例如，1110 0000 0000 0000 0011可以使用(1,3) , (0,19) , (1,2) 来编码，代表3个1，19个0，2个1。假设仅存在两个符号0和1，则还可以使用(0,3) , (22,2) 对字符串进行编码，表示位置0和位置22处的1的长度。

行程的长度变化也是可能的。行程编码的思想是：仅编码和存储连续数据元素的数量来代替原始数据元素，从而实现显着的数据压缩。行程编码是一种无损数据压缩技术，可以有效地用于压缩量化系数。对于图像数据而言，尤其是在丢弃图像的高频信息之后，量化系数会包含一系列连续的0和1。根据Shannon的源编码定理，统计冗余可以达到的最大压缩率如下：

$$C = \frac{\text{average bit rate of the original signal (B)}}{\text{average bit rate of the encoded data (H)}}$$

这里, H 表示原信号的熵 (每个符号的比特位数)。尽管通过设计诸如矢量量化或块编码的编码方案可以实现该理论的极限值,但是对于实际视频帧而言,其码本 (codebook) 规模可能非常大,例如:尺寸为 1920×1080 像素,且每像素为24位的视频帧²。因此,国际标准通常使用熵编码方法来任意接近理论极限。

¹. M. Rabbani and P. Jones, Digital Image Compression Techniques. ↪

². A. K. Jain, Fundamentals of Digital Image Processing. ↪

熵编码

考虑使用**B bit**来表示每个像素的一组量化系数。如果量化系数不是均匀分布的，则它们的熵将小于**B bit/像素**。对于一个**M**像素的图像块，假设每个bit可以是两个值之一，则总共有 $L = 2^{MB}$ 个不同的像素块。

对于给定的数据集，特定的块*i*出现的概率为 p_i ，其中*i* = 0, 1, 2, ..., *L* – 1。熵编码是一种无损编码方案，其目标是使用 $-\log_2 p_i$ bits编码该像素块，从而使的其平均码率为该M个像素块的熵： $H = \sum_{i=0}^{L-1} p_i (-\log_2 p_i)$ 。熵编码为**M**个像素的每个块提供了一个可变长度的代码，将较小的代码长度分配给了最可能出现的像素块。在大多数视频编码算法中，量化系数通常是行程编码的，而所得的数据则会再经过一次熵编码以进一步减少统计冗余。

对于给定的块大小，霍夫曼编码是最有效和最受欢迎的可变长度编码方法，可以接近可实现的最大压缩的香农极限。其他著名的熵编码技术是算术编码和Golomb-Rice编码。

当已知近似熵特征时（例如，在输入流中，小值比大值更频繁出现时）Golomb-Rice编码特别有用。通过使用样本到样本的预测，Golomb-Rice编码算法可为每个像素产生0.25 bit的一维差分熵（每个像素的熵值范围为0~8 bits），而无需存储任何码字（code words）。Golomb-Rice编码本质上是最佳游程编码。为了比较，我们主要讨论霍夫曼编码和算术编码。

霍夫曼编码

David Huffman在1952年开发的霍夫曼编码已经成为最流行的无损熵编码算法。霍夫曼编码使用可变长度码表（code table）对源符号进行编码，而该码表是基于源符号中每个可能值的出现概率计算得出。霍夫曼编码以这样的方式表示每个源符号：对出现频率最高的源符号给以最短的编码，频率最低的源符号给以最长的编码。赫夫曼码的码字（各符号的代码）是异前置码字，即任一码字不会是另一码字的前缀，从而使其可唯一解码。

为了了解霍夫曼编码的工作原理，考虑一组分别具有概率{0.47, 0.29, 0.23, 0.01}的源符号 $\{a_0, a_1, a_2, a_3\}$ 。首先，从左到右生成一个二叉树，将两个最不可能的符号合并到一个新的等效符号中，其概率等于两个符号的概率之和。因此，在我们的示例中，我们采用 a_2 和 a_3 合并形成概率为 $0.23 + 0.01 = 0.24$ 的新符号 b_2 。重复该过程，直到只剩下一个符号。

然后从右到左向后遍历二叉树，并将码字分配给不同的分支。在此示例中，将码字 0 (1 bit) 分配给符号 a_0 ，因为这是源字母中最可能出现的符号，而码字 1 留给了 c_1 。 c_1 是其所有分支码字的前缀，从而确保独特的可解码性。在下一个分支级别，将码字 10 (2 bits) 分配给下一个可能的符号 a_1 ，而将 11 分配给 b_2 并作为其分支的前缀。因此， a_2 和 a_3 分别接收码字 110 和 111 (3 bits)。图2-13显示了该过程以及最终的霍夫曼代码。

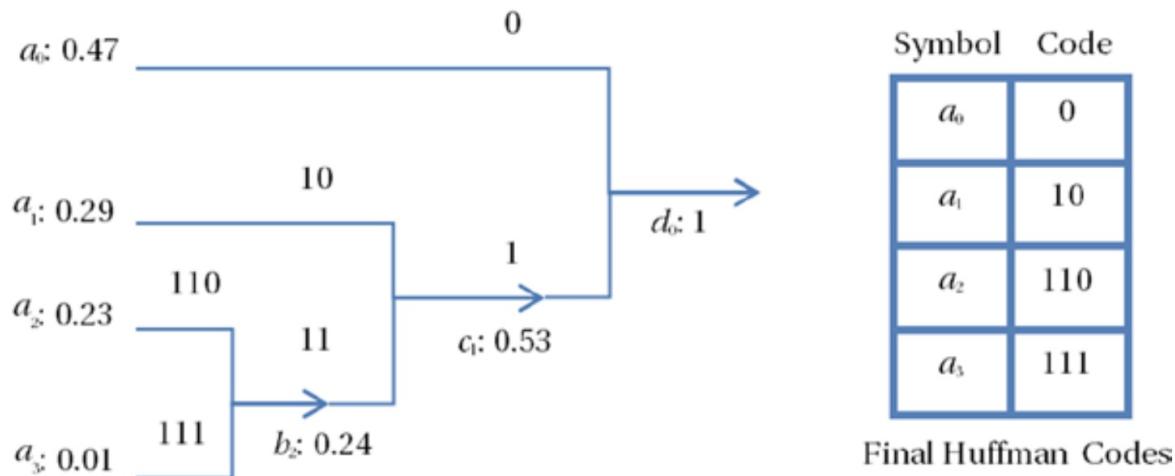


图2-13. 霍夫曼编码的例子

尽管可以使用两位码字来编码这四个符号 (00, 01, 10, 11)，但由于这4个符号的概率分布并不均匀，并且每个符号的熵仅为1.584 bits，因此还有可提升的空间。如果使用霍夫曼码字，则每个符号需要1.77 bits。尽管使用霍夫曼码字，每个符号的码字长度仍然比理论的最小值 (1.584 bits) 还多0.186 bits，但与固定长度码字相比，霍夫曼编码已经提供了约12%的压缩率。通常，最大概率符号和最小概率符号之间的概率差异越大，霍夫曼编码提供的编码增益越大。当每个输入符号的概率是2的幂的倒数时，霍夫曼编码是最佳的。

算术编码

算术编码是无损熵编码技术。算术编码与霍夫曼编码的不同之处在于，算术编码将整个输入消息编码为0.0~1.0之间的小数，而不是将输入分成单个符号并用码字单独编码每个符号。当带编码的符号的概率分布未知，不是独立且分布不均等时，算术编码可以提供更好的压缩能力。虽然算术编码提供了最佳的压缩效果，但是却非常复杂，实际中需要专用的硬件引擎来加速执行速度。

为了描述算术编码的工作方式，让我们考虑三个事件（例如，文本中的三个单词）的示例：第一个事件是 a_1 或 b_1 ，第二个事件是 a_2 或 b_2 ，第三个事件是 a_3 或 b_3 。为简单起见，尽管该算法也适用于多事件，但每个步骤仅在两个事件之间进行选择。假设输入文本为 $b_1a_2b_3$ ，并且每个符号的概率如图2-14所示。

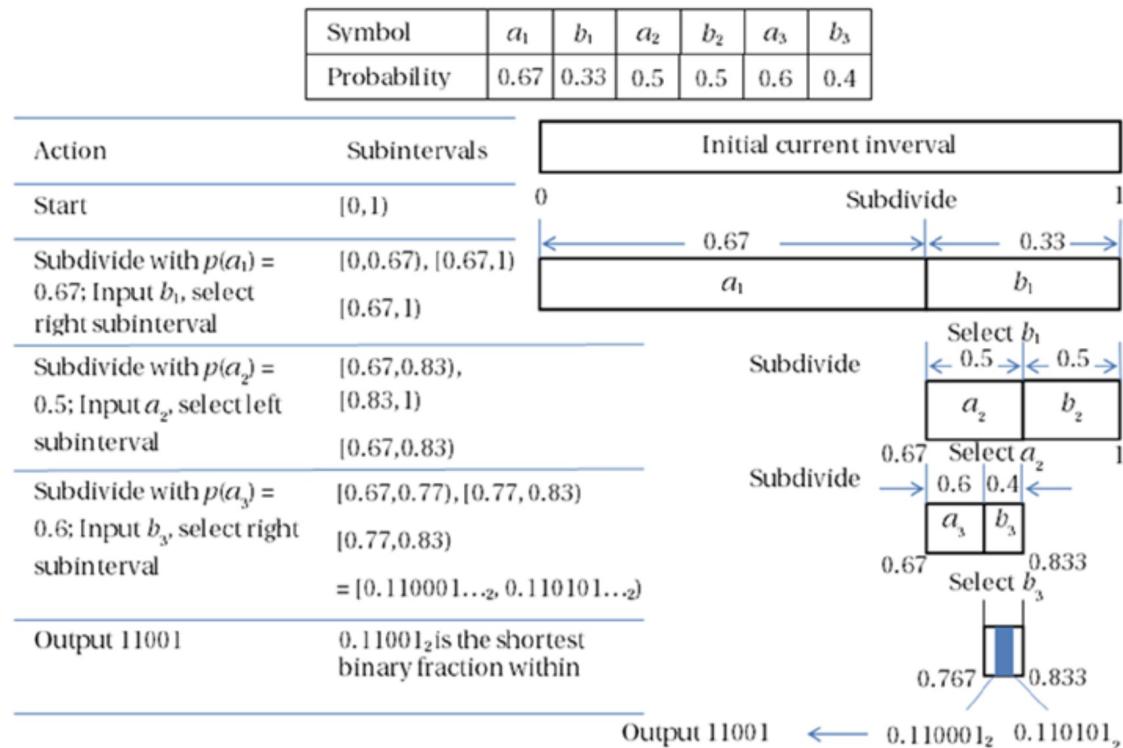


图2-14. 算术编码的例子

压缩技术: 成本-收益分析

在本节中，我们将讨论几种常用的视频压缩技术，并分析这些压缩算法的优缺点。

变换编码技术(TC)

如前所述，块中的像素彼此相似并且具有空间冗余。但是，像素数据块没有太多的统计冗余，并且不容易适用于可变长度编码。变换域中的去相关表示具有更多的统计冗余，并且更适合使用可变长度代码进行压缩。

在变换编码中，通常将大小为 $N \times M$ 的视频帧细分为较小的 $n \times n$ 块，并将可逆线性变换应用于这些块。该变换通常具有一组完整的正交离散基函数，而其目标是去相关原始信号并在一组变换系数之间重新分配信号能量。因此，可以在编码剩余的几个系数之前通过量化过程丢弃许多具有低信号能量的系数。变换编码的框图如图2-15所示。

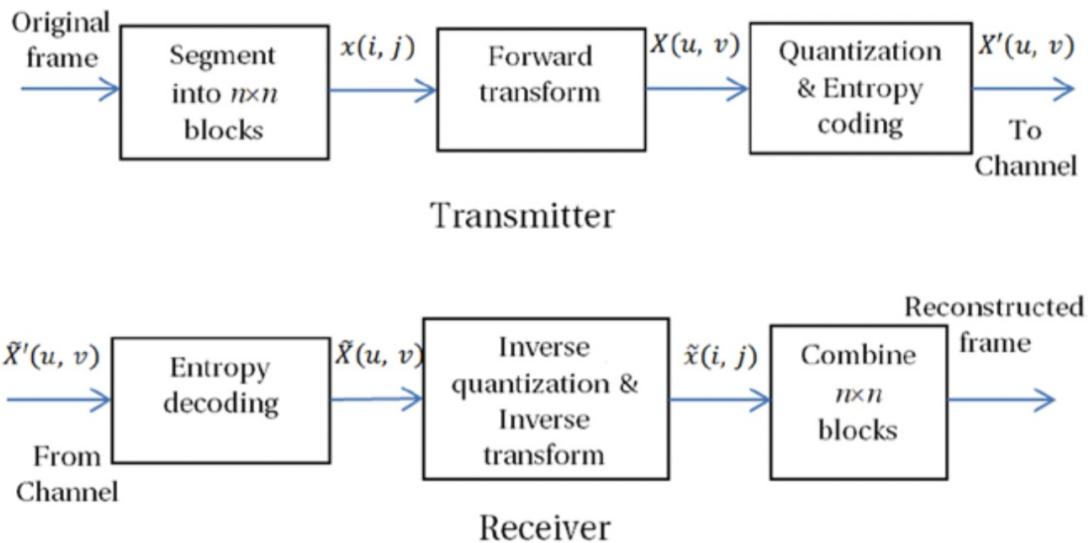


图2-15. 传输系统中变换编码的框图

离散余弦变换

离散余弦变换 (DCT) 根据具有不同频率和幅度的余弦函数之和表示离散数据点的有限序列。DCT是一种线性，可逆，无损的变换，可以非常有效地将像素块中存在的冗余去相关。实际上，DCT是可用于此目的的最高效、实用的变换，它接近理论上最理想的 Karhunen-Loeve变换 (KLT)，因为几乎不需要余弦函数来逼近典型信号。因此，DCT被广泛用于视频和音频压缩技术。DCT有四种表示形式，其中DCT-II¹⁷是最常见的形式：

$$X(k) = \sum_{n=0}^{N-1} x(n) \cos\left[\frac{\pi}{N} \left(n + \frac{1}{2}\right) k\right] \quad k = 0, \dots, N - 1$$

这里， X_k 是变换后的DCT系数， x_n 是输入信号。该一维DCT可以一个接一个地在垂直和水平方向上单独使用，以获得二维DCT。对于图像和视频压缩，DCT通常在 8×8 像素块上执行。 8×8 二维DCT可以表示为：

$$X(u, v) = \frac{1}{4} \alpha(u) \alpha(v) \sum_{m=0}^7 \sum_{n=0}^7 x(m, n) \cos\left[\frac{(2m+1)u\pi}{16}\right] \cos\left[\frac{(2n+1)v\pi}{16}\right]$$

这里， u 和 v 是水平和垂直空间频率， $0 \leq u, v < 8$ ； $\alpha(k)$ 是归一化因子，当 $k = 0$ 时等于 $1/2$ ，否则等于 1 。 $x(m, n)$ 是空间位置的像素值 (m, n) ； $X(u, v)$ 是频率坐标 (u, v) 的DCT系数。DCT将 8×8 的输入值块转换为64个二维DCT基本函数的线性组合，这些线性函数以64种不同的模式表示，如图2-16所示。

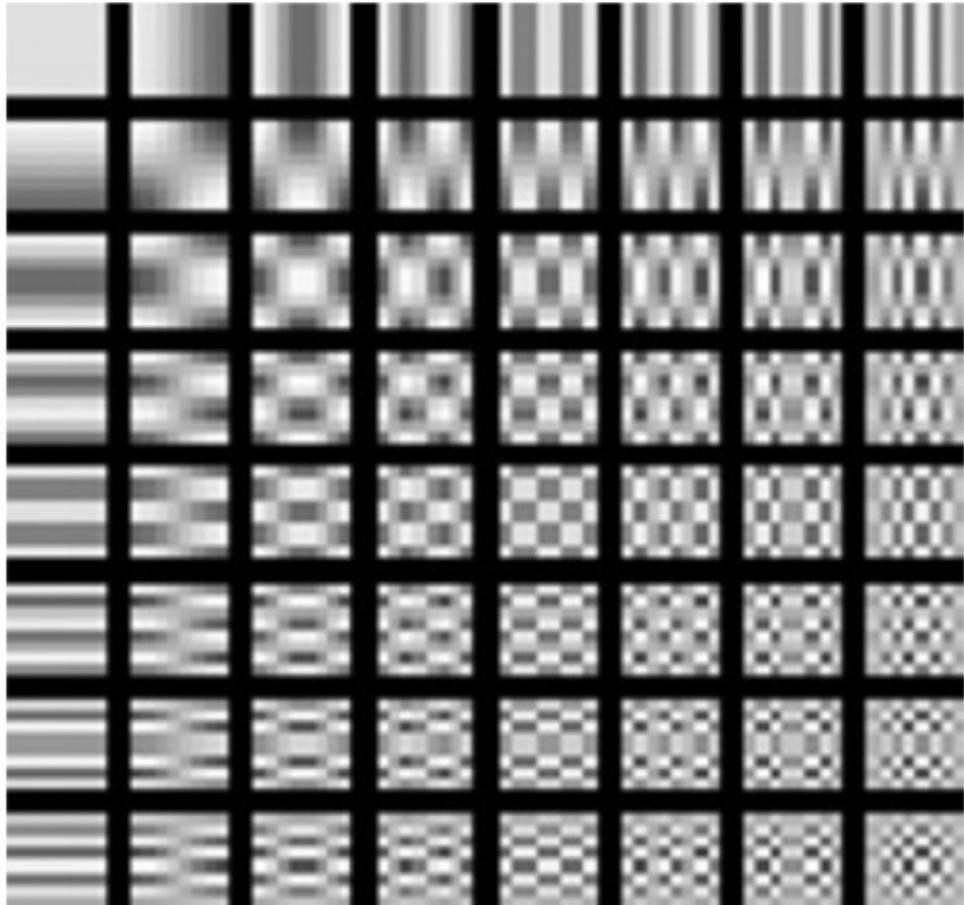
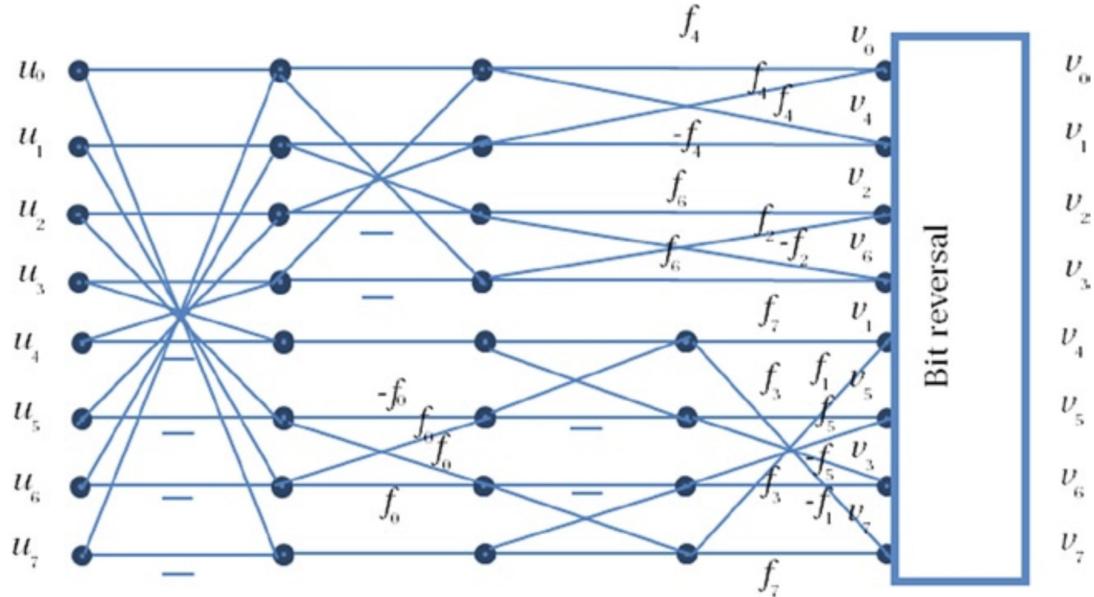


图2-16. 8×8 输入块的64个二维DCT基函数

尽管该变换是无损的，但是由于计算系统的算术精度的限制，它可能会引入不准确性，从而在进行逆运算时可能无法获得相同的准确输入。为了处理这种不准确性，标准委员会经常采取诸如定义IEEE标准1180之类的措施，本章稍后将对此进行描述。图2-17中显示

了八点DCT（和逆DCT）的信号流程图，代表一维DCT，其中输入数据集为 (u_0, \dots, u_7) ，输出数据集为 (v_0, \dots, v_7) 和 (f_0, \dots, f_7) 是中间结果的基于余弦函数的乘法因子。文献中有许多快速的DCT算法和实现，因为几乎所有国际标准都采用DCT作为减少空间冗余的选择。



Here, $f_0 = \cos(\pi/4)$, and $f_k = \frac{1}{2} \cos(k\pi/16)$, $k = 1, \dots, 7$.

图2-16. 八点DCT的信号流程图，从左到右（以及IDCT从右到左）

可以使用硬件或软件轻松实现DCT。经过优化的软件实现可以利用多媒体指令集（如MMX或SSE）中可用的单指令多数据（SIMD）并行构造。此外，在基于Intel集成图形处理器的编解码器解决方案中还有专用的硬件引擎。

图2-18给出了一个像素块及其DCT转换系数的示例^{[18](#)}。

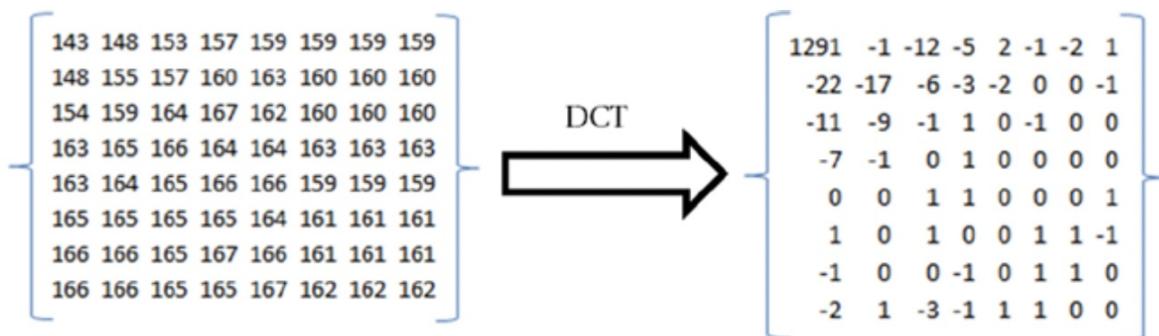


图2-18. 一个像素块及其DCT转换版本

量化

由于DCT具有典型的无损特性，因此它本身并不存在压缩。它仅对输入数据进行解相关。但是，DCT通常后面是量化过程，该过程会利用视频帧中存在的空间冗余来截断变换后的数据块的高频信息。量化器是一种阶梯函数，可将具有许多值的连续输入信号或离散输入信号映射为较小的有限数量的输出电平。如果 x 是一个实数标量随机变量，其中

$p(x)$ 是其概率密度函数，量化器将 x 映射到离散变量 $\hat{x} \in \{r_i, i = 0, 1, \dots, N - 1\}$ ，其中每个水平 r_i 被称为 a 重建水平。映射到特定 \hat{x} 的 x 的值由一组决策级别 $\{d_i, i = 0, \dots, N - 1\}$ 。根据量化规则，如果 x 位于间隔 $(d_i, d_{i+1}]$ 中，则将其映射（即量化为 r_i ），该 r_i 也位于相同的间隔中。给定 $p(x)$ 和给定的优化准则。图2-19显示了一个示例八阶非线性量化器。在此示例中， $(-255, 16]$ 之间的 x 的任何值都映射到-20，类似地 $(-16, -8]$ 之间的 x 的任何值都映射到-11， $(-8, -4]$ 之间的任何 x 的值都被映射到-6，依此类推。对于 $(-255, 255)$ 之间的任何输入值，此量化过程仅导致八个非线性重构级别。量化之后，一个 8×8 变换数据块通常从64个系数减少到5个到10个系数，通常实现大约6到12倍的数据压缩。但是，请注意，量化是一种有损过程，其中在执行逆运算后无法重新获得被丢弃的高频信息。尽管高频信息通常可以忽略不计，但并非总是如此。因此，变换和量化过程通常会引入质量损失，这通常称为量化噪声。所有国际标准都详细定义了转换和量化过程，并要求符合定义的过程。在通常是对像素块执行量化的二维信号（例如图像或视频帧）的情况下，由于块是独立变换和量化的，因此在块边界处会产生轮廓效果。结果，块边界变得可见。这通常称为阻塞或阻塞伪影。尽管量化级别越粗糙，数据压缩程度越高，但值得一提的是，信号的量化级别越粗糙，将引入更多的阻塞伪像。

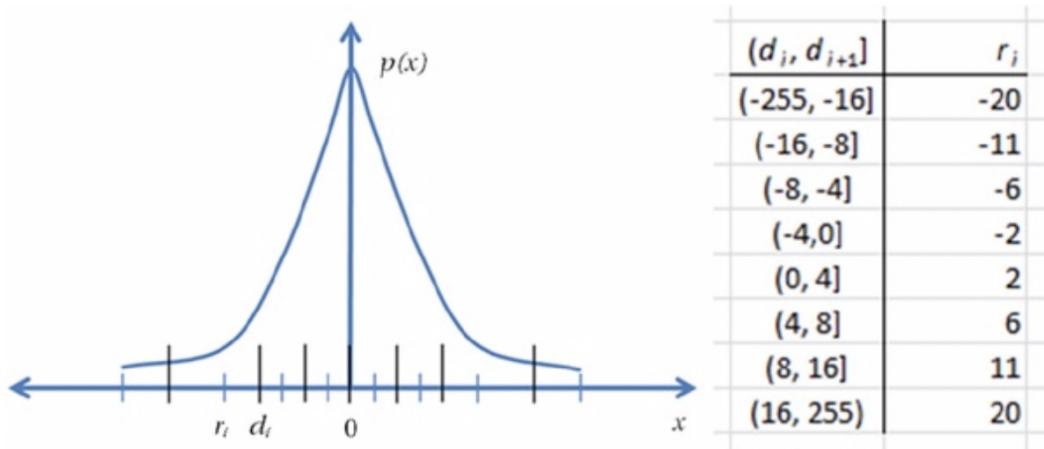


图2-19. 八阶非线性量化器

Walsh-Hadamard和其他的变换

Walsh-Hadamard变换（WHT）是线性，正交和对称变换，通常对 $2m$ 实数进行运算。它仅具有适度的去相关能力，但是由于其简单性，它是一种流行的转换。WHT基函数由+1或-1的值组成，并且可以从正交Hadamard矩阵的行中获取。可以从相同类型的最小 2×2 矩阵（即大小为2的离散傅里叶变换（DFT））递归构造正交哈达玛矩阵，如下所示：

$$H_2 = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}, \text{ and } H_{2n} = \begin{bmatrix} H_n & H_n \\ H_n & -H_n \end{bmatrix}$$

Hadamard变换可使用快速算法进行计算，使其适用于许多应用，包括数据压缩，信号处理和数据加密算法。在视频压缩算法中，通常以绝对变换差之和（SATD）的形式使用它，这是一种视频质量度量，用于确定一个像素块是否与另一个像素块匹配。在各种视频压缩方案中还有其他不常用的变换。其中值得注意的是离散小波变换（DWT），其最简单的形式称为Haar变换（HT）。HT是基于Haar矩阵的可逆线性变换。可以将其视为一种采样过程，其中Haar矩阵的行充当越来越精细的采样。它提供了一种简单的方法与非局部WHT相对，分析信号的局部方面非常有效，并且在算法（例如子带编码）中非常有效。 4×4 Haar矩阵的一个示例是：

$$H_4 = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 1 & -1 & -1 \\ 1 & -1 & 0 & 0 \\ 0 & 0 & 1 & -1 \end{bmatrix}$$

¹⁷: K. R. Rao and P. Yip, *Discrete Cosine Transform: Algorithms, Advantages, Applications* (New York: Academic Press, 1990).

¹⁸: S. Akramullah, I. Ahmad, and M. Liou, "Optimization of H.263 Video Encoding Using a Single Processor Computer: Performance Tradeoffs and Benchmarking," *IEEE Transactions on Circuits and Systems for Video Technology* 11, no. 8 (2001): 901–15.

预测编码技术

预测是一种重要的编码技术。在传输系统的接收端，如果解码器可以用某种方式预测信号，即使预测信息有错误（一般称之为残差，*residuals*），解码器就可以重建输入信号的近似版本。但是，如果残差是已知的，或者将残差传输到解码器，此时解码器就可以更完美的重建原始信号。预测编码技术利用了残差信息这一原理。预测编码可以是有损的，也可以是无损的。接下会介绍几种预测技术。

无损预测编码

通过利用空间冗余，像素可以通过相邻的像素预测。因为相邻像素之间的差异很小，编码差异而不是编码实际像素是一种更有效的方式。这种方法称为差分脉冲编码调制（DPCM）技术。在DPCM中，存储了最可能的估计，并且计算实际像素 x 与其最可能的预测 x' 之间的差异。该差异 $e = x - x'$ 称为误差信号，并且通常使用可变长度编码进行熵编码。

为了获得更好的估计，可以用之前的少量像素的线性组合来预测。由于解码器已经完成对先前的像素的解码，因此可以使用它们来预测当前像素以获得 x' ，并且在接收到误差信号 e 时，解码器可以执行 $e + x'$ 以获得真实像素值。图2-20显示了这个概念。

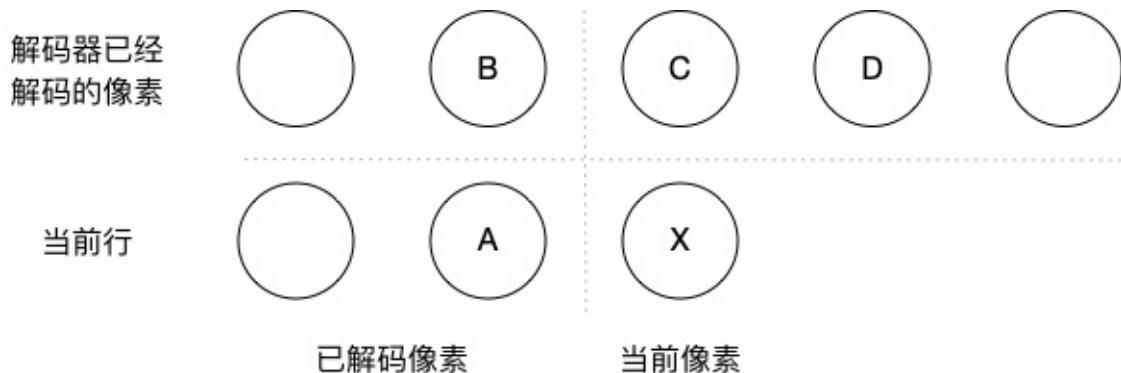


图2-20. DPCM预测两行像素的示例

在图2-20的例子中，可以根据先前解码的像素的线性组合来预测当前像素 X 。例如，根据相关性， $X = 0.75A - 0.25B + 0.5C$ 。一般而言，预测的图像和原始图像之间会存在有一定的误差，但是与原始图像相比，预测后的图像有更小的方差以及更少的空间相关性。因此，在DPCM中，可以使用诸如霍夫曼编码或算术编码的可变长度编码对预测后的图像进行编码。由此，这种方法产生的压缩是无损压缩。

很多应用会综合利用无损压缩和有损的预测算法实现更短的传输时间，例如医学成像等。然而，这些应用可能只能容忍很小的质量下降，并且期望具备高质量的图像重建。为了实现这一目的，需要首先使用有效的有损压缩算法来构造图像的低带宽版本；然后，计算有损版本和原始图像之间的差异来生成残差（*residual*）图像；最后对残差图像进行无损编码。

有损预测编码

在有损编码中，允许部分视觉质量损失以适应更低的比特率，更多的质量降级可以实现更大的数据压缩。图2-21显示了有损编码的一般概念，其中将原始图像分解/变换到频域空间，并对频域信息进行量化处理，然后利用熵编码技术来编码剩余信息。



图2-21. 有损编码方案

分解和变换会降低信号的动态范围并且还使信号失去相关性，从而可以产生一种可以高效编码的数据格式。分解和变换通常是可逆的、无损的。然而，在接下来的量化过程，会引入信息损失并因此引入质量降级。但是量化处理也同时实现了数据压缩。接下来的熵编码也是无损处理，但是却通过统计冗余来提供一些压缩。

有损DPCM

如前所述的在结合无损DPCM编码的预测编码中，可以基于参考形成预测或估计，然后计算预测信号和原始信号的误差并对误差进行编码。然而，DPCM方案也可以用于有损编码并产生有损预测编码。

可以采用原始信号作为预测的参考帧，然而，传输信道的接收端的解码器将仅能基于到目前为止接收到的数据重现部分信号。需要注意的是，接收到的信号是从信号的量化版本重建，因此会包含量化噪声。故而，在重建信号和原始信号之间会存在差异。

为了确保传输信道收、发两端具有相同的预测，编码器还需要基于重建值形成其预测。如图2-22所示，为实现这一点，需要将量化器包含在预测循环中，这实际上相当于将解码器和编码结构合并在了一起。

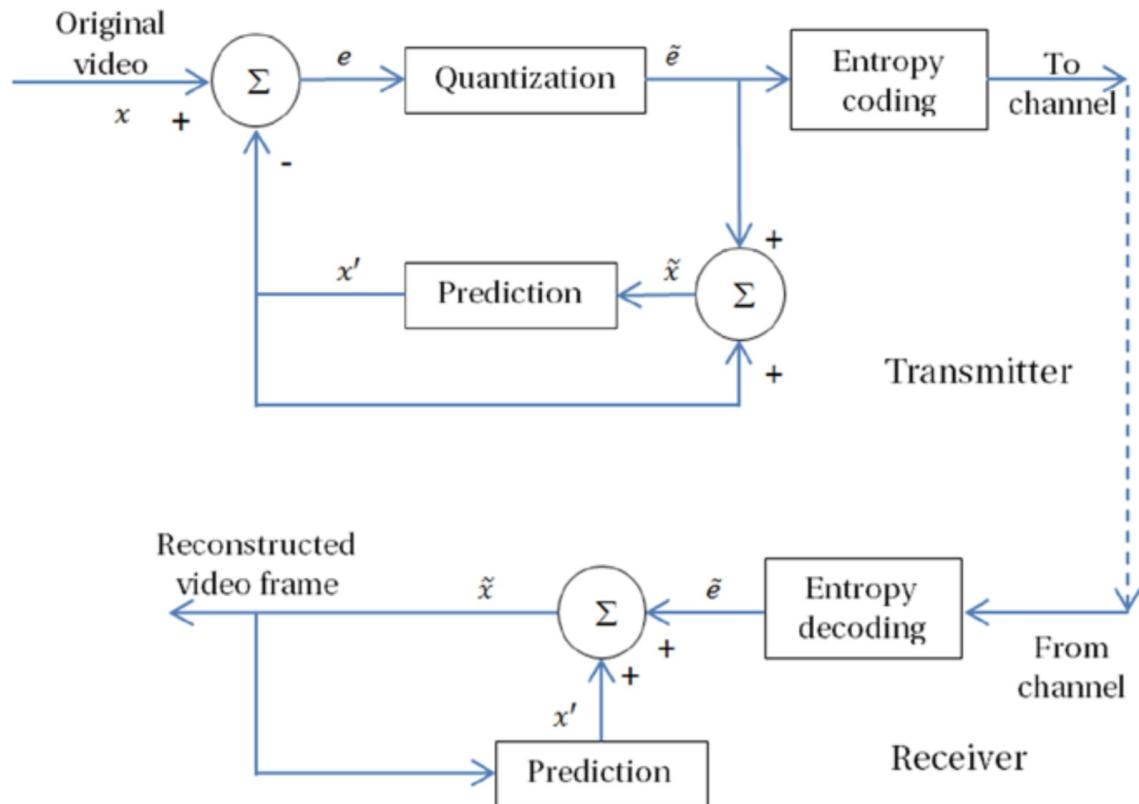


图2-22. 有损DPCM流程图

时域预测

除了利用相邻像素之间的空间冗余预测外，还可以利用时间冗余从相邻帧形成预测。除了物体在帧之间的微小移动外，相邻帧是相似的，因此可以捕获帧的差异信号并且可用通过运动来补偿帧的残差信息。

将帧分为几块时，每个块可能会移动到不同的位置在下一帧中。因此，通常为每个块定义运动矢量，以指示水平和垂直维度上的运动量。运动矢量是整数，并表示为 $mv(x, y)$ ；然而，可以将来自子采样残差帧的运动矢量与来自残差帧的原始分辨率的运动矢量组合，以使得将子采样运动矢量表示为分数。图2-23说明了此概念，其中最终运动矢量在当前帧中与参考帧中的原始位置 $(0, 0)$ 水平相距12.5像素；使用运动矢量的半像素（或半像素）精度。

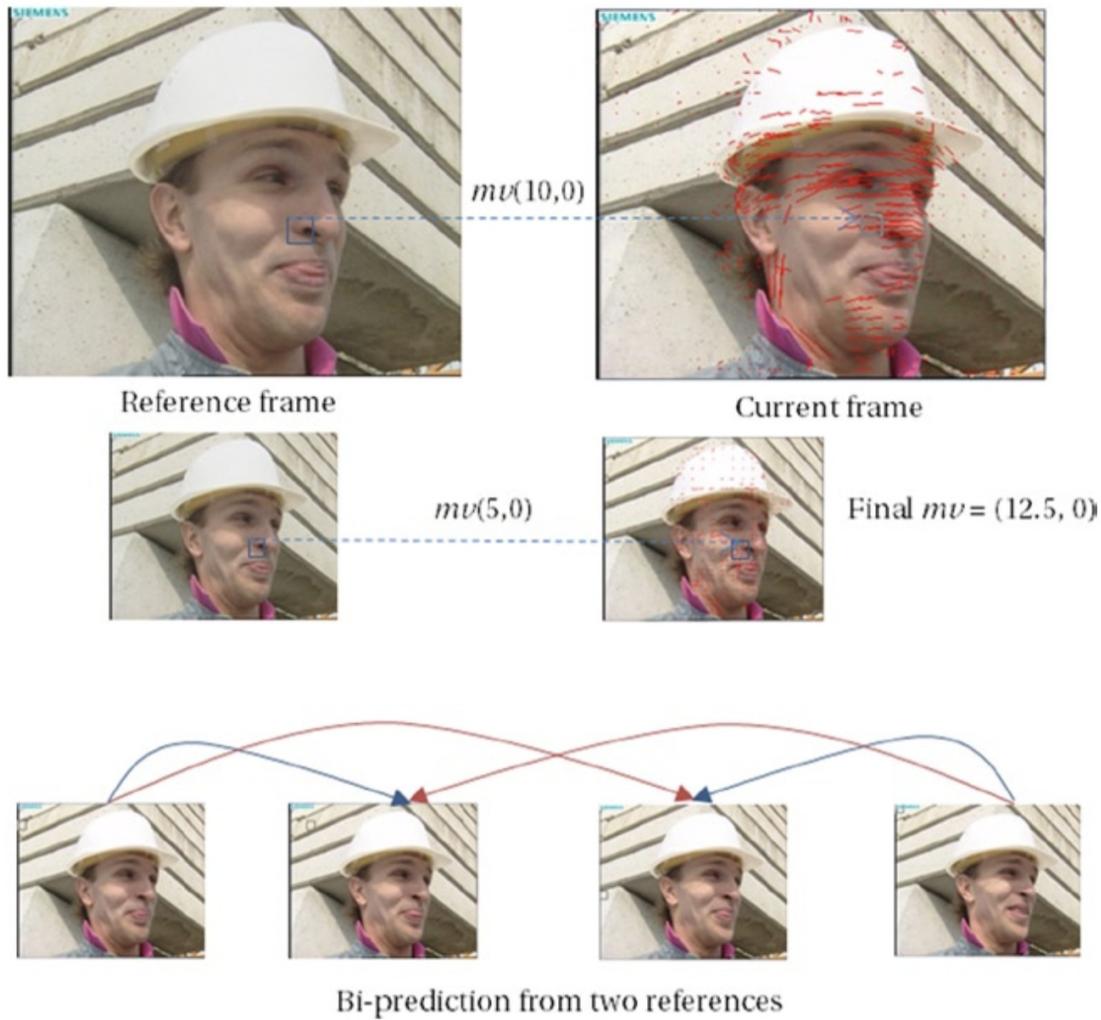


图2-23. 时域预测的例子

为了执行运动补偿，使用称为运动估计的过程来确定运动矢量，该过程通常针对 16×16 像素块或其他形状的图片分区进行。运动估计处理在参考帧中定义搜索窗口，在该搜索窗口中搜索相对于当前帧中的当前块的最佳匹配块。搜索窗口通常在同位置(0, 0)周围形成，与当前帧中的当前块相比，该位置在参考帧中具有相同的水平和垂直坐标。然而，在一些算法中，搜索窗口也可以围绕预测运动向量候选者形成。定义一个匹配标准（通常是失真度量）来确定最佳匹配。这种块匹配运动估计的方法不同于象素递归运动估计，后者涉及以递归方式匹配帧的所有像素。图2-24示出了块匹配运动估计的示例。

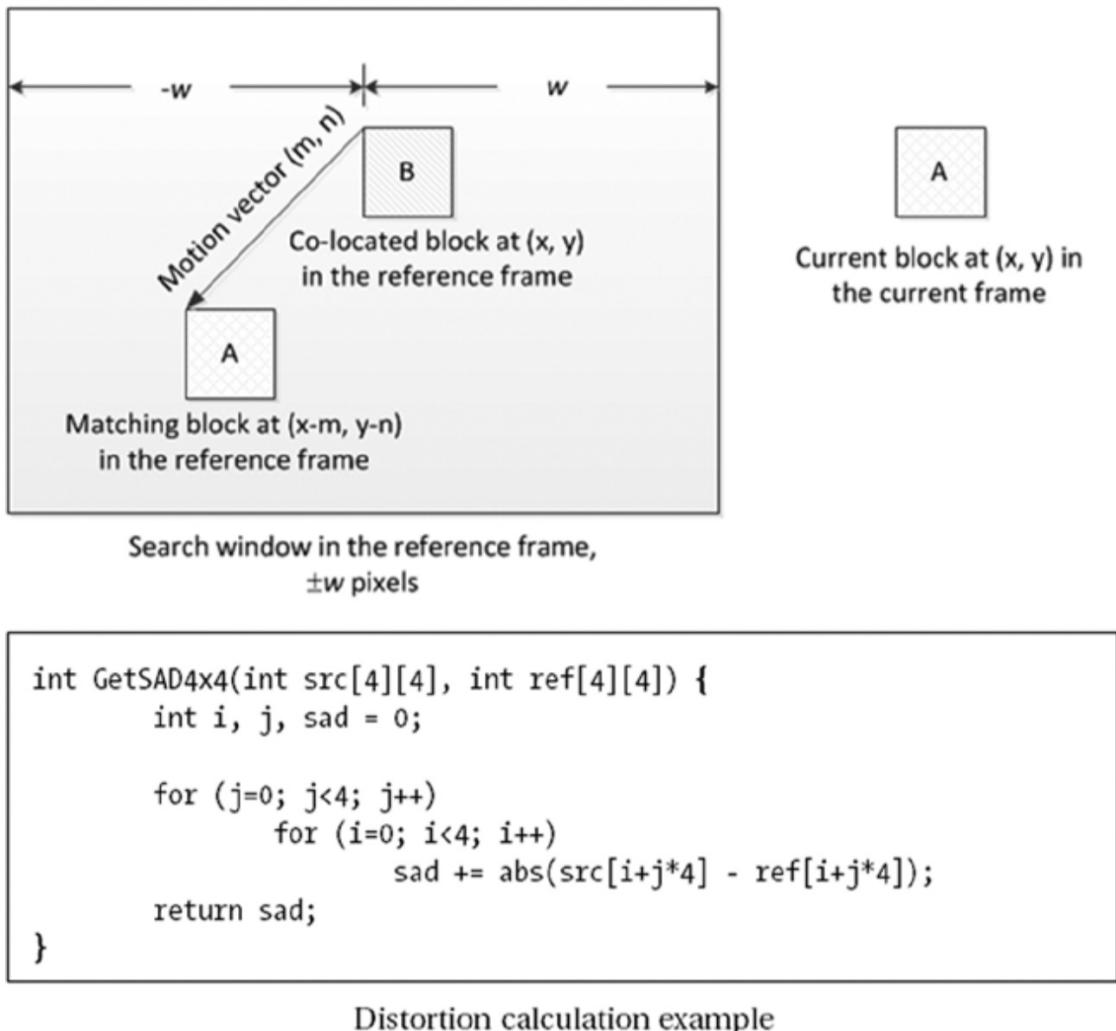


图2-24. 块匹配运动估计

文献中有大量的运动估计算法。块匹配技术试图最小化失真度量并尝试找到搜索区域内两个块之间的全局最小距离。典型的失真度量是平均绝对差 (MAD)，绝对差之和 (SAD) 和涉及Haar变换的绝对变换差之和 (SATD)，具有不同的计算复杂性和匹配能力。运动估计是一个计算量很大的过程，以至于编码速度在很大程度上取决于该过程。因此，失真度量的选择在有损预测视频编码中很重要。

其他编码技术

除了之前章节提到的编码算法，还有其它的流行的编码算法，包括矢量量化和子带编码。这些编码算法也因为各自的特殊特性而广为人知。

向量量化

在矢量量化（VQ）中，视频帧被分解为n维矢量。例如，Y'CbCr分量可以形成三维矢量，或者帧的每一列都可以用作形成w维矢量的矢量元素，其中w是帧的宽度。将每个图像矢量 X 与几个代码矢量 Y_i , $i = 1, \dots, N$ 进行比较。其取自先前生成的码本。

根据最小失真准则（例如均方误差（MSE）），比较可得出 X 和 Y （第 k 个代码向量）之间的最佳匹配。指标 k 使用 $\log_2 N$ 位发送。在接收端，密码本的副本已经存在可用，其中解码器仅从码本中查找索引 k 以重现 Y_k 。VQ框图如图2-25所示。

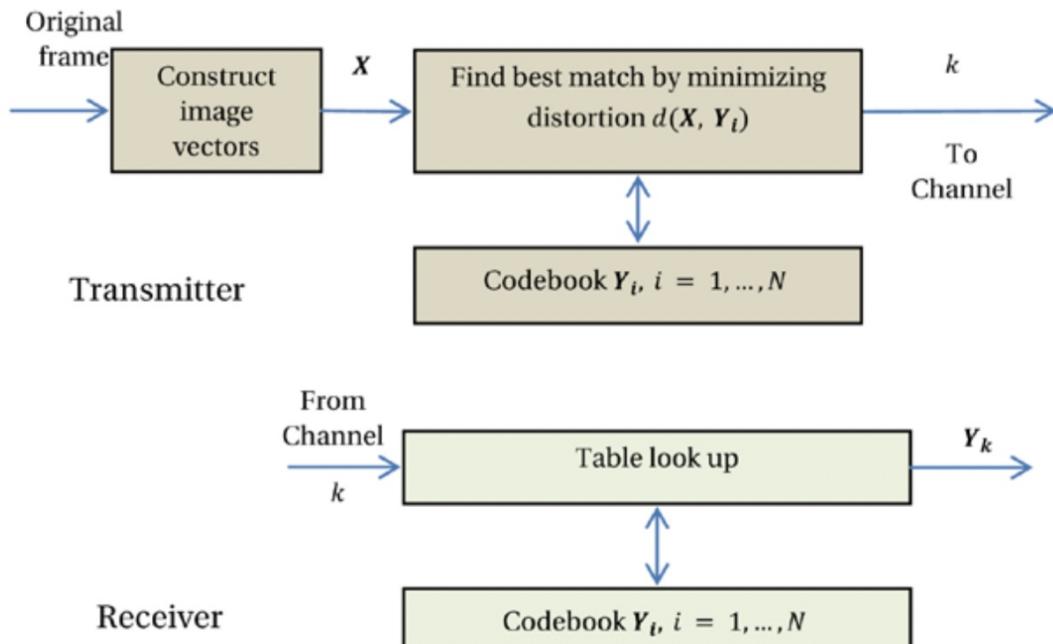


图2-25. 矢量量化方案的框图

之所以能够实现压缩，是因为与可能的代码矢量的数量相比，使用了代码矢量相对较少的代码本。尽管理论上VQ可以达到接近率失真界限的压缩效率¹⁹，但实际上需要一个不合理的大 n 值。但是，使用适当的尺寸，使用智能训练算法仍可以实现合理的压缩效率。有关VQ的详细讨论，请参见Rabbani和Jones²⁰。

子带编码

在子带编码 (SBC) 技术中，对图像进行滤波以创建称为子带的一组图像，每个子带具有有限的空间频率。由于每个子带的带宽都减小了，因此将原始图像的子采样版本用于每个子带。

过滤和二次采样的过程称为分析阶段。然后，使用一个或多个编码器，可能使用不同的编码参数，对子带进行编码。这允许将编码误差分布在不同的子带之间，从而可以通过执行相应的子带的上采样，滤波和随后的组合来实现视觉上最佳的重建。这种重建方式称为合成阶段。子带分解本身不提供任何压缩。但是，与原始图像相比，可以更有效地对子带进行编码，从而提供总体压缩优势。图2-26给出了该方案的框图。许多编码技术可以用于编码不同的子带，包括DWT，Haar变换，DPCM和VQ。在Rabbani和Jones中可以找到关于SBC的详尽讨论²¹。

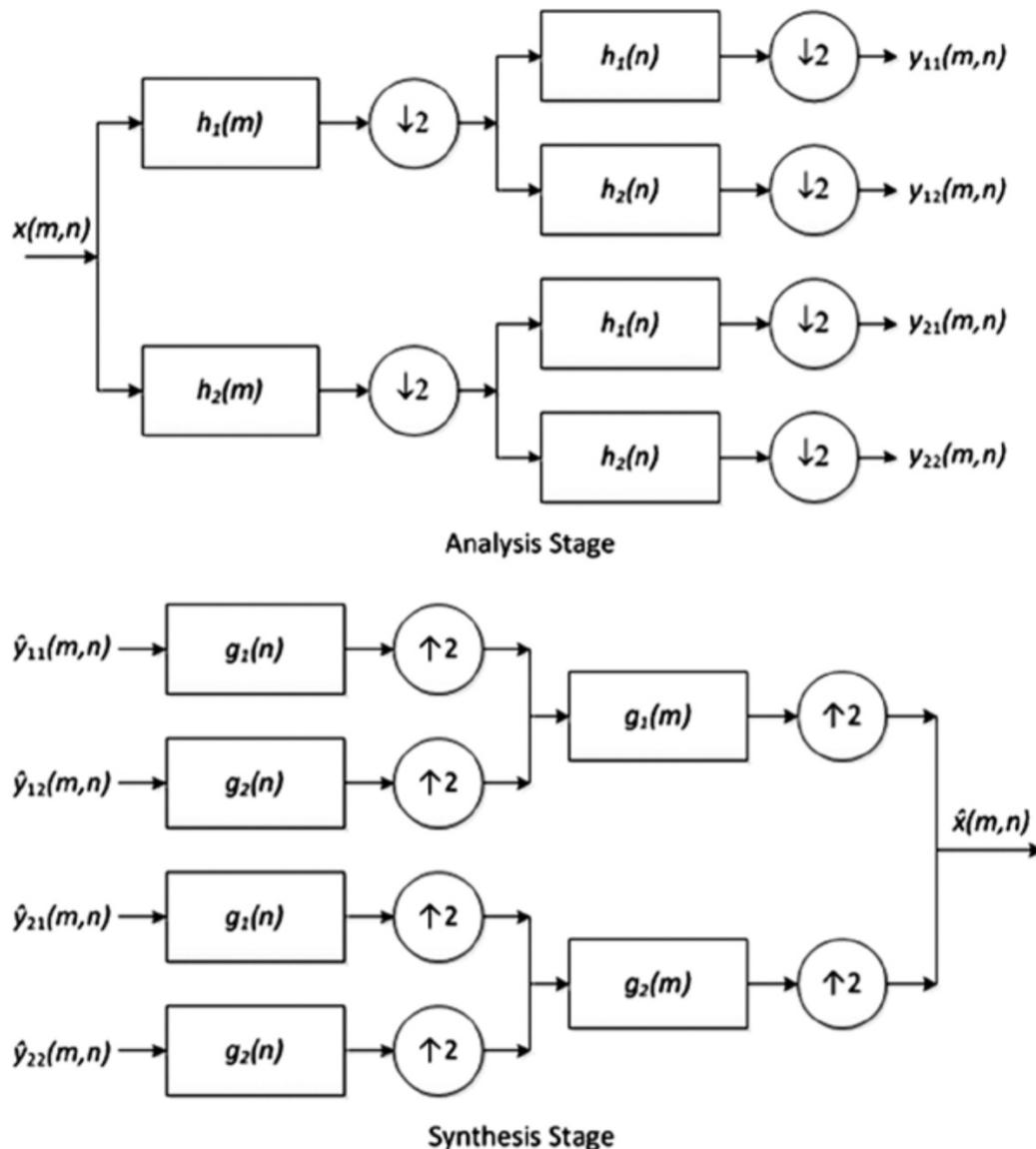


图2-26. 二维子带编码方案的框图

¹⁹. Compression efficiency refers to the bit rate used for a certain distortion or video quality. ↪

²⁰. Rabbani and Jones, Digital Image Compression. ↪

²¹. Rabbani and Jones, Digital Image Compression. ↪

率失真理论 (*Rate-Distortion Theory*)

信源的熵定义了编码图像或视频帧所需的最小bit数。但是，这种定义仅适用于无损编码。实际上，由于HVS的特性，人们可以容忍某些不可逆的视觉质量损失。视觉质量损失的程度可以通过调整编码参数（例如量化级别）来实现。

率失真理论是信息论的一个分支，其目的为：确定在给定码率下的可接受的视觉质量损失量。该理论为有损数据压缩提供压缩效率的理论界限，例如：在特定的码率下能达到的最小失真；或者为了满足特定的失真限制，最小码率应该是多少。率失真理论根据信号保真度准则，通过为各种失真度量和信源模型定义率失真函数— $R(D)$ 。 $R(D)$ 函数具有以下属性：

- 对于给定的失真 D ，存在一种编码方案，对于该编码方案而言，其失真为 D ，其码率为 $R(D)$ 。
- 对于任何编码方案， $R(D)$ 代表可以达到 D 失真的最小码率。
- $R(D)$ 是U型凸函数，并且是关于 D 的连续函数。

图2-27给出了典型的率失真函数的例子。对于无损压缩，其最小码率为 $D = 0$ 时的R值，该值可能等于或小于信源的熵值，具体的数值取决于失真度量。

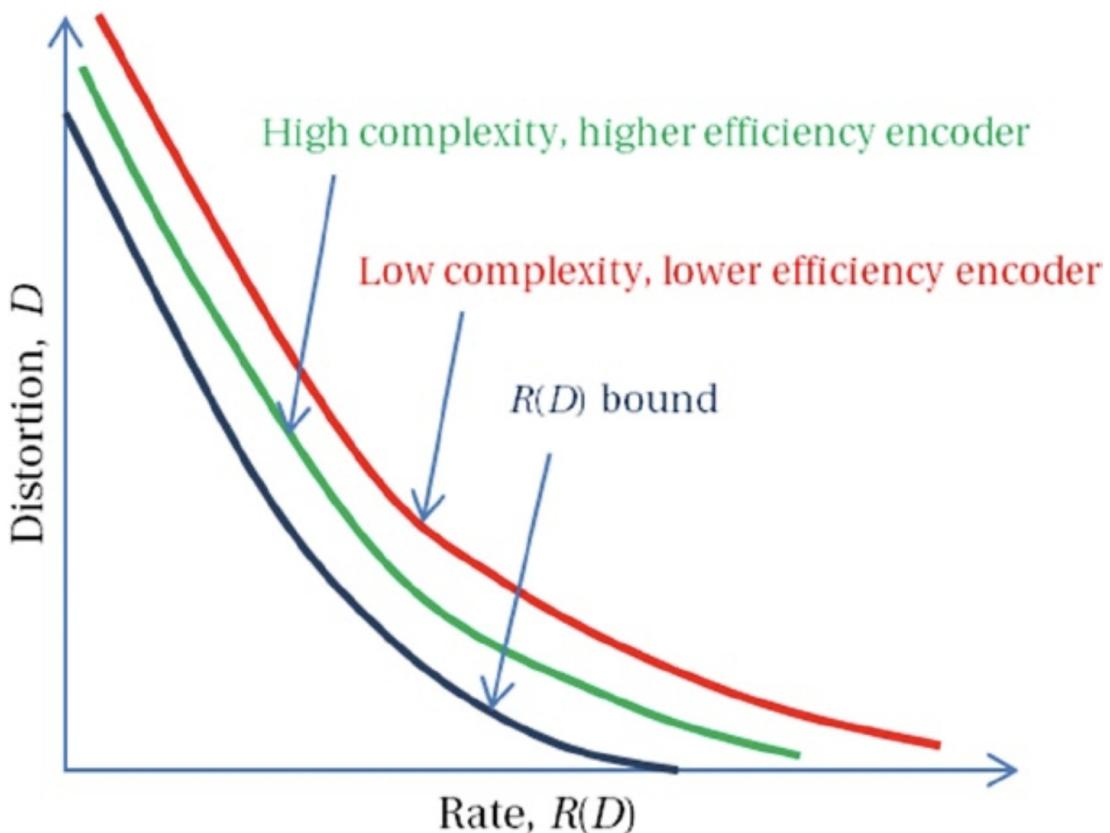


图2-27. 率失真曲线的例子

$R(D)$ 范围取决于信源模型和失真度量。通常，编码器可以用更高的复杂度为代价实现接近 $R(D)$ 的压缩效率。更好的编码器则使用较低的码率来获取可容忍的、较少的失真，但与其他编码器相比可能具有更高的复杂度。为了确定相对于 $R(D)$ 的压缩效率，经常使用具有单位相关系数的二维高斯-马尔可夫源图像模型作为参考。然而，对于自然图像和视频，寻找良好的信源模型以及寻找与人类视觉系统相关的、良好的失真标准是一个热门的研究领域。

有损压缩

影响和区别压缩算法的因素有多个。在为特定的应用模型调整或选择压缩算法时，应认真分析这些因素。这些因素包括：

- **对输入帧类型的敏感性：**压缩算法对于输入帧的不同特征可能会具有不同的压缩效率。输入帧特征包括：动态范围，相机噪声，像素间像素相关性，分辨率等。
- **目标码率：**由于带宽的限制，某些应用程序需要保持一定的码率，但是如果需要，则会牺牲视觉质量以降低码率。压缩算法在率失真曲线上具有不同的最佳点，并且如果目标码率超出其最佳点则将导致较差的视觉质量。某些算法可能无法在低于特定的码率的情况下运行，例如，无论时空复杂度如何，用于HD分辨率的AVC算法必须使用不低于1.5 Mbps的带宽。对于30 FPS的1920×1080的视频，1.5 Mbps的码率的压缩率大约为500倍。
- **恒定码率 vs 恒定质量：**某些算法更适合无缓冲传输，因为它们以恒定的码率运行。但是，对于复杂的场景，可能需要以牺牲视觉质量为代价来维持码率。视频复杂度随场景的变化而变化，此时，恒定的码率会导致重建质量的变化。另一方面，某些算法会保持恒定的质量，码率则会发生变化，这需要在传输时保证足够的缓冲。
- **编码器/解码器的不对称：**某些算法（例如矢量量化方案）使用非常复杂的编码器，而解码器通过简单的表查找来实现。与矢量量化相比，其他方案（例如MPEG算法）需要更复杂的解码器，但其编码器则可以简化。但是，MPEG编码器通常比解码器复杂，因为MPEG编码器中还包含完整的解码器。取决于最终的用户平台，某些方案可能比其他方案更适合特定的应用程序。
- **复杂性和可执行性：**计算复杂性，内存需求以及对并行处理的开放性是压缩算法的硬件或软件实现的主要区别因素。尽管基于软件的实现能够更灵活地进行参数调整以获得最高的可实现质量，并且可以适应未来的变化，但是硬件实现通常会更快且还可以优化功耗。需要根据终端服务平台和应用模型做出适当的权衡。
- **错误恢复能力：**压缩数据容易受到信道错误的影响，但是不同的算法对这种信道错误的敏感性不尽相同。基于DCT的算法可能会因为信道错误而丢失一个或多个块，而带有可变长度代码的简单DPCM算法可能会丢失整帧数据。纠错码（*error-correcting codes*）可以以复杂性为代价来补偿某些错误，但是纠错码在出现突发错误时的效果

并不理想。

- **伪像：**有损压缩算法通常会产生各种伪像。即使在相同的码率下，伪像的类型及其严重性也可能因算法而异。在视觉上，某些伪像可能比随机噪声或较柔和的图像更令人反感，例如可见的块边界，锯齿状的边缘，对象的振铃伪像等。而且，伪像还取决于视频内容的特性和观看条件。
- **多代编码的效果：**视频编辑等应用可能需要多代编码和解码，其中解码后的输出再次用作编码器的输入。编码器的输出是第二代压缩输出。某些应用程序支持此类的压缩。某些压缩算法则不适和多代压缩方案，并且对同一帧执行第二代编码后，通常会导致较差的视频质量。
- **系统兼容性：**并非所有标准都适用于所有系统。尽管标准化的目标之一是在整个行业中获得通用格式，但是某些厂商可能会仅强调某种压缩算法。尽管标准化成果通常认可诸如AVC和HEVC之类的格式，但是厂商可以选择推广类似的算法，例如VC-1，VP8或VP9。总体而言，这是一个更大的问题，涉及诸如蓝光（*Blu-ray*）与HD-DVD之类的技术定义。但是，在选择压缩方案时，必须考虑与目标生态系统的兼容。

总结

本章首先讨论了不同网络下对视频传输的压缩要求。接下来讨论了人类视觉系统的特性如何为视频压缩提供可能。然后，提出了各种压缩数字视频的方法以便使读者熟悉各种视频压缩的方法和概念（特别是最流行的技术）。本章解释了一些与视频压缩相关的技术术语和概念。然后，提出了各种压缩技术用以减少数字视频中可用的空间冗余，时间冗余和统计冗余。本章简要描述了重要的视频编码技术，例如变换编码，预测编码，矢量量化和子带编码。这些技术通常用于目前可用的视频压缩方案中。然后，介绍了率失真曲线，该指标可以用来度量压缩效率，并可用来比较两种编码解决方案。最后，介绍了影响压缩算法的各种因素，了解这些因素将有助于选择合适的压缩算法。

视频编码标准

- 视频的高度普及，归功于视频信号通用表示方法的建立。该通用表示方法是为了实现不同厂商的产品达到格式及可操作性的统一，而根据国际标准建立的。
- 19世纪80年代后期，人们意识到数字视频标准化的必要性。来自计算机、通信、视频行业的特别专家组聚在一起，为数字视频的存储和传输制定实用、低成本且易实现的标准。为了确定这些标准，专家组审阅了大量的视频数据压缩技术、视频数据结构和算法，最终在少数的通用技术上达成一致。本章将详细介绍这些通用技术。
- 在数字视频领域，存在很多国际标准来满足各种行业需求。例如，各种产品及应用程序之间进行数字视频转换时，标准的视频格式就是必不可少的。由于呈现数字视频所需的数据量特别大，因此这些视频数据需要被压缩转换--这也促使视频数据压缩标准成为了必需品。对于不同行业来说，标准的制定主要在于应对各种复杂的终端应用表现。例如，计算机行业的分辨率标准化，电视行业的数字影响标准化，以及远程通讯行业的网络协议标准化。数字视频的广泛应用，促使这些行业紧密的联系在一起；同时标准化工作也在致力于满足这些交叉行业的需求。本章我们将讨论国际视频编码标准的里程碑，以及业界广泛使用的编码算法。

视频编码国际标准概述

- 视频编码国际标准是由来自国际标准化组织(ISO)及国际电信联盟(ITU)等组织的专家委员会共同制定的。其目的是制定一套行业内通用的视频格式，来实现不同厂商、不同视频编码相关软硬件制造商之间的互通。
- 视频编码算法标准化，是从图像压缩方案开始的，例如应用于传真等应用中的二值图像压缩标准JBIG(*ITU-T Rec. T82 and ISO/IEC 11544, March 1993*)，以及可用于彩色图像的普通图像压缩标准JPEG(*ITU-T Rec. T81 and ISO/IEC 10918-1*)。JPEG标准化开始于1986年，但是该标准直到1992年才被ITU-T批准，1994年被ISO批准。19世纪80年代开始的一项重要的视频压缩算法标准化，ITU-T H.261，1988年被批准，是视觉通信领域第一个里程碑式的标准。在此基础上，随着电视、电影、计算机、通信、信号处理领域的发展，以及各厂商新的使用要求提出，视频标准化活动得到了极大的促进。随后产出了MPEG-1，H.263，MPEG-2，MPEG-4第二部分，AVC/H.264，以及HEVC/H.265算法。在接下来的章节中，我们将简要介绍图像及视频编码相关的几个主要国际标准。

JPEG 标准

JPEG是一种连续色调静止图像压缩标准，专为桌面出版，图形艺术，彩色传真，报纸线照片传输，医学成像等应用而设计。基线JPEG算法使用基于DCT的编码方案，其中将输入分为 8×8 像素块。每个块都经过二维正向DCT，然后进行均匀量化。最终的量化系数以Z字形扫描，以形成一维序列，其中高频系数（可能为零值）被放置在序列的后面，以方便游程编码。经过行程编码后，所得的符号将进行更有效的熵编码。

第一个DCT系数（通常称为直流系数，因为它是像素块平均值的一个度量），相对于前一个块进行了差分编码。使用可变长度的霍夫曼码对具有非零频率的AC系数的游程长度进行编码，为更可能的符号分配较短的代码字符。图3-1显示了JPEG编解码器框图。

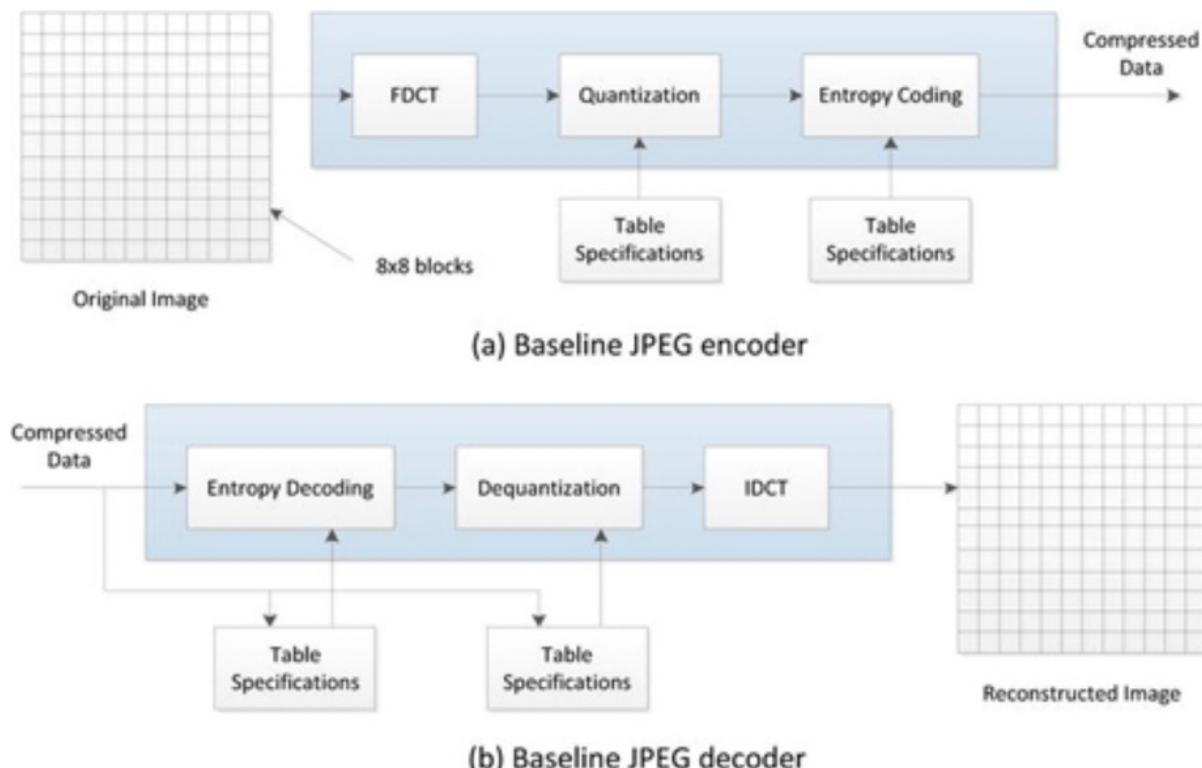


图3-1.基线JPEG编解码器框图

H.261标准

H.261是业界采用的第一个重要且实用的数字视频编码标准。它是 $p \times 64\text{ kbps}$ 视听服务的视频编码标准（其中 $p = 1, \dots, 30$ ），旨在通过ISDN提供视频电话和视频会议服务，并将此类应用视频传输的各个方面统一到同一标准上¹。目标包括以最小的延迟实时传输视频（通常是15-30fps，延迟小于150ms）。尽管作为一个成功的数字视频编码标准，其提供了全行业的互操作性、通用格式和压缩技术，但是H.261已过时，且如今已经很少使用了。

在H.261标准中，所有编解码器必须以四分之一公共中间格式（Quarter Common Intermediate Format, QCIF）的视频格式进行操作，而CIF的使用是可选的。由于CIF和QCIF在29.97fps下的未压缩视频比特率分别为26.45 Mbps和9.12 Mbps，因此在提供正常的视频质量的同时，使用ISDN通道传输这些视频信号非常困难。为了实现此目标，H.261将视频划分为分层的块结构，该结构包括图片，块组（GOB），宏块（MB）和块。一个宏块由四个 8×8 亮度块和两个 8×8 色度块组成； 3×11 的宏块阵列又构成一个块组。一个QCIF图片具有三个块组，而一个CIF图片则有12个。分层数据结构如图3-2所示。



图3-2.H.261视频多路复用编码器数据结构

H.261源编码算法是帧内和帧间编码的混合，利用了空间和时间上的冗余。帧内编码类似于基线JPEG，其执行基于块的 8×8 DCT变换，并对DCT系数进行量化。量化系数经过可变长度霍夫曼编码的熵编码，利用信号的统计特性实现了比特率的降低。

帧间编码涉及运动补偿帧间预测，并消除图片之间的时间冗余。帧间预测仅在正向执行，没有双向预测的概念。在以整数像素精度执行运动补偿的同时，可以将环路滤波器切换到编码器中，以在必要时通过消除编码后的高频噪声来改善图像质量。图3-3给出了H.261源编码器的框图。

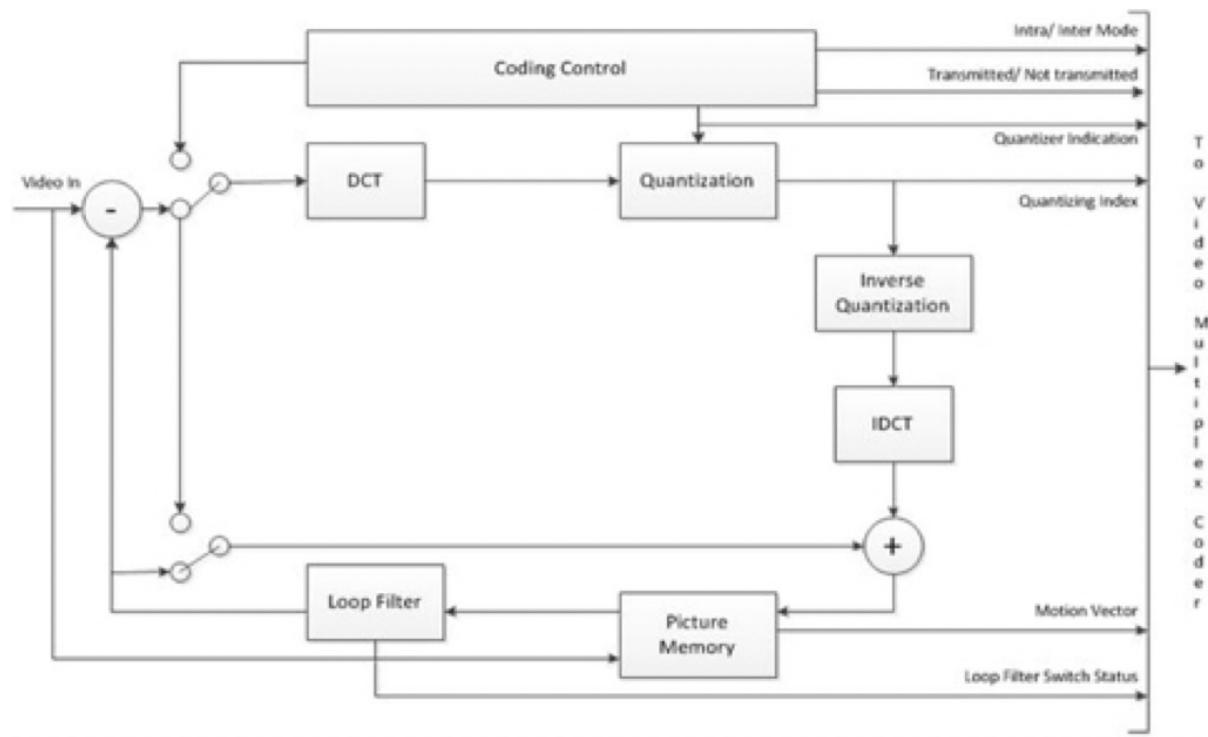


图3-3.ITU-T H.261源编码器框图

¹. M. L. Liou, "Overview of the px64 kbit/s Video Coding Standard," Communications of the ACM 34, no. 4 (April 1991): 60–63. ↪

MPEG-1标准

在19世纪90年代，受光盘数字音频市场成功的鼓舞，CD-ROM进入了数据存储领域。这促使MPEG-1标准的诞生，该标准针对要求1.2至1.5 Mbps视频家庭系统（VHS）品质视频的应用进行了优化。最初的动机之一是将压缩视频装入广泛可用的CD-ROM中；但是，出现了数量惊人的新应用，以利用标准算法提供的高压缩比、质量正常的视频。MPEG-1仍然是视频编码标准历史上最成功的发展之一。但是，可以说，MPEG-1标准最著名的部分是它引入的MP3音频格式。MPEG-1的预期应用包括CD-ROM存储，计算机多媒体等等。MPEG-1标准在1991年被批准为ISO/IEC11172。该标准包括以下五个部分：

1. 系统：处理视频，音频和其他数据的存储和同步
2. 视频：定义压缩视频数据的标准算法
3. 音频：定义压缩音频数据的标准算法
4. 符合性：定义测试方法以检查标准实施的正确性
5. 参考软件：提供标准关联的软件，作为正确实施编码和解码算法的示例

MPEG-1比特流语法很灵活，由六层组成，每层执行不同的逻辑或信号处理功能。图3-4描绘了洋葱结构中排列的各个层。

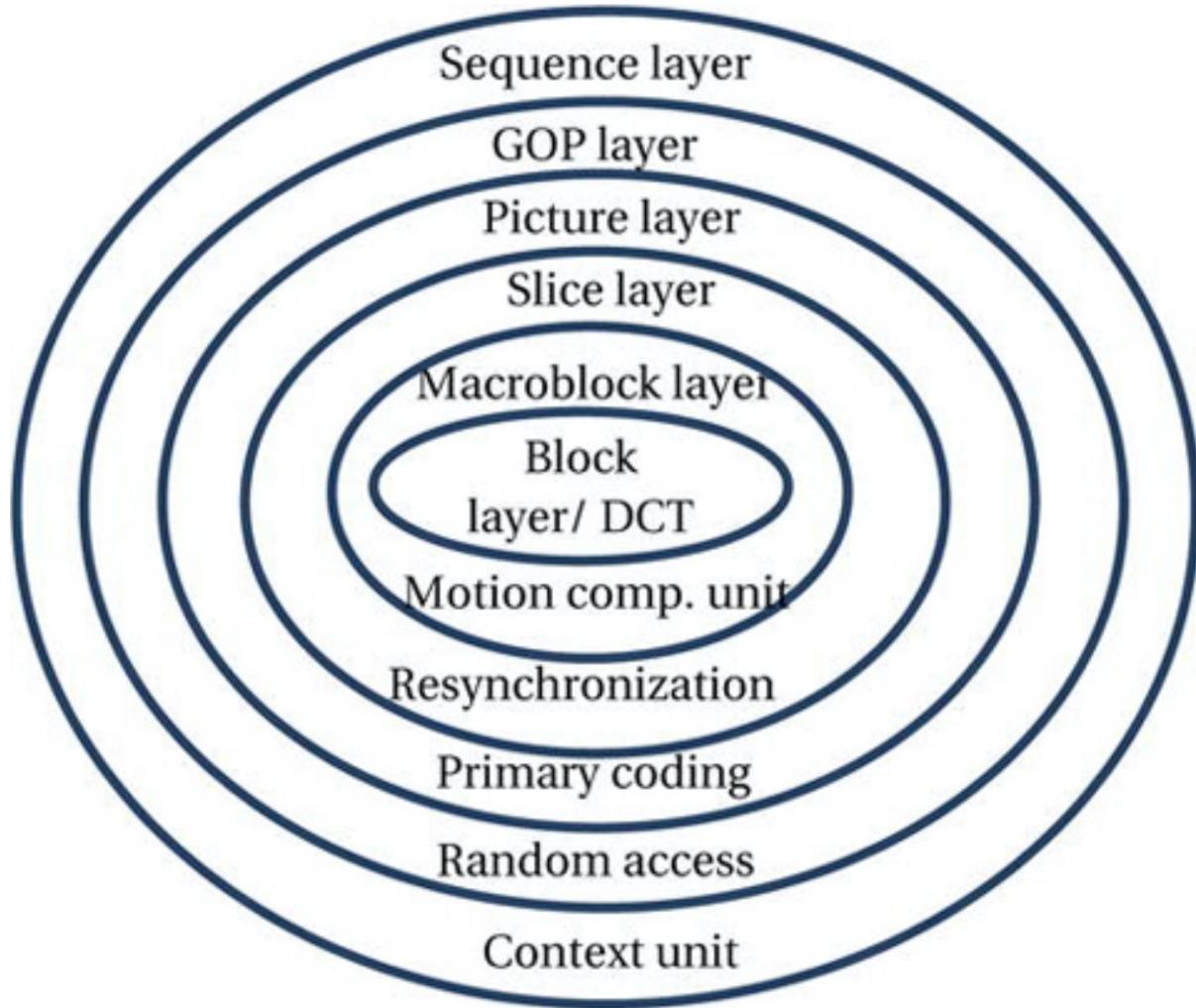


图3-4.MPEG-1比特流语法的洋葱结构

MPEG-1是为编码渐进视频序列而设计的，推荐的图片大小为约1.5Mbps、360×240（或352×288，又称为CIF）。但它不限于此格式，并且可以应用于更高的比特率和更大的图像尺寸。预期的色度格式为4: 2: 0，像素深度为8位。该标准要求实时解码，并支持促进与存储的比特流交互。MPEG-1仅指比特流和解码过程的语法，从而为编码器实现提供足够的灵活性。编码器通常是为满足特定的使用需求而设计的，但是它们有望在编码效率和复杂性之间提供足够的权衡。

与任何其他标准一样，MPEG-1视频算法的主要目标是在给定比特率下实现最高的视频质量。为了达到这个目标，MPEG-1采用与H.261相似的压缩方法：它也是帧内和帧间减少冗余技术的混合。对于帧内编码，将帧划分为8×8个像素块，使用8×8 DCT将其转换到频域，进行量化，Z形扫描，并使用可变长度霍夫曼码对生成位的游程长度进行编码。

通过计算原始帧与其根据重构参考帧构造的运动补偿预测之间的差异信号，即预测误差，可以减少时间冗余。但是，MPEG-1的时间冗余减少与H.261在几个重要方面存在差异：

- MPEG-1允许进行双向时间预测，从而为给定质量的图片提供了比仅使用前向预测所能达到的更高的压缩率。对于双向预测，某些帧使用显示顺序的过去或将来帧作为预

测参考进行编码。可以从过去参考帧中的一个块、未来参考帧中的一个块、或从两个块的平均中（每个参考帧一个块）预测一个像素块。在双向预测中，以更大的编码器复杂性和额外的编码延迟为代价实现了更高的压缩。但是，它对于数据存储和其他脱机应用程序仍然非常有用。

- 此外，MPEG-1引入了半像素精度进行运动补偿，并剔除了环路滤波器。半像素精度部分补偿了H.261环路滤波器所提供的好处，同时不会传播高频编码的噪声，也不会牺牲编码效率。

视频序列层指定一系列参数，例如视频帧的大小、帧率、码率等。图片组（GOP）层支持随机访问、快速搜索以及编辑。GOP的第一帧必须进行帧内编码（I帧），其只能使用DCT，量化和可变长度编码在空间维度上实现压缩。在I帧之后是前向预测编码帧（P帧）和双向预测编码帧（B帧）的排列。I帧被独立编码并用作解码的入口点，从而为随机访问比特流和快速搜索（例如类似VCR的特技播放，快进和快退）提供了能力。

图片层处理特定的帧，并包含帧类型（I, P或B）的信息以及帧的显示顺序。对应于运动矢量和量化DCT系数的比特位在切片层，宏块层和块层中的包中。切片是宏块的连续段。如果发生位错误，切片层有助于在解码期间重新同步位流。宏块层包含相关的运动矢量位，然后是块层，该块层由编码的量化DCT系数组成。图3-5在编码和显示顺序上展示了MPEG图片结构，该结构适用于MPEG-1和MPEG-2。

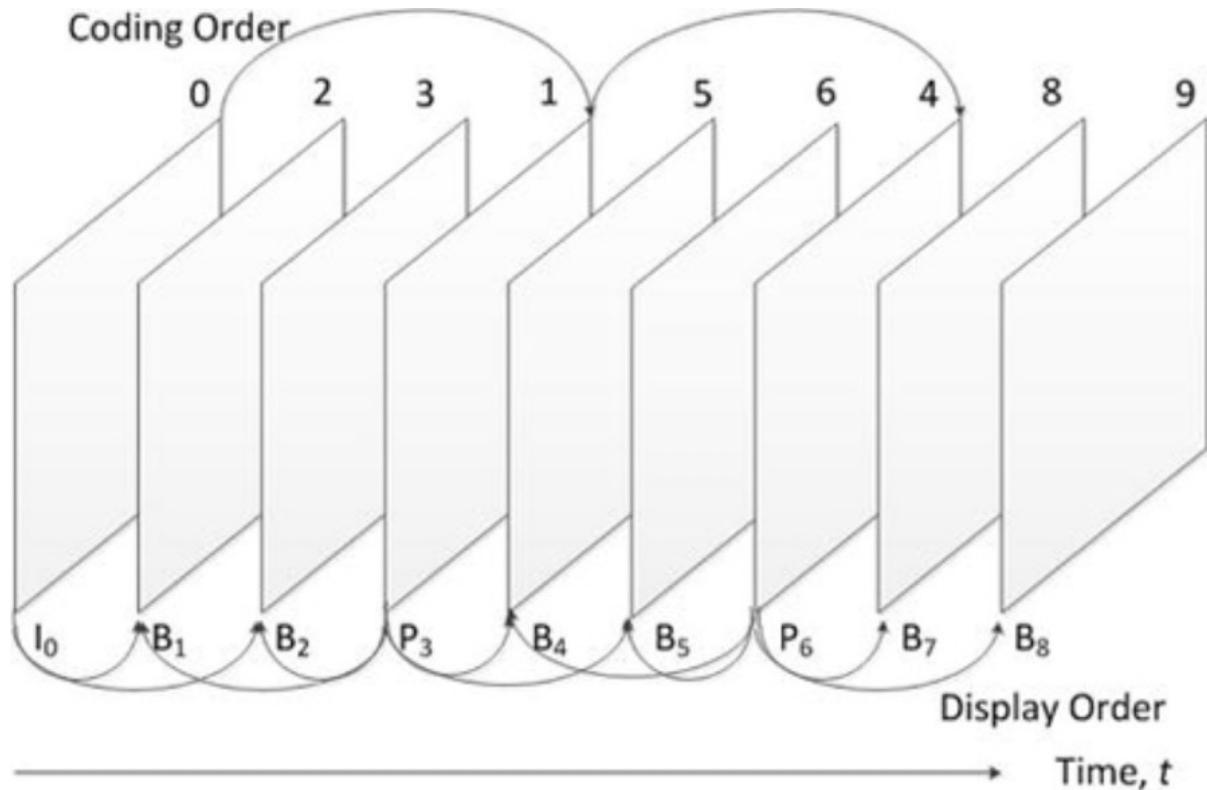


图3-5.I/P/B帧结构、预测关系、编码顺序及显示顺序

MPEG-2标准

MPEG-2被定义为运动图像和相关音频的通用编码标准。该标准由国际标准化组织/国际电工委员会（ISO/IEC）和电信标准化部门（ITU-T）的联合技术委员会指定，并于1993年被批准为ISO/IEC国际标准13818和ITU-T H.262建议书。为了解决MPEG-1中的现有问题，MPEG-2的标准化活动主要基于以下考虑：

- 将音频压缩通道的数量从2个通道扩展到5.1个通道。
- 为广播应用中的交错式视频添加标准化支持。
- 除了支持MPEG-1中的“约束参数”比特流以外，还提供更多标准配置文件，以支持更高分辨率的视频内容。
- 从对4:2:0颜色采样的支持，扩展到4:2:2和4:4:4。

对于MPEG标准，标准委员会讨论了视频和音频压缩以及视听数据压缩复用的系统注意事项。在MPEG-2应用中，压缩的视频和音频基础流通过多路复用来构成节目流；几个节目流在传输之前被打包并组合成传输流。在接下来的讨论中，我们将专注于MPEG-2视频压缩标准。

MPEG-2主要针对于2Mbps或更高码率的各种应用，包括高质量（NTSC）制式及高清电视（HDTV）。尽管广泛用作无线、有线和卫星直播电视系统的数字电视信号格式，但其他典型应用包括数字盒式磁带录像机（VCR），数字视频光盘（DVD）等。作为支持2Mbps至40Mbps范围内各种应用程序的通用标准，MPEG-2的压缩目标在30到40范围内。为提供应用程序独立性，MPEG-2支持多种视频格式，分辨率范围从源输入格式（短距离传输）到高清电视（HDTV）。表3-1展示了MPEG-2应用中使用的一些典型视频格式。

图3-1.典型的MPEG-2参数

格式	分辨率	压缩后码率（Mbps）
SIF	360x240, 30fps	1.2~3
ITU-R 601	720x480, 30fps	5~10
EDTV	960x480, 30fps	7~15
HDTV	1920x1080, 30fps	18~40

MPEG-2的目的是提供更好的图像质量，同时保留对编码比特流的随机访问。但是，这是一项相当困难的任务。由于目标码率所要求的高压缩率，仅通过帧内编码不能获得良好的图像质量。相反，用纯帧内编码可以最好地满足随机访问要求。这种困境需要在帧内编码和帧间编码之间达到微妙的平衡。与MPEG-1类似，这导致了I帧，P帧和B帧定义的

出现。I帧的压缩程度最低，并且在频域中以量化形式大约包含图片的全部信息，通过增强鲁棒性减少错误。P帧是根据过去的I或P帧来进行预测的，而B帧通过使用过去和将来的I或P帧进行运动补偿来提供最大的压缩。但是，B帧最容易出现信道错误。

MPEG-2编码器首先为信号选择合适的空间分辨率，然后进行块匹配运动估计，以找到当前帧中宏块（ 16×16 或 16×8 像素区域）相对于宏块的位移，该宏块从先前或将来的参考系或其平均值中获得。最佳匹配块的搜索基于平均绝对差（MAD）失真准则；当所有宏块的像素差的累积绝对值最小时，就会发生最佳匹配。然后，运动估算过程将定义一个运动矢量，该运动矢量表示当前块位置相对于最佳匹配块位置的位移。为了减少时间冗余，运动补偿既用于根据先前参考图片对当前图片进行因果预测，又用于根据过去和未来参考图片进行非因果插值预测。图片的预测是基于运动矢量构建的。

为减少空间冗余，使用块变换编码技术进一步压缩差异信号（即预测误差），该技术采用二维正交 8×8 DCT变换去除空间相关性。生成的变换系数以交替或锯齿扫描模式排序，然后以不可逆的过程对其进行量化，从而丢弃不太重要的信息。在MPEG-2中，在宏块层使用了自适应量化，从而实现了平稳的码率控制和感知上统一的视频质量。最后，将运动矢量与残差量化系数组合在一起，使用变长霍夫曼码进行传输。霍夫曼编码表是预先确定和优化好的，其适用于某些特定应用的有限压缩比范围。图3-6显示了MPEG-2视频编码框图。

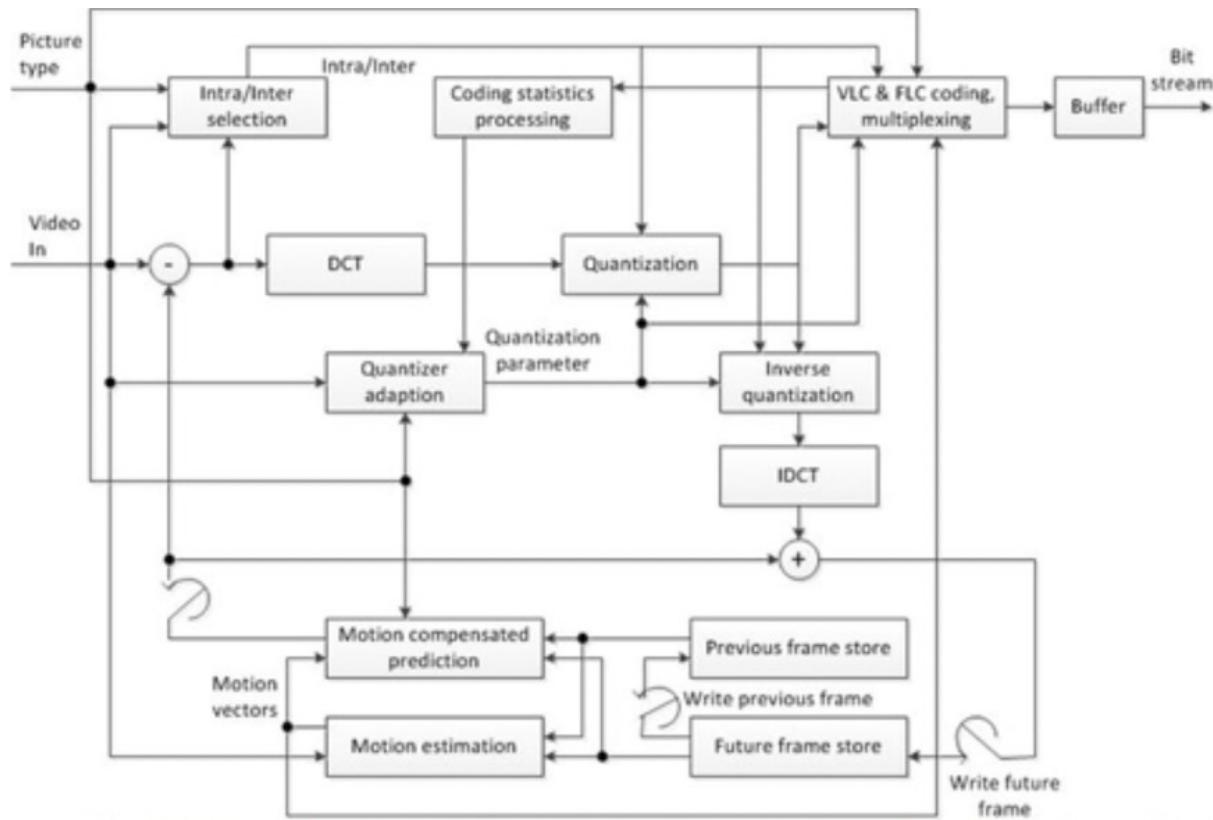


图3-6.MPEG-2视频编码框图

MPEG-2中的比特流语法被划分为配置文件子集，这些子集指定了对语法的约束。配置文件进一步分为多个级别，这些级别是对比特流中的参数施加的一组约束集。MPEG-2中定义了五个配置文件：

- 主要：以标清图像的最高质量为目标。
- 简单：直接通过不插入图片来节省内存。
- SNR可扩展：旨在通过使用多于一层的量化来按需提供更好的信噪比。
- 空间可扩展：旨在通过使用加权和重构参考图片的附加层来按需提供可变分辨率。
- 高：旨在支持4: 2: 2色度格式和完全可伸缩性。

在每个配置文件中，最多定义了四个级别：

- 低：提供与H.261或MPEG-1的兼容性。
- 主要：对应于传统电视。
- 1440高：大致相当于高清电视，每行有1,440个样本。
- 高：大致相当于高清电视，每行1,920个样本。

主要配置文件，主要级别（MP @ ML）反映了MPEG-2在娱乐应用方面的最初关注点。允许的配置文件级别组合为：具有主要级别的简单配置文件，具有所有级别的主要配置文件，具有低级别和主要级别的SNR可扩展配置文件，具有1440高级别的空间可扩展配置文件以及具有除低级别之外的所有级别的高配置文件。

比特流语法也可以划分如下：

- 不可扩展的语法：MPEG-1的超集，具有用于隔行视频信号的额外压缩工具以及可变码率、交替扫描、隐藏运动矢量、DCT转换内部格式等。
- 可扩展语法：类似于不可扩展语法的基础层，以及一个或多个增强层，能够实现有用视频的重构。

压缩比特流的结构如图3-7所示。这些分层类似于MPEG-1。压缩的视频序列以包含图像分辨率、图像速率、码率等的序列头开始。MPEG-2中有一个序列扩展头，其中包含视频格式、颜色基色、显示分辨率等。序列扩展头之后可以是具有时间码的可选GOP头，其后是包含时间参考、帧类型、视频缓冲验证器（VBV）延迟等的帧头。帧头可以由包含隔行、DCT类型和量化标度类型信息的图片编码扩展所接替，通常在其后跟随切片头以便于重新同步。在切片内，几个宏块被分组在一起，其中宏块的地址和类型、运动矢量、编码块模式等被放置在宏块中所有块的实际VLC编码的量化DCT系数之前。这些分片可以在任何宏块位置开始，并且不限于宏块行的开头。

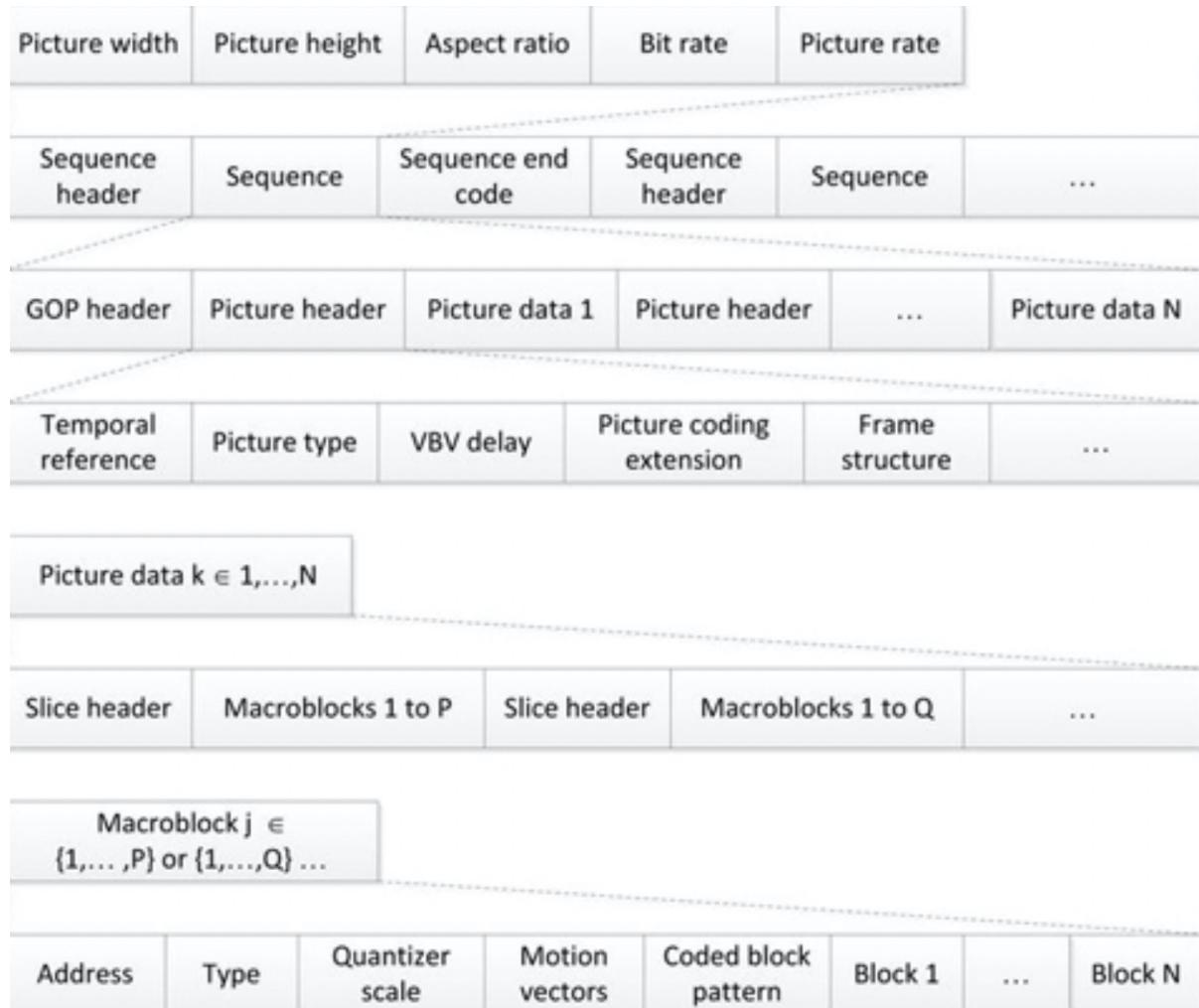


图3-7.MPEG-2视频比特流语法压缩结构

由于MPEG-2基础层是MPEG-1的超集，因此符合标准的解码器可以解码MPEG-1比特流，从而提供向后兼容性。此外，MPEG-2能够为运动补偿预测选择最佳模式，从而可以从整个参考帧或从参考帧的顶部或底部场预测当前帧或场，从而找到更好的场之间的关系。MPEG-2还添加了替代扫描模式，比之字形扫描模式更适合隔行视频。此外，还可以在线性和非线性量化表之间进行选择，并且内部宏块最多支持11位DC精度。相对于MPEG-1不支持非线性量化表，并且仅提供8位的DC内精度，这些是对MPEG-1的改进。在相同的比特率下，MPEG-2的质量比MPEG-1更好，尤其是对于隔行视频源。而且，MPEG-2对于给定码率下的参数变化更加灵活，使缓冲区控制更平滑。但是，这些好处和改进是以增加复杂性为代价的。

H.263标准

虽然ITU-T定义的H.263标准致力于低码率视频编码，但未规定对视频码率的限制；这种约束是由终端或网络给出的。H.263的目标是提供比其前身H.261更好的图像质量。从概念上讲，H.263是独立于网络的，并且可以用于广泛的应用程序，但是它的目标应用程序是诸如PSTN，ISDN和无线网络之类的低比特率网络上的可视电话和多媒体。H.263考虑的一些重要因素包括开销小、导致成本低下低复杂度、与现有视频通信标准（例如H.261，H.320）的互操作性、对信道错误的鲁棒性以及服务质量（QoS）参数。基于这些考虑，开发了一种有效的算法，该算法为制造商提供了在图像质量和复杂度之间进行权衡的灵活性。与H.261相比，它以不到一半的码率提供了相同的主观图像质量。

与其他标准类似，H.263使用图片间预测来减少时间冗余，并对残差预测误差进行变换编码以减少空间冗余。变换编码基于 8×8 DCT转换。用标量量化器对变换后的信号进行量化，并且在发送之前对所得符号进行可变长度编码。在解码器处，接收到的信号进行逆量化和逆变换以重建预测误差信号，将其添加到预测中，从而重构图像。重构的图片被存储在帧缓冲器中，以用作下一张图片预测的参考。编码器包含一个嵌入式解码器，其执行相同的解码操作，以确保在编码器和解码器上都进行相同的重构。

H.263支持五种标准分辨率：sub-QCIF (128×96)，QCIF (176×144)，CIF (352×288)，4CIF (704×576) 和16CIF (1408×1152)，涵盖了空间分辨率范围的大部分。解码器中必须同时支持sub-QCIF和QCIF格式，并且编码器必须支持这些格式之一。这一要求是高分辨率和低成本之间的折衷。

图片被分为 16×16 个宏块，由四个 8×8 亮度块和两个空间对齐的 8×8 色度块组成。一个或多个宏块行被组合成一块组（GOB），以在发生传输错误时实现快速重新同步。与H.261相比，GOB结构得到了简化；GOB标头是可选的，并且可以根据错误恢复力和编码效率权衡使用。

为了提升帧间预测，H.263解码器具有块运动补偿功能，而运动补偿功能在编码器中的使用是可选的。每个宏块发送一个运动矢量。与使用全像素精度和环路滤波器的H.261相比，H.263使用半像素精度进行运动补偿。在变长编码之后，运动矢量与变换系数一起被传送。可以通过预处理或通过更改以下编码器参数来控制编码视频的码率：量化器比例大小、模式选择和图像速率。

除了上述的核心编码算法，H.263还包括四个可选的编码选项，如下所示。前三个选项用于改善帧间预测，而第四个选项与无损编码有关。编码选项增加了编码器的复杂度，但改善了图像质量，从而允许在图像质量和复杂度之间进行权衡。

- 无限制运动矢量（UMV）模式：在UMV模式下，允许运动矢量指向编码图片区域之

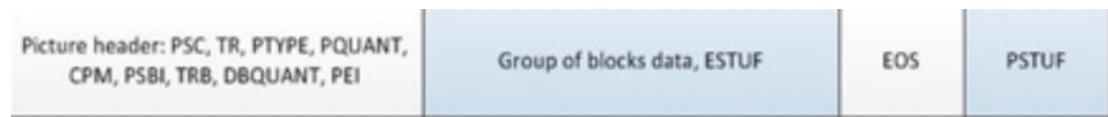
外，从而可以实现更好的预测，尤其是位于图片区域之外的参考宏块中的一部分不可用于预测。通常，对于那些不可用的像素，我们使用其周边像素来预测。但是，此模式允许利用完整的参考宏块，从而产生质量增益，尤其是对一些图片边界附近有运动的较小的图片格式。请注意，对于sub-QCIF格式，所有宏块中约有50%位于边界或边界附近。

- 高级预测（AP）模式：在此可选模式下，重叠块运动补偿（OBMC）用于亮度，从而减少了块状伪像并提高了主观质量。一些宏块使用四个 8×8 运动矢量替代 16×16 矢量，从而以更多的比特为代价提供了更好的预测。
- PB帧（PB）模式：PB帧模式的主要目的是在不显着增加码率的情况下增加帧率。一个PB帧由两张编码图像组成，这两张图片作为一个单元。从最后解码的P帧预测P帧，从上一个P帧和当前P帧来预测B帧。尽管P帧和B帧来源于MPEG标准，但是H.263标准中的B帧用于完全不同的目的。B帧的质量有意保持较低水平，从而最大程度地减少双向预测的开销，而这种开销对于低比特率应用很重要。B帧仅用所分配比特率的15%到20%，但可以带来如平滑运动般更好的主观感觉。
- 基于语法的算术编码（SAC）模式：H.263已针对非常低的码率进行了优化。严格来讲，它允许使用可选的基于语法的算术编码模式，该模式用可变长度编码的算术编码替换了霍夫曼编码。霍夫曼码必须使用整数位，但算术编码消除了这种限制，从而产生了降低码率后的无损编码。

H.263的视频比特流以分层结构排列，该分层结构由以下几层组成：图像层，块组层，宏块层和块层。每个编码图像包括图像头，其后是布置为块组的编码图像数据。一旦图片的传输完成，就会发送序列结束（EOS）码和填充位（ESTUF）（如果需要）。比特流中有一些可选元素，例如，仅当图片类型（PTYPE）表示B图片时，B帧的时间参考（TRB）和量化参数（DBQUANT）才可用。对于P帧，将传输量化参数PQUANT。

GOB层由一个GOB标头和宏块数据组成。每一帧中的第一个GOB不需要标头，而对于其他GOB，标头是可选的，并根据可用带宽使用。GOB起始代码（GBSC）可能需要组填充（GSTUF）。组号（GN），GOB帧ID（GFID）和GOB量化器（GQUANT）可以出现在GOB标头中。

每个宏块都由宏块头和后面的编码块数据组成。编码宏块通过一个称为COD的标志来标识；对于P帧，所有宏块都被编码过。当由COD标识时或当PTYPE标识为I帧时，会存在色度的宏块类型和编码块模式（MCBPC）。对于PB帧的非帧内宏块，存在B帧（MODB）的宏块模式。可以根据MCBPC来提供亮度编码的块模式（CBPY），以及用于差分量化器（DQUANT）和运动矢量数据（用于高级预测的MVD或MVD2-4）的代码。仅当编码模式为B（MODB）时，才会显示B块（CBPB和MVDB）的CBP和运动矢量数据。如前所述，在正常模式下，宏块由四个亮度块和两个色度块组成。但是，在PB帧模式下，可以将一个宏块视认为包含12个块。块结构由内部DC以及变换系数（TCOEF）组成。对于帧内宏块，为宏块中的每个P块发送帧内DC。图3-8显示了H.263层的每层结构。



Group of Blocks layer



Macroblock layer



Block layer



图3-8.H.263比特流多层结构

MPEG-4标准（第二部分）

MPEG-4，于1999年3月被国际标准化委员会/国际电工委员会（ISO/IEC）批准为多媒体数据表示和编码的标准，正式成为ISO/IEC 14496标准。除了音视频编码和多路复用之外，MPEG-4还解决了各种二维或三维合成媒体的编码问题，以及视听场景和组合的灵活表示问题。随着多媒体用途的发展和多样化，MPEG-4的关注范围从最初有限视听材料的低码率编码扩展到了新的多媒体功能。

与MPEG-1或MPEG-2中基于像素的视频处理不同，MPEG-4支持基于内容的通信、访问，及对数字视听对象操纵，用于实时或非实时交互操作或非交互式应用程序。MPEG-4提供了扩展的功能，并改善了先前标准提供的编码效率。例如，它支持可变的像素深度，基于对象的传输以及包括无线网络和Internet在内的各种网络。多媒体创作和编辑功能是MPEG-4特别吸引人的功能，有望取代现有的文字处理器。从某种意义上说，H.263和MPEG-2嵌入在MPEG-4中，确保了对数字电视和可视电话等应用程序的支持，同时还用于基于Web的媒体流。

MPEG-4与较早的视频编码标准不同，它引入了基于对象的表示，及真实或虚拟视听（AV）对象的编码方法。每个AV对象都有其本地3D+T坐标系系统，用于对时间和空间操纵的管理。通过指定从对象的本地坐标系到公共的全局3D+T坐标系（称为场景坐标系）的坐标转换，编码器或最终用户都可以将AV对象放置在场景中。MPEG-4的组合特性使在压缩域中执行比特流编辑和创作成为可能。

一个或多个AV对象（包括其时空关系）从编码器被传输到解码器。在编码器上，AV对象被压缩、错误保护、多路复用并向下游传输。在解码器上，这些对象被解复用、纠错、解压缩、合成并呈现给终端用户。终端用户有机会与演示文稿进行交互。交互信息可以在本地使用，也可以在上游传输到编码器。

传输的流可以是包含连接设置、配置文件（编码工具的子集）和类定义信息的控制流，也可以是包含所有其他信息的数据流。控制信息至关重要，因此必须通过可靠的渠道进行传输；但是数据流可以通过具有不同服务质量的各种通道进行传输。

标准的第2部分涉及视频压缩。随着支持各种配置文件和级别的需求不断增长，引入了该标准的第10部分来处理这种需求，该需求很快在业界变得比第2部分更加重要和司空见惯。但是，MPEG-4第10部分可以看作是独立的。标准化工作，因为它不提供与MPEG-4第2部分类似的向后兼容性。下一节将讨论MPEG-4第10部分，也称为高级视频编码（AVC）。

MPEG-4第2部分是基于对象的自然和合成混合编码标准。（为简单起见，在下面的讨论中，我们将MPEG-4第2部分简称为MPEG-4。）MPEG-4视频的结构本质上是分层的。在顶层是由一个或多个视频对象（VO）组成的视频会话（VS）。一个VO可以由一个或多个视频对象层（VOL）组成。每个VOL包含一个称为视频对象平面（VOP）的有序时间快照序列。视频对象平面组（GOV）层是VOL和VOP层之间的可选层。比特流可以具有任意数量的GOV标头，而GOV标头的频率是一个编码器问题。由于GOV标头指示绝对时间，因此它可用于随机访问和错误恢复。

视频编码器由许多编码器和相应的解码器组成，每一个都专用于一个单独的视频对象。重建的视频对象被合成在一起并呈现给用户。用户与对象的交互（例如缩放，拖动和链接）可以在编码器或解码器中进行处理。

为了描述任意形状的VOP，MPEG-4通过称为VOP窗口的边界矩形定义了VOP。视频对象由最紧密的VOP窗口限制，以便对最少数量的图像宏块进行编码。每个VO包含三个主要功能：形状编码，运动补偿和纹理编码。在矩形VOP的情况下，MPEG-4编码器的结构类似于MPEG-2编码器的结构，并且可以跳过形状编码。图3-9显示了视频对象编码器的结构。

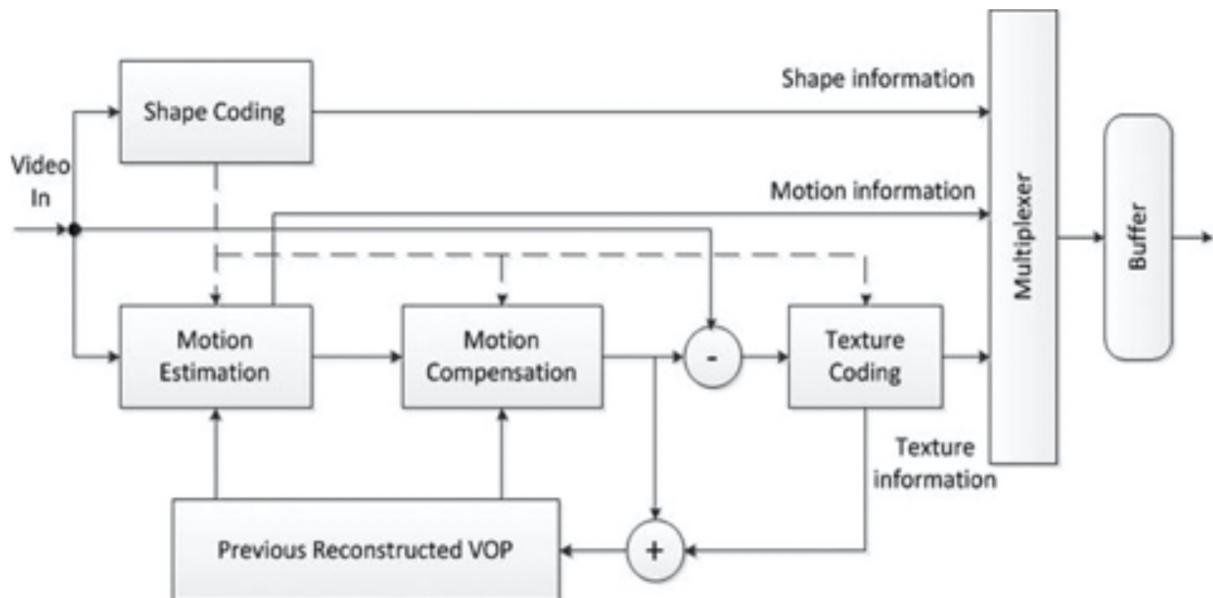


图3-9.MPEG-4 视频对象编码器结构

VOP的形状信息在MPEG-4中称为alpha平面。Alpha平面与亮度的格式相同，并且无论像素是否在视频对象内，其数据都表示相关像素的特性。形状编码器压缩alpha平面。二进制alpha平面通过改进的基于内容的编码算法（CAE）进行编码，而灰度alpha平面则通过类似于纹理编码的运动补偿DCT转换进行编码。完全位于对象外部的宏块（透明宏块）不进行运动或纹理编码；因此，不需要任何开销来表示该模式，因为可以从形状编码中获得该透明度信息。

运动估计和补偿用于减少时间冗余。填充技术应用于参考VOP，该技术允许多边形匹配替代矩形图像的块匹配。填充方法旨在通过填充与信号外推相对应的丢失数据，将任意形状的图像段扩展到规则的块网格，从而可以应用常见的基于块的编码技术。除了基本的运动补偿技术外，MPEG-4视频还支持无限制的运动补偿、高级预测模式和双向运动补偿，从而以增加非常少的复杂性为代价获得了质量上的显着改善。

VOP运动补偿后的内部数据和残差数据，使用类似于以往标准的基于块的DCT方案进行编码。VOP内的宏块使用与H.263相同的技术进行编码。轮廓宏块（即具有对象边缘的宏块）内VOP外部的区域可以以常规DCT变换进行填充，也可以使用形状自适应DCT（SA-DCT）变换进行填充。透明块将被跳过，并且不会在比特流中进行编码。

MPEG-4支持在空域和时域对视频对象进行可伸缩编码，并提供跨媒体的错误恢复能力。视频包同步重试、数据分区、标头扩展编码和可逆VLC这四个主要工具提供了丢失恢复能力，例如重新同步、错误检测、数据恢复和错误隐藏。

AVC编码

高级视频编码（AVC），也称为ITU-T H.264标准（ISO/IEC 14496-10），目前是业界用于视频记录和分发的最常见的视频压缩格式。它也被称为MPEG-4第10部分。AVC标准由ITU-T视频编码专家组（VCEG）和ISO/IEC运动图像专家组（MPEG）标准化组织的联合视频小组（JVT）于2003年批准。AVC标准之所以广为人知的原因之一是，它是蓝光的三种压缩标准之一（其他是MPEG-2和VC-1），并且还被YouTube和iTunes等互联网流应用、Flash Player等软件应用程序、Silverlight等软件框架以及在地面、有线和卫星频道上播放的各种HDTV广播等广泛应用。

AVC视频编码标准与之前的MPEG-4 Part 2, MPEG-2, H.263, MPEG-1和H.261标准具有相同的基本功能元素。它使用有损预测、基于块的混合DPCM编码技术。其需要减少空间相关性的变换、用于码率控制的量化、用于减少时间相关性的运动补偿预测、以及用于减少统计相关性的熵编码。但是，为了实现比以前的标准更好的编码性能，AVC合并了每个功能元素的细节更改，包括图片内预测、新的 4×4 变换、多个参考图片、可变块大小和运动补偿的四分之一像素精度、去滤波器和改善熵编码。AVC还引入了诸如通用B切片之类的编码概念，该概念不仅支持双向前后预测对，而且还支持向前和向后预测对。AVC还定义了其他几种工具，包括直接模式和加权预测，以获得对源信号的非常好的预测，从而使误差信号的能量最小。这些工具可帮助AVC在各种应用程序方面的性能明显优于先前的标准。例如，与MPEG-2相比，AVC通常以一半的码率获得相同的质量，特别是对于以高码率编码的高分辨率内容。

但是，提高编码效率是以牺牲编码器和解码器的额外复杂性为代价的。因此，作为补偿，AVC利用一些方法来降低实现的复杂性，例如，将用于变换和量化的乘法运算相结合，引入了无乘法器整数变换。此外，为了促进在嘈杂的信道条件和容易出错的环境（例如无线网络）上的应用，AVC利用一些方法对网络噪声的进行错误恢复。其中包括灵活的宏块排序（FMO）、交换片、冗余片方法和数据分区。

编码的AVC比特流具有两层，网络抽象层（NAL）和视频编码层（VCL）。NAL提取VCL数据以便在各种通信通道或存储媒体上传输。NAL单元同时指定字节流和基于数据包的格式。字节流格式为应用程序定义了唯一的开始代码，这些应用程序将NAL单元流封装在网络数据包（如MPEG-2传输流）中，作为字节或位的有序流传输。先前的标准在每个元素的开头都包含有关切片、帧和序列的标头信息，在有损环境中丢失这些关键元素会使其余元素数据变得无用。AVC通过将序列和图片参数设置保持在非VCL NAL单元中，使用更加强大的错误保护进行传输来解决此问题。VCL单元包含核心视频编码数据，包括视频序列、帧、切片和宏块。

配置文件与级别（Profile and Level）

配置文件是编码算法的一组功能，通过标识这些配置以满足应用程序的某些要求。这意味着编码算法的某些功能在一些配置下是不支持的。该标准针对特定类别的应用程序定义了21组功能。

对于不可缩放的二维视频应用，以下是必要的配置：

- 受约束的基准配置：针对低成本的移动和视频通信应用程序，受约束的基准配置使用与基准配置、主要配置和高级配置相同的功能子集。
- 基准配置：此配置针对需要额外的错误恢复能力的低成本应用程序。因此，除了“受约束的基准配置”所支持的功能之外，它还具有三个功能来增强鲁棒性。但实际上，“受约束的基准配置”比“基准配置”更常用。这两个配置的比特流共享相同的配置标识符代码值。
- 扩展配置：这是用于视频流。与基准配置相比，它具有更高的压缩能力和鲁棒性，并且支持服务器流切换。
- 主要配置：主要配置文件用于标清数字电视广播，但不用于HDTV，因为HDTV主要使用高级配置。
- 高级配置：这是HDTV广播和光盘存储（例如蓝光光盘存储格式）的主要配置。
- 渐进式高级配置：此配置与高级配置类似，不同之处在于它不支持现场编码工具。它适用于使用逐行扫描视频的应用程序和视频展示。
- 高级10配置配置：主要用于每个样本解码图像精度为10位的高级内容，此配置相比较高级配置，添加了10位的精度支持。
- 高级4:2:2配置：此配置针对使用隔行视频的专业应用程序。在High 10配置的之外，它增加了对4:2:2色度子采样格式的支持。
- 高级4:4:4预测配置：除了High 4:2:2配置之外，此配置还支持高达4:4:4色度采样，每个采样精度高达14位。它还支持无损区域编码，并将每个图片编码为三个单独的色彩平面。

除了上述配置之外，可伸缩视频编码（SVC）扩展定义了另外五个可伸缩配置：可伸缩受约束基准配置、可伸缩基准配置、可伸缩高级配置、可伸缩受约束高级配置和可伸缩高级内部配置。同时，多视图编码（MVC）扩展为三维视频添加了三个配置，即“立体声高级配置”，“多视图高级配置”和“多视图深度高级配置”。此外，为专业编辑应用程序定义了四个帧内配置：高级10帧内配置，高级4:2:2帧内配置，高级4:4:4帧内配置和CAVLC 4:4:4帧内配置。

级别是用于指定配置所需的解码器性能程度的约束；例如，一个级别指定解码器必须遵守的最大图片分辨率、比特率、帧速率等。表3-2给出了一些级别限制的示例。有关完整说明，请参见标准规范²。

表3-2.AVC级别限制示例

级别	每秒最大亮度采样次数	最大宏块数	最大数量	最大视频码率, kbps (基准配置、扩展配置、主要配置)	最大视频码率, kbps (高级配置)	帧采样率 (
1	380,160	1,485	99	64	80	176x144@
1.1	768,000	3,000	396	192	240	320x240@
1.2	1,536,000	6000	396	384	480	352x288@
1.3	3,041,280	11,880	396	768	960	352x288@
2	3,041,280	11,880	396	2,000	2,500	352x288@
2.1	5,068,800	19,800	792	4,000	5,000	352x576@
2.2	5,184,000	20,250	1,620	4,000	5,000	720x480@
3	10,368,000	40,500	1620	10,000	12,500	720x480@
3.1	27,648,000	108,000	3,600	14,000	17,500	1280x720@
3.2	55,296,000	216,000	5,120	20,000	25,000	1280x720@
4	62,914,560	245,760	8,192	20,000	25,000	1280x1080
4.1	62,914,560	245,760	8,192	50,000	62,500	2048x1024
4.2	133,693,440	522,240	8,704	50,000	62,500	2048x1080
5	150,994,944	589,824	22,080	135,000	168,750	2560x1920
5.1	251,658,240	983,040	36,864	24,000	300,000	4096x2048
5.2	530,841,600	2,073,600	36,864	24,000	300,000	4096x2160

帧结构 (Picture Structure)

视频序列含有帧图片或现场图片。图片通常包括三个样本阵列，一个亮度和两个色度样本阵列（仅在高级4:4:4配置中支持RGB阵列）。AVC支持逐行扫描或隔行扫描，它们可以按相同顺序混合。基准配置仅限于渐进式扫描。

图片分为多个切片。切片是图片内宏块的序列，宏块数量灵活多变。多个切片可以形成切片组；存在宏块到切片组的映射，以确定哪个切片组包括特定的宏块。在4:2:0格式中，每个宏块具有一个 16×16 亮度和两个 8×8 色度样本阵列，而在4:2:2和4:4:4格式中，色度样本阵列为 8×16 和 16×16 。图片可以被划分为具有各种形状的 16×16 或更小的分区，例如 16×8 、 8×16 、 8×8 、 8×4 、 4×8 和 4×4 。这些分区用于预测目的。图3-10显示了不同的分区。

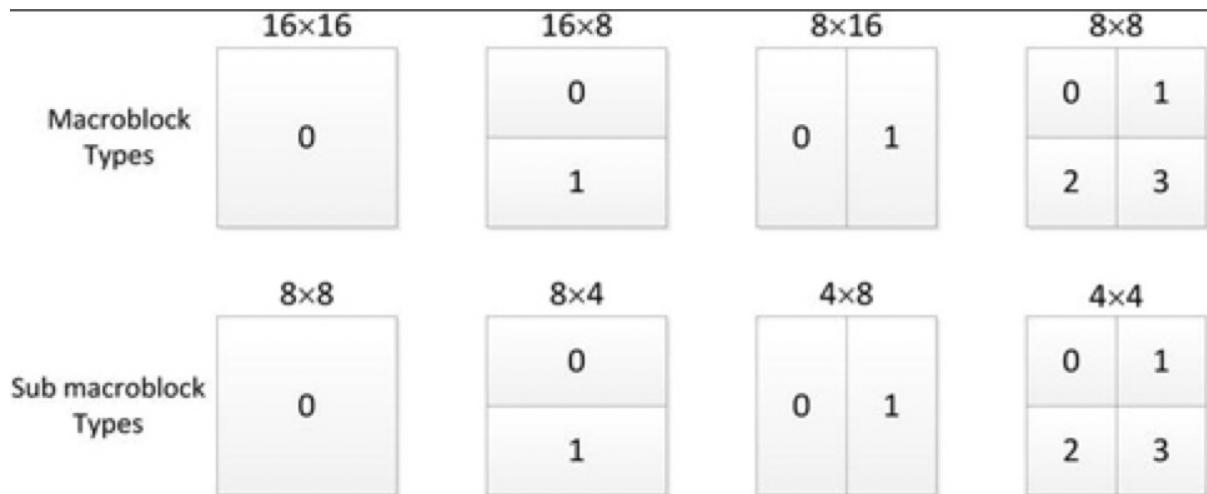


图3-10.AVC宏块及块分区

编码算法

在AVC算法中，编码器可以为每个图片的各个分区选择帧内编码，还是帧间编码。帧内编码（I）在比特流中提供随机访问点，解码可以从随机访问点开始并正常进行。帧内编码使用各种空间预测模式来减少图片内的空间冗余。另外，AVC定义了帧间编码，该帧间编码使用运动矢量进行基于块的画面间预测以减少时间冗余。帧间编码有两种类型：预测（P）和双向预测（B）。因为帧间编码使用像素块的帧间预测，而这些像素块与前序解码过的视频帧相关，因此帧间编码具有更好的编码效率。预测结果是从被用作预测参考的先前重建图片的解块版本中获得的。使用解块滤波器是为了减少块边界处的块效应。可以为图片中的各种块大小指定运动矢量和帧内预测模式。通过对预测残差进行变换以在量化之前消除块中的空间相关性，可以实现进一步的压缩。帧内预测模式、运动矢量和量化的变换系数信息使用熵编码进行编码，例如上下文自适应可变长度编码（CAVLC）或上下文自适应二进制算术编码（CABAC）。如图3-11所示的AVC编码算法的框图，显示了编码器和解码器模块。

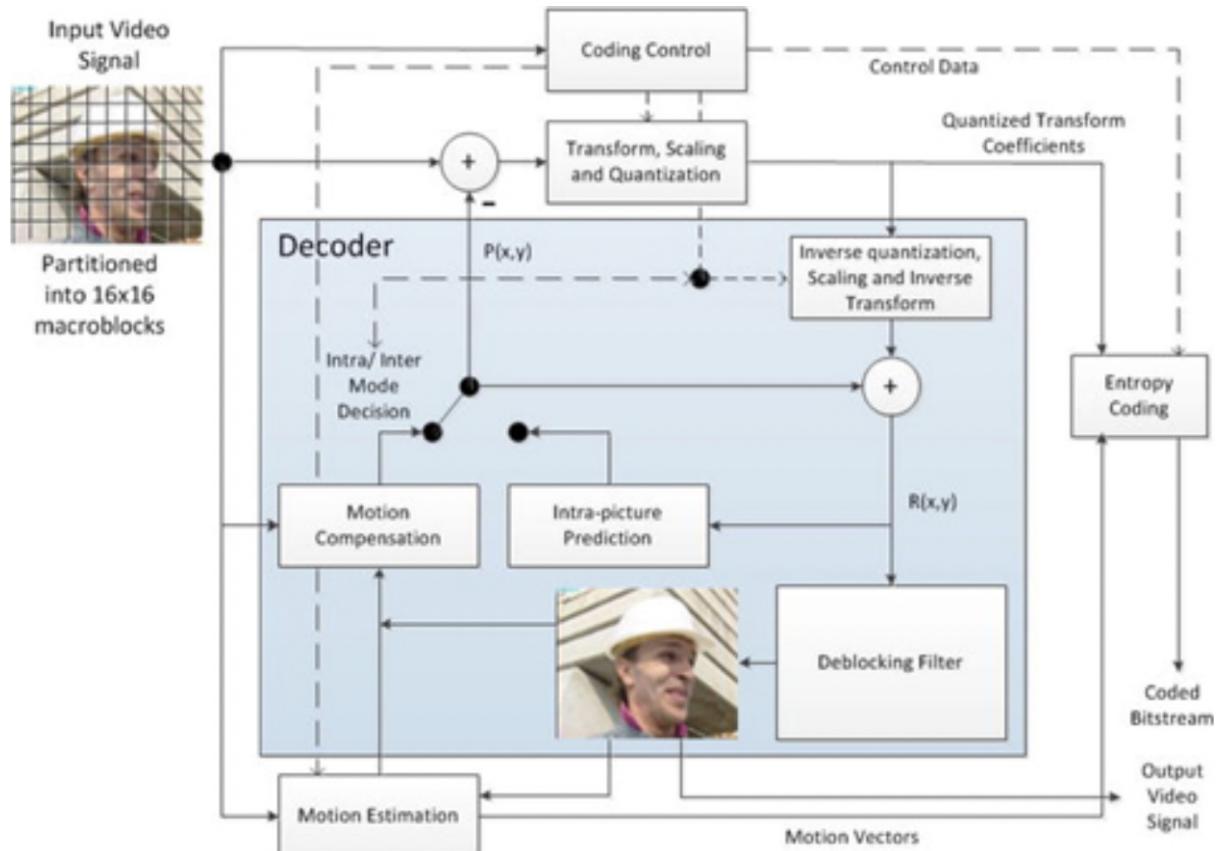


图3-11.AVC编码算法框图

帧内预测

在以往的标准中，对帧内编码的宏块都是独立编码的，没有任何参考，并且对于要预测的宏块没有一个好的预测方法，因此必须使用帧内编码的宏块。由于帧内宏块使用的位数多于预测所用的位数，因此压缩效率通常较低。为了缓解该问题（即减少编码帧内图片所需的位数），AVC引入了帧内预测，从而根据属于同一图片的先前重建的块来形成预测块。与对当前块本身进行编码相比，需要更少的位来对当前块和预测块之间的残余信号进行编码。

亮度样本的帧内预测块的大小可以是 4×4 、 8×8 或 16×16 。存在几种帧内预测模式，从中选择一种模式并在比特流中进行编码。AVC总共为 4×4 和 8×8 亮度块定义了9种帧内预测模式，为 16×16 亮度块定义了4种模式，为每个色度块定义了4种模式。图3-12示出了用于 4×4 块的帧内预测模式的示例。在此示例中， $[a, b, \dots, p]$ 是当前块的预测样本，其是从已经解码的左块和上块利用样本 $[A, B, \dots, M]$ 来预测的；箭头表示预测的方向，每个方向都表示为编码比特流中的一种帧内预测模式。对于模式0（垂直模式），通过外推上边的样本（即 $[A, B, C, D]$ ）来形成预测。类似地，对于模式1（水平模式），外推左边的样本 $[I, J, K, L]$ 。对于模式2（DC预测模式），将上面和左边样本的平均值用作预测。

对于模式3（向左斜下模式），模式4（向右斜下模式），模式5（向右模式），模式6（水平向下模式），模式7（向左垂直模式）和模式8（向左上水平模式），预测样本由预测样本A至M的加权平均值形成。

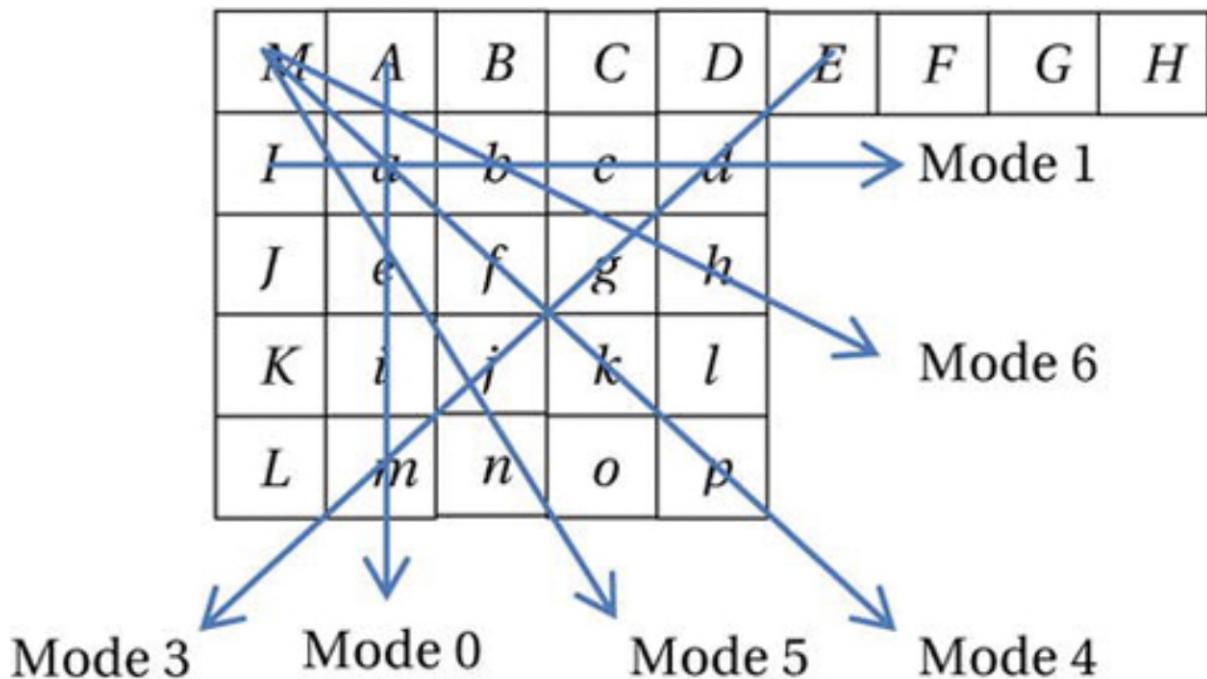


图3-12.4×4的帧内预测模式示例（仅显示了几种模式作为示例）

帧间预测

帧间预测通过使用运动估计和运动补偿来减少时间相关性。如之前所述，对于这样的预测，AVC将图片分成从 16×16 到 4×4 的几种形状。尽管对于较小的分区，运动矢量和信令分区类型会产生额外比特开销，但运动补偿仍会导致残留信号中的信息减少。

帧内预测可以应用于小至 4×4 的亮度样本块，具有高达四分之一像素的运动矢量精度。与单独使用整数像素相比，亚像素运动补偿可提供更好的压缩效率；虽然四分之一像素优于半像素，但它涉及更复杂的计算。对于亮度，首先生成半像素样本，并使用权重为 $(1, -5, 20, 20, -5, 1) / 32$ 的六抽头有限脉冲响应（FIR）滤波器从相邻的整数像素样本进行内插。利用可用的半像素样本，使用相邻的半像素或整数像素样本之间的双线性插值来生成四分之一像素样本。对于4:2:0色度，八分之一像素样本对应于四分之一像素亮度，并且是从整数像素色度样本的线性插值获得的。相对于来自相邻运动向量的预测，亚像素运动向量被差分编码。图3-13显示了相对于完整像素的亚像素预测的位置。

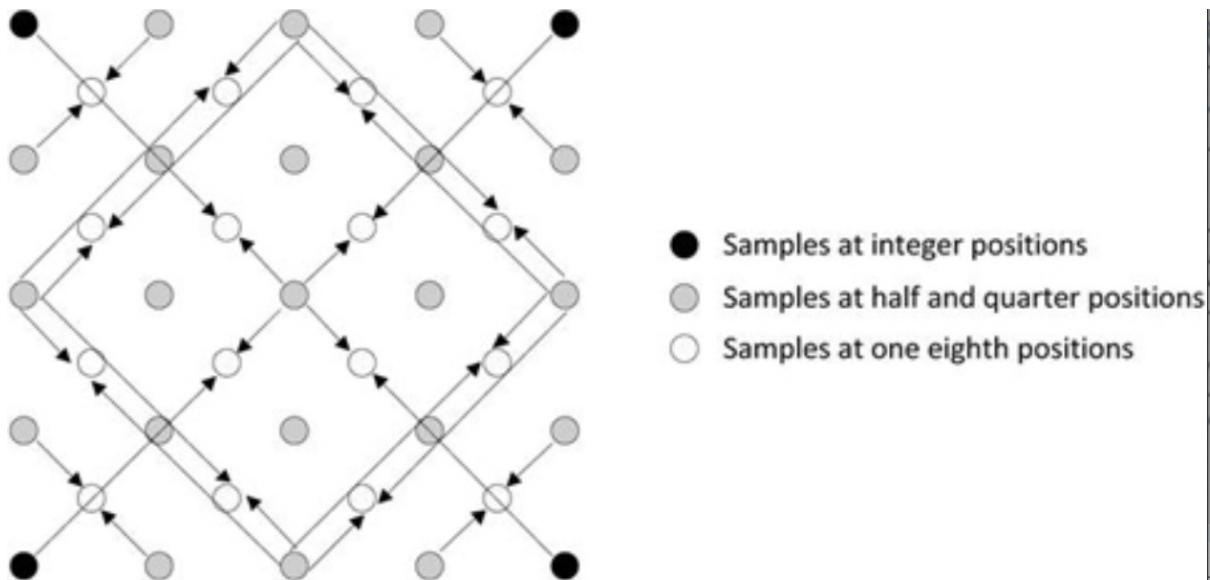


图3-13.亚像素预测的位置

转换和量化

AVC算法使用基于块的转换来消除空间冗余，因为来自帧内或帧间预测的残差信号被分为 4×4 或 8×8 块（仅限高级配置），在对它们进行量化之前将其转换为转换域。与之前使用固定 8×8 DCT转换的标准相比，在AVC中使用 4×4 整数变换可以减少振铃失真。同样，在这个较小的尺寸上不需要乘法。AVC引入了分层转换结构的概念，其中将相邻的 4×4 亮度变换的DC分量组合在一起以形成 4×4 块，然后使用哈达玛积（Hadamard）变换再次对其进行变换，以进一步提高压缩效率。

AVC中的 4×4 和 8×8 变换都是基于DCT的整数变换。整数变换、后缩放和量化在编码器中是组合在一起的；而在解码器中，转换序列是逆量化、预缩放和逆整数变换。为了更深入地了解该过程，请参考下面的矩阵H。

$$H = \begin{bmatrix} a & a & a & a \\ b & c & -c & -b \\ a & -a & -a & a \\ c & -b & b & -c \end{bmatrix}$$

使用此矩阵和公式 $X = HFH^T$ 可以完成 4×4 DCT转换，其中HT是矩阵H的转置，F是输入 4×4 数据块，X是得到的 4×4 变换后的结果块。对于DCT，变量a, b和c如下：

$$a = \frac{1}{2}, b = \sqrt{\frac{1}{2}} \cos\left(\frac{\pi}{8}\right), c = \sqrt{\frac{1}{2}} \cos\left(\frac{3\pi}{8}\right)$$

AVC算法通过估算来简化这些系数，并通过使用以下方法保持正交性：

$$a = \frac{1}{2}, b = \sqrt{\frac{1}{2}}, c = \frac{1}{2}$$

通过使用缩放变换 $X = \hat{H}F\hat{H}^T \otimes SF$, 将变换与量化步骤结合起来, 可以进一步简化运算以避免乘法, 其中,

$$\hat{H} = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 2 & 1 & -1 & -2 \\ 1 & -1 & -1 & 1 \\ 1 & -2 & 2 & -1 \end{bmatrix}, \text{ and } SF = \begin{bmatrix} a^2 & \frac{ab}{2} & a^2 & \frac{ab}{2} \\ \frac{ab}{2} & \frac{b^2}{4} & \frac{ab}{2} & \frac{b^2}{4} \\ a^2 & \frac{ab}{2} & a^2 & \frac{ab}{2} \\ \frac{ab}{2} & \frac{b^2}{4} & \frac{ab}{2} & \frac{b^2}{4} \end{bmatrix}$$

SF是一个 4×4 矩阵, 代表正交性所需的比例因子, 而\bigotimes 表示逐元素相乘。使用52个可用量化器级别之一 (也称为量化步长, Qstep) , 通过适当的量化, 获得具有分量 $Y_{i,j}$ 的经过变换和量化的信号Y:

$$Y_{i,j} = X_{i,j} \text{round} \left(\frac{SF_{ij}}{Qstep} \right), \text{ where } 0 \leq i, j \leq 3$$

在解码器中, 使用Qstep和SF作为逆量化和一部分逆变换来缩放接收信号Y, 以获得具有分量 $X'_{i,j}$ 的逆变换块X':

$$X'_{i,j} = Y_{i,j} Qstep SF^{-1}_{ij}, \text{ where } 0 \leq i, j \leq 3$$

4x4重构块为: $F' = \hat{H}_Y^T X' \hat{H}_Y$, 其中, 证书逆置矩阵为:

$$\hat{H}_Y = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & \frac{1}{2} & \frac{-1}{2} & -1 \\ 1 & -2 & -1 & 1 \\ \frac{1}{2} & -1 & 1 & -\frac{1}{2} \end{bmatrix}$$

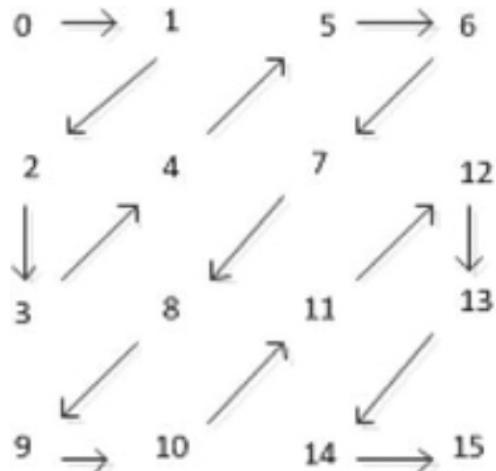
此外, 在用于 16×16 帧内模式的分层变换方法中, 使用Hadamard变换对 4×4 亮度帧内DC系数进行进一步变换:

$$\tilde{H} = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 \\ 1 & -1 & 1 & -1 \end{bmatrix}$$

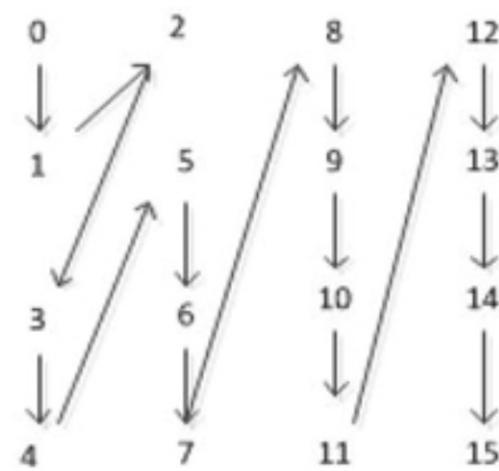
在4:2:0颜色采样中, 对于色度DC系数, 变换矩阵如下:

$$\tilde{H} = \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}$$

为了增加游程编码提供的压缩增益, 定义了两个扫描顺序以在熵编码之前排列量化系数, 即之字形扫描和场扫描, 如图3-14所示。之字形扫描适用于逐行扫描的信号源, 而交替场扫描则有助于隔行扫描的内容。



Zigzag scan



Alternate field scan

图3-14.4×4块的之字形扫描和交替场扫描顺序

熵编码

早期的标准使用固定的可变长度编码表来提供熵编码，这些表是由基于一组通用视频的概率分布标准预先定义的，无法为特定视频源来优化霍夫曼表。相反，AVC使用不同的VLC根据上下文特征为每个源符号找到更合适的编码。除残差数据外的语法元素使用指数哥伦布码进行编码。残余数据通过之字形扫描或交替场扫描进行重新排列，然后使用上下文自适应可变长度编码（CAVLC）进行编码，或者对于主要配置和高级配置，可选择使用上下文自适应二进制算术编码（CABAC）。与CAVLC相比，CABAC以更高的复杂性为代价提供了更高的编码效率。

CABAC使用自适应二进制算术编码器，在编码每个码元之后更新概率估计，并因此适应于上下文。CABAC熵编码包含三个主要步骤：

- 二进制化：在算术编码之前，将非二进制符号（例如，变换系数或运动矢量）唯一地映射到二进制序列。这种映射类似于将数据符号转换为可变长度代码，但是在这种情况下，二进制代码在传输之前由算术编码器进一步编码。
- 上下文建模：基于先前编码的语法元素，选择称为上下文模型的二进制化符号概率模型。
- 二进制算术编码：在这一步骤中，算术编码器根据所选的上下文模型对每个元素进行编码，然后更新模型。

灵活的隔行编码

为了提供增强的隔行编码能力，AVC支持宏块自适应帧场（MBAFF）编码和图片自适应帧场（PAFF）编码技术。在MBAFF中，宏块对结构用于编码为帧的图片，从而允许在场模式下使用 16×16 宏块。这与MPEG-2相反，在MPEG-2中，帧编码图片中的场模式处理只能支持 16×8 个半宏块。如果使用PAFF，则可以将编码为完整帧的图像，与包含这些单独的单个场的合并场进行图像混合。

环内解块

由于帧内和帧间预测编码中基于块的变换以及变换系数的量化，特别是对于更高的量化尺度，产生了可见且令人讨厌的阻塞伪像。为了减轻在块边界处的此类伪像，AVC提供了解块滤波器，这也防止累积的编码噪声传播。

解块过滤器并不是新事物；它是作为可选工具在H.261中引入的，并且在减少编码噪声的时间传播方面取得了一些成功，因为仅运动补偿中的整数像素精度不足以减少这种噪声。然而，在MPEG-1和MPEG-2中，由于其高复杂度而未使用去块滤波器。相反，半像素精确运动补偿（通过对整数像素样本进行双线性滤波获得半像素）起到了平滑编码噪声的作用。

尽管复杂，但AVC还是使用了去块滤波器来获得更高的编码效率。作为预测环路的一部分，其从预测图片中消除了块状伪影，获得了更接近的预测，从而导致了能量误差信号的减少。去块滤波器应用于 4×4 块的水平或垂直边缘。亮度滤波在四个16样本边缘上执行，色度滤波在两个8样本边缘上执行。图3-15显示了解块边界。

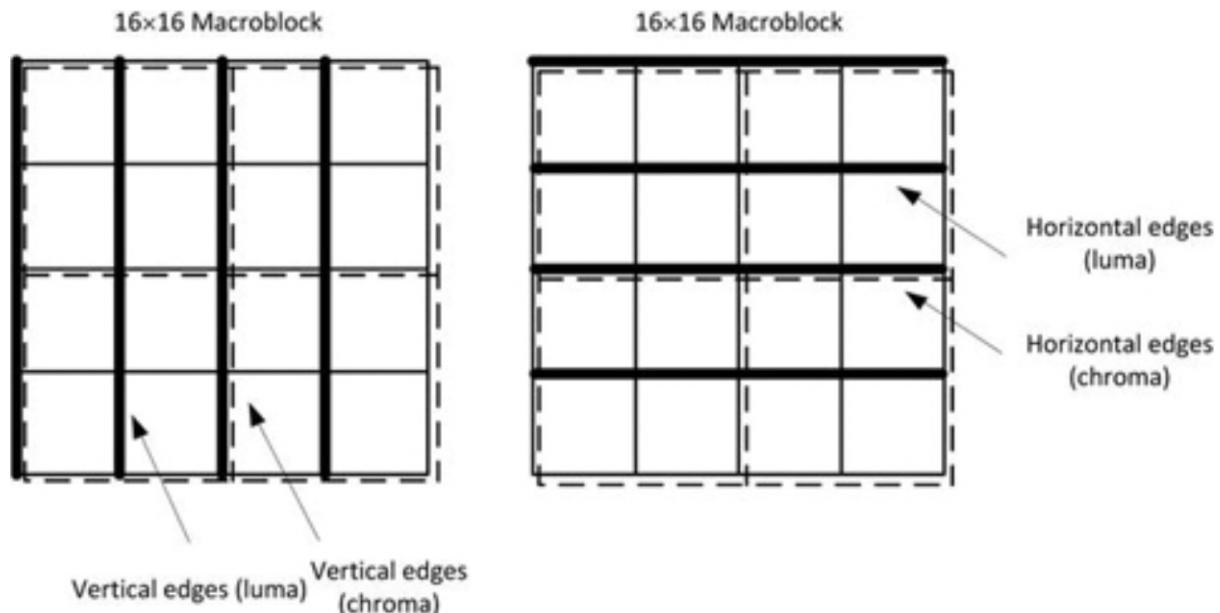


图3-15.在宏块中沿垂直和水平边界进行解块

容错

AVC提供了增强信道错误恢复能力的功能，其中包括NAL单元、冗余片、数据分区、灵活的宏块排序等。其中一些功能如下：

- 网络抽象层（NAL）：通过定义NAL单元，AVC允许在许多网络环境中使用相同的视频语法。在以往的标准中，标头信息是语法元素的一部分，因此，在包含标头的单个数据包接收到出错的情况下，整个语法元素将变得无用。相反，在AVC中，通过从媒体流中解耦与多个切片相关的信息来生成自包含数据包。高级关键参数（即序列参数集（SPS）和图片参数集（PPS））以具有更高级别错误保护的NAL单元保存。在整个编码视频序列中，有效的SPS保持不变，而有效的PPS在编码的图片中保持不变。
- 灵活的宏块排序（FMO）：FMO也称为切片组。与任意切片排序（ASO）一起，该技术对图片中的宏块进行了重新排序，因此丢失数据包不会影响整个图片。丢失的宏块可以通过从相邻的重构宏块进行插值来重新生成。
- 数据分区（DP）：此功能可根据其重要性将语法元素分离为不同的数据包。它使应用程序具有不平等的错误保护（UEP）。
- 冗余切片（RS）：这是AVC中的错误恢复功能，允许编码器发送切片数据的额外表示形式，通常以较低的保真度发送。万一主要数据片因通道错误而丢失或损坏，可以使用此表示形式。

². ITU-T Rec. H.264: Advanced Video Coding for Generic Audiovisual Services (Geneva, Switzerland: International Telecommunications Union, 2007). ↪

HEVC

高效视频编码（HEVC）或H.265标准（ISO/IEC 23008-2）是ITU-T视频编码专家组（VCEG）和ISO/IEC运动图像专家组（MPEG）标准化组织在2013年批准的视频编码标准。它遵循先前称为AVC或H.264的标准，该标准也由同一MPEG和VCEG视频编码联合协作小组（JCT-VC）定义，其目标是解决越来越流行的高分辨率视频，例如高清（HD， 1920×1080 ），超高清（UHD， $4k\times2k$ ）及以上。特别是，HEVC解决了两个关键问题：提高视频分辨率和增加并行处理体系结构的使用。因此，HEVC算法的设计目标是实现比AVC高两倍的压缩效率。

图片分割与结构

在较早的标准中，宏块是基本的编码构建块，它包含一个 16×16 亮度块，通常是两个 8×8 色度块，用于4:2:0色彩采样。在HEVC中，类似的结构是编码树单元（CTU），也称为最大编码单元（LCU），其中包含亮度编码树块（CTB）、相应的色度CTB和语法元素。在CTU中，亮度块大小可以为 16×16 、 32×32 或 64×64 ，在位比特流序列参数集中指定。可以使用树结构和四叉树信令将CTU进一步划分为较小的正方形块。

四叉树指定了编码单位（CU），编码单元构成预测和变换的基础。遍历编码树块中的编码单元并以Z顺序对其进行编码。图3-16显示了在 64×64 CTB中的排序示例。

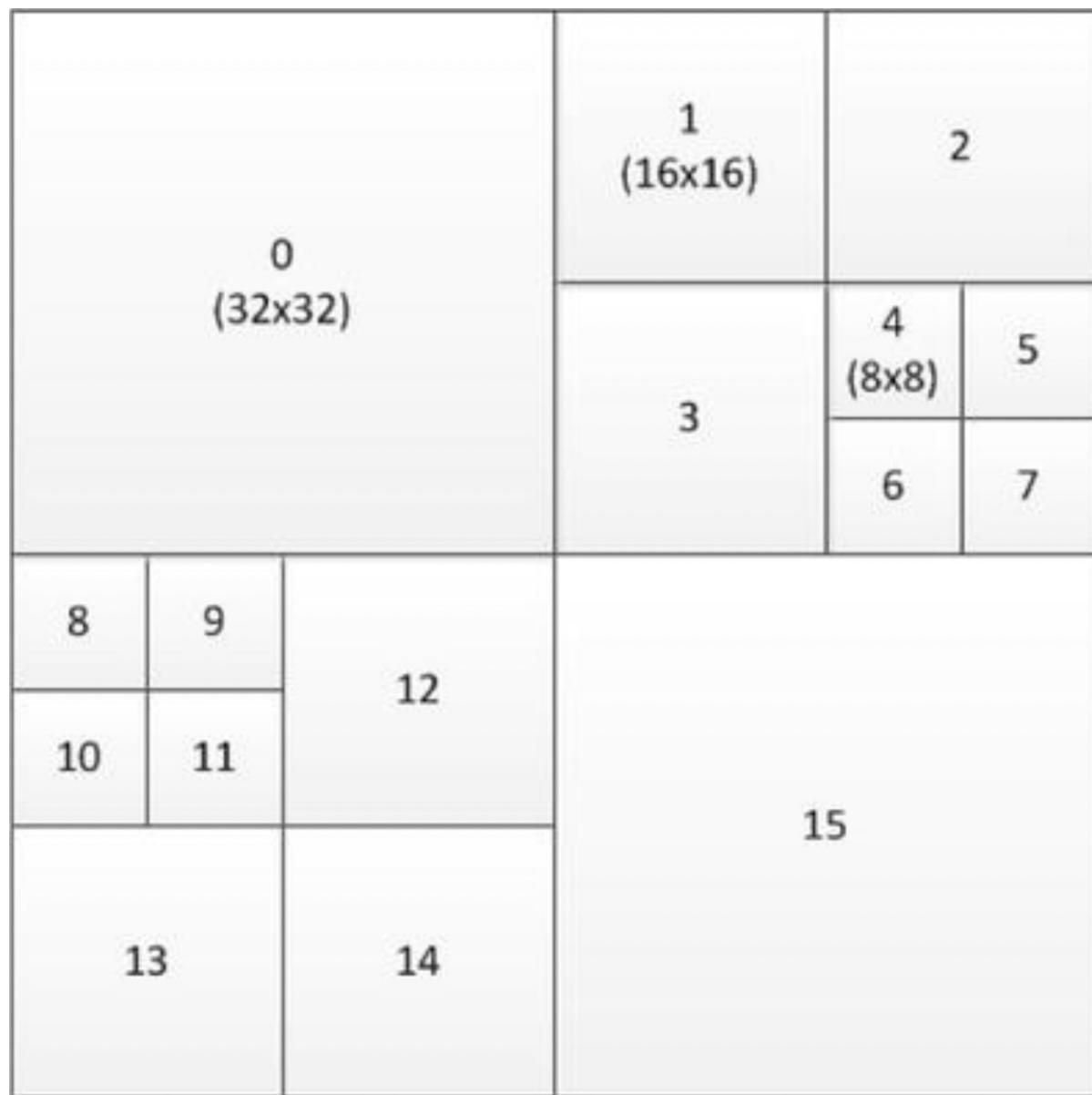


图3-16.64×64编码树块中编码单元的排序示例

一个编码单元具有一个亮度和两个色度编码块（CB），根据预测类型，它们可以进一步拆分大小并可以从相应的预测块（PB）进行预测。HEVC支持可变的PB大小，范围从64×64到4×4样本。使用变换单元（TU）树结构对预测残差进行编码。亮度或色度编码块残差可以与对应的变换块（TB）相同，也可以进一步分成较小的变换块。变换块只能具有4×4、8×8、16×16和32×32的正方形大小。对于画面内预测残差的4×4变换，除了基于常规DCT的整数变换之外，还指定基于离散正弦变换（DST）形式的整数变换。通常认为这种四叉树结构是HEVC相对于AVC的编码效率增益的最大贡献者。

HEVC简化了编码，并且不支持任何隔行扫描工具，因为隔行扫描不再用于显示器中，并且隔行视频在分发中变得越来越不常见。但是，隔行视频仍可以编码为一系列的现场图片。HEVC中提供了元数据语法，以使编码器可以通过编码以下内容之一来指示隔行扫描的视频已发送：

- 隔行扫描视频的每个场（即每个视频帧的偶数或奇数行）作为单独的图片
- 每个隔行扫描帧作为一个HEVC编码图片

这提供了一种对隔行视频进行编码的有效方法，而不会给解码器带来麻烦，因为该解码器需要支持特殊的解码过程。

配置文件和级别

HEVC定义了三个配置文件：“主要”配置文件，“主要10”配置文件和“静态图片”配置文件，其中“主”是目前最常用的配置文件。它需要4:2:0的颜色格式，并施加了一些限制；例如，位深度应为8，形成矩形图块的LCU组必须至少为 256×64 ，依此类推（在本章的稍后部分中，将对并行处理工具进行详细说明）。HEVC指定了许多级别，范围从1到6.2。6.2级别比特流在120fps时可支持 8192×4320 的高分辨率³。

帧内预测

除了平面和DC预测模式外，帧内预测还支持33种定向模式，而H.264/AVC中只有8种定向模式。图3-17显示了定向帧内预测模式。



图3-17. HEVC中的方向帧内预测模式

编码单元中的帧内预测正好遵循TU树，这样，当使用 $N \times N$ 分区模式对帧内编码单元进行编码时，会强制将TU树至少拆分一次，从而确保帧内编码单元与TU树之间的匹配。这意味着始终对 32×32 、 16×16 、 8×8 或 4×4 大小执行内部操作。与AVC相似，帧内预测需要两个一维数组，其中包含上、左相邻样本以及左上样本。数组的长度是内部块大小的两倍，在块的下面向右边延伸。图3-18显示了一个 8×8 块的示例数组。

140	132	131	139	133	150	152	167	168	167	171	189	190	167	178	177	178
139	136	132	138	142	149	137	170	149								
138	133	129	151	155	156	171	167	159								
135	129	130	153	159	150	157	169	172								
142	141	150	160	172	155	151	182	151								
147	149	137	170	149	146	132	140	142								
132	133	151	181	180	153	135	148	160								
135	150	157	189	172	149	145	153	163								
120	155	151	182	151	151	155	160	172								
120																
120																
120																
120																
120																
120																

图3-18.HEVC中的Luma内部样本和预测结构

帧间预测

对于帧间预测，HEVC提供了两个参考列表L0和L1，每个参考列表可容纳16个参考帧，其中最多八个图片是唯一的。这意味着一些参考帧将会出现重复。这有助于从相同的图片以不同的权重进行预测。

运动矢量预测

HEVC中的运动矢量预测非常复杂，因为它会建立候选运动矢量列表，并使用在比特流中编码的列表索引从列表中选择一个候选对象。运动矢量预测有两种模式：合并和高级运动矢量预测（AMVP）。对于每个预测单元（PU），编码器决定使用哪种模式，并在比特流中通过标记位来指示该模式。AMVP过程使用增量运动矢量编码，并且可以产生任何期望的运动矢量值。HEVC在16×16网格上对时间运动矢量进行二次采样。这意味着解码器只需要为16×16像素区域的时间运动矢量缓冲区中的两个运动矢量（L0和L1）分配空间。

运动补偿

HEVC以四分之一像素的粒度指定运动矢量，但对亮度使用8抽头滤波器，对色度使用4抽头八像素滤波器。这是对AVC中定义的六抽头亮度和双线性（两次抽头）色度滤镜的改进。由于八抽头滤波器的长度较长，因此需要为每个块读取所有侧面的三个或四个额外像素。例如，对于 8×4 的块，需要将 15×11 的像素区域读入存储器，而对于较小的块，其影响会更大。因此，HEVC将最小预测单位限制为单向且大于 4×4 。HEVC支持单向和双向PU的加权预测。但是，权重始终在分片标头中明确传输；与AVC不同，没有隐式加权预测。

熵编码

在HEVC中，熵编码是在CTU级别使用上下文自适应二进制算术代码（CABAC）执行的。HEVC中的CABAC算法在AVC的基础上进行了改进，并做了一些小的改进。上下文状态变量大约是AVC中的一半，并且初始化过程要简单得多。通过设计比特流语法，以便将旁路编码的模块尽可能地分组在一起。CABAC解码本质上是一个顺序操作。因此，很难并行化或快速硬件化。但是，可以一次解码多个旁路编码的模块。这与旁路模块分组一起，极大地促进了硬件解码器中的并行实现。

回路解块与样本自适应偏移

在HEVC中，可以将两个滤波器应用于重构后的像素值：环路解块（ILD）滤波器和样本自适应偏移（SAO）滤波器。可以选择将这两个过滤器之一或全部应用于图块边界和切片边界。HEVC中的环路解块滤波器与H.264/AVC相似，而SAO是一个新的滤波器，并在环路解块滤波器之后应用。

与在 4×4 网格边缘进行解块的AVC不同，HEVC仅在 8×8 网格上进行解块。图片中的所有垂直边缘都首先被解块，然后是所有水平边缘。滤波器本身与AVC中的滤波器相似，但是在HEVC中，仅支持边界强度为2、1、0的块。其边缘之间的距离为8像素，因此它们之间没有依赖性，从而实现了高度并行化的实现。例如，可以使用图片中每8像素列一个线程过滤垂直边缘。仅当对特定边缘的任一侧上的一个PU进行帧内编码时，色度才被解块。

作为解块后的辅助滤波器，SAO通过使用CTB级别的查找表来执行非线性幅度映射。它在CTB的每个像素上运行一次，每个 64×64 CTB总共运行6,144次。 $(64 \times 64 + 32 \times 32 + 32 \times 32 = 6144)$ 。对于每个CTB，在比特流中编码一个滤波器类型和四个偏移值，例如对于8位的视频，其范围从-7到7。编码器选择这些参数的目的是为了更好地匹配重建的图片和源图片。

并行处理语法和工具

HEVC标准中具有三个新功能，以支持增强的并行处理能力或出于打包目的而修改切片数据的结构。

碎片

有一个称为碎片的选项，可以将图片划分为矩形区域。碎片是图片的独立可解码区域，并使用一些共享的标头信息进行编码。碎片主要是指定用于提高并行处理能力，尽管也可以归因于某些抗错能力。碎片在图片和子图片级别提供粗粒度的并行性，并且不需要使用复杂的线程同步。切片的使用，需要减小行缓冲区的大小，这对于在缓存受限的硬件和较便宜的CPU上进行高分辨率视频解码而言是有利的。

波前并行处理

启用波前并行处理（WPP）时，会将碎片分为CTU行。第一行以常规方式处理；但是只有在第一行做出一些决定之后，才能开始第二行的处理。同样，只要在第二行中做出了一些决定，就可以开始第三行的处理，依此类推。每行的熵编码器的上下文模型可以从前行中推断出，并且处理延迟很小。WPP在碎片内提供细粒度的并行处理。与碎片相比，WPP通常提供更好的压缩效率，同时避免了因使用碎片而导致的潜在视觉伪影。

切片片段和相关切片

编码树块的序列称为切片。构成视频帧的图片可以分为任意数量的切片，或者整个图片可以只是一个切片。依次地，每个切片被分成一个或多个切片片段，每个切片片段以其自己的NAL单元。切片中只有第一个切片片段包含完整的切片标头，其余片段称为从属切片片段。这样，解码器必须能够访问第一切片片段以成功解码。切片的这种划分允许图片的低延迟传输，而无需支付由于许多切片头而原本会引起任何编码效率损失。例如，摄像机可以发出属于第一CTB行的切片片段，以便网络另一侧的播放设备可以在摄像机发出下一CTB行之前开始解码。这在诸如视频会议之类的低延迟应用中很有用。

³. ITU-T Rec. H.265: High Efficiency Video Coding (Geneva, Switzerland: International Telecommunications Union, 2013). ↪

视频质量国际标准

ITU-T和ITU-R视觉质量专家组已经指定了几种视频质量标准。尽管它们不是编码标准，但由于它们与本书的主题有关，因此值得一提。此外，由于IEEE标准1180与所有上述标准中使用的通用IDCT技术的计算精度有关，因此在此也进行了简要说明。

VQEG Standards

1997年，由ITU-T和ITU-R研究组组成的视频质量专家小组成立了视觉质量专家组（VQEG），以期推进视频质量评估领域。该小组研究了新的和先进的主观评估方法以及客观质量指标和测量技术。

VQEG采用了一种系统化的验证测试方法，通常包括几个视频数据库，需要使用这些视频数据库来预测主观视觉质量的客观模型，并且他们定义了执行客观模型验证的测试计划和程序。最初的标准由ITU-T第9研究组于2000年作为J.144建议书发布，但是所研究的各种方法都没有优于众所周知的峰值信噪比（PSNR）。2003年，发布了J.144的更新版本，其中推荐了四个客观模型用于有线电视服务。ITU-R第6研究组的镜像标准作为ITU-RBT.1683建议书发布，用于基带电视服务。

针对数字视频和多媒体应用的最新研究（称为多媒体第一阶段（MM-I）于2008年完成。MM-I测试集用于验证全参考（FR）、简化参考（RR）、无参考（NR）客观模型（将在第4章中详细讨论这些模型）。使用单刺激呈现方法和绝对类别评分（ACR）等级对主观视频质量进行评估。无需识别视频，对包含源视频的视频序列进行一次主观评估，并且根据国际电联五级质量等级，进行独立评估。计算平均意见分数（MOS）和差异平均意见分数（DMOS），其中DMOS是已处理视频的分数与源视频的分数相比的算术差别的平均值，以便删除任何隐藏的内容。用于控制和运行VQEG多媒体测试的软件称为AcrVQWin⁴。（主观质量评估方法和技术的详细信息已收录在ITU-T建议书P.910，P.912等，并在第4章将会进行讨论）

IEEE标准1180-1990

主要用于可视电话和类似应用中，重构环路使用了8x8逆离散余弦逆变换（IDCT）的结果。IEEE标准1180-1990⁵指定了8x8IDCT的数字特性。这些规范确保了IDCT的不同实现之间的兼容性。来自不同制造商的编码器和解码器中不同的IDCT实现可能会导致失配

错误；由于系统中的重建循环，失配误差可能会在视频持续时间内传播。IEEE标准不是将所有制造商限制在一个严格的实施范围内，而是允许在特定时间段内出现小的失配，同时使用帧内编码帧定期刷新视频，例如，对于ITU-T可视电话应用，持续时间是132帧。

标准规定失配误差应满足以下要求：

- 对于任何像素位置，峰值误差（ppe）的大小不得超过1。
- 对于任何像素位置，均方误差（pmse）均不得超过0.06。
- 总体而言，均方误差（omse）不得超过0.02。
- 对于任何像素位置，平均误差（pme）的大小不得超过0.015。
- 总的来说，平均误差（ome）的大小不得超过0.0015。
- 对于全零输入，建议的IDCT将生成全零输出。

⁴. K. Brunnstrom, D. Hands, F. Speranza, and A. Webster, "VQEG Validation and ITU Standardization of Objective Perceptual Video Quality Metrics," IEEE Signal Processing (May 2009):96-101; software available at www.acreo.se/acrvqwin. ↪

⁵. The IEEE Standard Board, IEEE Standard Specifications for the Implementations of 8x8 Inverse Discrete Cosine Transform (New York, USA: Institute of Electrical and Electronics Engineers, 1991). ↪

其他行业标准和格式概述

除了由ISO, ITU或电子电气工程师协会（IEEE）定义的国际标准外，还有业内众所周知的标准。下面描述了其中一些标准。

VC-1

众所周知的VC-1格式，最初是由微软开发的专有视频格式，在2006年由美国电影电视工程师协会（SMPTE）定义为SMPTE 421M视频编解码器标准。蓝光，当前已过时的HD-DVD，Microsoft Windows Media，Microsoft Silverlight框架，Microsoft X-Box 360和Sony PS 3视频游戏机以及各种基于Windows的视频应用程序均支持它。自第二代Intel (R) Core (TM) 处理器（2011）和Raspberry Pi（2012）起，Intel集成处理器图形中就可以使用VC-1格式的硬件解码。

VC-1使用类似于国际标准的常规基于DCT设计，并支持逐行和隔行视频。该规范定义了三个配置文件：简单，主要和高级，以及最多五个级别。表3-3中显示了每个配置文件支持的主要工具。

配置	等级	最大码率	分辨率 @ 帧率	工具集
简单	低	96	176x144 @ 15 (QCIF)	Baseline intra frame compression; variable sized, 16-bit and overlapped transform; four motion vectors per macroblock; quarter pixel luma motion compensation
简单	中	384	240x176 @ 30, 352x288 @ 15 (CIF)	
主要	低	2,000	320x240 @ 24 (QVGA)	In addition to Simple profile: Start codes; quarter-pixel chroma motion compensation; extended motion vectors; loop filter; adaptive quantization; B-frames; dynamic resolution change; intensity compensation; range adjustment
主要	中	10,000	720x480 @ 30(480p30), 720x576 @ 25(576p25)	
主要	高	20,000	1920x1080 @ 30(1080p30)	
高级	L0	2,000	352x288 @ 30 (CIF)	In addition to Main profile: GOP layer; field and frame coding modes; display metadata
高	L1	10,000	720x480 @ 30(NTSC-SD),	

级	L1	10,000	720x576 @ 25(PAL-SD)	
高级	L2	20,000	720x480 @ 60(480p60), 1280x720 @ 30(720p30)	
高级	L3	45,000	1920x1080 @ 24(1080p24), 1920x1080 @ 30(1080i30), 1280x720 @ 60(720p60)	
高级	L4	135,000	1920x1080 @ 60(1080p60), 2048x1536 @ 24	

VP8视频压缩技术

通过收购On2科技公司，Google成为VP8视频压缩格式的所有者。2011年11月，国际工程任务组（IETF）将VP8数据格式和解码指南发布为RFC 6386⁶⁶。谷歌还根据BSD许可证发布了VP8编解码器库软件libvpx。Opera, FireFox和Chrome浏览器以及各种基于硬件和软件的视频编解码器（包括Intel集成处理器图形硬件）目前支持VP8。

像许多现代视频压缩方案一样，VP8的基础是将帧分解为像素的方形子块，使用先前构造的块对此类子块进行预测，并使用离散余弦变换（DCT）调整这种预测，或者在一种特殊情况下，Walsh-Hadamard变换（WHT）。该系统旨在通过指定前一帧的视觉相似部分的位置来利用视频信号的时间相干性来降低数据速率，并利用DCT和WHT提供的频率隔离以及信号的容忍度来降低空间相干性。为了减轻保真度编解码器的损失，人类视觉系统使用三种不同的参考帧进行帧间预测：前序帧，黄金帧和altref帧，以提供时间可伸缩性。

VP8编解码器将数据分区应用于编码数据。每个编码的VP8帧分为两个或多个分区，包括未压缩部分，后跟压缩头信息和每个宏块信息，这些信息指定如何预测每个宏块。第一分区包含所有宏块的预测模式参数和运动矢量。其余分区均包含残差的量化DCT或WHT系数。每帧可以有一个，两个，四个或八个DCT/WHT分区，具体取决于编码器设置。可以在RFC 6386中找到该算法的详细信息。

谷歌提出了适用于传输使用VP8视频编解码器编码的视频流的RTP有效负载规范⁷⁷。RTP有效载荷格式可用于低比特率对等和高比特率视频会议应用程序中。RTP有效负载格式考虑了帧分区边界，以提高针对丢包的鲁棒性并促进错误隐藏。它还使用高级参考帧结构来实现有效的错误恢复和时间可伸缩性。此外，标记非参考帧是为了使服务器或支持媒体的网络能够根据需要丢弃适当的数据。

用于浏览器应用程序编程接口（API）的IETF国际草案标准，称为Web实时通信（WebRTC）⁸⁸，指定如果支持VP8，则使用双线性过滤器和无重建过滤器，帧速率至少为10帧每秒，并且必须支持从320×240到1280×720的分辨率。谷歌Chrome、Mozilla、FireFox和Opera浏览器支持WebRTC API，旨在用于基于浏览器的应用程序，包括视频聊天。谷歌Chrome操作系统也支持WebRTC。

⁶. J. Bankoski, J. Koleszar, L. Quillio, J. Salonen, P. Wilkins, and Y. Xu, “VP8 Data Format and Coding Guide, RFC 6386,” November 2011, retrieved from <http://datatracker.ietf.org/doc/rfc6386/>. ↪

⁷. P. Westin, H. Lundin, M. Glover, J. Uberti, and F. Galligan, “RTP Payload Format for VP8 Video,” Internet draft, February 2014, retrieved from <http://tools.ietf.org/html/draft-ietf-payload-vp8-11>. ↪

⁸. C. Bran, C. Jennings, J. M. Valin, “WebRTC Codec and Media Processing Requirement,” Internet draft, March 2012, retrieved from <http://tools.ietf.org/html/draft-cbran-rtcweb-codec-02>. ↩

VP9视频压缩技术

视频压缩标准VP9是VP8的后继产品，也是谷歌开发的开放标准。最新规范于2013年2月发布，目前可通过IETF的因特网草案⁹获得。最终规格尚未批准。VP9视频编解码器是专门为满足因特网上视频消费需求而开发的，包括专业和业余制作的视频点播和会话视频内容。WebM媒体容器格式主要通过使用VP9编解码器来替换HTML5视频的免版税，开放视频压缩，该VP9编解码器替代了最初支持的VP8编解码器。

VP9草案包括许多增强功能和新的编码工具，这些编码工具已添加到VP8编解码器中，以提高编码效率。草案中描述的新工具包括更大的预测块尺寸，最大可达到 64×64 ，各种形式的复合帧间预测，更多的帧内预测模式，八分之一像素运动矢量，8抽头可切换子像素插值滤波器，改进的运动参考生成和运动矢量编码，改进的熵编码，包括对各种符号的帧级熵自适应，改进的环路滤波，非对称离散正弦变换（ADST）的合并，更大的 16×16 和 32×32 DCT，以及改进的帧-级别细分。但是，VP9目前正在开发中，VP9规范的最终版本可能与规范草案有很大差异，此处描述了其中的一些功能。

图片分区

VP9将图片划分为 64×64 超级块（SB），并按从左到右，从上到下的光栅扫描顺序进行处理。与HEVC相似，尽管 8×8 块大小是模式信息的最典型单位，但可以使用递归四叉树将超级块细分为最小 4×4 。与HEVC相比，VP9中没有切片结构。

人们希望能够并行执行编码或解码任务，或者使用多线程，以便有效地利用可用资源，尤其是在诸如智能手机之类的资源受限的个人设备上。为此，VP9通过frame_parallel_mode标志和基于两个或四个列的平铺提供帧级并行性，同时允许跨图块边界执行环路过滤。VP9中的平铺仅在垂直方向上完成，而每个平铺具有整数个块。相邻图块之间没有数据依赖性，并且可以按任何顺序处理帧中的任何图块。在除最后一个瓦片之外的每个瓦片的开始处，传输一个4字节的大小，指示下一个瓦片的大小。这允许多线程解码器通过向前跳到适当的图块来启动特定的解码线程。每帧有四个图块，便于在硬件和软件实现中实现数据并行化。

比特流功能

VP9比特流通常以WebM之类的容器格式提供，它是Matroska媒体容器的子集。容器格式对于随机访问功能是必需的，因为VP9并未为此提供起始代码。VP9位流从包含所有内部编码块的关键帧开始，这也是解码器重置点。与VP8不同，VP9中没有数据分区。所有数

据类型都按照超级块编码顺序进行交织。进行此更改是为了方便硬件实施。但是，类似于VP8，VP9还使用8位非自适应算术编码（也称为bool编码）压缩比特流，对于该模型，概率模型是固定的，并且在帧解码开始之前先验地知道所有符号概率。每个概率都有一个已知的默认值，并作为1字节数据存储在帧上下文中。解码器维护四个这样的上下文，以及用于帧解码的比特流信号。一旦帧被解码，基于解码帧中某些符号的出现，可以使用新的概率分布更新上下文，以供将来的帧使用，从而提供有限的上下文适应性。

每个编码帧都有三个部分：

- 未压缩的报头：很少包含图像大小，环路滤波器强度等的字节。
- 压缩报头：布尔型编码的报头数据，包含帧的概率，用与默认概率值的差异表示。
- 压缩帧数据：重建帧所需的布尔编码帧数据，包括分区信息、帧内模式、运动矢量和变换系数。

VP9比特流除了提供具有合理复杂性的高压缩效率外，还包括旨在支持各种特定用例的功能，这些用例涉及通过因特网传输和消费视频。例如，对于在不可靠的网络上以低延迟进行对话视频的通信，必须支持一种编码模式，即使在丢失任意帧的情况下，解码也可以继续进行而不会损坏。具体来说，即使帧缓冲区已损坏，算术解码器也应能够继续正确解码符号，从而导致编码器与解码器不匹配。

VP9支持帧级error_resilient_mode标志，以允许在编码器和解码器之间可能存在可管理的漂移的编码模式，直到关键帧可用或选择可用的参考图片来纠正错误为止。尤其是，在容错性能模式下，以下限制适用，而性能预期会有所下降：

- 在每一帧的开始，熵编码上下文的概率被重置为默认值，从而阻止了前向或后向更新的传播。
- 对于运动矢量参考选择，来自先前编码参考帧的同位运动矢量不能再包含在参考候选列表中。
- 对于运动矢量参考选择，在搜索参考帧缓冲区上，基于运动矢量参考候选的初始列表的排序被禁用。

VP9比特流不提供任何安全功能。尽管VP9不受外部对象和相关安全漏洞的影响，但必须通过双流之外的功能来确保完整性和机密性。

残差信号编码

如果一个块不是跳过的块（以 8×8 粒度表示），则为它编码并发送残差信号。与HEVC相似，VP9还支持不同大小（ 32×32 、 16×16 、 8×8 和 4×4 ）的整数转换，该整数转换近似于DCT。但是，根据内部残基的特定特性，垂直和水平变换通道之一或二者都可以是

ADST。在位流中，转换大小经过编码，以使使用 8×8 转换的 32×16 块将会具有由一个 8×8 转换系数的 4×2 网格，和两个 16×8 色度残差组成，其中每个残差由一个 8×8 转换系数的 2×1 网格组成。

从左上角开始扫描变换系数，遵循“弯曲的之字形”模式向更高的频率扫描，而具有DCT/DST混合的变换块使用相应偏斜的扫描模式¹⁰。但是，扫描模式并不简单，并需要进行表查找。此外，每个变换系数都是使用布尔编码进行编码的，并具有与之相关的几个概率，这是由各种参数（例如，块中的位置、变换的大小、相邻系数的值等）导致的。

反量化只是将亮度和色度的直流和交流系数的四个缩放因子之一进行简单的相乘，它们对于单帧保持不变；不允许进行块级QP调整。此外，VP9使用 4×4 沃尔什·阿达玛变换在帧级别提供了无损模式。

帧内预测

VP9中的帧内预测类似于AVC和HEVC中的帧内预测，并且在与变换块分区相同的分区上执行。例如，具有 8×8 转换的 16×8 块将导致两次 8×8 亮度预测操作。有10种不同的预测模式：DC、TM（真实运动）、垂直、水平和六个角度预测，分别对应于27、45、63、117、135和153度角。像其他编解码器一样，帧内预测需要两个一维数组，其中包含相邻块重构的左像素和上像素。对于大于 4×4 的块大小，水平阵列的后一半包含与前一半的最后一个像素相同的值 图3-19中给出了一个示例。

105	100	102	100	96	98	101	93	80	80	80	80	80	80	80	80	80
88	87	86	85	86	87	89	91	93	94	95						
85	85	86	87	89	91	93	94	95	96	97						
85	87	89	91	93	94	95	96	97	98	99						
89	91	93	94	95	95	96	97	98	99	100						
93	94	95	95	94	93	92	91	90	90	90						
95	95	94	93	91	90	90	90	90	90	90						
95	93	91	90	90	90	90	90	90	90	90						
90	90	90	90	90	90	90	90	90	90	90						

图3-19.亮度内部样本，D27_PRED模式

帧间预测

VP9中的帧间预测使用八像素运动补偿，提供了相对于其他大多数标准两倍的精度。对于运动补偿，VP9主要每个块使用一个运动矢量，但可选地允许每个块具有两个运动矢量的复合预测，从而产生两个预测样本，这些预测样本被平均在一起。复合预测仅在用作参考帧的不可显示帧中启用¹¹。VP9允许将这些不可显示帧背负于可显示帧，一起形成要在容器中使用的超帧。

VP9定义了三个8抽头滤波器系列，可以在比特流的帧或块级别选择：

- 8抽头常规滤波器：8抽头拉格朗日插值滤波器
- 8抽头锐化滤波器：基于DCT的插值滤波器，通常在较锐利的边缘使用
- 8抽头平滑（非插值）滤波器：一种平滑非插值滤波器，在某种意义上，在整数像素对齐的位置处的预测是参考帧像素的平滑版本。

运动矢量指向三个可能的参考帧之一，这些帧称为Last（时间上前一帧），Golden（时间上任一帧）和AltRef（参考帧，不显示，质量高于普通帧）。参考帧以8×8的粒度应用-例如，两个4×8块（每个块具有各自的运动矢量）将始终指向同一参考帧。

在VP9中，运动矢量是从候选参考运动矢量的排序列表中预测的。使用共享相同参考图片的最多八个周围块来构建候选对象，然后使用来自前一帧的同位运动矢量的时间预测值。如果此搜索过程未填充列表，则会再次搜索周围的图块，但这一次引用不必匹配。如果此列表仍不完整，则将推断出(0, 0)个向量。

与块相关，以下四个运动矢量模式之一会被编码：

- NEW_MV预测模式：此模式使用预测列表的第一项以及在比特流中传输的增量运动矢量。
- NEAREST_MV预测模式：此模式按原样使用预测列表的第一项。
- NEAR_MV预测模式：此模式按原样使用预测列表的第二个条目
- ZERO_MV预测模式：此模式使用(0, 0)作为运动矢量值。

VP9解码器始终保持一列八张参考图片的列表，其中一帧将其中三张用于帧间预测。预测的帧可以选择将自身插入这八个插槽中的任何一个，以淘汰现有帧。VP9支持参考帧缩放；可以以与前一帧不同的分辨率对新的帧间帧进行编码，同时根据需要按比例放大或缩小参考数据。缩放滤镜是具有第16像素精度的8抽头滤镜。该功能在可变带宽环境中非常有用，例如通过因特网进行视频会议，因为它允许快速，无缝地实时调整比特率。

环路滤波器

VP9引入了各种新的预测块和变换大小，它们需要附加的环路过滤选项来处理大量边界类型的组合。VP9还在环路滤波器中集成了平坦度检测器，该平坦度检测器可检测平坦区域并相应地改变滤波器的强度和尺寸。

VP9环路滤波器应用于解码后的图片。环路滤波器在超级块上运行，使垂直边缘平滑，然后使水平边缘平滑。超级块以光栅扫描顺序进行处理，而不考虑可能用信号通知的任何图块结构。这与HEVC环路滤波器不同，在HEVC环路滤波器中，帧的所有垂直边缘在任何水平边缘之前被过滤。VP9环路过滤中使用四种不同的过滤器：16宽、8宽、4宽和2宽，其中在边缘的每一侧分别处理八个、四个、两个和一个像素。根据在帧头中发送的阈值应用每个过滤器。尝试在以下条件下进行滤镜：边缘两边的像素应该相对平滑，并且边缘两边的亮度差异必须明显。满足这些条件后，将使用滤镜对边缘进行平滑处理。如果不满足条件，则尝试使用下一个较小的过滤器。 8×8 或 4×4 的块大小从8宽或更小的过滤器开始。

分割

VP9中的分割机制提供了一组灵活的工具，可以有针对性地使用这些工具，以在给定的压缩率下提高某些区域的感知质量。它是VP9的一项可选功能，允许块指定块所属的段ID（0至7）。帧头可以传达以下任何功能，适用于具有相同段ID的所有块：

- AltQ：属于具有AltQ功能的片段的块可以使用与其他片段中的块不同的逆量化比例因子。这在许多速率控制方案中很有用，特别是对于前景和背景区域中的不均匀位分布。
- AltLF：属于具有AltLF功能线段的块可以使用不同的平滑强度进行环路滤波。这对于特定的一组有针对性的目标平滑很有用。
- 模式：属于具有活动模式功能段的块，被当做具有相同的编码模式。例如，如果在一个段中激活了跳过模式，则所有块都将没有残留信息，这对于帧的静态区域很有用。
- 参考：假定启用了参考功能的属于某个段的块均指向特定的参考帧（last, golden或AltRef），不必采用参考帧信息的常规传输。
- EOB：属于具有块系数结尾（EOB）标记编码功能段的块可以对属于该段的所有块使用相同的EOB标记编码。这样就无需分别解码EOB标记。
- 变换大小：也可以为段中所有的块指示块变换大小，这对于一个段可能是相同的，但允许在同一帧中使用不同的变换大小。

为了最小化信令开销，分割图跨帧进行了差分编码。细分与拼贴无关。

⁹. A. Grange and H. Alvestrand, “A VP9 Bitstream Overview,” Internet draft,

February 2013, retrieved from <http://tools.ietf.org/html/draft-grange-vp9-bitstream-00>. ↵

¹⁰. P. Kapsenberg, “How VP9 Works: Technical Details and Diagrams,” Doom9’s forum, October 8, 2013, retrieved from <http://forum.doom9.org/showthread.php?p=1675961>. ↵

¹¹. 通过使用没有残差的 64×64 块和指向该参考帧的运动矢量 $(0, 0)$ ，可以从此类参考构建低成本的可显示帧。 ↵

概要总结

本章简要概述了业界可用的主要视频编码标准。为了保证互操作性、易实现以及整个行业的格式通用，这些标准相对于其他标准，指定或优先使用某些技术。由于本章前面已进行了讨论，因此这些偏见很容易理解。视频编码标准的另一个目标是通过单个标准，解决视频实际传输、存储或广播的所有方面。这是从H.261到MPEG-2的标准中完成的。MPEG-4和更高的标准不仅继承了成功的内容，而且在早期的技术和算法上得到了改进。

尽管在本章中，我们没有尝试比较各种标准算法提供的编码效率，但文献中已有此类研究。例如，在MPEG-2、AVC、WMV-9和AVS之间进行了有趣的比较¹²。多年来，这种比较，尤其是确定后一代标准与上一代标准相比节省比特率的方法，已经成为现实。格罗斯¹³等人在比较HEVC、VP9和AVC标准时证实了这一点。



开放权限 本章根据知识共享署名-非商业性-无衍生品4.0国际许可 (<http://creativecommons.org/licenses/by-nc-nd/4.0/>)，允许您以任何媒介或格式进行任何非商业性的使用、共享、分发和复制，只要您对原始作者和原始资料给予适当的感谢，请提供知识共享许可的链接，并指出您是否修改了许可材料。在没有获得此许可的情况下，不能共享本章及其改编内容。

除在材料的信用栏中另有说明的内容外，本章中的图像及其他第三方材料已包含在本章的知识共享许可中。如果本章的创用许可中未包含这些材料，并且法律规定不允许您使用或超出许可的使用范围，则您需要直接从版权所有者那里获得许可。

¹². S. Kwon, A. Tamhankar, and K. Rao, "Overview of H.264/ MPEG-4 Part 10," Journal of Visual Communication and Image Representation 17 (April 2006): 186–216. ↪

¹³. D. Grois, D. Marpe, A. Mulayoff, B. Itzhaky, and O. Hadar, "Performance Comparison of H.265/MPEG-HEVC, VP9, and H.264/MPEG-AVC Encoders," Proceedings of the 30th Picture Coding Symposium. (San Jose, CA, USA: Institute of Electronics and Electrical Engineers, 2013). ↪

视频质量指标

质量通常是卓越的代名词，在很多领域，我们用通用的社会规范来获取最高的质量。然而，在数字视频领域，可以采取精细的、量化的的方案来允许出现用户无法识别出来的质量缺陷。可感知的视觉质量方面的这种让步为视频压缩的价值提供了空间。

视频信号会通过一系列的变换或处理系统后进行传输，视频质量就是指通过这种传输之后的信号较之于原始信号而言，可感知的质量降级的度量。视频处理系统常常会引入一些失真 (*distortions*) 或伪像 (*artifacts*)，但是这些失真的数量可能会因为视频内容的复杂度、处理视频的参数的不同而不同。不同级别的质量对于终端用户的可接受程度并不一致，因此确定所有用户都能够接受的视频质量是非常困难，但是这件事情仍然是视频质量评估研究的重要目标。所以，理解那些导致用户不满的各种类型的视觉退化或伪影和评估终端用户的视频质量是一件非常重要的事情。

在本章中，我们首先关注由于压缩导致的信息损失（这些信息损失是刻意为之并且经过了精细的计算）以及由此产生的伪像 (*artifacts*)。然后，讨论影响视频压缩和视觉质量的各种因素（视频压缩和处理阶段）。

基于对这些内容的认识和理解，接下来将着眼于测量视频质量并讨论各种主观和客观质量指标（这些指标主要基于ITU-T的各种标准）。然后会进一步讨论视频编码效率评估指标和一些基于标准算法的评估示例。

在本章的最后，我们将讨论影响视频质量的主要参数、需要调整哪些参数以便在视频质量和压缩速度之间达到良好的权衡。这些内容在设计某些视频应用时非常有用。尽管部分参数由应用依赖的可用的系统资源或网络资源决定，但是，终端用户也可以设置或调整这些参数中的部分参数。

压缩损失，伪像，视觉质量

当压缩视频经过解压缩处理并呈现给用户时，压缩伪像 (*compression artifacts*) 是压缩视频中比较明显的失真现象。其它的压缩信号中也有可能会存在压缩伪像。这些失真是由所涉及的有损压缩技术引起的。压缩算法的目标之一是在最大化压缩量的同时最小化其信号失真。但是，根据压缩算法和压缩量级的不同，压缩的数据会具有不同程度的质量损失或引入伪像。有的质量评估算法可以区分那些主观影响很小的失真和用户难以接受的失真，并且可以采取措施来优化最终的视觉质量。

压缩损失：量化噪声

压缩损失以多种不同的方式表现出来并会导致某种视觉损失。本节将讨论最常见的压缩损失——量化噪声，以及和量化噪声相关的伪像。量化是将大量输入值映射到较小集合的过程，例如，将输入值四舍五入为某个精度单位的值。执行量化的设备或算法称为量化器。量化过程引入的舍入误差即量化误差或量化噪声。换句话说，量化误差就是输入信号和量化信号之间的差异。视频应用中有两种主要的量化噪声源：第一，模拟信号转换为数字格式时；第二，数字信号的有损压缩期间丢弃高频信息时。接下来将详细阐述如上的内容。

采样的量化

图像的数字化处理将传感器每个采样点的连续值亮度信息转换为不同灰度级的离散整数集。传感器的特性（例如动态范围和线性度）将会显著影响亮度测量和量化的整个过程。传感器可以处理的信号具备有限的范围，它们仅能处理处于某个区间内的光强度。真实传感器也是非线性的，但也可能存在一些区域，在传感器的这些区域的两端是非线性区域，而该区域内部或多或少是线性的。

A/D (*analog-to-digital*) 转换器的量化位数决定了量化器输出的动态范围。 n 位的量化器可以表示从最小到最大的 $N = 2^n$ 个级别。8位量化器通常表示0到255之间编号的256个灰度级别，其中0表示黑色，255表示白色。更多的量化位数可以提供更精细的量化级别，因此可以产生更少的噪声，从而使得量化信号更接近原始信号。图4-1显示了2位4级和3位8级量化信号的对比。2位4级量化仅是输入信号的粗略近似，而3位8级量化信号则比2位4级量化更精准。

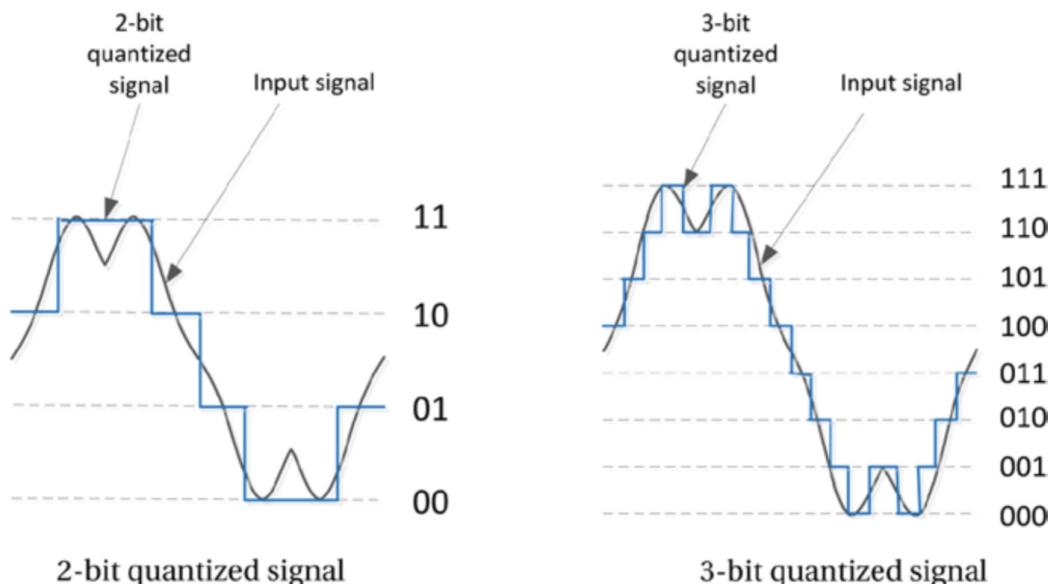


图4-1. 不同量化基数的对比

对于图像而言，像素亮度的真实输入与量化后的数字级别表示之间的差异表示该像素的量化误差（*quantization error*）。量化误差可以取正值或负值。对于均匀量化（*uniform quantization*）而言，量化级别是等间隔的，但是对于非均匀量化而言，量化间隔并非是等间隔的。如果量化级别以步长 b 等间隔，数字图像的量化误差可以近似为均匀分布，且量化误差的期望为0，方差为 $\frac{b^2}{12}$ 。

均匀量化器通常是无记忆的，也就是说像素的量化级别在计算时会独立于其他像素。

频率量化

在频率量化中，图像或视频帧通过变换（例如离散余弦变换，*DCT*）将图像数据转换为频域数据。对于 8×8 的像素块，DCT将产生64个变换系数。第3章中提到的有损压缩技术会使用相同大小的量化矩阵对变换系数执行量化操作。量化矩阵对于高频信息和低频信息的量化并非是线性的，一般会精细量化低频信息，对高频信息的量化则比较粗糙。实际上，大多数高频分量在量化后变为零，这一方面有助于压缩，但是另一方面会导致信息发生不可逆转地丢失，进而导致解压缩期间，无法根据量化系数回复原始值。原始像素块和重建像素块之间的差异表示引入的量化误差，图4-2展示了量化误差的过程。

$$f = \begin{bmatrix} 139 & 144 & 149 & 163 & 155 & 155 & 155 & 155 \\ 144 & 151 & 153 & 156 & 159 & 156 & 156 & 156 \\ 150 & 155 & 160 & 163 & 158 & 156 & 156 & 156 \\ 159 & 161 & 162 & 160 & 160 & 159 & 159 & 159 \\ 159 & 160 & 161 & 162 & 162 & 155 & 155 & 155 \\ 161 & 161 & 161 & 161 & 160 & 157 & 157 & 157 \\ 162 & 162 & 161 & 163 & 162 & 157 & 157 & 157 \\ 162 & 162 & 161 & 161 & 163 & 158 & 158 & 158 \end{bmatrix}$$

$$F = \begin{bmatrix} 1260 & -1 & -12 & -5 & 2 & -2 & -3 & 1 \\ -23 & -17 & -6 & -3 & -3 & 0 & 0 & 1 \\ -11 & -9 & -2 & 2 & 0 & -1 & -1 & 0 \\ -7 & -2 & 0 & 1 & 1 & 0 & 0 & 0 \\ -1 & -1 & 1 & 2 & 0 & -1 & 1 & 1 \\ 2 & 0 & 2 & 0 & -1 & 1 & 1 & -1 \\ -1 & 0 & 0 & -1 & 0 & 2 & 1 & -1 \\ -3 & 2 & -4 & -2 & 2 & 1 & -1 & 0 \end{bmatrix}$$

对原始像素矩阵 f 执行DCT变换后得到 F , 对其进行量化如图4-2。

$\begin{bmatrix} 16 & 11 & 10 & 16 & 24 & 40 & 51 & 61 \\ 12 & 12 & 14 & 19 & 26 & 58 & 60 & 55 \\ 14 & 13 & 16 & 24 & 40 & 57 & 69 & 56 \\ 14 & 17 & 22 & 29 & 51 & 87 & 80 & 62 \\ 18 & 22 & 37 & 56 & 68 & 109 & 103 & 77 \\ 24 & 35 & 55 & 64 & 81 & 104 & 113 & 92 \\ 49 & 64 & 78 & 87 & 103 & 121 & 120 & 101 \\ 72 & 92 & 95 & 98 & 112 & 100 & 103 & 99 \end{bmatrix}$	$\begin{bmatrix} 79 & 0 & -1 & 0 & 0 & 0 & 0 & 0 \\ -2 & -1 & 0 & 0 & 0 & 0 & 0 & 0 \\ -1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$
量化矩阵	量化系数

$\begin{bmatrix} 1264 & 0 & -10 & 0 & 0 & 0 & 0 & 0 \\ -24 & -12 & 0 & 0 & 0 & 0 & 0 & 0 \\ -14 & -13 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$	$\begin{bmatrix} -4 & -1 & -2 & -5 & 2 & -2 & -3 & 1 \\ 1 & -5 & -6 & -3 & -3 & 0 & 0 & -1 \\ 3 & 4 & -2 & 2 & 0 & -1 & -1 & 0 \\ -7 & -2 & 0 & 1 & 1 & 0 & 0 & 0 \\ -1 & -1 & 1 & 2 & 0 & -1 & 1 & 1 \\ 2 & 0 & 2 & 0 & -1 & 1 & 1 & -1 \\ -1 & 0 & 0 & -1 & 0 & 2 & 1 & -1 \\ -3 & 2 & -4 & -2 & 2 & 1 & -1 & 0 \end{bmatrix}$
重建系数	量化误差

图4-2. 转换系数的量化过程

量化矩阵和输入到量化器的变换系数的大小一致。为了获得量化系数, 变换系数的每一个元素需要除以对应的量化系数, 然后对结果取整。例如, 在图4-2中, 通过舍入(1260/16)给出量化系数79。量化之后, 位于左上角的低频系数得到保留, 而高频系数则变为零, 并在传输之前被丢弃。量化系数乘以相同的量化矩阵来重建变换系数。但是, 重建的结果会包含量化误差, 如图4-2所示。

通常, 系数的量化取决于其相邻系数的量化方式。在这种情况下, 在量化下一个系数之前会保存并考虑邻域上下文。这是具有存储器的量化器的一个例子。

实际上, 量化系数矩阵中出现的大量零元素并非偶然。以这种方式设计的量化矩阵, 可以从信号中去除HVS不太敏感的高频分量, 从而在视觉感知没有明显降低的情况下, 提升视频信号的压缩率。

颜色量化

颜色量化是用于减少图像中的颜色数量。由于HVS对颜色信息的损失不敏感, 因此颜色量化是一种有效的压缩技术。此外, 对于只能支持有限颜色的设备而言, 颜色量化通常也是一种有效的手段。通常将颜色量化技术 (最近邻颜色算法) 与抖动 (Dithering, 量化误差

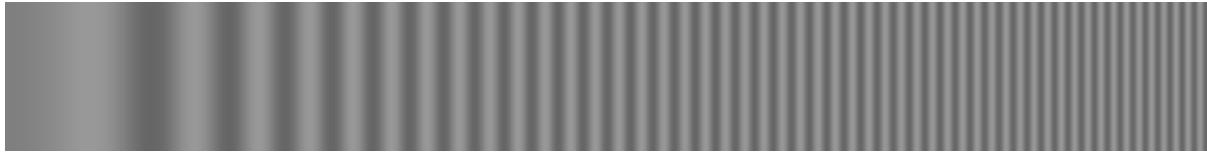
的随机化技术) 组合以产生比实际可用颜色更多的颜色, 以及避免色带伪影 (*color banding artifacts*) 的现象。色带伪影主要因为连续变化的色调量化之后由于部分颜色缺失带来的色调突变。

常见的伪影

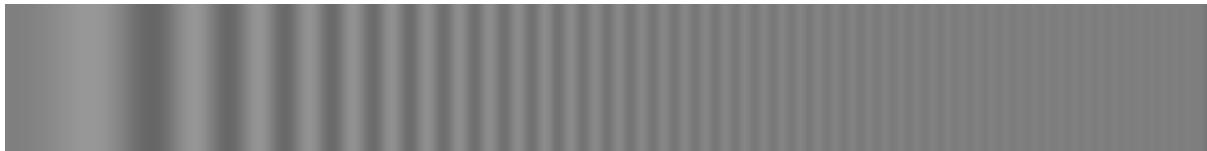
本节将介绍几种经常出现在各种图像和视频压缩应用中的视觉误差（visual artifacts）。

模糊效应（Blurring Artifact）

图像模糊（*blurring*）指的是平滑图像的细节和边缘，模糊对于图像来说就是一个低通滤波器（*low-pass filter*）。模糊对象（*object*）看起来像是对象的镜头焦外成像，但是实际上模糊效果和镜头焦外成像是不同的。一般而言，用户更趋向于获取更清晰的图像，因此会极力避免模糊的产生。但是，有时候也会使用高斯模糊（*gaussian blurring*）来降低图像噪点或者增强图像在不同比例大小下的图像效果。通常，模糊会作为压缩之前的预处理操作，从而减少图像的高频信号来产生更有效的压缩。另外，边缘检测对于环境噪点非常敏感，因此，模糊在边缘检测算法中也很有用。图4-3展示了模糊的例子¹。



频率从左到右线性增加的单色光栅



对如上的单色光栅的高斯模糊效果

图4-3.一个模糊频率斜坡（frequency ramp）的例子

从图4-3中可以看到，对于左边的低频率信号，模糊几乎没有任何效果，但是对右边的高频率信息，高斯模糊则由很明显的效果。

运动模糊出现在长时间暴光或场景内的物体快速移动的情况下。摄影机的工作原理是在很短的时间里把场景在胶片上暴光。场景中的光线投射在胶片上，引起化学反应，最终产生图片。这就是暴光。如果在暴光的过程中，场景发生变化，则就会产生模糊的画面。运动模糊通常是具有快速运动的运动内容中的伪影。然而，对于体育题材的视频内容，运动模糊常常会引起人们的不满。运动模糊会导致运动员在慢动作的场景下，无法确认其准确的位置，因此，运动模糊会带来一定程度上的不便。可以采用平移相机以跟踪移动物体的方法来避免运动模糊，在这种情况下，运动物体保持清晰但背景是模糊的。图形、图像

或视频编辑工具也可能出于艺术原因产生运动模糊效果。当添加计算机生成的图像（CGI, *computed generated imagery*）到视频镜头时，会频繁的合成运动模糊从而模拟真实世界中存在的模糊或表现运动物体的速度感。图4-4展示了一个运动模糊的例子。



图4-4.运动模糊的例子

通过显示器的隔行扫描和工作室的电视电影处理可能会导致运动速度的不规则。另外，对于快速运动的物体，数字视频中的压缩失真（*compression artifacts*）也可能带来额外的模糊。LCD固有的采样-保持（*sample-and-hold*）工作模式与人眼跟踪效应的综合效果决定了LCD在显示运动画面时会存在运动模糊。对于LCD显示器而言，运动模糊是一个较为严重的问题。在LCD显示器中，可以通过控制背光减少运动模糊的影响。

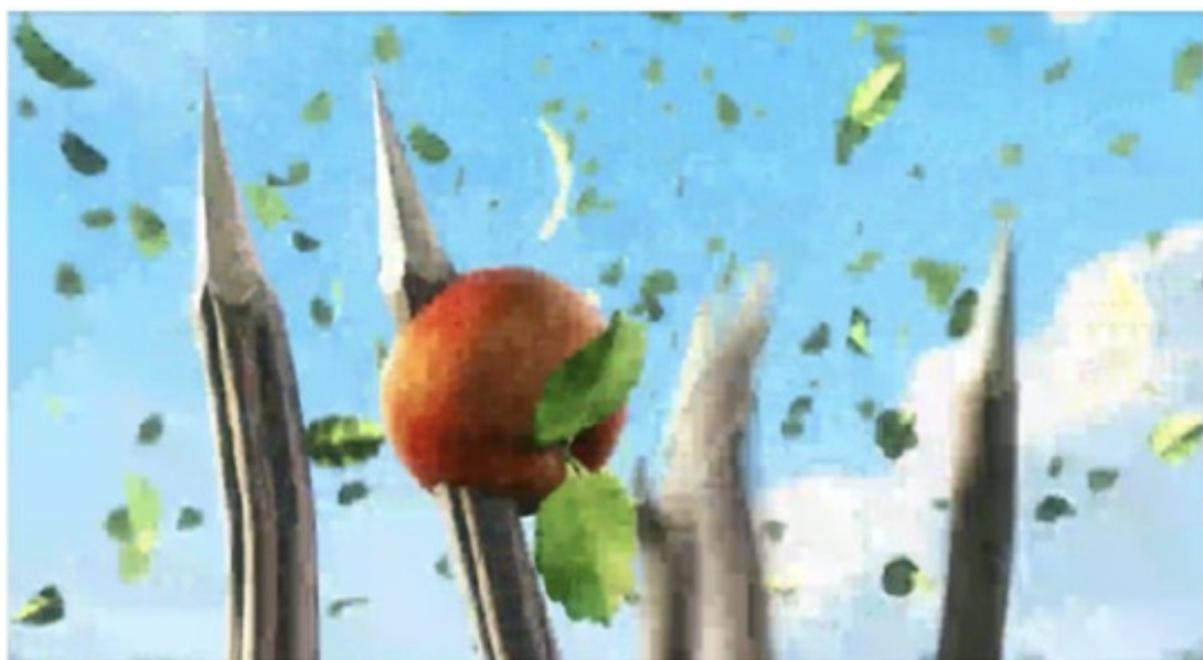
块效应（Block Boundary Artifact）

基于块的有损编码方案，包括所有的主要视频、图像编码标准，以较低的码率编码块的边缘像素从而在块的边缘处引入了可见的伪像。在基于块的变换编码中，使用DCT或类似的变换将像素块映射到频域空间，并且在量化过程中会丢弃高频系数。码率越低，对编码块的量化越粗糙，从而产生模糊的、低分辨率的图像块。在极端情况下，编码块只剩下代表数据平均值的DC系数，因此对该编码块的重建图像只是一个单色区域。

块效应伪像是对各图像块进行独立量化变换系数的结果。相邻的图像块分别量化系数，从而导致重构图像块的边界不连续。图像块边界的不连续性通常是可见的，尤其是在天空、人脸等平坦的颜色区域中。这些平坦的颜色区域中几乎没有任何细节可以掩盖图像块边界的不连续性。压缩算法通常执行去块（*deblocking*）操作以平滑重构的块边界，甚至使用没有此伪像的参考帧来执行去块操作。图4-5显示了块效应伪像的示例。



原始图像



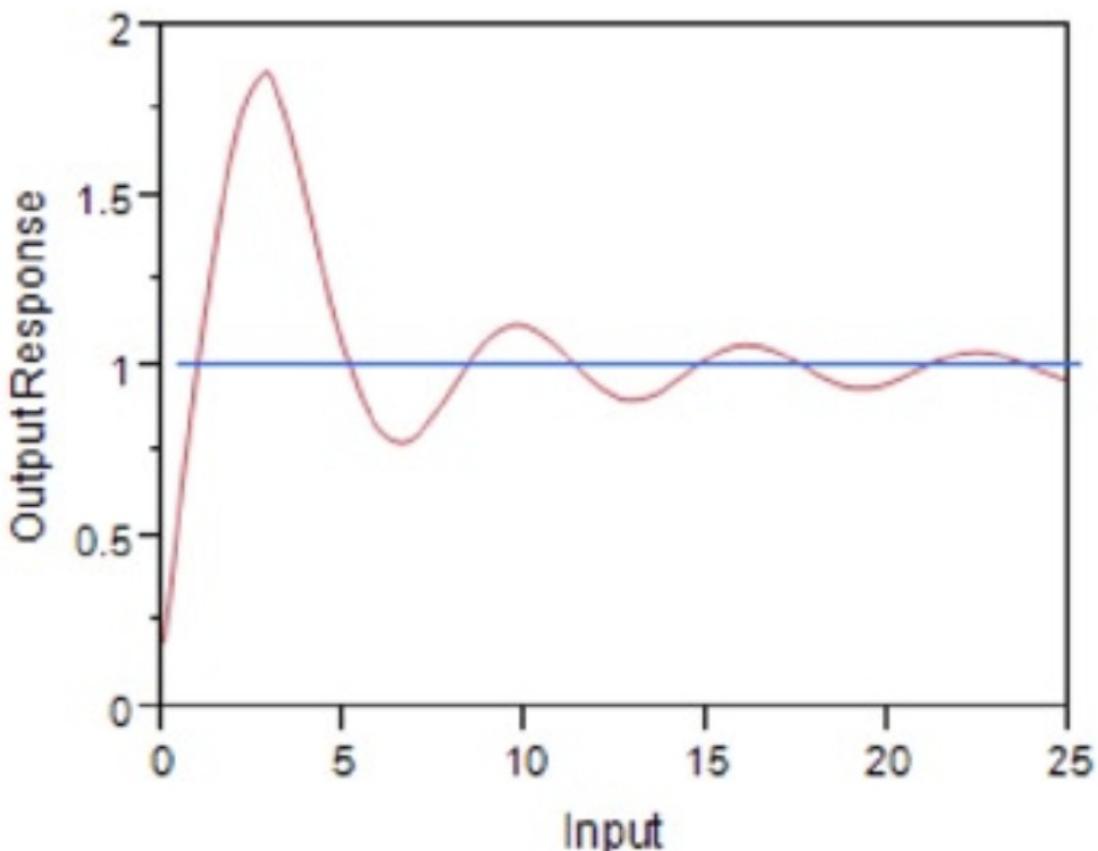
重构的存在明显的块效应伪像的图像

图4-5. 块边界效应伪像的例子

块边界伪像非常普遍，以至于经常使用许多名称来对其进行描述。尽管不连续性可能会（也可能不会）和视频/图像编码标准中定义的宏块的边界对齐，但是宏块 (*macroblocking*) 是此伪像的常用术语。其他名称包括平铺 (*tiling*)，镶嵌 (*mosaicing*)，缝 (*quilting*) 和棋盘格 (*checkerboarding*)。

振铃效应 (Ringing Artifact)

振铃是指因为输入图像中存在的剧烈变化而导致的输出信号产生的不必要的振荡。图像处理中，需要对图像进行滤波处理，若选用的频域滤波器具有陡峭的变化，则会使滤波图像产生振铃。所谓振铃，就是指输出图像在灰度剧烈变化处或轮廓处产生的震荡而导致的重影或回声效果，就好像钟被敲击后产生的空气震荡一样。振铃是由于众所周知的吉布斯 (Gibb's) 现象引起的，即滤波器在不连续点附近的脉冲响应的振荡行为，其中输出的比相应的输入值高 (*overshoots, higher value*) 或小 (*undershoots, lower value*)，并且幅度 (*magnitude*) 不断减小直到达到稳态为止。输出信号以渐弱的频率振荡。图4-6给出了一个振铃效应的例子。



吉布斯现象的振铃输出



原始图像



包含振铃伪像的图像

图4-6. 振铃伪像的例子

混叠效应 (Aliasing Artifacts)

让我们考虑一个时变信号 $x(t)$ 及其采样信号 $x(n) = x(nT)$, 采样周期 $T > 0$ 。当 $x(n)$ 为2倍的下采样时, 其他所有采样都将被丢弃。在频率 (ω) 域中, 信号 $X(e^{j\omega})$ 的傅立叶变换被扩展了2倍。在这种情况下, 变换后的信号通常可以与其移位后的副本重叠。在这种重叠的情况下, 原始信号无法从其下采样版本中明确恢复, 因为重叠区域同时代表了转换信号的两个副本。这种重叠的效果称为混叠。图4-7显示了下采样 (包括混叠) 的变换域效果。

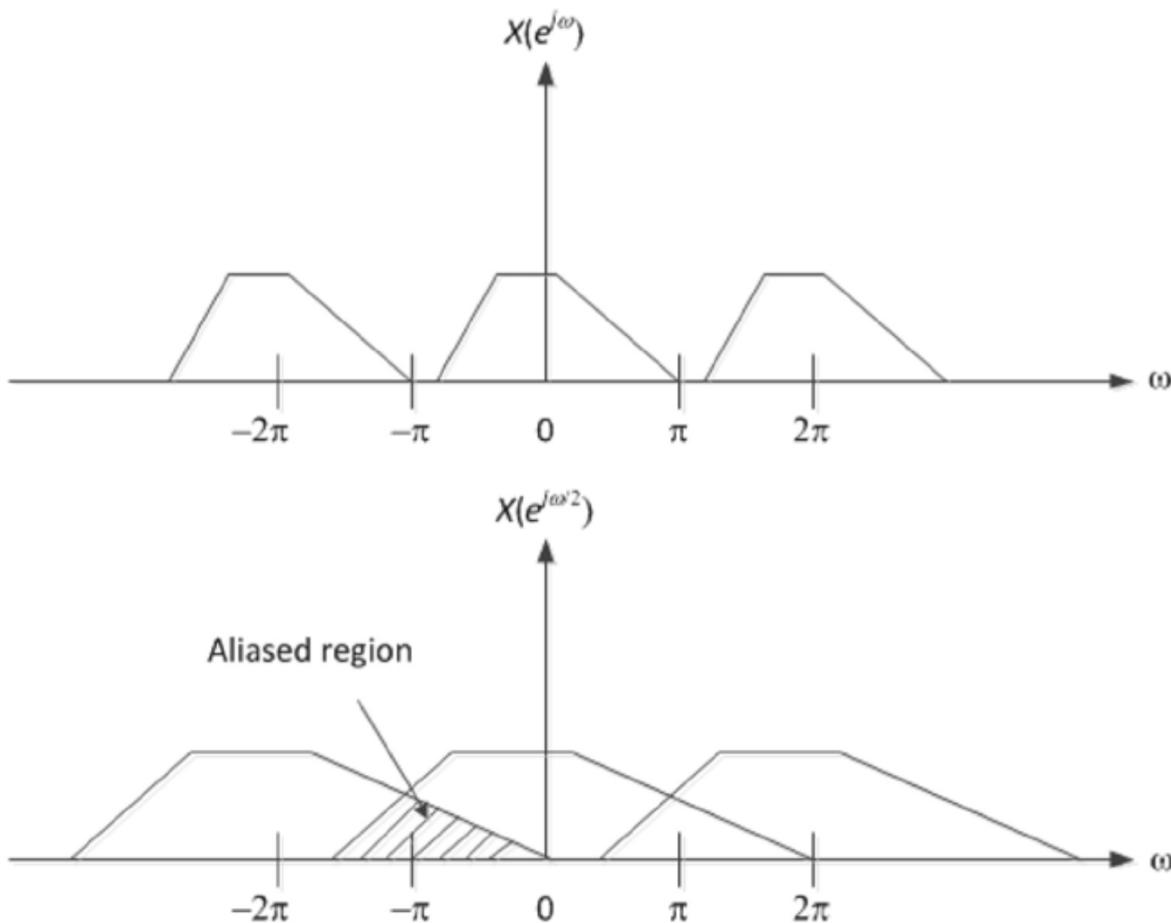


图4-7. 变换域的下采样导致的混叠效应

通常，混叠是指对抽样信号的模棱两可的重构而导致的伪影或失真。混叠可能发生在实时采样的信号中，例如数字音频，这被称为时间混叠。混叠也可能发生在空间采样信号（例如，数字图像或视频），在这种情况下称为空间混叠。

当采样具有有限持续时间的实际信号时，总是会发生混叠。这是因为这些函数的频率内容没有上限，导致它们的傅立叶变换表示始终与其他变换函数重叠。另一方面，具有受限频率内容（带宽受限）的函数具有无限的持续时间。如果以高于奈奎斯特采样率的速度采样，则原始信号就可以从采样信号中完全恢复。从图4-7可以明显看出，如果将原始信号限制在 $|\omega| < \frac{\pi}{M}$ ，其中 M 是下采样倍数。在这种情况下，可以使用上采样器从下采样的版本中恢复原始信号，然后进行滤波。

实际的数字信号处理中，信号的采样和其他处理过程由不同模块完成，他们对于信号的采样率需求可能不同。对于抽样后的信号 $x(nT)$ ，抽样频率 f_s 的改变需要使用上采样和下采样来完成。

上采样 ($L = 2$) 又称为信号插值，即在原信号序列 $x(n)$ 的两个点之间插入 L 个 0，等效于在其频域上做了频谱压缩。下采样 ($M = 2$) 又称为信号抽取，即在原信号序列 $x(n)$ 中每隔 M 个点抽取一个点，等效于在其频域上做了频谱扩展。

锯齿效应 (Jaggies)

有一种常见的混叠伪影被称为锯齿 (*jaggies*)，其会在数字图像中的平滑直线或曲线上产生可见的阶梯状线条。这些阶梯或台阶是像素规则的、正方形布局的结果。随着图像分辨率的提高，该伪像会变得越来越不可见。同样，抗锯齿 (*anti-aliasing*) 滤镜可用于减少锯齿边缘的可见性，而锐化 (*sharpening*) 则会提高这种可见性。

图4-8显示了混叠伪像（例如锯齿和摩尔纹图案）的例子。

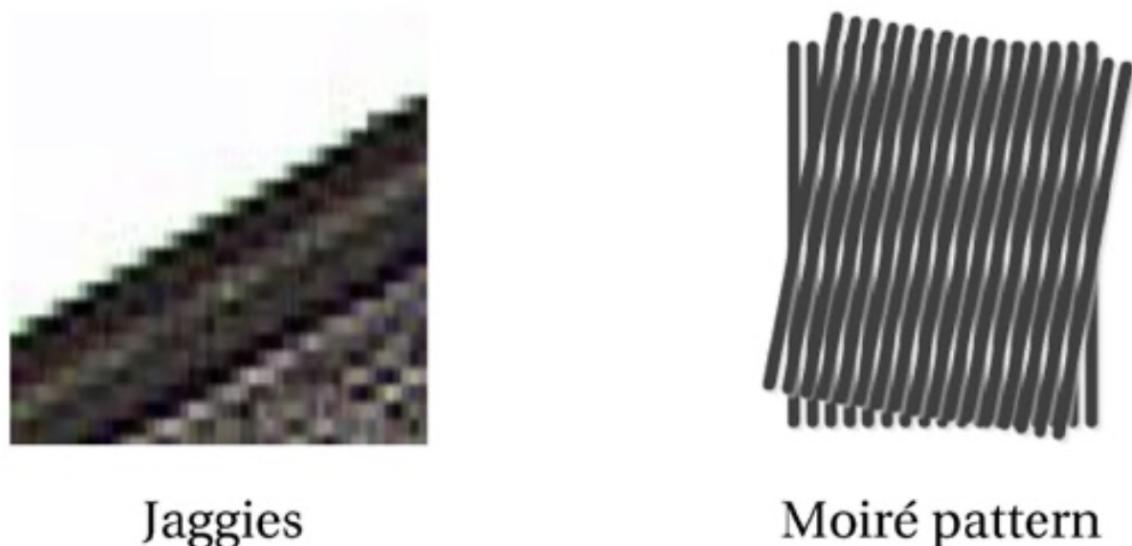


图4-8. 混叠伪影的例子

摩尔纹 (*Moiré pattern*)

由于精细的常规图案的下采样，出现了摩尔纹的混叠现象。摩尔纹是各种数字成像和计算机图形技术产生的图像中所不希望存在的伪像。摩尔效果是明显不同的视觉感知，这是由两个相似图案的不精确叠加引起的。在图4-8中，摩尔效果可以看成是起伏的图案，而原始图案则包括紧密间隔的直线网格。

闪烁效应 (Flickering Artifacts)

在视频显示期间的足够长的时间内（例如，大约100毫秒），可以察觉到亮度中断现象，该现象称之为闪烁。闪烁是一种令人不悦的闪光效果。当以较低的刷新率驱动旧显示器时（例如CRT，阴极射线管），就会发生闪烁现象。由于液晶显示器（LCD）中，用于每个像素的快门保持稳定的不透明性，因此即使刷新图像也不会闪烁。

抖动（Jerkiness）

抖动是一种类似闪烁的伪像，描述了对视频中的单个静止图像的感知。可以发现，感觉到闪烁和抖动的频率取决于许多条件，包括环境照明条件。对于以24 FPS或更高的帧率正常播放视频而言，无法感知到抖动。但是，在视觉通信系统中，如果解码器延迟则会导致视频帧被视频解码器丢弃，或者由于网络错误导致解码失败，此时则将继续显示前一帧。直到成功解码下一个无错误的帧之后，显示屏上的场景才会突然更新。此时，就会导致可见的抖动现象。

电视电影转换抖动（Telecine Judder）

电视电影转换抖动是另一种类似闪烁的伪像。为了将24 FPS的电影转换为30 FPS的视频，通常需要做电视电影转换（telecine）或采用2 : 3的下拉（pulldown）。该过程将每四帧电影帧转换为五帧隔行扫描的视频帧。某些DVD或蓝光播放器，或录像机可以检测电视电影转换，并应用反向电视电影转换过程重建原始的24 FPS的视频内容。图4-9显示了电视电影转换的过程。

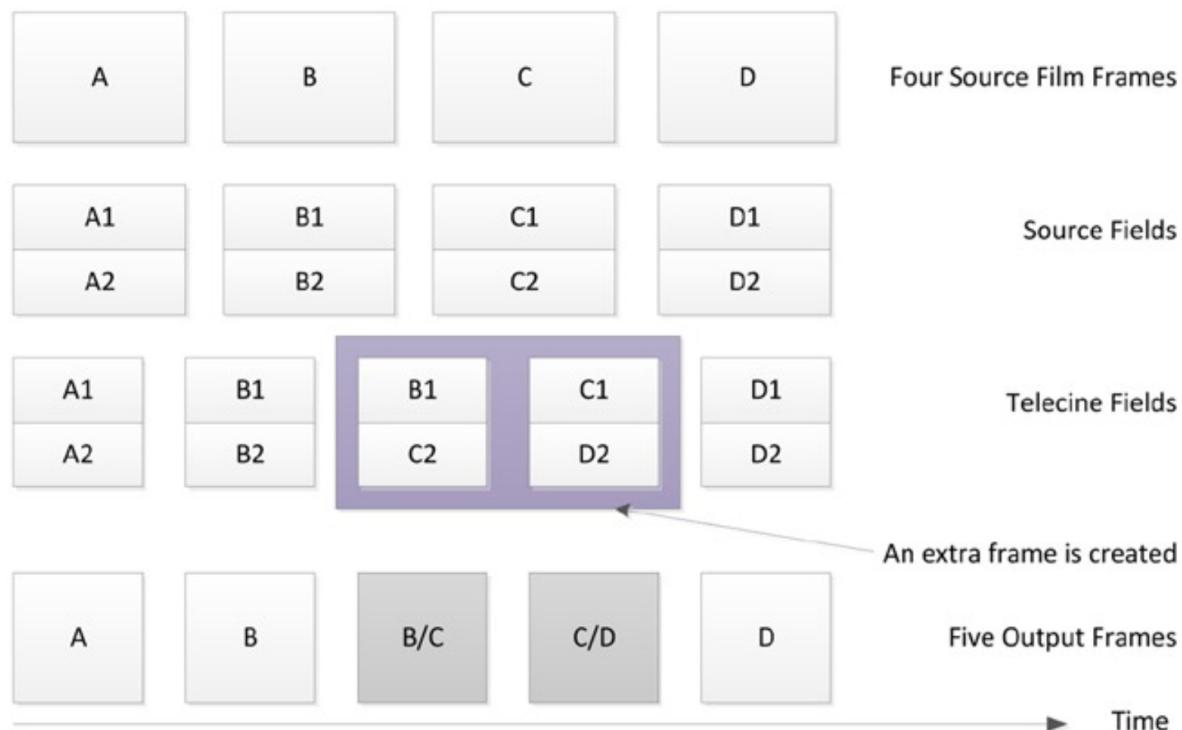


图4-9. 电视电影转换过程

需要注意的是，电视电影转换处理创建了两个新的帧B/C和C/D，它们并不属于原始的视频帧。因此，与原始胶卷相比，电视电影处理会在视频信号中产生少量的错误。这曾经导致在NTSC制式的电视上观看电影时的体验问题，即电影看起来不像在电影院中观看时的那样平滑。尤其是在相机缓慢、稳定的运动时，该问题尤为明显。

其它图像伪像 (Other Image Artifacts)

在压缩视频中，还存在其它类型的可视误差，接下来将对其进行简单介绍。

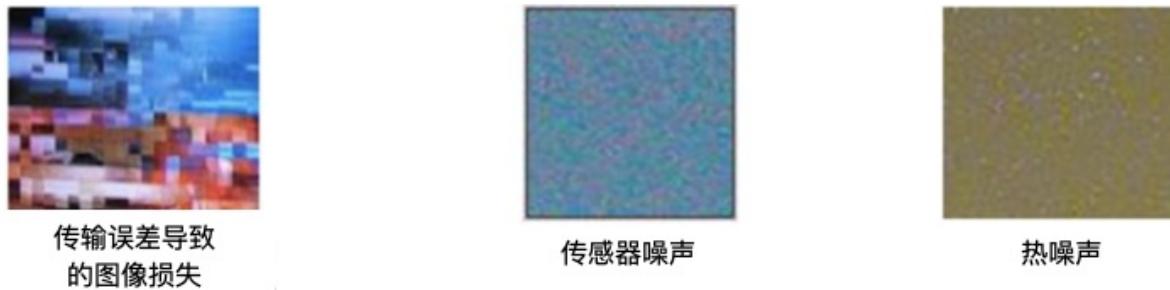


图4-10.各种图像噪声的示例

传输错误导致的数据损坏

压缩比特流中的传输错误会导致在重建信号时产生数据损坏。传输错误还会破坏比特流的解析，导致图像的部分解码或解码的图像存在丢失块。在严重错误的情况下，解码器会在短时间内继续对损坏的图片进行更新，从而产生重影图像效果 (*ghost image effect*)。重影效果会一直持续到下一个无错误的独立压缩帧到来为止。在露天电视信号中，重影是一种比较常见的效果。

图像噪声

每个像素的相机传感器包含一个或多个光敏光电二极管用于将入射光转换成电信号，电信号则被处理成图像的像素的颜色值。然而，这个过程并不总是完全可重复的，并且会存在一些统计差异。此外，即使没有入射光，传感器的电活动也可能产生一些信号。图像噪声则来源于这些非期望的信号和差异。这种噪声随每个像素和时间而变化，并随温度而增加。胶片颗粒也会导致图像噪声。数字图像中的噪声在均匀表面中最明显，例如在天空和阴影中的单色颗粒和彩色波。热像素噪声是另一种类型的噪声。热像素噪声因为持续一秒以上的长时间曝光而出现，并且显示为略大于单个像素的彩色点。然而，在现代相机中，热像素噪声越来越少。

¹ <https://aty.sdsu.edu/bibliog/latex/scan/blur.html>. ↵

影响视觉质量的因素

数字视频信号处理过程中的信息丢失而导致的视觉误差 (*visual artifacts*) 通常会降低可感知的视觉质量。除了之前提到的视觉误差之外，还有其它的重要因素会影响视觉质量，接下来会一一介绍。

Sensor noise and pre-filtering (传感器噪声&预过滤)

传感器噪声是在图像捕获过程中产生的、影响视觉质量的、非预期的副产物。噪声不仅会在视觉上影响用户体验，而且噪声的存在也会影响后续的图像处理，从而导致或加剧误差的产生。例如，预过滤通常在捕获图像之后、编码之前完成。在预过滤阶段，可能出现混叠效应 (*aliasing artifacts*) 或振铃效应(*ringing artifacts*)。因此，即使接下来采用无损编码技术，也会出现这些视觉误差。

Characteristics of video (视频特征)

视觉质量会受到数字视频本身特性的影响，这些特征包括：位深，分辨率，帧率和帧的复杂度。

通常，一般的视频帧采用8位的位深来表示像素的每个通道的数据，而更高级的视频则采用为10~16位的位深。由于格式不同，高清 (HD) 视频帧的大小是标清 (SD) 视频帧的4~6倍。超高清 (UHD) 视频的分辨率比标清视频高24~27倍，因此超高清视频的质量最高。

帧率是另一个重要因素。虽然HVS可以感知10 fps的慢动作，也可以感知24 fps的平滑运动，但是更高的帧率意味着更平滑的运动。尤其是对于快速移动的物体而言，更高的帧率能够使其看起来更流畅。例如，运动中的羽毛球可能在30 fps时是模糊的，但在120 fps时会较为清晰。更快的运动需要更高的帧率才能得到较高质量的视频。例如，蜂鸟的机翼运动在30 fps甚至120 fps时都会变得模糊，可能需要1000 fps才能清晰地看到这种快速运动。更高的帧率也用于产生特殊的慢动作效果。可以用帧的细节量或者空间业务量来度量帧的复杂性。低复杂度和低细节的帧中的误差一般比更高复杂度的帧更明显。

视频的空间信息 (*spatial information, SI, 细节*) 和时间信息 (*temporal information, TI, 运动*) 是视频的关键参数。SI和TI对于确定视频可能达到的压缩量中起到重要作用。同时，对于确定在固定速度的数字传输服务信道传输信息时导致的信息折损等级，SI和TI

也具有至关重要的作用。

Amount of compression (压缩量)

压缩一般通过对视觉质量的折中来实现。因此，对于压缩的数字视频而言，压缩量是一个很重要的指标。一般而言，压缩度越高，视频的质量则越低。对于低码率的视频而言，压缩失真 (*compression artifacts*) 更为明显，并且会降低用户体验。同时，基于可用比特数的限制，图像的不同块和不同图像会采用不同的量化。此时，压缩质量也会因为帧的复杂性的不同而不同。另外，尽管色度亚采样的压缩技术会利用HVS的特性，但是4:4:4的图像格式比4:2:0的图像格式具有更好的视觉质量。

Methods of compression (压缩方法)

无损压缩保留了视频信号中的所有信息，因此不会导致质量下降。有损压缩则通过在视觉质量和压缩量之间进行仔细权衡来控制质量损失。在有损压缩中，压缩模式的选择也会影响质量。在诸如无线网络这种容易抖动的环境中，帧内模式可以用作错误的恢复点，但是需要使用更多的比特数。

Multiple passes of processing (多次处理)

在不需要实时处理的离线视频应用中，视频信号可能经历多次通过。分析第一遍的统计数据，可以调整参数以用于后续传递。这些技术通常在最终产生的信号中产生更高的质量。然而，由于各种处理造成的误差仍可能导致一些质量损失。

Multiple generations of compression (多代压缩)

视频应用可以采用多代压缩。在多代压缩中，先将压缩视频信号解压缩，然后用不同的参数再次进行压缩。由于每代使用不同的量化图，可能导致质量下降。通常，在第二代之后，视觉质量会急剧恶化。为了避免这种质量损失，量化参数的稳健设计是必要的。

Post-production (后期制作)

后期制作中涉及的视频特效和视频剪辑可能会使编码视频序列的不同部分具有不同的质量等级。

视频质量的评估方法和指标

需要通过评估视频质量确定系统要求、比较竞品提供的服务、确定传输规划、确定网络维护，确定基于客户端的视频质量等。在文献中已经提出了很多方法来解决各种场景下的质量评估问题。随着许多方法和算法的出现，ITU在一些建议中一直致力于满足业界对准确可靠的客观视频指标的需求。ITU的每个建议都会针对特定的行业，如标清和高清广播电视。

随着移动广播，互联网流视频，IPTV等现代应用的发展，标准化工作也在不断扩展。标准用于解决各种问题，包括定义，职权范围，要求，推荐实践和测试计划。本节重点关注质量评估算法的定义，方法和指标。尤其是从用户体验的角度来观察视频的体验质量（*QoE, quality of experience*）。用户体验是用户的感知，而不是用于评估数据传输和网络性能的服务质量（*QoS, quality of Service*）的度量。

有两种方法可以解释视频质量：

1. 第一种方法很简单，基于人类的主观评估来确定图像或视频内容的实际视觉质量。
2. 第二种方法等价于在某些感知空间中评估信号的保真度，或者评估信号与参考信号或图像之间的相似性。有很多复杂的模型可以捕获自然视频信号的统计数据。基于这些模型，开发了客观信号保真度标准。该标准将视频质量和参考/失真视频信号之间共有的信息量关联起来。

接下来将详细介绍评价视频质量的主观指标和客观指标。

主观视频质量评估

视频处理系统需要执行各种任务，包括视频信号采集，压缩，恢复（*restoration*），增强（*enhancement*）和再现（*reproduction*）。在这些任务中，为了在可用的系统资源的约束下获得最佳视频质量，系统设计者通常基于某些质量标准进行各种权衡。

衡量质量的一个明显方法就是征求人类的意见。因此，视频质量的主观评价方法就是利用人类受试者来执行评估视觉质量的任务。但是，主观评估应用程序处理的每个视频的质量是一件不可能的事情。此外，由于不同的人类观察者对质量判断存在天然的不一致性，通常需要从多个角度进行评估以便使得主观研究更有意义。此外，观察者的观看条件也会影晌对视频质量的主观感受。这些观看条件包括：环境照明，显示设备，观看距离等。因此，必须在精心控制的环境中进行视频质量评估的主观研究。

尽管可以使用主观评估技术来跟踪真实的感知视频质量，但是该过程是一个繁琐的、非自动化的过程。并且因为不同的个体对相同的视觉对象有不同的感知，因此主观评估的结果可能会随着观看者的不同而变化。尽管如此，主观评价方法仍然是一种有价值的方法，并且可以为自动评估或客观视频质量评估算法提供参考。

可以用客观算法来估计用户的感知，并且可以使用主观测试的结果评估客观算法的性能。媒体降级（*media degradations*）会影响观众的感知质量。因此，必须精心设计主观测试，从而能够准确捕捉退化对用户感知的影响。主观测试需要进行全面的实验，以产生一致的结果。为了准确评估客观质量算法，主观测试需要包含如下方面：

- 参与主观评估的用户必须是无专业知识的非专家的普通用户。利用客观质量模型来评估参与用户的感知。参与用户按照测试设计者的指示对主观质量进行投票。但是，对于特定应用（例如开发新的编解码器），选择有经验的参与用户更为合适。
- 对于每个视频测试样本而言，参与主观评估的用户数量应该符合ITU-T相关建议中给定要求，一般而言，参与用户数至少是24人。
- 为了保证实验的一致性和可重复性，同时为了在实验中可以调整质量范围和失真类型，实验需要一个参考样本库（*anchor¹ pool of samples*），该样本库能够代表需要评估的特定应用。因为并非总能在参考条件中包含所有失真类型，即使使用参考样本，在不同实验中也可能会经常存在偏差。

ITU-T第9研究组（SG9, study group 9）提出了若干建议作为主观和客观质量的评估方法，例如ITU-R BT.500-13²和ITU-T P-系列建议。这些建议提供了：标准观察条件，观察者和测试材料的选择标准，评估程序和数据分析方法。建议ITU-T P.910³至P.913⁴用于处理多媒体应用中的主观视频质量评估。建议的早期版本（例如，P.910, P.911⁵）主要针对

多媒体应用中的固定视频服务的范例而设计。这些范例认为视频信号通过可靠的连接传输到位于安静且无干扰的环境（例如起居室或办公室）中的固定阴极射线管（CRT）电视机。ITU-T引入了P. 913建议，用于适应新的应用（例如互联网视频）。

ITU-T P.913于2014年1月获得批准，建议描述了非交互式的主观评估方法，该方法用于评估单向视频（*one-way overall*）质量，音频质量和视听质量。P.913旨在覆盖一种新的视频范例，例如，视频点播服务，通过不可靠的链接传输至位于嘈杂环境的、使用LCD或其它平板显示器的、各种移动设备和固定设备。这种新范式影响了主观测试的关键特征，例如观看环境，收听环境和要回答的问题。新范例中的主观质量评估会增加先前建议中未考虑的问题。但是，P.913并未满足广播公司的专业需求。

测试镜头的数量、类型、持续时间以及主题数量，对于主观评估结果的解释至关重要。P.913建议测试视频的持续时间为5~20秒，最好为8~110秒。场景数量可以控制在4~6个。P.913要求需要至少24名受试者在受控环境中进行评估，并且至少35名受试者在公共环境中进行评估。较少的受试者可用于试点研究以指示视频质量的趋势。

主观质量评估方法和指标

ITU-T P-系列定义了很多最常用的主观质量评估的方法。本节将会对其进行介绍。

ACR，绝对等级评分

绝对等级评分方法是一种对测试序列一次呈现一个，且在质量等级量表内进行独立评分的等级判断⁶。ACR也称为单刺激方法，其中观看者观看一个刺激（例如，视频剪辑）然后对其进行独立评级。ACR方法受到观看者对内容看法的影响，例如，观看者如果不喜欢制作的内容，他可能会给该视频较差的评分。

ACR使用下列的5种量级评价视频的总体质量：

分数	等级	解释
5	Excellent	优秀
4	Good	良好
3	Fair	普通
2	Poor	较差
1	Bad	差

带有隐参考的绝对等级评分方法绝对是一种对测试序列一次呈现一个，且在等级量表内进行独立评分的等级判断。

还有一种ACR方法的变体——具有隐藏参考的ACR (ACR-HR)。对于ACR-HR而言，实验包括每个视频片段的参考版本，参考版本不是作为一对的一部分，而是作为评级的独立刺激。在进行数据分析时，将计算每个测试序列和其对应的(隐)参考间的差异质量评分(DMOS, differential mean opinion score)。这个步骤就是“隐参考”。

差异观测者(DV, differential viewer)分数在每个被试者/每个处理后的视频序列(PVS, processed video sequence)的基础上计算得出。通过下述公式，合适的隐参考(REF)被用于计算DV。

$$DV(PVS) = V(PVS) - V(REF) + 5$$

其中V是观测者的ACR分数。

使用此公式时，DV为5时表示质量“优秀”，为1时表示质量“差”。任何大于5的DV值(即，处理后的序列评分比其相关的隐参考序列高)，通常情况下被认为是有效的。否则，可应用如下的公式来预防这些单独的ACR-HR观测者分数(DV)过度影响总体平均意见分数：

$$crushed_DV = (7 * DV) / (2 + DV), \ s.t. \ DV > 5$$

ACR-HR方法消除了来自ACR评级的内容的一些影响，但程度小于双刺激方法，这将在下面讨论。

DCR, 劣化等级评分

劣化等级评分 (DCR) 也称为双刺激损害量表 (DSIS, double stimulus impairment scale) 方法，DCR中测试视频序列会成对呈现。首先呈现参考刺激，然后呈现经过处理和质量降级后的版本。在这种情况下，要求受试者对第二序列相对于参考序列的损伤进行评级。对于观看者对内容的影响而言，DCR受到的这种影响是最小的。因此，DCR能够检测ACR方法可能遗漏的颜色损伤和跳过错误。但是，因为DCR始终首先显示参考序列，因此DCR可能略有偏差。在DCR中，使用以下五种量级评价相对损害的等级：

分数	等级	解释
5	Imperceptible	无感知的
4	Perceptible but not annoying	可感知但不令人厌烦
3	Slightly annoying	轻微令人厌烦
2	Annoying	令人厌烦
1	Very annoying	非常令人厌烦

比较等级评分

比较等级评分 (CCR) 是一种双刺激方法，其中以随机顺序呈现相同刺激的两个版本。例如，可以有一半的时间首先显示参考序列，以及有一半的时间最后显示参考序列，并且以随机方式确定参考序列的显示顺序。CCR也称为双刺激比较量表 (DSCS, double-stimulus comparison scale) 方法。它可用于比较参考视频和处理过的视频，或比较两种不同的损伤。与DCR一样，CCR观察者对内容的看法的影响微乎其微。然而，在CCR中，受试者可能会无意间交换了CCR中的分数，这将导致DCR或ACR中不存在的一种错误。在DCR中，使用以下七种量级。

分数	等级	解释
-3	Much worse	甚差
-2	Worse	较差
-1	Slightly worse	稍差
0	The same	相同
1	Slightly Better	稍好
2	Better	较好
3	Much Better	甚好

MOS，平均主观得分

平均主观得分 (MOS) 是主观视频质量评估中最常用的指标。MOS构成了主观质量评价方法的基础，也可作为客观评价指标的参考。在电话网络领域，使用MOS评估用户对网络质量的看法已有数十年之久。MOS也可用于音频领域的主观音频质量测量。在所有受试者进行实验之后，需要获取每个视频的平均得分以计算MOS或差异质量评分 (DMOS) 。

对于经过压缩和传输的视频而言，MOS提供了数字等级来表示用户对这些视频的感知质量。MOS通常表示为1~5的单个数字，其中1是最低感知质量，5是最高感知质量。MOS用于诸如ACR/ACR-HR的单刺激方法。在ACR/ACR-HR评估方法中，受试者对测试视频中的每一个视频进行单独评估。相反，DMOS分数测量相同刺激的两个版本（例如，源视频和处理后的视频）之间的质量变化。在平均差异观察者得分情况下的ACR-HR方法，DCR方法和CCR方法通常产生DMOS分数。

比较不同实验的MOS值需要仔细考虑实验内和实验间的变化。通常，只有来自相同测试类型的MOS值才能比较。例如，使用ACR标度MOS值不能直接与来自DCR实验的MOS值进行比较。另外，即使对相同测试类型的MOS进行比较，此时，即便对于相同的参与者，每个实验的结果也会略有不同。这种情况将会导致以下问题：

- 即使以相同的样本，相同的顺序进行重复实验，受试者给出的分数也很少保持一致。

通常这被认为是MOS分数上的一种噪声。

- **短期依赖：**因为受试者会受到他们先前评分的样本的短期影响，因此MOS会受到短时背景的影响。例如，在一个或两个不良样本之后，受试者倾向于将平庸的样本给出更高的评分。如果一个平庸的样本出现在非常好的样本之后，则受试者倾向给予平庸样本较低的评分。为了平衡这种短时依赖性，应该针对不同的受试者调整测试样本的呈现顺序。但是，这种策略并没有消除统计上的不确定性。
- **中期依赖：**与平均质量、质量分布有关的中期背景导致了主观实验之间的差异。例如，如果实验主要由低质量的样本组成，那么人们往往会被给予它们一个更高的评分，反之亦然。这是因为人们倾向于使用实验中提供的全部质量量级，并使该量级适应实验中提出的质量。
- **长期依赖：**长期依赖性反映了受试者对类别标签的文化解释、对质量的文化态度以及语言依赖性。例如，某些人可能比其他人更频繁地使用视频内容。此外，对质量的期望可能会随着时间而改变。随着人们对数字视频的失真现象越来越了解，数字噪音也成为人们日常体验的一部分。

虽然如上的限制会导致实验之间的差异，但是却无法避免。但是，通过提供信息说明，均衡的测试设计，足够数量的参与者以及混合的演示顺序，可以最大限度地降低它们的影响。

¹. anchor字面意思是锚，是个把船固定的东西。anchor在计算机视觉中有锚点或锚框，目标检测中常出现的anchor box为锚框，表示固定的参考框。 ↵

². ITU-R BT.500-13: [Methodology for the Subjective Assessment of the Quality of Television Pictures.](#) ↵

³. ITU-T P.910: [Subjective video quality assessment methods for multimedia applications.](#) ↵

⁴. ITU-T P.913: [Methods for the subjective assessment of video quality, audio quality and audiovisual quality of Internet video and distribution quality television in any environment.](#) ↵

⁵. ITU-T P.911: [Subjective audiovisual quality assessment methods for multimedia applications.](#) ↵

⁶. ITU-T P.910(C), 6.1, P12, 绝对等级评分法。 ↵

客观视频质量评估和指标

视频质量评估（VQA, *Video quality assessment*）旨在设计一套自动评估视频质量的算法。这些自动评估算法可以与人类的主观评估在感知上保持一致。自动评估算法可以跟踪视频质量的指标。因为这些指标不需要人工现场试验，因此这些指标是可自动化的，并且可以重复执行来获得结果。然而，这些算法并非总是很完美，仅仅试图预测人类的主观体验而已。客观评估算法可能会因某些不可预测的内容而失败。因此，客观质量评价不能取代主观质量评价，客观质量评价只是作为质量评估的工具而已。ITU-T P.1401¹为各种媒体类型的客观质量评估提供了算法评估的框架。

在ITU-T P.1401中，客观质量评估的统计指标²需要涵盖三个主要方面——准确性（accuracy），一致性（consistency）和线性相关性（linearity），这三个方面的指标计算需要和主观评估结果对应起来。其中，可以用预测误差来（prediction error）计算准确性，利用离群点率（outlier ratio）或者残差分布（residual error distribution）来计算一致性，利用皮尔森相关系数（Pearson correlation coefficient）来计算相关性。

算法预测结果残差³的均方根误差（RMSE, root mean square error）可以用如下的公式计算。

$$(4-1) \text{ RMSE of } P_{error} = \sqrt{\frac{1}{N-1} \sum_{i=1}^N (MOS(i) - MOS_{predicted}(i))^2}$$

4-1中， N 是样本的数量， $N-1$ 用于确保RMSE的无偏差估计。

残差（ $MOS - MOS_{predicted}$ ）的分布通常符合二项分布的特征。低于某个预定义的阈值（通常该阈值需要满足具备95%的置信区间）的残差的概率的期望为 $P_{th} = \frac{N_{th}}{N}$ ，标准差为 $\sigma_{th} = \sqrt{\frac{P_{th}(1-P_{th})}{N}}$ ，其中 N_{th} 表示残差低于指定阈值的样本数。

客观视频/图像质量指标在视频应用中扮演着各种不同的角色，特别是如下的角色：

- 用于质量的动态监控和调整。例如，网络数字视频服务器可以基于实时的视频质量评估来适当的地分配、控制和折衷视频流资源。
- 用于优化视频处理系统的算法和参数。例如，在视频编码器中，质量度量可以促进预滤波和码率控制算法的最佳设计。在视频解码器中，可以帮助优化重建算法、错误消除算法以及后期滤波（post-filtering）算法。
- 用于视频处理系统和算法的基准测试。
- 用于比较两个不同的视频系统解决方案。

视频质量客观指标分类

可以根据视频质量评估算法需要的参考信息的数量将客观视频质量评估方法分成三类：完全参考（*FR, full reference*），半参考（*RR, reduced reference*）和无参考（*NR, no reference*）。其中，FR方法可进一步划分为如下几类：

- 基于误差灵敏度的方法（*Error sensitivity based approaches*）
- 基于结构相似性的方法（*Structural similarity based approaches*）
- 基于信息保真度的方法（*Information fidelity based approaches*）
- 时空方法（*Spatio-temporal approaches*）
- 基于显著性的方法（*Saliency based approaches*）
- 网络感知方法（*Network aware approaches*）

接下来的章节将会具体讨论如上的评估方法，并且在介绍的时候会增加对应的具体例子。

全参考

数字视频信号会经历很多处理步骤。在这些过程中，视频质量可能已被各种需求（压缩，速度或其他标准）所折中，从而使得用户可以获得可用的失真信号。在客观质量评估中，经常会测量失真信号的保真度。为了精确测量发生了多少退化，需要对参考信号（假定参考信号具有完美的质量）进行信号保真度测量。但是，参考信号也并不总是万能的。

全参考（*FR*⁴）指标度量失真视频相对于参考视频的视觉质量劣化程度。*FR*要求参考视频没有质量损失并且以非压缩的形式提供，同时失真信号和参考信号需要具备精确的空间和时间对齐，并且具备相同的亮度和颜色标准。只有这样，才可以直接比较视频中的每一帧的每个像素。

一般以某种合理的方式计算参考信号和失真信号之间的距离来确定信号的保真度。*FR*质量评估方法试图利用HVS的重要生理、心理视觉特征进行建模，并使用该模型来评估信号保真度，进而实现客观质量预测和用户感官之间的一致性。随着保真度的增加，感知质量也会随之提高。*FR*在视频质量分析方面非常有效，并且广泛用于视频质量分析和基准测试。但*FR*要求质量评估期间对参考信号的可访问性，这种要求在实践中可能无法实现。这可能会限制*FR*的应用。

半参考

当参考信号不能完全可用时，也可以设计模型和评估标准来客观评估视频质量。该领域的研究产生了各种使用部分参考信号的半参考（*RR*⁵）方法。*RR*需要从参考/失真的测试视频中提取很多特征。提取的特征构成了比较两个视频的基础，因此*RR*不需要完整的参考

信号。RR即避免了在没有任何参考信息的情况下所必须进行的假设操作，又保证了参考信息数量的可管理性。

无参考

无参考（NR）指标不依赖于显式参考视频，其仅分析失真的测试视频。因此，NR指标不受对齐问题的影响。然而，NR方法的主要挑战是区分当前信号是失真信号还是实际视频信号。因此，NR指标必须对视频内容和失真类型给出假设。

图4-11给出了FR和NR的流程图。

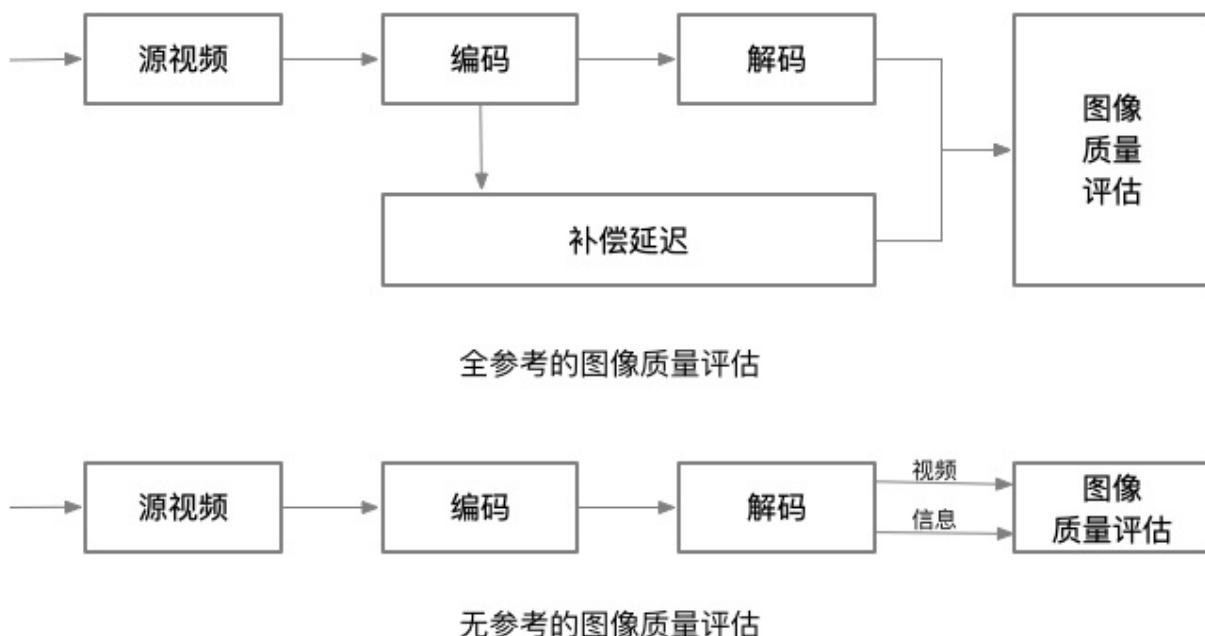


图4-11.基于参考的的分类举例：全参考和无参考算法流程图

王等人在论文中提出了一种NR测量算法⁶。该算法将模糊效应和块效应视为JPEG压缩过程中产生的最重要的噪声，并建议提取可反映这些噪声相对大小的特征。该算法然后将提取的特征进行组合，从而生成质量预测模型，接下来使用主观实验结果对该模型进行训练。算法的提出者希望训练之后的模型也适用于其它图像的评估。虽然类似的算法可以不使用任何参考而仅根据可用的图像内容进行评估，但在有可用的参考信号时，最好还是使用FR方法进行评估。接下来的讨论将集中在各种FR方法。

在所有条件下设计与感知视觉质量一致的客观度量算法是一个难题。许多可用的指标可能无法解释损坏图像的所有失真类型，也无法解释图像的内容以及失真的强度。但这些指标提供了与人类主观感受相对一致的结果。因此，客观质量度量仍然是一个非常活跃的研究领域。接下来将介绍各种不同的FR质量评估算法。

¹. ITU-T P.1401, *Methods, Metrics and Procedures for Statistical Evaluation, Qualification and Composition of Objective Quality Prediction Models.* ↵

- 2. ITU-T P.1401, 7.5 Statistical evaluation metrics, P14. ↵
- 3. 残差：观测值与拟合值的偏离；误差：即观测值与真实值的偏离。 ↵
- 4. ITU-T J.247: Objective perceptual multimedia video quality measurement in the presence of a full reference. ↵
- 5. ITU-T J.246: Perceptual visual quality measurement techniques for multimedia services over digital cable television networks in the presence of a reduced bandwidth reference. ↵
- 6. No-reference Perceptual Quality Assessment of JPEG Compressed Images. ↵

基于误差灵敏度的方法

当图像或视频帧经过有损处理时，就会产生失真的图像或视频帧。有损处理过程引入的误差量或失真量决定了视觉质量下降的程度。许多质量评估度量基于失真图像和参考图像之间的误差评估视频质量。在FR中，最简单、最广泛使用的度量指标是均方误差（MSE, mean squared error）。峰值信噪比（PSNR, peak signal-to-noise ratio）就是简单地通过均方误差（MSE）来定义。这些算法易于计算，具有明确的物理意义，并且在优化领域具备很好的数学便利性。但是，它们的结果与感知的视觉质量之间的匹配程度不高。

在基于误差灵敏度的图像或视频质量评估中，通常假设感知质量的损失与误差信号的可见性直接相关。MSE就是这个概念最简单的实现，MSE客观地量化了误差信号的强度。但是具有相同MSE的两个失真图像可能是不同类型错误，并且，在这些错误中，其中一些错误比其它的错误更明显。在过去的四十年中，已经开发出了许多利用人类视觉系统（HVS）特征的质量评估方法。这些模型中的大多数都会修改MSE，以便根据误差信号的可见性对误差信号的不同方面进行加权。这些模型也基本上都是基于一个通用框架。接下来将介绍该通用框架。

General Framework

图4-12描述了基于误差灵敏度的图像或视频质量评估方法的通用框架。对于大多数基于误差灵敏度的质量评估模型而言，尽管具体细节可能有所不同，但是基本上都可以用类似的框图来描述。

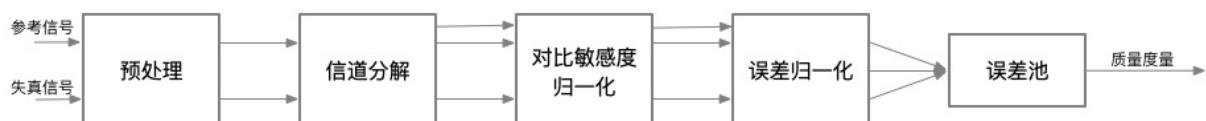


图4-12. 基于误差灵敏度的评估方法的通用框架

如图4-12，通用框架一般包括如下步骤：

- **预处理（Pre-processing）**：预处理会消除已知的失真并准备图像，以便在失真图像和参考图像之间进行公平的比较。例如，两个图像都经过适当的缩放和对齐。必要的情况下，还需要执行更适合于HVS的颜色空间转换或伽马校正。此外，可以用低通滤波器来模拟眼睛光学系统的点扩散函数（PSF, point spread function）⁷。另外，可以使用非线性点操作（point operation）⁸修改图像来模拟人眼的光适应（light adaptation）⁹现象。
- **信道分解（Channel Decomposition）**：通常将图像分解为对特定空间频率、时间频率以及方向敏感的子带¹⁰或信道。复杂的方法试图在初级视觉皮层（primary

visual cortex) ¹¹ 中尽可能的模拟神经反应，而其他方法仅简单使用离散余弦变换 (*DCT*) 或小波变换(*wavelet transforms*)进行信道分解。

- **对比敏感度归一化 (*Contrast Sensitivity Normalization*)**：对比敏感度函数 (*CSF*) ¹² 描述了HVS对视觉刺激中存在的不同空间和时间频率的灵敏度。通常用线性滤波器实现CSF的频率响应。很多早期的方法会在信道分解之前根据CSF对信号进行加权，但是最近的方法使用CSF作为每个信道的基本灵敏度归一化因子。
- **误差归一化 (*Error Normalization*)**：图像的不同分量可能在空间、时间位置，空间频率等方面非常接近，因此一个图像分量可能会减小或掩盖另一个分量的可见性。当计算并归一化每个信道中的误差信号时，需要考虑这种掩蔽效应 (*masking effect*)。归一化过程通过空间变化的视觉门限 (*visibility threshold*) ¹³ 对信道中的误差信号进行加权。对于每个通道，基于通道的基本灵敏度以及空间邻域中的参考或失真图像系数的能量确定视觉门限。归一化过程仅利用差别感觉阈限 (*JND, just noticeable difference*) ¹⁴ 表示错误。有的方法还会考虑对比度响应的饱和度的影响。
- **误差池 (*Error Pooling*)**：模型的最后阶段会将归一化的误差信号组合成单个值。为了获得组合值，通常计算如 $4 - 2$ 的明可夫斯基范数 (*Minkowski norm*)。其中， $e_{i,j}$ 为第*i*个频率通道的第*j*个空间系数的归一化错误值， β 为1~4的一个常量。

$$(4 - 2) E(e_{i,j}) = \left(\sum_i \sum_j |e_{i,j}|^\beta \right)^{\frac{1}{\beta}}$$

Limitations

尽管基于误差灵敏度的方法通过模拟HVS的特性来估计误差信号的可见性，但是这些模型中的大多数都是基于受限刺激和简单刺激表征的线性或拟线性算子。实际上，HVS是一个复杂且高度非线性的系统。因此，基于误差灵敏性的方法需要利用一些假设条件，因此也导致了以下的限制：

- **质量定义 (*Quality definition*)**：基于误差灵敏度的图像或视频质量评估方法仅跟踪图像保真度，较低的保真度并不总是意味着较低的视觉质量。误差信号的可见性转换为质量下降的假设可能并不总是有效的。有些扭曲是可见的，但并不那么令人反感。通过全局增加亮度值来增亮整个图像就是一个这样的例子。因此，图像保真度仅与图像质量有一定的相关性，而不是完全正比的相关性。
- **模型泛化 (*Generalization of models*)**：许多误差敏感模型都是基于实验来估计误差门限，然而实验中刺激 (*stimulus*) 的误差几乎是不可见的。这些阈值用于定义误差灵敏度量，例如对比敏感度函数。然而，在典型的图像或视频处理中，感知失真发生时，其误差值一般会远远高于实验给定的的阈值。因此，心理物理学 (*suprathreshold psychophysics*) ¹⁵ 中的近阈值模型 (*near-threshold models*) 的

泛化模型一般不是很精确。

- **信号特征 (Signal characteristics)** : 大多心理物理学 (*psychophysical*) 实验使用相对简单的图案, 例如斑点, 条形或正弦光栅。例如, 经常利用全局正弦图像的阈值实验来获取CSF。然而, 实践中的自然图像与简单图案的特征并不相同。因此, 简化模型的适用性可能会在实践中受到限制。
- **依赖关系 (Dependencies)** : 错误池 (*error pooling*) 会因为其使用的假设 (不同信道和空间位置的错误信号是独立的) 而轻而易举的受到质疑。对于诸如小波变换的线性信道分解方法, 自然图像的信道内和信道间小波系数之间存在强依赖关系。转换 (*transformation*) 和掩蔽模型 (*masking models*) 的优化设计会减少这种统计和感知的依赖性。但是, 此类设计对VQA模型的影响尚未确定。
- **认知交互 (Cognitive interaction)** : 众所周知, 诸如眼球运动之类的交互式视觉处理会影响感知质量。此外, 认知理解会对感知质量产生巨大影响。例如, 对于不同的指令, 用户可能会给同一图像打出不同的分数。对图像的先验知识和偏见也可能会影晌图像质量的评估。因为认知交互难以理解、难以量化, 大多数基于误差敏感度的评估方法都不会考虑认知交互的影响。

7. **点扩散函数 (PSF, point spread function)** 描述了一个成像系统对一个点光源 (物体) 的响应。PSF的一般术语就是系统响应, PSF是一个聚焦光学系统的冲击响应。在大多情况下, PSF可以认为像是一个能够表现未解析物体的图像中的一个扩展区块。函数上讲, PSF是成像系统传递函数的空间域表达。 ↫
8. **点运算**指的是对图像中的每个像素依次进行同样的灰度变换运算。 ↫
9. 人刚从暗处走到亮处的时候, 最初的一瞬间会感到强光耀眼发眩, 眼睛睁不开, 什么都看不清楚, 要过几秒钟才能恢复正常, 这就是**光适应现象**。 ↫
10. **子带编码技术**, 是将原始信号由时间域转变为频率域, 然后将其分割为若干个子频带, 并对其分别进行数字编码的技术。 ↫
11. **初级视皮层 (V1)** 又被称为纹状皮层, 由6层细胞构成, 发达的第4层又被分为 A、B、C 三个亚层。位于Brodmann 17区, 其输出信息有两条通道, 分别为背侧流 (Dorsal Stream) 和腹侧流 (Ventral Stream) 。 ↫
12. **对比敏感度函数(CSF, contrast sensitivity function)**是反映人眼辨认平均亮度下两个可见区域差别的能力指标。 ↫
13. **视觉门限**: 光刺激必须达到一定的数量才能引起感觉。能引起感觉的最低限度的光通量, 称为视觉的绝对阈限。同时人眼的视觉阈限又与空间和时间因素有关。 ↫
14. **差别感觉阈限**是指刚刚能引起差别感觉的刺激的最小差异量。也称为最小可觉差 (just noticeable difference), 简称JND。 ↫
15. **心理物理定律 (Psychophysical Law)** 是关于物理连续体上的变量和相应的感官

反应之间的函数关系及其数量化的描述。这些定律的目的是解释感官系统的活动和预测感觉行为。心理物理定律描述的现象主要有两类，一是对刺激探察力或阈限的测量，二是对阈限刺激分辨能力的测量。 ↪

峰值信噪比，PSNR

峰值信噪比（*PSNR*）是信号的最大可能功率与失真噪声（失真噪声会影响到信号质量）功率之间的比率的表达式。信号经过压缩，处理或传输后经常会产生影响其表示质量的失真噪声。由于许多信号具有非常宽的动态范围（可变数量的最大和最小可能值之间的比率），因此PSNR通常用对数分贝（*logarithmic decibel(dB)*）单位来表示。由于HVS的非线性行为，PSNR并非总能完美的表示可感知的视觉质量，但是只要视频内容和编解码器类型没有改变，PSNR就是一种有效的质量测量¹⁶。PSNR是有损环境中的视频信号保真度的良好指标。

对于原始信号 f 而言，经过一系列的处理和传输后，重建为近似信号 \hat{f} 。在这个过程中，会引入一些噪声。令 f_m 为信号的最大值或者峰值，对于用 n -bit表示的信号而言， $f_m = 2^n - 1$ 。对于8-bit的信号，其 $f_m = 255$ ，而10-bit的信号的 $f_m = 1023$ 。PSNR就是信号功率和噪声功率之间的比值，其数学定义如4 – 3所示。

$$(4-3) \text{ } PSNR = 10 \log_{10} \frac{(f_m)^2}{MSE}$$

4 – 3中的MSE用下式进行定义：

$$(4-4) \text{ } MSE = \frac{1}{N} \sum_{i=1}^N (f_i - \hat{f}_i)^2$$

4 – 4中， N 为样本的数量。

类似的，对于宽度为 M ，高度为 N 的图像或视频帧的二维信号而言，其MSE的定义如4 – 5。

$$(4-5) \text{ } MSE = \frac{1}{M * N} \sum_{i=1}^M \sum_{j=1}^N (f(i, j) - \hat{f}(i, j))^2$$

其中， $f(i, j)$ 为原图像 (i, j) 处的像素值， $\hat{f}(i, j)$ 为重建图像中对应的像素值。

通常针对图像平面（*image plane*）测量PSNR，例如视频帧的亮度（*luma*）或色度（*chroma*）平面。

应用

作为稳定的质量指标 (*consistent quality metric*)，PSNR一直应用于模拟视听系统。但是，在数字视频技术领域，PSNR会存在某些限制。然而，由于PSNR的低复杂性和易测量性，在评估有损视频压缩或处理算法时，PSNR仍然是最广泛使用的视频质量度量指标。PSNR还可用于评估压缩视频在特定比特率的情况下质量增益。PSNR还可用于检测帧丢失或严重帧数据损坏的情况，并且可以在自动化环境中定位丢弃或损坏帧的位置。类似的检测在视频编码或处理方案的调试和优化中非常有用。此外，PSNR还广泛应用于比较两种视频编码方案。

优势

PSNR的优势如下：

- PSNR是一种简单、易用的，基于图片的度量指标。PSNR的计算非常快并且计算过程可以并行化（例如使用SIMD并行计算）。
- 由于PSNR基于MSE，因此它与差异信号的方向无关。无论源信号和重建信号的运算顺序如何，都会产生一致的PSNR输出。
- 实践中，PSNR可以很容易的融入自动化质量测量系统中。这种灵活性使PSNR适用于大型测试套件。因此，对于建立评估的信心而言，PSNR非常有用。
- PSNR计算是可重复执行的。对于相同的源和重建信号，其PSNR总是相同的。另外，PSNR也不依赖于视频的宽度或高度，PSNR适用于任何分辨率的视频。
- 与繁琐的主观测试不同，PSNR不需要对评估环境进行特殊设置。
- PSNR被认为是开发其它客观视频质量指标的参考基准。
- 对于相同的视频源和相同的编解码器，PSNR是一种稳定的质量指标 (*consistent quality indicator*)。因此可利用PSNR优化编码器以最大化主观视频质量和编码器的性能。
- PSNR可单独用于亮度和色度信道。因此，PSNR可以非常方便地跟踪两个编码方案之间的亮度/颜色的变化。对于明确的质量要求，PSNR提供的单独信道的信息可用来确定哪个方案可以使用更少的比特位。
- PSNR的普及不仅源于它的简单性。作为度量标准，PSNR的性能也不容小觑。视频质量专家组 (VQEG, *Video Quality Experts Group*) 在2001年进行的验证研究发现：VQEG测试的九种VQA方法（包括当时的一些最复杂的算法）与PSNR方法“在统计上无法区分”¹⁷。

限制

对PSNR的常见的意见如下：

- 一些研究表明，PSNR与主观质量评估之间的关联度不高¹⁸。

- PSNR不考虑两个不同图像的可见性差异，而只考虑数值差异。PSNR不考虑视觉掩蔽现象¹⁹或HVS的特征。对于PSNR而言，无论差异的可见性如何，两个图像中的所有像素都会用来计算PSNR。
- 在特定情况下，在预测主观视频质量方面，其它的客观感知质量指标的表现优于PSNR²⁰。
- PSNR不考虑编码器在执行时间（*execution time*）或机器周期（*machine cycles*）方面的计算复杂性。PSNR也不考虑系统配置，例如数据高速缓存大小，存储器访问带宽，存储复杂性，指令高速缓存大小，并行性和流水线。因此，当PSNR用作主要标准时，两个编码器的比较会受到很大的限制。
- 单独的PSNR没有提供关于编码器编码效率方面的足够信息。通常在表示视频文件所使用的位数（*bits*）方面也需要相应的测量成本。除非视频的文件大小或码率是已知的，否则说某个视频具有一定级别的PSNR是没有意义的。
- PSNR通常在整帧图像进行平均计算，而不考虑帧内的局部区域的统计信息。对于视频序列而言，质量在不同场景之间的变化可能很大。因此，如果基于帧的PSNR结果进行聚合并平均化PSNR，则可能无法精确地获取质量结果。
- PSNR不考虑帧延迟或帧丢失等时间维度的质量问题。另外，PSNR仅是源编码测量，而不考虑各种信道编码问题。因此，在有损网络环境中，PSNR不是一种合适质量测量方法。
- PSNR是一种FR测量方法，因此在质量评估中需要提供参考视频。但是实践中，重建端通常并不能提供完美的参考视频。尽管如此，在评估、分析和评估视频质量方面，PSNR仍然有效且广受欢迎。

PSNR的改进

在很多文献中，已经进行了很多尝试来改善PSNR。给定失真的可见性取决于源图像的局部内容（*local content*）。图像变化不明显的平坦区域²¹和图像边缘的噪声比快速变化区域（*busy area*）的噪声更令人反感。因此，与简单的用PSNR测量信号能量来评估信号的失真相比，有可能会有一种更复杂的方式可以模拟失真本身的视觉效果。例如，可以在频域中应用加权函数，从而对误差的低频分量赋予更多的权重。Sarnoff在2003年基于视觉辨别模式提出了差别感觉阈限（JND, just noticeable difference）的方案²²。

运动图像的质量指标

PSNR不考虑视觉掩蔽现象。即使错误不可察觉，每个像素的误差也会导致PSNR的降低。通常需要结合HVS模型来解决该问题。在文献中已经深入研究了HVS的两个关键方面：对比敏感度（*contrast sensitivity*）和掩蔽（*masking*）。第一种现象是，只有当对比

度大于某个阈值时，眼睛才能检测到信号。眼睛的灵敏度会随空间频率、方向和时间频率的变化而变化。第二种现象与人类对多种信号组合的视觉反应有关。例如，考虑由前景和背景信号组成的刺激。此时，需要根据与背景的对比度来修改前景的检测阈值。

运动图像质量度量 (*MPQM, moving picture quality metric*)²³ 是用于运动图像的、基于误差灵敏度的时-空 (*spatio-temporal*) 客观质量度量。MPQM包含了上述讨论的HVS的2个特性。按照图4-12所示的通用框架，MPQM首先将原始视频及其失真版本分解为感知信道。然后计算基于信道的失真度量，并且在计算过程中需要考虑对比度灵敏度和掩蔽。在获得每个通道的失真数据之后，组合所有通道的数据以计算质量等级。得到的质量等级范围为1~5（从差到优秀）。MPQM与某些视频的主观测试具有良好的相关性。但和其它的基于误差敏感度的方法一样，MPQM也会对其它视频产生不良结果²⁴。

原始MPQM算法不考虑色度信息，因此引入了一种MPQM的变体——颜色MPQM (*CMPQM, color MPQM*) 算法。在CMPQM中，首先将颜色分量转换为与亮度成线性的RGB值。然后将RGB值转换为对应于亮度和两个色度通道的坐标值。然后通过滤波器组分析原始信号和失真信号的每个分量。由于HVS对色度不太敏感，因此这些信号仅需要9个空间滤波器和1个时间滤波器。CMPQM的其余步骤与MPQM中的步骤类似。

¹⁶. Q. Huynh-Thu & M. Ghanbari: [Scope of validity of PSNR in image/video quality assessment.](#) ↵

¹⁷. [Final report from the Video Quality Experts Group on the validation of objective models of video quality assessment.](#) ↵

¹⁸. Bernd Girod: [What's wrong with mean-squared error?](#) ↵

¹⁹. [掩蔽效应 \(Masking Effects\)](#)，指由于出现多个同一类别（如声音、图像）的刺激，导致被试不能完整接受全部刺激的信息。 ↵

²⁰. ITU-T J.144: Objective Perceptual Video Quality Measurement Techniques for Digital Cable Television in the Presence of a Full Reference. ↵

²¹. 图像的平坦区域：[图像中属性变化不明显的区域成为图像的平坦区域。](#) ↵

²². JEFFREY LUBIN, A visual discrimination mode for image system design and evaluation. ↵

²³. Christian J. Van den Branden Lambrecht, Olivier Verscheure: [Perceptual quality measure using a spatiotemporal model of the human visual system.](#) ↵

²⁴. <http://www.irisa.fr/armor/lesmembres/Mohamed/Thesis.pdf>. ↵

基于结构相似性的方法

图像是一种高度结构化的信号。图像像素之间（特别是空间上相邻的像素）存在很强的依赖性。像素之间的依赖关系包含镜头中对象结构的重要信息。基于误差敏感的方法中使用明可夫斯基范数度量距离，因此不考虑信号的基础结构。即使可以利用线性变换对图像信号进行分解（如基于误差灵敏度的大多数质量测量所做的那样），但是这种分解并不会消除像素之间的强依赖性。基于结构相似性的质量评估方法试图找到更直接的方法来比较参考信号和失真信号的结构。人类视觉对视野中的结构信息能够做出快速反应。利用HVS的这种特性，基于结构相似性的方法使用度量结构信息变化的方式来近似感知图像的失真，例如：通用图像质量指数（*UIQI, Universal Image Quality Index*）²⁵ 和结构相似性指数（*SSIM, Structural Similarity Index*）²⁶。为了更深入的理解，下面将详细讨论SSIM。

结构相似性指数

用于评估图像感知质量的客观方法尝试利用HVS的各种已知属性来测量失真图像和参考图像之间的可见差异。人类视觉感知系统高度适应从场景中提取结构化信息。在这种假设下，引入了基于结构信息的退化的质量评估方法。

图像中的结构信息被定义为表示场景中对象结构的那些属性，这些属性与亮度和对比度无关。由于亮度和对比度会发生变化，因此结构相似性指数（SSIM）仅考虑局部亮度和局部对比度。由于这三个部分相对独立，因此图像亮度或对比度的变化不会影响图像的结构。

基于结构相似性指标的质量评估系统的系统图如图4-13所示。

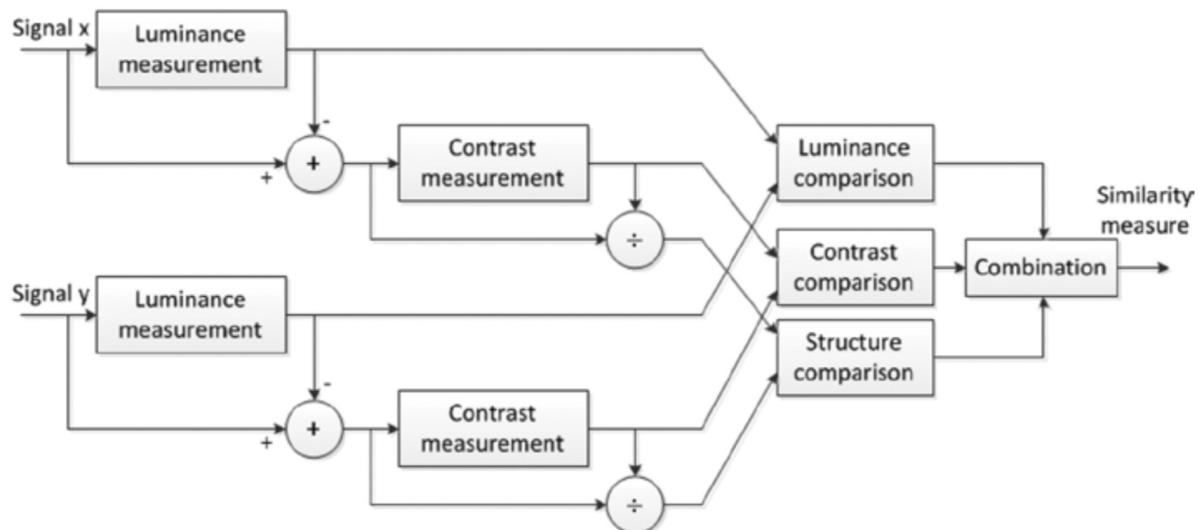


图4-13. SSIM评估系统的流程图

如图4-13所示，该系统由两个非负空间对齐的图像信号 x 和 y 组成。如果其中一个信号具有完美的质量，则相似性度量可以用作度量第二个信号质量的定量测量。该系统把相似性测量的任务分解为三个部分：亮度比较、对比度比较、结构比较。

亮度估计为图像的平均强度 (*mean intensity*) ²⁷: μ 。亮度比较函数 $l(x, y)$ 是关于两个图像 x, y 的平均强度 μ_x, μ_y 的函数。对比度估计为图像像素的标准差: σ 。对比度比较函数 $c(x, y)$ 可以近似为比较两个图像的像素标准差 σ_x, σ_y 。为了进行结构比较，首先将信号除以其自身的标准差来对图像进行归一化，使得两个图像都具有单位标准差。记下来利用归一化信号的 $\frac{x-\mu_x}{\sigma_x}, \frac{y-\mu_y}{\sigma_y}$ 来计算结构比较函数 $s(x, y)$ 。最后，结合这三个不同的结构产生一个整体的相似性度量结果：

$$(4-6) S(x, y) = f(l(x, y), c(x, y), s(x, y))$$

$S(x, y)$ 需要满足如下的条件：

- 对称性: $S(x, y) = S(y, x)$
- 有界性: $S(x, y) \leq 1$
- Unity maximum: $S(x, y) = 1$ 当且仅当 $x = y$; 对于离散信号而言, $S(x, y) = 1$, 当且仅当 $x_i = y_i, \forall i = 1, \dots, N$ 。

亮度比较函数4-7所示：

$$(4-7) l(x, y) = \frac{2\mu_x\mu_y + C_1}{\mu_x^2 + \mu_y^2 + C_1}$$

其中, C_1 为常数, 其目的是为了避免 $\mu_x^2 + \mu_y^2$ 趋于0时的不稳定性。并且, $C_1 = (K_1 L)^2$, L 为图像像素值的动态范围 (8-bit的灰度像素而言, 为255), $K_1 \ll 1$ 为一个很小的常数。从本质上讲, 4-7与韦伯定律²⁸是一致的。韦伯定律广泛用于模拟HVS中的光适应或亮度掩蔽。简而言之, 韦伯定律指出HVS对绝对亮度变化不敏感, 而对相对亮度变化敏感。如果 R 表示与背景亮度相比的相对亮度变化, 则可以用 $\mu_y = (1 + R)\mu_x$ 来表示失真信号的平均强度。此时, 4-7可以重写为:

$$(4-8) l(x, y) = \frac{2(1 + R) + \frac{C_1}{\mu_x^2}}{1 + (1 + R)^2 + \frac{C_1}{\mu_x^2}}$$

当 $C_1 \ll \mu_x^2$ 时, $l(x, y) = f(R)$ 。

对比度比较函数如4 – 9所示：

$$(4 - 9) \quad c(x, y) = \frac{2\sigma_x\sigma_y + C_2}{\sigma_x^2 + \sigma_y^2 + C_2}$$

其中， C_2 为常数，且 $C_2 = (K_2L)^2$ 。

结构比较函数的定义如4 – 10。

$$(4 - 10) \quad s(x, y) = \frac{\sigma_{xy} + C_3}{\sigma_x\sigma_y + C_3}$$

对于离散信号而言，可以用下式来估计 σ_{xy} ：

$$(4 - 11) \quad \sigma_{xy} = \frac{1}{N - 1} \sum_{i=1}^N (x_i - \mu_x)(y_i - \mu_y)$$

最后，结合4-8，4-9，4-10来计算信号 x, y 之间的SSIM值：

$$(4 - 12) \quad SSIM(x, y) = [l(x, y)]^\alpha [c(x, y)]^\beta [s(x, y)]^\gamma$$

$\alpha, \beta, \gamma > 0$ ，并且用来调整三个分量的重要程度。

一般而言，会使用 $\alpha = \beta = \gamma = 1$, $C_3 = \frac{C_2}{2}$ 的简化形式：

$$(4 - 13) \quad SSIM(x, y) = \frac{(2\mu_x\mu_y + C_1)(2\sigma_{xy} + C_2)}{(\mu_x^2 + \mu_y^2 + C_1)(\sigma_x^2 + \sigma_y^2 + C_2)}$$

UIQI是当 $C_1 = C_2 = 0$ 时的一种特性的SSIM。正因如此，当 $\mu_x^2 + \mu_y^2$ 或者 $\sigma_x^2 + \sigma_y^2$ 无限接近于0的时候，UIQI会产生不稳定的结果。

²⁵. Zhou Wang, A.C. Bovik: a universal image quality index. ↪

²⁶. Zhou Wang, A.C. Bovik, H.R. Sheikh, E.P. Simoncelli: image quality assessment: from error visibility to structural similarity. ↪

²⁷. intensity image：强度图，表示单通道图像像素的强度或值的大小。在灰度图像中intensity image指的就是图像的灰度。在RGB颜色空间中intensity image可以理解为是R通道的像素灰度值，或者G通道的像素灰度值，或B通道的像素灰度值，也就是RGB颜色有三个intensity image。在其他颜色空间类似，就是说每个通道图像的像素值。 ↪

²⁸. 韦伯定律：德国生理学家E.H.韦伯通过对重量差别感觉的研究发现的一条定律，

即感觉的差别阈限随原来刺激量的变化而变化，而且表现为一定的规律性。刺激的增量(ΔI)和原来刺激值(I)的比是一个常数(K)，用公式表达即 $K=\Delta I/I$ ，这个常数叫韦伯常数、韦伯分数或韦伯比率。 ↵

基于信息保真度的方法

图像和视频通常会涉及自然场景，对于自然场景而言，需要使用统计模型来对其进行表征。真实世界的失真过程会干扰这些统计数据并使图像或视频信号不自然。这使研究人员将自然场景统计（*NSS, natural scene statistics*）模型与失真模型结合，以量化失真图像和参考图像之间共享的信息量。这种共享信息是信号保真度的一个方面，并且与视觉质量密切相关。与HVS误差敏感度和结构方法相比，统计方法（信息论中也有应用）不依赖于任何HVS参数，也可以产生具有竞争力的FRQA方法（最先进的QA方法相比）。视觉信息保真度（*VIF, visual information fidelity*）方法就是基于信息保真度的视频质量评估指标。

视觉信息保真度

VIF²⁹是基于NSS的图像保真度测量的信息理论标准。VIF测量可以量化大脑从参考图像中提取的信息。然后，利用NSS、HVS和图像失真模型来量化该信息的损失。最终发现，图像的视觉质量与失真图像中存在的相对图像信息密切相关，并且VIF方法优于现有的质量评估算法。此外，VIF的特征仅有一个HVS参数，便于训练和优化以提高性能。

VIF利用NSS模型进行FR质量评估，并使用众所周知的高斯尺度混合（*GSM, Gaussian Scale Mixtures*）对小波域中的自然图像进行建模。图像的小波分析对于自然图像建模是有用的。GSM模型可以捕获自然图像的关键统计特征，例如自然图像中的线性依赖性。

完美的自然图像可以被建模为随机源的输出。在没有任何失真的情况下，该信号通过HVS进入大脑，然后大脑从中提取认知信息。对于失真图像，可以假设参考信号在通过了另一个失真的信道后才进入HVS。

失真模型捕获重要且互补的失真类型：模糊，加性噪声³⁰以及全局/局部的对比度变化。现实世界的失真可以看作是模糊噪声和加性噪声的组合。良好的失真模型对于失真图像和合成图像而言，具备同样的感知结果（*perceptual annoyance*）。失真模型的目标不是模拟图像伪像（*artifacts*）而是模拟伪像的感知。即使失真模型可能无法准确地捕获诸如振铃或块效应之类的失真，失真模型仍然能够捕获它们的感知结果。但是，对于模糊和白噪声之外的失真类型（例如，对于低比特率的压缩噪声），失真模型并不能充分地再现感知结果。

也可以在小波域中描述HVS模型。HVS模型是NSS模型的双重模型（*HVS models are duals of NSS models*），因此HVS的许多方面已经在NSS中捕获，包括：小波信道分解，响应指数和掩蔽效应。HVS会限制流经它的信息量，因此在VIF中，HVS被认为是一

种失真信道。HVS不确定性的所有来源都集中在一个称为视觉噪声的加性高斯白噪声 (*additive white Gaussian stationary noise*)³¹ 中。

VIF将大脑分别从参考中和失真图像的特定子带中提取的信息定义为

$I(C^N; E^N | s^N)$, $I(C^N; F^N | s^N)$ 。直观而言, 视觉质量应该与大脑可以从失真图像中提取的信息量相对于从参考图像中提取的信息量之间的差值有关。例如, 如果大脑可以从失真图像中提取2.0 *bits*/每像素的信息, 当大脑可以从参考图像中提取2.1 *bits*/每像素的信息时, 大部分信息都已被大脑获取, 因此相应的视觉质量应该非常好。相比之下, 如果大脑可以从参考图像中提取5.0 *bits*/每像素的信息, 那么对于每个像素而言, 有3.0 *bits*的信息已经丢失, 此时的视觉质量就应该非常差。

VIF的定义如4 – 14。

$$(4 - 14) \text{ VIF} = \frac{\sum_{j \in \text{subbands}} I(C^{N,j}; F^{N,j} | s^{N,j})}{\sum_{j \in \text{subbands}} I(C^{N,j}; E^{N,j} | s^{N,j})}$$

$C^{N,j}$ 为随机字段 C_j 的N个元素构成的向量, 并用来表示子带 j 中的系数。

$\text{VIF} < 0$, 表示所有信息在失真信道中丢失。如果测试图像只是它本身的一个副本, 并没有扭曲, 所以VIF为1。因此, VIF始终在[0, 1]的范围内。值得注意的是, 因为参考图像的线性对比度的增强不会增加噪声, 这将导致VIF值大于1, 从而表示增强图像的视觉质量优于参考图像。这是VIF独有的, 其它VQA指标所不具备的属性。

²⁹. Image Information and Visual Quality. ↪

³⁰. 加性噪声一般指热噪声、散弹噪声等, 它们与信号的关系是相加, 不管有没有信号, 噪声都存在。 ↪

³¹. 加性高斯白噪声 (AWGN) 从统计上而言是随机无线噪声, 其特点是其通信信道上的信号分布在很宽的频带范围内。 ↪

时空方法

传统的FR客观质量指标没有考虑时间维度的失真，例如帧丢失 (*frame drops*) 或抖动 (*jitter*)。时空 (*spatio-temporal*) 方法会考虑视频帧之间的运动信息，从而会捕获时间质量的劣化。因此，时空方法更适合于视频信号的质量评估。时空算法通常与HVS之间有很好地相关性。本节将介绍其中的一种方法：时空视频SSIM (*stVSSIM, spatio-temporal video SSIM*)。

时空视频结构相似性指数

时空视频SSIM (stVSSIM) ³² 算法是一种基于运动的视频完整性评估 (*MOVIE, motion-based video integrity evaluation*) ³³ 的全参考的视频质量评估算法。MOVIE利用多尺度时空Gabor滤波器组来分解视频并计算运动矢量。但是，MOVIE具有高计算复杂性，这使得在实际应用中很难实现。因此，stVSSIM提出了一种新的时空度量指标来解决MOVIE的计算复杂性问题。在VQEG的全参考数据集上对stVSSIM算法进行了评估，并且发现stVSSIM算法与人类的感知具备很好的相关性。

对于空间质量评估，stVSSIM使用单尺度结构相似性指数 (*SS-SSIM, single-scale structural similarity index*)，因为SS-SSIM与人类对视觉质量的感知有很好地相关性。对于时间质量评估，stVSSIM将SS-SSIM扩展到时空域并将其称为SSIM-3D。与MOVIE中使用的光流 (*optical flow*) 相反，SSIM-3D使用基于块的运动估计算法将运动信息融合到stVSSIM中。此外，SSIM-3D还引入了一种避免块运动估计的方法，从而降低了计算复杂度。

SS-SSIM逐帧计算视频的空间质量，并且使用百分位方法计算帧的质量指标。对于包含低质量区域的图像质量而言，人类倾向于给出更差的评级。因此使用百分位 ³⁴ 方法将提高算法准确性。Percentile-SSIM或P-SSIM应用于每帧获得的分数。具体而言，帧质量度量方法如下：

$$(4-15) \quad S_{frame} = \frac{1}{|\varphi|} \sum_{i \in \varphi} SSIM(i)$$

利用每一帧的得分的平均数来表示视频的空间得分，并用符号 S_{video} 表示。

利用视频的三维结构相似性 (SSIM-3D) 来评估时间质量，并对由运动信息 (运动信息从运动矢量推导而出) 计算而来的得分进行加权。在这种情况下，可以将视频看作一种三维信号。如果 x 和 y 分别代表参考视频和失真视频，那么该3维空间中的某个像素坐标 (i, j, k)

周围的空间可以用二维空间坐标(α, β)和时间维度来表示，其中时间维度会包含 γ 个视频帧。 (i, j) 表示空间位置， k 则表示对应的视频帧的序号。因此，SSIM-3D可以使用SSIM的3D扩展形式来表示，具体如下所示：

$$(4-16) SSIM_{3D} = \frac{(2\mu_{x(i,j,k)}\mu_{y(i,j,k)} + C_1)(2\sigma_{x(i,j,k)}y(i,j,k) + C_2)}{(\mu_{x(i,j,k)}^2 + \mu_{y(i,j,k)}^2 + C_1)(\sigma_{x(i,j,k)}^2 + \sigma_{y(i,j,k)}^2 + C_2)}$$

stVSSIM的本质是评估像素在各个方向上的时空质量，然后利用某个加权方案计算该像素的时空质量指数。加权因子取决于所使用的滤波器的类型。

为了考虑运动信息，需要使用块运动估计。运动估计在 8×8 的像素块上，使用ARPS算法（*Adaptive Rood Pattern Search*）在相邻帧之间计算运动矢量。一旦每个像素 (i, j, k) 的运动矢量可用，则对时空SSIM-3D分数进行加权。为了避免浮点数加权，需要执行贪心加权。像素 (i, j, k) 的时空分数是从四个滤波器产生的分数中筛选出来的（筛选的过程中会基于滤波器的类型），并且筛选出来的分数最近接像素 (i, j, k) 的运动方向。例如，如果像素的运动矢量是 $(u, v) = (0, 2)$ ，则该像素的时空分数将是由垂直滤波器产生的SSIM-3D值。如果运动矢量与两个滤波器平面等距，则时空分数是两个滤波器的SSIM-3D分数的平均值。在零运动的情况下，时空分数是所有四个SSIM-3D值的平均值。

视频的时间分数为帧级分数的平均值，并表示为 T_{video} 。视频的最终得分由 $S_{video} \times T_{video}$ 计算得出。

³². Efficient Motion Weighted Spatio-Temporal Video SSIM Index. ↪

³³. Motion-based Perceptual Quality Assessment of Video. ↪

³⁴. **percentile**, 百分位数，统计学术语，如果将一组数据从小到大排序，并计算相应的累计百分位，则某一百分位所对应数据的值就称为这一百分位的百分位数。可表示为：一组n个观测值按数值大小排列。如，处于p%位置的值称第p百分位数。 ↪

基于显著性的方法

适用于单个图像的质量评估方法通常也用于视频。然而，这些方法没有考虑视频序列的运动信息。这导致对视频质量评估而言，这些用于评估单个图像质量的方法表现很差。另外，大多数VQA算法忽略了人类视觉注意机制，这是HVS的一个重要特征。

视觉注意力机制实质上是一种生物机制，这种机制能够从外界复杂的环境中选出重要的、所需要关注的信息，并逐步排除相对不重要的信息。通过这种方式能够将十分复杂的外界视觉场景进行简化和分解，进而在接下来对重要的信息进行进一步处理。这种机制的优势在于它能够使得我们在十分复杂的外界视觉场景环境中，可以十分迅速的注意所需要关注的重要的信息和物体。在图像理解和分析中，人类视觉系统的视觉注意使得人们可以在复杂的场景中选择少数的感兴趣区域作为注意焦点（FOA, *Focus Of Attention*），并对其进行优先处理，从而极大地提高视觉系统处理的效率。在日常生活中，我们会常常感受到视觉注意机制的存在。比如说我们会轻易的发现，墙壁上的小坑和黑点，白色打印纸上的纸张缺陷，蓝色车牌上的车牌号码等等。³⁵

人眼的注意力通常会聚焦于与其相邻区域不同的、具有较高对比度或显著区域的边缘。例如上文提到的：我们可以轻易的发现，墙壁上的小坑和黑点。利用人眼注意力的这一特点，基于显著性的视频质量评估方法采用不对称地方法处理显著区域中发生的扭曲（显著区域和非显著区域区分对待）。SVQA就是一种这样的方法³⁶。

基于显著性的视频质量评估

在基于显著性的视频质量评估（SVQA）方法中，使用称为四元数傅里叶变换的相位谱（PQFT, *phase spectrum of quaternion Fourier transform*）的快速频域方法从参考图像或视频帧中提取空间显著图（saliency map）。当对图像相位谱进行逆傅里叶变换时，很容易识别出图像的显著区域。可以使用显著图来调整其他VQA的客观评估标准，例如PSNR, MSSIM, VIF等。和时空方法类似，SVQA也从相邻帧确定时间权重。

给定一个参考图和其对应的失真图，可以用PQFT算法提取参考图像的显著图（saliency map）。然后，利用显著图对原始索引进行加权处理，进而确定改进的质量评估指数——基于显著性的指数（S指数）。例如，对于显著区域中的第*i*个像素，其亮度值为*p_i*，其显著性权重*w_i*的定义如下：

$$(4 - 17) \quad w_i = \frac{p_i + b}{\frac{1}{M*N} \sum_{i=1}^{M*N} (p_i + b)}$$

其中， b 为一个非常小的常数，其目的在于保证 $w_i > 0$ 。 M, N 分别表示图像的宽和高。从公式可以看出，该权重的计算也会考虑图像的非显著区域。但是显著区域的权重会更大。利用这些权重，基于显著性的PSNR算法（*SPSNR, saliency-based PSNR*）可以定义为：

$$(4-18) \quad SPSNR(x, y) = 10 \log \frac{255^2}{SMSE(x, y)},$$

$$SMSE(x, y) = \frac{1}{M * N} \sum_{i=1}^{M*N} (x_i - y_i)^2 w_i$$

因此，显著区域中的像素的失真比其他区域中的像素更重要。基于显著性的MSSIM（*SMSSIM*）和基于显着性的VIF（*SVIF*）也以类似的方式定义。

由于SVQA不仅处理图像信号，还处理视频信号，因此需要考虑以下因素：

1. HVS对运动信息敏感，但对背景信息不敏感。因此，运动物体的失真比背景的失真更重要。SVQA在定位运动物体时需要区分固定摄像机和移动摄像机。
2. 由于视频帧是实时播放的，因此在观看视频时，人眼能注意到的区域比观看固定图像时要小得多。需要在帧内权重中考虑这个特性。
3. 由于运动掩蔽效应，在大规模场景变化期间或物体高速运动期间，视觉灵敏度会被抑制。因此，帧应根据运动掩蔽进行不同的加权。需要在帧间权重中考虑这个特性。
4. 考虑到视频序列的时空特性，最终质量指标需要综合考虑视频的空间域和时域中的显着性权重。

在SVQA中，帧内权重使用PQFT来计算具有非零运动的像素的显著性图。非零运动表示为两个相邻视频帧之间的图像差异。这样，就首先解决了关于运动物体的第一个需要考虑的因素。在较短的时间间隔中，允许显著图具有正方形的处理区域，从而解决了第二个需要考虑的因素。由于基于运动遮蔽来计算帧间权重，因此还解决了加权的第三个问题。最后，在SVQA中会同时考虑帧内权重和帧间权重，从而解决需要考虑的第4个问题。图4-14展示了SVQA的框架。

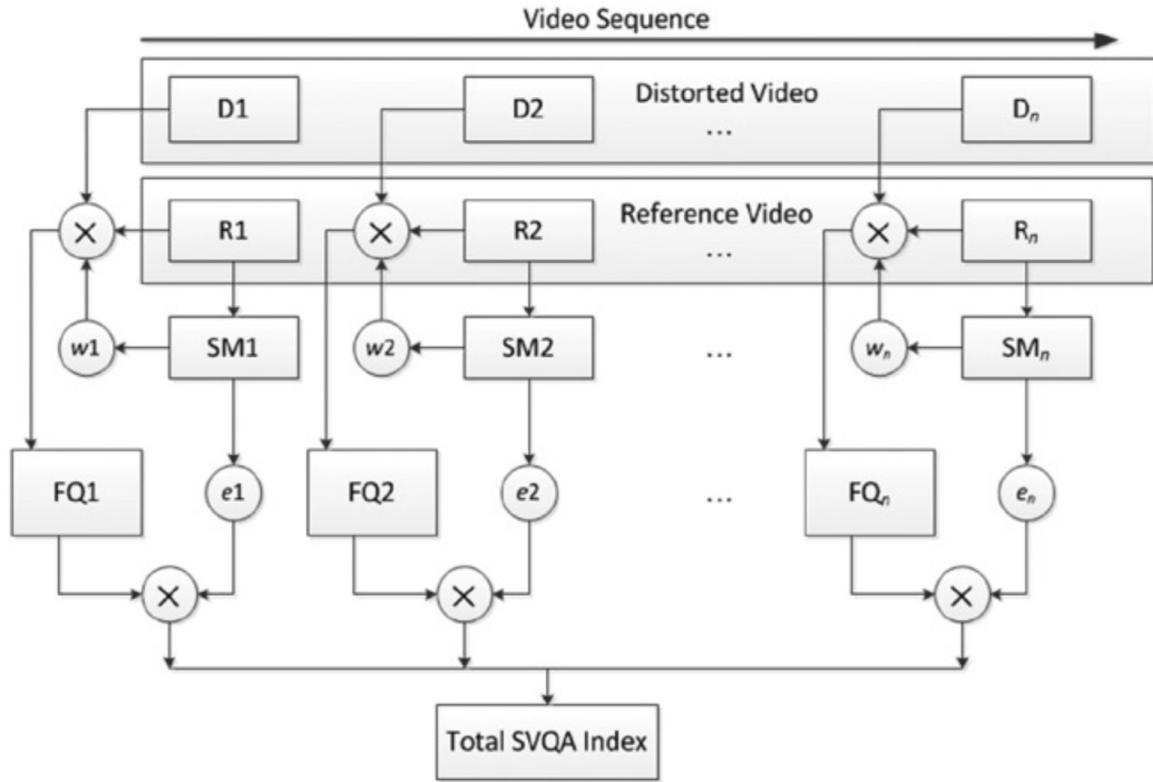


图4-14. SVQA的框架图

根据图4-14所示，SVQA的整体流程如下：

1. 参考视频和失真视频序列分解成视频帧。对于失真视频，每一个视频序列包含 $D_1 \sim D_n$ 的 n 张图像。对于参考视频而言，则为 $R_1 \sim R_n$ 的 n 张图像。如图4-14的上部分所示。
2. 在相机移动时，画面的所有像素都在移动，因此需要定义一个二元函数来检测这种运动。如果同一像素在相邻帧之间的亮度差超过某个阈值，则认为发生移动。如果所有的像素都在移动，则可以认为是相机在移动。否则，认为运动对象在静态背景中移动。帧的运动信息和三个颜色通道信息的加权组合构成帧的四元数（quaternion）。
3. PQFT利用运动信息计算每一个参考视频帧的显著图（SM）， SM_1, \dots, SM_n 。
4. 基于 SM 来计算视频帧的帧内权重： (w_1, \dots, w_n) 。
5. 利用基于显著的度量指标（例如SPSNR, SMSSIM, SVIF）来计算第 i 帧的质量 FQ_i 。
6. 基于 SM 计算视频帧的帧间权重： (e_1, \dots, e_n) 。
7. 利用4 – 19计算整个视频的质量，式中， n 为视频帧的数量。

$$(4 - 19) \quad SVQA = \frac{\sum_{i=1}^n (FQ_i e_i)}{\sum_{i=1}^n e_i}$$

35. 视觉注意机制理论分析,

[https://wenku.baidu.com/view/0bfdeeab453610661fd9f460.html ↵](https://wenku.baidu.com/view/0bfdeeab453610661fd9f460.html)

36. New Strategy for Image and Video Quality Assessment. ↵

网络感知方法

类似PSNR的客观视频质量指标，既不能与视觉的感知质量完美匹配，也不考虑有损网络环境中（例如，多跳无线网状网络）的数据分组丢失。虽然PSNR和类似指标可用于评估桌面编码应用程序中的视频质量和有线网络上的流式传输，但当它们用于评估无线网络上的视频质量时，会出现明显的不准确性。

例如，在无线环境中，与原始未失真视频相比，具有38dB左右的PSNR的视频流（在桌面视频编码应用中通常被认为是中高质量）实际上被认为具有同等的质量。这是因为无线视频应用通常使用UDP协议，UDP协议不保证可靠的传输并且允许数据丢失以满足延迟要求。通常，与有线网络相比，在不稳定的无线信道组成的无线局域网（WLAN）中的丢包的概率要高很多。在这样的网络环境中，丢失连续的分组数据可能导致整个帧的丢失。因此，与桌面视频编码应用相比，此时会进一步降低视频的感知质量。

修正的PSNR

为了解决视频帧的丢失问题，提出了修正的PSNR方法（*MPSNR, Modified PSNR*）³⁷。基于PSNR和主观MOS的线性回归，提出了两个客观指标。

1. 第一个指标称为基于PSNR的客观MOS（*POMOS, PSNR-based Objective MOS*）指标。POMOS利用平均PSNR来预测MOS，并且与MOS之间的相关性达到了0.87。
2. 第二个指标称为基于速率的客观MOS（*ROMOS, Rate-based Objective MOS*）指标。ROMOS增加了诸如帧丢失率之类的网络参数，并且与MOS之间的相关性高达0.94。

帧丢失现象在无线网络中非常普遍，但传统的PSNR计算中却没有考虑这个因素。流式传输期间的分组数据丢失可能会导致帧丢失。然而，用户通常无法识别这种视频帧的丢失。但是，丢失的帧会导致在PSNR计算期间将错误的帧与原始帧进行比较。这种偏离位置的比较会导致PSNR的降低。解决此问题的方法是在源视频中引入时序信息。但是对源视频的这种修改并不受欢迎。

要确定是否存在帧丢失的情况，另一种方法是将帧与原始帧进行匹配。该算法假设当所有帧匹配时所有帧的PSNR之和最大化，并且使用该PSNR的和来确定不匹配帧。*MPSNR*将流式视频中的每个帧与参考视频中的帧进行匹配，从而使得所有匹配帧之间的PSNR之和最大化。并且，需要一个移动窗口来确定匹配帧的位置。如果流式视频中的帧 j 与参考视频中帧 k 匹配，则认为帧 $(k - j)$ 丢失。但是需要注意的是，这种匹配的方法仅仅适用于无损传输的信道（此处的无损特指画面质量无损，而不是上文提到的数据丢失）中。对于存在画质损失的传输信道而言，因为画质已经收到折损，所以无法进行这种匹配查找。

MPSNR还测量了如下的视频流参数：

1. Distorted frame rate(d)：视频流中不匹配的帧的比例。
2. Distorted frame PSNR($dPSNR$)：所有不匹配的帧的平均PSNR。
3. Frame loss rate (l)：视频流中丢失的帧的占比。该参数通过比较接收的流式视频的帧数与参考视频中的帧数来计算。

一旦流式视频和参考视频中的相应帧完成匹配，并且已经计算出流式视频中的每个帧的PSNR，则很容易获取上述提到的所有参。

在MPSNR模型中，这种匹配方法可以应用于训练视频集，并且计算窗口 W 的平均PSNR。实验结果表明，平均PSNR与主观MOS呈线性关系。因此，可以用平均PSNR的线性模型来预测MOS得分。PSNR的线性模型如下所示：

$$(4 - 20) POMOS = 0.8311 + 0.0392(\text{average } PSNR)$$

需要注意的是，上述模型中使用了平均PSNR。对于完全匹配的帧而言，其平均PSNR为无穷大（或非常高的值），因此这会影响MOS的预测。为了缓解这个问题，提出了另一种不使用PSNR的线性模型：

$$(4 - 21) ROMOS = 4.367 - 0.5040 \frac{d}{dPSNR} - 0.0517l$$

³⁷. Metrics for Evaluating Video Streaming Quality in Lossy IEEE 802.11 Wireless Networks. ↪

基于噪声的质量指标

还有一种质量评估方法不是通过评估信号的保真度，而是通过评估引入的噪声来评估视频的质量。

噪声质量测量

在噪声质量测量 (*NQM, noise quality measure*)³⁸ 中，劣化图像被建模为线性频率失真和注入加性噪声 (*additive noise*) 的原始图像。NQM认为这两种噪声源是独立的，并且解耦为两种质量测量：

1. 频率失真导致的失真测量 (*DM, distortion measure*)
2. 加性噪声导致的噪声质量测量 (*NQM, noise quality measure*)

NQM是一种基于对比金字塔 (*CP, Contrast Pyramid*)³⁹ 的测量指标，并会考虑以下因素：

- 对比灵敏度会随着距离，图像尺寸以及空间频率的变化而变化
- 局部亮度平均值也会发生变化
- 空间频率之间的对比度之间会相互作用
- 对比度会产生掩蔽效应

对于加性噪声，非线性NQM比PSNR、线性质量测量的效果要更好。

计算DM分为三步：

1. 找到劣化图像中的频率失真
2. 计算该频率失真与单位增益 (*unity gain*)（即无失真）的全通响应的偏差
3. 利用HVS的频率响应模型对偏差进行加权，并且在可见频率上对得到的加权偏差进行积分

³⁸. Image Quality Assessment Based on a Degradation Model. ↪

³⁹. 对比度金字塔源于图像的高斯金字塔分解。最高级第N级对比度金字塔是其高斯金字塔本身，除此以外各级对比度金字塔是这一级的高斯金字塔的图像与上一级作扩大后图像之比，即上一级扩大后的图像被看作为背景，比值含有对比度的意义，故称对比度金字塔。[基于金字塔方法的图像融合原理及性能评价](#) ↪

客观编码效率指标

测量编码效率是另一种指标，主要用于视频编码应用中权衡视觉质量和码率成本。本节将讨论用于客观确定编码效率的流行的BD(*Bjontegaard delta*)指标。

BD-PSNR, BD-SSIM, BD-Bitrate

Bjontegaard delta PSNR (BD-PSNR) ⁴⁰ 是编码器相对于参考编码器的编码效率的客观测量。Gisle Bjontegaard于2001年4月在视频编码专家组 (VCEG) 会议上提出BD-PSNR方法。BD-PSNR考虑两种编码方案在实现特定质量时需要的比特数方面的相对差异。BD-PSNR计算一个区间上的两个码率-失真 (*R-D, rate-distortion*) 曲线之间的平均PSNR差异。因为BD-PSNR考虑了在实现特定视觉质量 (质量可以由PSNR来表示) 下的成本 (例如，所使用的比特位数)，因此，BD-PSNR是视频视觉质量的良好指示。在绘制R-D数据点时，可以使用 $\log_{10}(\text{bitrate})$ ，从而产生更直的R-D曲线和更均匀的数据点分布。

BD-PSNR使用三阶对数多项式来拟合给定的R-D曲线，PSNR中的重建失真如下：

$$(4 - 22) D_{PSNR} = D(r) = a + br + cr^2 + dr^3$$

其中， $r = \log_{10}(R)$ ， R 为输出的码率， a, b, c, d 为拟合参数。

该模型可以非常好的拟合R-D曲线，并且对于奇异点 (*singular points*) 也不会产生任何问题。利用从实际编码中获取的四个R-D数据点来求解式4 – 22，从而确定四个拟合参数 a, b, c, d 。因此，该等式可用于插值⁴¹两条R-D曲线（该曲线从两个编码器中获取），并且两条曲线之间的 $\Delta PSNR$ 可以用4 – 23得到。

$$(4 - 23) BD - PSNR = \frac{1}{r_H - r_L} \int_{r_L}^{r_H} \left(D_2(r) - D_1(r) \right) dr$$

其中， $r_H = \log_{10}(R_H)$, $r_L = \log_{10}(R_L)$ 分别为输出码率范围的最大值和最小值。 D_1, D_2 分别为两条R-D曲线。

类似的，可以使用4 – 24来表示关于SNR的码率函数。

$$(4 - 24) r = a + bD + cD^2 + dD^3$$

其中， $r = \log_{10}(R)$ ， R 为输出码率， a, b, c, d 为拟合参数， D 为PSNR定义的质量失真。因此，BD-bitrate可以用4 – 25表示。

$$(4 - 25) BDBitrate = \frac{1}{D_H - D_L} \int_{D_L}^{D_H} (r_2 - r_1) dD$$

因此，BD-PSNR可以同时包含如下的两点：

1. 整个码率区间中的平均PSNR差异
2. 整个PSNR区间中的平均码率差异

如果用SSIM而不是PSNR来表示失真，则可以用相同的方式获得BD-SSIM。BD-PSNR/BD-SSIM计算取决于基于一组R-D数据点的插值多项式。

优势

BD度量具有以下优点：与单独使用R-D曲线相比，BD能够更准确地表示质量差异。在大量测试的情况下，BD度量可以很容易的给出各种参数下的两个编码方案之间的差异。此外，BD度量可以将来自多个测试的结果合并到单个图表中，同时显示两个编码方案之间的视频质量差异。这种对比可以有效地传达两个编码方案之间质量对比的整体情况。

限制

BD指标在比较两种编码方案时非常有用。然而，对于超高清（UHD）视频序列，BD度量可能会给出非预期的结果⁴²。这些非预期的行为主要由于多项式曲线拟合和视频序列中的高频噪声。当使用高次多项式时，标准多项式插值容易受到龙格现象（Runge's phenomenon）⁴³的影响。即使仅有四个数据点（对于三次多项式而言），一些插值曲线也会发生可能导致BD-PSNR评估不准确的振荡现象。

用样条曲线（splines）⁴⁴替代插值法可以降低由龙格现象引起的误差，同时仍然可以通过测量的率失真数据点提供精准的拟合曲线。在有的视频样本中，使用分段三次样条插值将BD-PSNR计算的精度提高了近1 dB（和多项式插值的结果相比）。

当多项式插值发生振荡时，得到的BD-PSNR可能会显着偏斜。图4-15显示了两个编码样本的rate-PSNR曲线中的多项式插值问题。图表显示了使用多项式插值和三次样条插值的差异，同时还显示了使用两种方法时各自的PSNR值。

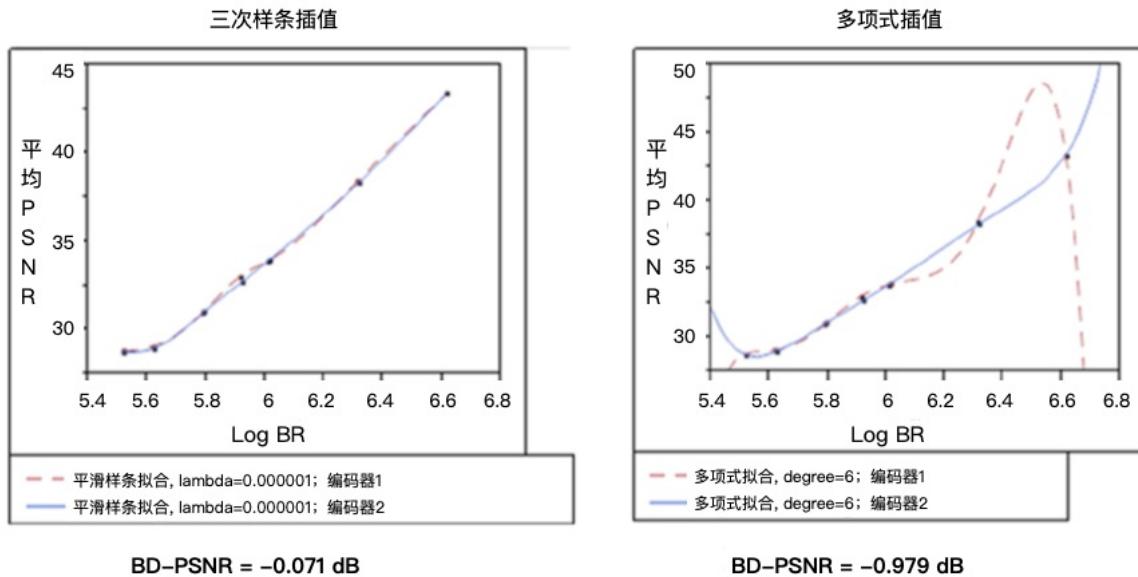


图4-15.多项式插值在R-D曲线中的震荡的例子

图4-15中，两个编码器的码率和平均PSNR非常相似，也显示出基于多项式插值的BD-PSNR并不能精准的评估两个编码器之间的差异。

另外，BD-PSNR没有考虑编码复杂性，这对于实际视频应用来说是一个关键问题，特别是对于那些计算能力，存储资源和电源都有限的移动设备而言。可以通过广义的BD-PSNR (*generalized BD-PSNR*) 度量来解决这些限制。关于广义的BD-PSNR，我们将在下一节中介绍。

广义BD-PSNR

广义BD-PSNR (*GBD-PSNR*) ⁴⁵是通过将BD-PSNR从R-D曲线拟合推广到速率-复杂度-失真 (*R-C-D, rate-complexity-distortion*) 的表面拟合而开发出的编码效率度量。GBD-PSNR涉及编码复杂度的测量，R-C-D表面拟合，以及计算两个R-C-D表面之间的PSNR差异。

通常，编码复杂性是多维的并且需要考虑若干因素，包括计算复杂度，数据高速缓存大小，存储器访问带宽，存储复杂性，指令高速缓存大小，并行性和流水线。但是，很难在实践中同时考虑所有的因素。广泛使用的替代方案是在给定平台上评估编码时间。这种方案不仅可以用来表示计算复杂性，而且还部分地反映了其它维度（例如编码过程中的存储器访问）的复杂性。

为了计算R-C-D曲面拟合，定义R-C-D函数如下：

定义4 – 1： 码率-复杂度-失真函数 $D(R, C)$ 是失真 D 的下限，从而使得在给定 (R, C) 对下，三元组 (R, D, C) 仍然位于信源的 (R, C, D) 空间区域中。

因此， $D(R, C)$ 函数在 R 坐标和 C 坐标中均为非增函数。与R-D凸函数类似，R-C-D函数也是凸的。基于这些属性，可以使用指数模型来逼近 $D(R, C)$ 。为了保持BD-PSNR的向后兼容性，同时在准确度和拟合复杂度之间获得良好的折衷， $D(R, C)$ 可以逼近为：

$$(4 - 26) D(R, C) = a_0 r^3 + a_1 r^2 + a_2 r^1 + a_3 + a_4 c^2 + a_5 c$$

其中， a_0, \dots, a_5 为拟合参数， $r = \log(r)$, $c = \log(C)$, R 为输出的码率， C 为编码复杂度， D 为PSNR中的失真。

为了利用该等式拟合R-C-D曲面，需要从际编码中获取至少六个 (R, C, D) 三元组数据点。然而，在实践中，更多的 (R, C, D) 数据点将带来更好的准确度。通常使用20个数据点来拟合这样的曲面。

和BD-PSNR类似，两个R-C-D曲面之间的平均PSNR差异可以用下式计算得出：

$$(4 - 27) \Delta P_{GBD} = \frac{\int_{r_L}^{r_H} \int_{c_L}^{c_H} (D_2(r, c) - D_1(r, c)) dc dr}{(r_H - r_L)(c_H - c_L)}$$

其中， DP_{GBD} 为广义BD-PSNR（GBD-PSNR）， $D_1(r, c), D_2(r, c)$ 为两个R-C-D曲面的拟合函数， r_H, r_L, c_H, c_L 构成R-C空间的有界区域并且分别为 R_H, R_L, C_H, C_L 的对数形式。

由于某些算法的复杂性，两个R-C-D表面可能没有R-C交叉点。在这种极端情况下，GBD-PSNR是未定义的。

限制

有些时候，GBD-PSNR可以覆盖的编码复杂度的范围是有限的。当两个编码器的编码复杂度差异很大时，会导致两个R-C-D曲面仅存在相对较小的重叠区域时，此时就会发生这种情况。

此外，编码复杂性取决于平台和实现。尽管GBD-PSNR在不同平台上显示出良好的一致性，但在不同的平台上，GBD-PSNR值仍然略有差异。

⁴⁰. B.Bjontegaard: Calculation of Average PSNR Differences between RD curves.



⁴¹. 在离散数据的基础上补插连续函数，使得这条连续曲线通过全部给定的离散数据点。插值是离散函数逼近的重要方法，利用它可通过函数在有限个点处的取值状况，估算出函数在其他点处的近似值。



⁴². On the Calculation of PSNR and Bit Rate Differences for the SVT Test Data.



43. 在《计算方法》和《计算机图形学基础》中讲到插值（线性插值、抛物线插值、高次lagrang插值）的拟合度，在三种自由曲线的图形中，是上升趋势。我们总以为次数越高精度越高。实际上，当点数n增大（次数 $m=n-1$ 也增大）时，会在两端产生激烈的震荡，出现函数不收敛的现象，即所谓的龙格现象。在数值分析中，高次插值会产生龙格现象。即在两端处波动极大，产生明显的震荡。[←](#)

44. 所谓样条曲线(Spline Curves)是指给定一组控制点而得到一条曲线，曲线的大致形状由这些点予以控制，一般可分为插值样条和逼近样条两种，插值样条通常用于数字化绘图或动画的设计，逼近样条一般用来构造物体的表面。[←](#)

45. Rate-Complexity-Distortion Evaluation for Hybrid Video Coding.[←](#)

基于ITU-T标准的客观的质量度量方法

除了前面几节介绍的客观指标外，还有一些基于ITU-T标准的客观的质量度量方法。

视频质量指标

视频质量指标（VQM, *video quality metric*）⁴⁶是在国家电信和信息管理局（NTIA）开发的感知视频质量的客观测量。由于其在VQEG第2阶段验证测试中的出色性能，美国国家标准协会（ANSI）将VQM作为国家标准，就像ITU将ITU-T Rec J. 144⁴⁷作为标准一样。VQM测量视频中损伤的感知效果（包括模糊，抖动，全局噪声，块效应和颜色失真等），并将它们组合成单个指标。测试结果表明，VQM与主观视频质量评估之间具有高度相关性。

VQM算法把源视频剪辑和处理过的视频剪辑作为输入，并分四步计算VQM：

1. 校准：在此步骤中，校准采样视频以准备接下来的特征提取。相对于源视频对处理后视频的空间/时间偏移，对比度和亮度偏移进行估计和校正。
2. 质量特征提取：在该步骤中，使用数学函数，从视频流的时空子区域提取表征空间、时间、颜色属性中的感知变化的质量特征集合。
3. 质量参数计算：在该步骤中，通过比较处理视频提取的特征与源视频提取的特征来计算描述视频质量的感知变化的一组质量参数。
4. 计算VQM：使用先前步骤计算出的参数的线性组合来计算VQM。

可以使用基于某些优化标准的各种模型来计算VQM。这些模型包括电视模型，视频会议模型，通用模型，开发人员模型和PSNR模型。VQM通用模型会使用七个参数的线性组合来计算最终的VQM。其中，四个参数基于从亮度分量的空间梯度中提取的特征计算而来，两个参数基于从由两个色度分量形成的矢量中提取的特征计算而来，并且最后一个参数基于对比度和绝对时间信息（两者均提取自亮度分量）。测试结果显示，主观测试与VQM通用模型（VQMG, *VQM general model*）之间的相关系数高达0.95。

ITU-T G.1070 and G.1070E

ITU-T G.1070⁴⁸是用于体验质量（QoE）规划的标准计算模型。G.1070模型最初是为双向视频通信而开发的，目前已经受到广泛使用、研究、扩展和增强。在G.1070中，视觉质量模型会基于多个因素，包括：帧率，码率和分组数据丢失率。对于固定帧率和固定分

组数据丢失率的场景，码率的降低将导致G.1070视觉质量的降低。然而，码率的降低并不一定意味着质量的降低。如果视频内容具有低复杂度且易于编码，则即便码率较低，也可能不会存在质量损失。但G.1070却无法区分如上的这两种情况。

给定视频的码率，帧率和分组数据丢失率，G.1070模型可以用这些信息估计用户端对视频的感知质量，该估计的质量通常以质量分数的形式存在。一般而言，高码率的压缩视频，该质量得分一般较高；而较低码率的压缩视频，该分数一般也较低。

为了计算G.1070的视觉质量，评估系统会包含用于分析压缩比特流相关信息的数据收集器或估计器。该数据收集器用于提取有用信息，估计码率，帧率和分组数据丢失率。根据这三个估计，G.1070视频质量估计器根据G.1070建议的第11.2节中定义的函数来计算视频质量估计。

尽管G.1070模型也适用于估计网络质量（例如预期的分组数据丢失率），但G.1070一般不考虑视频的内容信息。例如，具有复杂背景和高速运动的视频场景，以及包含相对较少的活动或纹理的另一场景，即使它们以相同的码率和帧率进行编码，也会有显著不同的感知质量。而且，简单场景的高质量编码所需的编码码率本来就相对较低。由于G.1070模型通常为低码率视频提供较低的分数，因此对于简单的场景，G.1070存在这种不合理地惩罚。实际而言，该视频场景的感知质量可能很高。同理，G.1070也可能会高估高码率视频的感知质量。因此，G.1070模型可能与最终用户的主观质量得分无关。

为了解决上文提到的G.1070与主观质量得分之间可能存在的不相关性的问题，提出了G.1070的改进模型——G.1070E⁴⁹。G.1070E会考虑帧复杂度（相当于考虑帧的内容复杂度），并且提供了帧复杂度的估计方法。基于帧复杂度，然后执行码率归一化处理。最后，G.1070使用归一化的码率，帧率以及分组数据丢失率计算视频质量的估计。

G.1070E是一种无参考的压缩域下的客观视频质量度量模型。实验结果表明，G.1070E模型与主观MOS得分具有较高的相关性。G.1070E能够比G.1070更好地反映视频的体验质量。

ITU-T P.1202.2

ITU-T P.1202系列文件规定了基于基于IP的视频服务的视频质量的模型，该模型通过分组报头和比特流信息来监控视频的质量。ITU-T P.1202.2⁵⁰建议书规定了具备更高视频分辨率应用的质量模型。P.1202.2具有两种模式：

- 模式1：视频比特流被解析但是并未被解码成像素
- 模式2：视频比特流被完全解码成像素以用于分析

P.1202.2是一种无参考的视频质量指标。该算法的执行需要如下的步骤：

1. 提取视频的基本参数，例如：帧分辨率，帧级别的量化参数，帧的大小，帧的数量。

2. 将这些基本参数聚合到图像级别，从而确定视频帧的复杂度。
3. 将基本参数聚合到模型级别，以获得视频序列复杂度以及视频序列级别的量化参数。
4. 利用4 – 28所示的质量评估模型来评估MOS。

$$(4 - 28) P.1202.2 \text{ MOS} = f(\text{frame QP}, \text{frame resolution}, \text{frame size}, \text{frame number})$$

研究发现，P.1202.2算法的估计的MOS结果和VQEG JEG(*Joint Effort Group*)估计的MOS结果类似，均具有相似的皮尔逊线性相关系数 (*Pearson linear correlation coefficient*) 和斯皮尔曼等级相关系数 (*Spearman ranked order correlation coefficient*) ⁵¹。这种线性关系如下所示：

$$(4 - 29) VQEG JEG \text{ MOS} = -0.172 * \text{frame QP} + 9.249$$

但是，P.1202.2和VQEG JEG的预测结果逗比MS-SSIM(*multi-scale structural similarity method*, 多尺度结构相似性)要差。研究还发现，P.1202.2模型并不能捕获压缩噪声 (*compression artifacts*)。

因此，提出了一种基于MS-SSIM的改进的FR MOS估计器。特别是，开发了基于MS-SSIM的重映射功能。得到的MOS估计是关于MS-SSIM和帧参数（例如帧级量化参数，帧大小，帧类型和分辨率）的函数。该算法首先执行设备和内容分析，然后进行空间复杂度计算。

然后，使用逻辑函数 (*logistic function*) 执行非线性模型拟合。然后MOS估计器利用这些结果以及MS-SSIM值来计算MOS的估计。实验结果表明，对于一组测试集，估计的MOS与MOS之间的皮尔逊相关系数 > 0.9 ，这比MS-SSIM的0.7265要好得多。

⁴⁶. A New Standardized Method for Objectively Measuring Video Quality. ↵

⁴⁷. ITU-T Recommendation J.144: Objective Perceptual Video Quality Measurement Techniques for Digital Cable Television in the Presence of a Full Reference. ↵

⁴⁸. ITU-T Recommendation G.1070: Opinion Model for Video-Telephony Applications. ↵

⁴⁹. Efficient Frame Complexity Estimation and Application to G.1070 Video Quality Monitoring. ↵

⁵⁰. ITU-T Recommendation P.1202.2: Parametric Non-intrusive Bitstream Assessment of Video Media Streaming Quality – Higher Resolution Application Area. ↵

⁵¹. "Extending the Validity Scope of ITU-T P.1202.2" in Proceedings of the 8th International Workshop on Video Processing and Quality Metrics for Consumer

Electronics, retrieved from www.vpqm.org. ↵

视频质量测量

我们详细阐述了视频质量测量（主观测量和客观测量）需要重点考虑的各种因素。此外，为了清楚起见，我们从典型的应用场景讨论了各种客观测量方法。

主观测量

主观测量中使用的指标是MOS和DMOS。但是，在获得原始分数后，这些得分还不能直接使用。通常需要使用接下来的测量程序来消除得分之间的偏差。

令 s_{ijk} 表示主观测试的第 k 个阶段中，第 i 个用户给第 j 个视频的主观评分。主观测试一般会有两个测试阶段。

在处理原始评分的过程中，在每个测试阶段中计算视频的评分差异 (*difference scores*)

d_{ijk} 的计算方式为：在同一个测试阶段中，受试者对视频的质量评分减去同一个受试者对该视频的参考视频的质量评分。每一个测试阶段都计算评分差异，有助于解释受试者在不同测试阶段使用质量量表的任何可变性。得分差异的计算如4 – 30。

$$(4 - 30) \quad d_{ijk} = s_{ijk} - s_{ij_{ref}k}$$

对于参考视频而言，在任何测试阶段，参考视频的 d_{ijk} 均为0，因此可以将其从最终的评分差异中删除。然后利用4 – 33将每个阶段的评分差异转换为Z-scores的形式。

$$(4 - 31) \quad \mu_{ik} = \frac{1}{N_{ik}} \sum_{j=1}^{N_{ik}} d_{ijk}$$

$$(4 - 32) \quad \sigma_{ik} = \sqrt{\frac{1}{N_{ik} - 1} \sum_{j=1}^{N_{ik}} (d_{ijk} - \mu_{ik})^2}$$

$$(4 - 33) \quad z_{ijk} = d_{ijk} - \mu_{ik}\sigma_{ik}$$

其中， N_{ik} 为受试者*i*在第*k*阶段的测试中评估的视频序列个数。

对于每一位受试者而言，在整个的测试中，数据库中的测试视频仅会出现一次。因此，将所有测试阶段的Z-scores合并后可以生成一个Z-scores矩阵 $\{z_{ij}\}$ 。可以使用ITU-R BT.500-13（附件2的第2.3.2节）中定义的程序筛选并去掉不置信的受试者。

然后分析评分的分布。如果评分是正态分布的，则当该受试者给出的评分中，如果有5%以上的评分不在 $[\mu_i - \sigma_i, \mu_i + \sigma_i]$ 区间中，则ITU-R BT.500-13（附件2的第2.3.2节）中定义的筛选程序会拒绝该受试者的评分。如果评分不是正太分布的，则当该受试者给出的评

分中，如果有5%以上的评分不在 $[\mu_i - 2.235\sigma_i, \mu_i + 2.235\sigma_i]$ 区间中时，则会拒绝该受试者的评分。然而，在这两种情况下，如果受试者对质量的判断始终保持悲观或乐观，则不会拒绝该受试者的得分。

然后将Z-scores线性调整至[0,100]的区间内。最后，利用重新调整后的、剩余的可信受试者的Z-scores的平均数作为每个视频的DMOS。

客观测量及其应用

客观测量在自动化环境中非常有用。例如，自动化比较两个视频编码器的质量。图4-16展示了使用全参考的客观视频质量指标进行编码器比较的流程图。

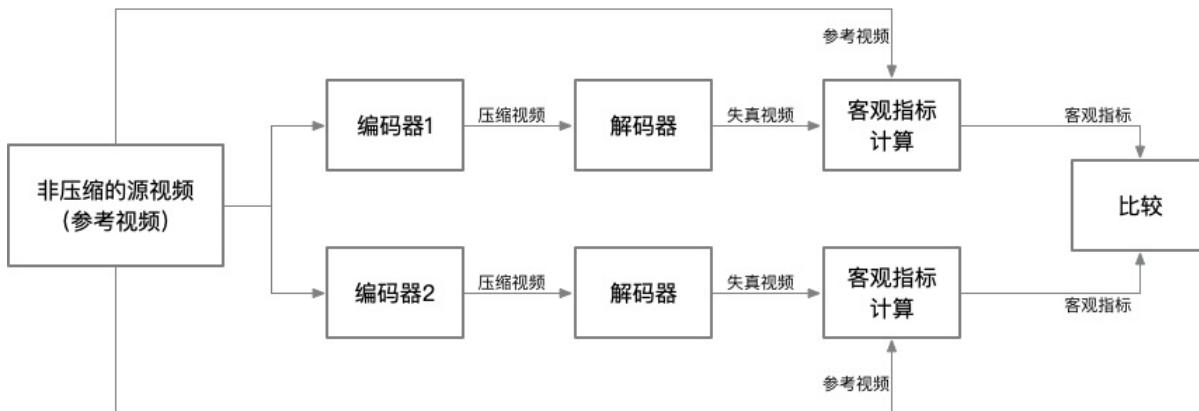


图4-16.一个典型的全参考的编码器比较方案

对于使用全参考的客观视频质量指标的应用而言，需要考虑如下的几个因素：

- 原视频和失真视频需要在时间维度上对齐，这样才能在相同的视频帧之间比较质量。
- 为了避免解码期间的变数，需要使用相同的解码器解码压缩的视频。
- 为了保证比较的公平性，编码器的参数必须一致或者尽可能的一致。
- 假定在编码过程不进行预处理。尽管可以在编码器之前使用预处理步骤，但在这种情况下，对于每个编码器而言，必须使用相同的预处理器。

需要注意的是，如上的方案可以利用自动化的优势，并使用大量视频剪辑来比较不同编码器，进而挖掘出各种工作负载复杂度下的不同编码器的优缺点。在不考虑网络或信道错误的情况下，这种利用源信号来比较编码器是相对公平的。在实际应用中（例如，对两个不同的移动设备上的视频拍摄器而言，计算客观质量之前会存储和解码其记录的视频），也应该尽可能在一致的环境（例如，对比过程中使用到的无线网络环境应该具备相似的丢包率或误码率）中进行编码器的质量比较。

客观视频质量测量也被广泛应用于确定视频应用中的丢帧现象。例如，如果逐帧跟踪源视频和失真视频之间的PSNR，则可以检测到失真视频中的帧丢失。在低失真的环境中，产生的压缩视频应该完美地匹配源视频。此时的PSNR应该是25 dB~40 dB之间的数字，该数值取决于引入错误的各种信道的有损特性。然而，在丢帧的情况下，会用错误的帧与源视频帧进行比较，此时将获得非常低的PSNR值，从而来指示存在帧丢失的情况。包含频繁的场景变化的视频会夸大这种丢帧检测的效果。

可以应用相同的概念来检测视频中的突发的低质量帧（帧被损坏或帧存在其他伪影）。网络错误或编码器问题，可能会导致帧损坏。但是，自动化环境中的PSNR或其它客观测量的突然下降，可能意味着此处就是发生帧损坏的位置。

调参

在视频通信应用中，视频编解码器是失真的主要来源之一。由于视频解码器必须遵循各种标准所定义的某些规范，因此解码器通常不会导致视频质量的显著下降。但是，编码器可以自由设计并实现算法，从而可以根据系统资源、应用程序要求、以及应用程序环境的各种因素来控制视频的压缩量，进而控制信息丢失量。因此，在视频编码应用中，有几个参数决定了信息丢失的数量，并影响最终的视频质量。在这些参数中，部分参数可由系统架构师在算法级别调整；部分参数可以由算法实现者调整；而可供用户调整的参数一般很少。

影响视频质量的参数

理解本节介绍的各种参数对最终的视觉质量产生的影响非常重要。特别是对于视频编码方案的基准测试，优化或比较分析而言，理解这些参数对视觉质量的影响尤其重要。

Number of lines in the vertical display resolution (显示设备的垂直分辨率)

高清电视（HDTV）的分辨率为1,080或720行像素。相比之下，标清数字电视（DTV）是480行像素（对于NTSC，其中525条扫描线中有480条可见）或576行像素（对于PAL/SECAM，其中625条扫描线中有576条可见）。例如，DVD的分辨率是标清，而蓝光光盘则是高清。编码器可以根据需要来降低视频的分辨率，这取决于可用的比特数和需要达到的质量等级。然而，最近的编码器在大多数应用中通常处理视频的全分辨率。

Scanning type (图像扫描类型)

数字视频使用两种类型的图像扫描模式：逐行扫描（*progressive scanning*）或隔行扫描（*interlaced scanning*）。在刷新帧时，逐行扫描重绘视频帧的所有行，并且通常表示为：720p或1080p。隔行扫描一次绘制一个场（*field*），在第一次刷新操作期间绘制奇数行，然后在第二次刷新期间绘制剩余的偶数行。因此，隔行刷新率是逐行刷新率的两倍。例如，隔行扫描的视频通常表示为480i或1080i。

物体的运动会对隔行扫描视频的感知质量产生影响。在逐行扫描的显示器上，由于较高的刷新率，隔行扫描视频为帧中的静止对象产生更好的质量；但是当帧中的对象移动时，由于隔行扫描视频失去高达50%的分辨率，因此会产生梳状噪声（*combing artifacts*）。请注意，当且仅当两个场编织在一起形成单个帧，然后显示逐行扫描的显示设备上时，才会发生梳状噪声。当在隔行扫描显示器上显示隔行内容时或者使用不同的去隔行算法（例如bob算法）用于逐行扫描显示器时，不会发生梳状噪声。

实际上，因为由帧的奇数行和偶数行组成的两个场在时间上发生移位，因此两个隔行扫描场构成单个帧。Frame pulldown和segmented frames是一种允许通过隔行视频流的方式发送全帧视频的特殊技术。为了在传输系统的接收端进行适当的重建和显示，必需跟踪首先发送哪个场。

Number of frames or fields per second (帧率)

欧洲的电视广播系统一般采用50赫兹的帧率，而美国则为60赫兹。众所周知的720p60格式的含义是：分辨率为 1280×720 像素，采用逐行扫描，每秒60帧。1080i50或者1080i60格式的含义是：分辨率为 1920×1080 像素，采用隔行扫描，每秒50/60场。如果不能维持正确的帧率/场率，就会导致产生可见的闪烁噪声 (*flickering artifact*)。帧丢失 (*Frame drop*) 和帧抖动 (*frame jitter*) 是由帧率管理不当引起的视频质量问题。

Bit rate (码率)

可以通过为每秒视频数据分配一定数量的比特位来控制数字视频的压缩量。码率是定义视频质量的主要因素。较高的码率通常意味着更高质量的视频。利用可跳过宏块 (*skippable macroblocks*) 的优势，可以实现有效的比特分配，同时有效的比特分配机制还基于宏块的时空 (*spatio-temporal*) 复杂性。量化级数也是由可用的码率来确定，并且量化级数会高度影响变换块 (*transform block*) 边界处的块效应。

Bit-rate control type (码率控制类型)

码率控制取决于传输系统的某些限制和应用的性质。有的传输系统具有固定的信道带宽并且需要以固定码率 (*CBR, constant bit rate*) 传送视频内容。但是，其它的传输系统允许动态码率 (*VBR, variable bit rate*)，其中数据量可以在每个时间段上发生变化。CBR意味着视频的解码速率是恒定的。通常，解码缓冲区用于保存已经解码完成的比特流，直到帧的数据已经显示给用户为止。CBR在流视频应用中非常有用。在流视频应用中，为了满足固定码率的需求，可能需要发送没有有用信息的填充比特 (*stuffing bits*)。

VBR允许为视频的复杂部分分配更多的比特位，而为相对简单的部分分配更少的比特位。用户给定视频的主观质量值，编码器会根据用户的质量需求来分配比特位，从而实现用于预期的质量水平。因此，使用VBR可以获得更好的观看体验。但是，VBR产生的压缩视频仍然需要满足可用的信道带宽，因此需要最大的码率限制。VBR编码方法通常允许用户指定最大/或最小码率范围。与CBR相比，VBR通常更适合于存储类型的应用。

除了CBR和VBR之外，还可以使用平均码率 (*ABR, average bit rate*) 编码来保证输出的视频流具有可预测的、长期的平均码率。

Buffer size and latency (缓冲区大小和延迟)

正如前面所述，因为输入视频可能以固定码率或动态码率到达，因此解码器的缓冲区会临时存储接收的视频。缓冲区将会在特定的时刻删除缓存的帧的数据。这个特定的时刻就是从缓冲器中取出一帧的数据用于显示给用户时。而删除的比特位数则根据帧的类型而变化。对于固定容量的缓冲区，则必须小心地保持数据的到达速率和删除速率，从而保证缓冲区不会溢出数据或缺少数据。缓冲区的管理通常由速率控制机制来完成。速率控制机制会控制量化级数并管理所得到的帧的大小。如果缓冲区溢出，则最新到达的数据将会丢失，此时则可能无法显示接下来的一个或多个帧（无法显示的帧的数量取决于帧的依赖性）。如果缓冲区没有数据，则解码器将没有需要解码的数据。此时，显示器将继续显示先前显示的帧，并且解码器必须等到解码器刷新信号到达时才能纠正该情况。缓冲区开始填充数据的时间和从缓冲区中取出第一帧的时间之间存在一个最初的延迟。该延迟也就是所谓的解码延迟。通常，在解码器开始消耗缓冲区数据之前，缓冲区的填充比例通常控制在缓冲区大小的50%~90%。

Group of pictures structure (GOP结构)

帧的预测结构决定了视频序列中帧的依赖顺序。在第二章中我们介绍过：因为I帧通常作为一组图像的锚点 (*anchor*)，因此I帧的编码是独立进行的，并且需要给其分配更多的比特位。P帧和B帧通常会有更高的量化级别，导致其具有更高的压缩率，随之而带来的代价是单个图像的质量相对较差。因此，图像组的排列非常重要。对于典型的广播应用，每秒需要发送2次I帧。并且，在两个I帧之间，使用P帧和B帧进行填充，从而达到在I帧或P帧之间存在2个B帧。使用更多的B帧通常并不能改善视觉质量，但是B帧的使用还是取决于应用。值得注意的是，具有长期参考的预测并非适用于场景高速变化的视频。高效的编码器需要先执行场景分析从而确定最终图片组的结构。

Prediction block size (预测块的大小)

可以使用各种尺寸的预测块来执行帧内预测或帧间预测。预测块的尺寸一般为 $16 \times 16 \sim 4 \times 4$ 。为了有效编码，必须根据视频帧中的细节模式选择合适的预测块大小。例如，具有更精细细节的区域需要更小尺寸的预测块，而平坦区域则可使用更大尺寸的预测块。

Motion parameters (运动参数)

运动估计的搜索类型、搜索区域、以及代价函数 (*cost function*) 在确定视觉质量中起着重要作用。全搜索算法 (*full search algorithm*) 检查每个搜索位置以找到最佳匹配块，但代价是计算复杂度非常高。研究表明，编码计算中，超过50%的时间都消耗在块匹配过程。块匹配计算的复杂度会随着搜索区域的变大（捕获大运动或适应更高分辨率的视频）

呈指数增长。此外，匹配标准可以从如下的技术中进行选择：绝对误差和（*SAD, sum of absolute difference*）¹ 和hadamard变换后的绝对误差和（*SATD, sum of absolute transformed differences*）²。使用SATD作为匹配标准可以使用更高的计算复杂度为代价而提供更好的视频质量。

Number of reference pictures (参考帧数量)

对于运动估计，可以从前向或后向参考的列表中使用一个或多个参考图像。多个参考图像会增加获取到更好匹配的概率，进而使得差异信号（*difference signal*）更小并且可以提升编码效率。因此，在视频具有相同总比特数的条件下，多参考图像的最终质量会更好。此外，视频的最终质量还取决于视频内容，在某些视频中，某一帧可能与非前后邻近帧或非近邻帧之间存在更好地匹配。在这种情况下，就需要长期（*long-term*）参考。

Motion vector precision and rounding (运动向量的精度)

可以在各种精度级别执行运动补偿：全像素，半像素，四分之一像素等。精度越高，找到最佳匹配块的概率就越大。更精确的匹配导致使用更少的比特来编码误差信号（*error signal*）。因此，与全像素运动补偿相比，四分之一像素运动补偿为相同数量的比特数提供了更好的视觉质量。舍入的方向和数量对于保持足够的数据细节也很重要，从而实现更好的质量。舍入参数一般因为预测块类型（帧内或帧间）的不同而不同。

Interpolation method for motion vectors (运动向量的插值方法)

可以使用不同类型的滤波器来完成运动矢量插值（*motion vector interpolation*）。典型的插值方法采用双线性、4抽头³（4-tap）或6抽头（6-tap）滤波器。这些滤波器产生不同质量的运动矢量，进而导致最终视觉质量的差异。一般而言，6抽头滤波器产生最佳质量，但其在处理周期和功耗方面更昂贵。

Number of encoding passes (编码通道数)

单通道编码（*single-pass encoding*）实时分析和编码数据。在编码速度是重要因素的场景（例如，在实时编码应用程序）中，需要使用单通道编码。但是，当编码质量最重要时，需要使用多通道编码。多通道编码一般会在两个通道中执行编码操作。由于输入数据

在每个通道中都会经过额外处理，因此多通道编码会花费更长的时间。在多通道编码中，使用一个或多个初始通道来收集视频特征数据，并且最后的通道使用该数据以便在指定的目标码率下实现均匀的质量。

Entropy coding type (熵编码类型)

熵编码类型（如CABAC⁴或CAVLC⁵）一般不会影响视频质量。但是，如果存在码率限制（尤其是对于低目标码率）时，为了获取更高的编码效率，CABAC可以产生更好的视觉质量。

¹. SAD即绝对误差和，就是以左目图像的源匹配点为中心，定义一个窗口D，统计其窗口的灰度值的和，然后在右目图像中逐步计算其左右窗口的灰度和的差值，最后搜索到的差值最小的区域的中心像素即为匹配点。SAD仅反映残差时域差异，影响PSNR值，不能有效反映码流的大小。 ↪

². SATD即将残差经哈德曼变换的4×4块的预测残差绝对值总和，可以将其看作简单的时频变换，其值在一定程度上可以反映生成码流的大小。 ↪

³. 抽头数是变压器中的一个概念，抽头数越多，我们就可以得到越多的电压信号。对于滤波器，每一级都保存了一个经过延时的输入样值，各级的输入连接和输出连接被称为抽头。 ↪

⁴. CABAC (Context-based Adaptive Binary Arithmetic Coding)，基于上下文的自适应二进制算术编码。CABAC是H.264/AVC标准中两种熵编码中的一种，它的编码核心算法就是算术编码 (Arithmetic Coding)。 ↪

⁵. CAVLC(Context Adaptive VariableLength Coding)是在H.264/MPEG-4AVC中使用的熵编码方式。在H.264中，CAVLC以zig-zag顺序用于对变换后的残差块进行编码。CAVLC是CABAC的替代品，虽然其压缩效率不如CABAC，但CAVLC实现简单，并且在所有的H.264profile中都支持。 ↪

参数之间的权衡

视频编码器通常具有可调参数，以实现编码器的最佳质量或最佳编码速度。有的参数允许编码器分析输入视频并收集输入视频特征的详细信息。基于编码器收集的信息，编码器做出某些决定：例如要执行的压缩量，或要使用的编码模式。通常，使用多通道来分析并执行后续的编码工作。因此，对于给定的算法，编码器能够有效地压缩视频并实现最佳的质量。但是，编码器的这种分析需要一定时间并且会降低编码速度。另外，分析工作还将增加编码设备的功耗。因此，在某些情况下，并不会试图通过调整某些参数以适应给定的视频特性，以便提高编码器性能或满足系统资源限制。相反，会使用这些参数的预定义的值来达到相同的目的，从而减少分析工作并且实现可能的最佳速度。

上一节中提到的影响视觉质量的大多数参数也会影响编码速度。对于给定的视频编码器，需要调整多个参数以便在编码质量和速度之间取得良好的权衡。对于此处列举的参数而言，虽然并不是所有的参数都是用户可调整的，但某些参数可能会根据编码器的具体实现而暴露给最终用户。

Bit rate, frame rate, resolution

具有高码率，帧率和分辨率的视频通常需要更长的编码时间，但却可以提供更好的视觉质量。应该仔细设置这些参数以适应具体应用的要求。例如，可以在仅具有特定参数的特定设备上满足实时编码和实时处理的要求。

Motion estimation algorithm

文献中提出了大量的快速运动估计 (*fast-motion estimation*) 算法，所有这些算法都基于一个共同的目标：在提供合理的质量的同时提高运动估计的速度。由于运动估计是视频编码中最耗时的部分，因此仔细选择运动估计的算法非常重要。

Motion search range

为获得最佳质量，应将运动搜索范围设置为一个较高的数值，以便捕获较大的运动。另一方面，搜索窗口越大，在要进行的计算量越大。因此，较大的搜索区域会直接影响编码速度，内存带宽，帧延迟和功耗。另外，较大的运动矢量也需要更多的比特位来编码。如

果源块和预测块之间的差别信号具有相当大的能量，则更应该采用块帧内编码模式而不是使用大的运动矢量。因此，需要根据每分贝质量增益 (*decibel of quality gain*) 所花费的比特数对搜索参数和编码效率进行权衡。

Adaptive search

为了获得更好的质量，通常运动搜索算法可以适应视频的运动特性，并且可以有效地控制搜索过程以获得显著的编码速度。例如，为了加速运动搜索，算法可以避免搜索静止区域，使用可切换形状的搜索模式，利用运动矢量中的相关性。因此，通过这些算法可以增加编码速度，并且无需求助于次优搜索同时也不会牺牲视觉质量。

Prediction types

P帧和B帧具备不同级别的计算复杂度，并且通过视觉质量损失的方式实现数据压缩。然而，它们也提供了视觉上令人愉悦的平滑运动外观。因此，帧的预测类型是权衡编码质量和编码速度的重要考虑因素。

Number of reference frames

多个参考帧比单个参考帧提供更好的视觉质量，但是根据多个参考帧计算运动矢量却更耗时。在资源受限的环境中，也需要权衡考虑这些参数。

Transform mode and partition size

可以使用 8×8 或 4×4 的块用于变换，并且可以利用不同的块大小预测运动。在某些平台上，处理四个 4×4 的块可能比处理一个 8×8 的块要慢。但是，根据视频中可用的详细信息量，确定块尺寸的决策可能会影响视觉质量（因为 4×4 的块更能适应更精细的细节）。

Skip conditions

如果满足特定条件，则可以跳过某个块。与简单的启发式算法相比，可以基于对量化变换系数特性的分析来做出更好的跳过决策，从而产生更好的质量。但是这种复杂的算法需要大量的计算。对于资源受限的设备来说，需要进行一定的权衡以获得最佳的效果。

Deblocking filter parameters

编码速度通常对去块滤波器 (*deblocking filter*) 的参数比较敏感。执行强去块 (*strong deblocking*) 会降低编码速度。但是根据内容和块效应的数量，去块滤波器可以提供更好的视觉质量。

总结

本章讨论了视觉质量的相关问题以及影响人类视觉感知质量的相关因素。首先，研究了导致视觉质量下降的各种压缩和处理噪声，以及影响视觉质量的各种因素。接下来，讨论了各种主观和客观的质量评估方法和指标，并特别关注了各种ITU-T标准。并且详细讨论了几种客观的质量评估方法。这些客观方法基于各种因素：误差敏感度，结构相似性，信息保真度，时空性，显著性，网络感知和噪声。我们还讨论了视频编码效率的评估指标和基于标准的算法的视频质量指标的一些示例。

在本章的最后部分，我们介绍了影响视频质量的编码参数。调整参数可在视频质量和压缩速度之间提供良好的折衷机会。这些参数包括码率，帧率，分辨率，运动估计参数，图像组结构，参考帧数和去块滤波器参数。并且最终用户可以调整其中的部分参数。

视频编码性能

20世纪80年代到21世纪初，台式PC机是主要的计算平台。PC机具备独立的组件部分，如CPU，芯片组和独立显卡。在此期间，Intel(R) 810(TM)掀起了集成显卡的起步阶段，并且集成显卡主要针对低端市场。此时的电量消耗也不是一个需要特别关心的事情。平台迭代的最大区别就是CPU速度。因此，当设计CPU的微架构时，关键问题之一是如何实现更高的性能。实现这一目标的传统方法是不断提高时钟频率。然而，晶体管速度的增长已接近其物理极限，这意味着处理器时钟频率无法继续增加。在过去几年中，台式机和平板电脑的最高CPU速度趋于稳定，分别为3~3.5 GHz和1.5~2 GHz。随着具有更小外形尺寸的平台的出现，保持处理器频率受限已成为新的常态，同时焦点已经转向降低系统功耗并更有效地利用可用系统资源。

数字视频应用需要大量的处理步骤。并且，实时处理和回放需求要求系统需要具备某些功能和性能级别。仅仅几十年前，实时视频编码主要还集中在非商业学术解决方案中。在这些方案中，需要使用高性能的专用的硬件，或基于通用处理器的大规模的并行计算，才能实现实时视频编码。对于实时视频而言，硬件和软件都需要在系统和应用程序级别进行仔细的性能优化和调整，才能实现合理的质量。然而，随着近年来处理器速度和系统资源利用率的大幅提升，如今的处理器可以实现更高数量级的编码速度，同时还能获得更好的质量。

本章首先简要讨论CPU时钟速度，并考虑为什么无限期增加时钟速度是不切实际的。然后讨论提高视频编码速度的动机，以及达到该编码速度所需的权衡。然后我们讨论影响编码速度的因素，可能遇到的性能瓶颈以及优化方法。最后，我们介绍了各种性能测量注意事项，工具，应用程序，方法和指标。

CPU速度和限制

以下是CPU时钟速度无法无限增加的主要原因：

- 高频电路的功耗随着频率的增加而增加，对随着而产生的热量的散热将会在某一点变得不可能。英特尔CTO Pat Gelsinger在2001年曾经预测：“十年后，微处理器将以10 GHz~30 GHz的频率运行。”然而如果是这样的话，“这些芯片将产生与核反应堆一样多的热量。”高频电路的散热是常规冷却技术的基本问题。无论从经济角度还是从工程角度来看，无限增加频率都是不可行的。
- 诸如时钟门控和功率门控之类的现代节能技术不适用于高频电路。时钟门控中，在每个状态之前插入一个clock-enable，这样，如果数据保持不变，则不会为该状态计时。这样可以避免在写同一位时存在的大量的充电/放电而带来的浪费。但是，如上的操作会给关键时钟路径带来额外的延迟，这不适用于高频设计。在功率门控中，大型晶体管充当处理器各种功能块的电压源，当某些功能模块没有使用时则可能会关闭该功能块。但是，由于功率门控晶体管中的额外压降，其开关速度会降低。因此，该技术也不适合高频设计。
- 晶体管的速度已达到平稳状态。尽管晶体管变得越来越小，但它们却没有变得更快。为了理解原因，让我们考虑一下电子设备的以下事实：较薄的栅极电介质会导致跨晶体管通道的电场增强，从而使其开关速度更快。晶体管栅极面积的减小意味着可以将栅极做得更薄，而无需增加为了控制节点充电而创建电场所需的负载电容。但是，在45 nm制造技术下，栅极电介质的厚度已经约为0.9 nm，约为单个 SiO_2 分子的大小。用相同的材料将它制成任何更薄的材料是根本不可能的。凭借22 nm技术，英特尔利用创新的三门（*tri-gate*）技术来克服这一限制。此外，改变栅极电介质和连接材料有助于提高晶体管速度，但会导致解决方案比较昂贵。基本上，在20世纪80年代和90年代，晶体管尺寸的每一次缩小都会导致更快的晶体管，但是现在晶体管尺寸的缩小已经非常困难。
- 晶体管不再是影响处理器速度的主要因素。连接晶体管的导线正成为最重要的延迟因素。随着晶体管变小，连接线变细，从而导线的电阻变高，进而使得电路的电流更低。考虑到较小的晶体管能够驱动较小的电流这一事实，很容易看出，电路路径延迟仅部分取决于晶体管的开关速度。为了克服这个问题，在芯片设计期间尝试将时钟和数据信号路由到相似的路径上，从而为这两个信号获得大约相同的传播时间。这对于需要大量数据、控制信号的任务（例如固定功能视频编解码器引擎）较为有效。但是，通用微处理器的设计很复杂，交互作用不规则，数据传输到多个位置，而这些位置并不总是紧跟时钟。不仅有反馈路径和循环，而且还有大量控制集中的资源，例如调度，分支预测，寄存器文件等。可以使用多个内核并行执行此类任务，但是当处理器频率提高时，需要使用更细的线。

提升性能的动机

在视频领域，性能是一个非常复杂的术语，其涵盖的内容远超出了“性能”这个词语本身。

- 在一些文献中，编码器性能是指用于获得特定视觉质量的比特数的压缩效率。测试编码器比参考编码器可以节省的平均码率被认为是客观编码的性能标准¹²。
- 还有另外一种看法就是：编码器的性能是指以FPS (*frame per second*) 为单位的编码每帧图像的速度。

在本书中，性能一般指的是编码视频帧的速度。我们还注意到FPS可用于不同目的。视频剪辑中的帧率也称为FPS（例如，24 FPS或30 FPS），这意味着应该以指定的帧率 (FPS) 播放 (*played back in real time/real-time play back*)³ 该视频，从而提供平滑的运动感知（如果播放的帧率低于规定的帧率，则会产生视频卡顿的现象）。但是，当生成压缩视频时，压缩速度比实时播放速度快许多倍，称该速度（也以FPS表示）为编码速度或编码器性能。需要注意的是，在某些实时应用程序（如视频会议）中，视频帧仅用于实时显示。因此，较快的编码速度并不是必需的，虽然如此，编码速度也需要足够快。因为更快的编码速度使得处理器可以进入空闲状态，从而节省电量。

但是，有很多视频应用要求编码器的编码速度要快于播放速度。例如：

- 可以在更短的时间内压缩较长的视频。这对于视频编辑人员非常有用。视频编辑人员通常会处理大量的视频内容，并且他们的工作需要在指定的时间限制内完成。
- 视频归档应用程序需要压缩、存储大量视频。因此，视频归档应用可以从快速编码中受益。
- 视频录制应用程序能够以合适的压缩格式存储录制的视频，快速编码可以实现共享处理单元的目的，从而实现同时运行编码和（或）非编码任务。
- 快速编码有益于将视频从一种格式转换为另一种格式。例如，可以使用诸如 Handbrake⁴ 之类的转码器同时将多个MPEG-2标准的DVD视频转换为AVC标准。
- 视频转码可用于：视频创作，视频编辑，上传视频到Internet，刻录视频内容到光盘，或者视频内容分发。对于这些场景而言，视频转码速度越快越好。特别是，可以通过高速编码（实时编码的几倍速度）使得基于云的视频点播服务能够同时处理多个请求。当然，同时处理多个请求的能力还需要网络带宽的优化。
- 快速编码还有益于视频传输应用（例如有限电视等）。将视频转换到较低的码率水平，则可以通过电缆、电信、以及卫星有效地分发视频，并且可以在相同的信道带宽中容纳更多的视频节目。转换 (*transrating*) 和重新封包 (*repacketization*) 系统中的总延迟通常是恒定的，并且只有实时处理需要关注这种整体延时。尽管如此，从调度灵活性和资源利用的角度来看，仍然需要提升转换速度和编码任务的速度。

典型的视频应用涉及一系列任务，例如捕获视频数据；压缩，传输或存储视频数据；视频数据解压缩以及显示视频数据。在处理这一些列任务的同时，需要保持一个恒定的整体系统延迟。相机和显示设备引入的延迟通常可以忽略不计。一般而言，性能需要重点关注：解码、编码、以及处理时间。其中，解码任务通常由视频标准指定，并且每秒需要一定数量的操作。但是视频编码和处理任务中的计算量大大超过了解码任务所需的计算量。因为编码和处理任务具有更高的复杂性，因此，根据应用要求，它们更适合于性能优化。

视频编码需要大量的信号处理操作，大约每秒数十亿次操作。幸运的是，视频压缩可以很容易地分解为流水线任务。在各个任务中，视频数据可以在空间或时间维度上进一步分解为一组相互独立的部分，使其适合并行处理。利用这种并行特性，通过并行使用多个处理单元，可以获得比实时视频编码更快的性能。这些处理单元可以是专用、固定功能硬件单元和（或）可编程硬件单元的组合。专用硬件的优势在于对特定任务的优化，因此任务可以通过性能和功耗优化的方式完成。可编程单元则提供了一定的灵活性，并且不会轻易过时。同时，可编程单元的性能调整也比专用硬件单元更便宜。因此，有效地将专用单元和可编程单元组合成混合解决方案，可以提供比实时性能更高的性能。例如，Intel(R) Core(TM)和Intel (R) Atom (TM)等CPU，可以将繁重的编码任务交给集成GPU执行。

1. Performance Analysis of HEVC-Based Intra Coding for Still Image Compression.



2. Performance Comparison of H.265/ MPEG-HEVC, VP9, and H.264/MPEG-AVC Encoders.



3. real-time play back, 实时回放实际上就是全帧率播放。回放就是从某种已存在的记录来源进行播放，也就是通常所说的播放。在监控领域把全帧率称作实时，非全帧的称作非实时，PAL制式全帧是25帧，意思就是录像文件1秒钟由25张图片组成。缺帧多的话肉眼会感觉到画面卡顿。



4. [HandBrake](#), The open source video transcoder. HandBrake is a tool for converting video from nearly any format to a selection of modern, widely supported codecs.



对性能的考虑

在视频编码和处理应用中，性能优化的目的在于通过适当地改变编码器的设计或实现，以提高编码速度。单纯增加处理器的频率并不会获得性能最佳的编码方案。并且如前所述，CPU的频率增长会存在上限。因此，需要探索其他的性能增强方法。需要注意的是，某些技术所做的必要的设计修改相对便宜，而其他技术则可能需要大量投资。例如，获得更高性能的低成本的方法包括：编码任务并行化，调整任务的调度，优化各个任务的资源利用率等。当然，也可以利用更复杂的专用硬件单元实现更高的性能。但是，这些专用硬件单元的制造成本更高。性能优化需要考虑：如何明智地选择技术方案，从而能够以最低开销提供最高性能。但是，根据应用程序的性质和可用的资源，可能需要支付大量的资金才能获得预期的性能。例如，更大的缓存可能有助于达到某些性能目标，但同时意味着可能会花更多的钱。因此，在任何性能优化的方案中，必须做出仔细的权衡。

一般不会只考虑性能优化，性能优化一般会与视觉质量和功耗放在一起研究。例如，具备更高频率的CPU或GPU可以提供更快的编码速度，但同时也将消耗更多的电量。因此，在系统设计和架构级别上，需要在能量消耗和编码速度之间进行权衡。如今的视频应用程序一般运行在资源受限的计算平台之上。当可用的系统资源处于活动状态时最大化其利用率，当系统资源再不需要时则使其进入休眠状态。可以通过如上的方式来获得性能和功耗之间的权衡。

调整某些视频编码参数（例如码率，量化参数）也可以实现更高的编码速度。丢弃大部分的高频细节后，只需要处理更少的信息，从而提升编码速度。但是，这种调整编码参数的方法会直接影响得到的视频的视觉质量。因此，使用该技术时，需要在视觉质量和性能之间做好权衡。

在给定的时间段内，编码性能最大化的方式主要有三种：

- 确保在工作负载期间充分利用可用的系统资源（CPU和内存）。但是，根据工作负载的不同，资源利用也会有所不同。例如，编码应用程序应以CPU执行周期的100%的占空比(*duty cycle*)¹的方式运行（100%占空比意味着CPU在整个周期内均处于运行状态）。如前所述，这种性能最大化需要考虑电量优化。例如，CPU在必要时以100%的占空比运行，并在之后快速进入休眠状态。但是，实时播放应用可能只会使用一小部分的资源（比如10%的占空比）。此时需要重点考虑的是应用的节能，而不是应用的性能优化。
- 如果可以，那就使用专门的资源。专用资源通常权衡任务的性能和功耗，因此可以在不需要明确权衡的情况下提供性能改进。
- 根据应用的需求来调整部分视频参数，从而提高编码速度。但是，编码参数的调整可能会影响视频的视觉质量、编码器的压缩率、以及应用的功耗。此时，需要对各种因

素做仔细的权衡。

1. 占空比 (*duty ratio*) 指的是信号周期中，高电平和该周期内高、低电平所占的时间的比率，占空比越大，电路开通时间就越长，整机性能就越高。 ↵

资源利用率最大化

应用程序，服务，驱动程序和操作系统会竞争重要的系统资源：包括处理器时间，内存，硬盘，网络带宽和电池电量。而这些系统资源意味着的资金的投入，从这个层面讲，为了发挥资金的最大效能，必须在尽可能短的时间内最大限度地利用可用的系统资源。从而实现，以最小功耗获得最大性能。为此，通常会采用以下技术。

任务并行

许多任务彼此之间相互独立，不存在资源的相互等待，因此可以并行运行。任务并行化能够使CPU得到充分利用。通常而言，任务流水线也可以在任务执行期间保持资源繁忙，从而实现资源利用率的最大化。

寄存器，缓存和内存利用率

存储器是分层次的，离CPU越近的存储器，速度越快，每字节的成本越高，同时容量也因此越小。

寄存器速度最快，离CPU最近，成本最高，所以个数容量有限。其次是高速缓存（缓存也是分级，有L1，L2等缓存），再次是主存（普通内存），再次是本地磁盘。

对于性能而言，优化内存层级的使用是一个重要的因素。

寄存器的操作 (*register transfer operations*) 由CPU控制并以CPU的处理速度完成。高速缓存通常为静态随机存取存储器 (SRAM)，并且由存储器管理单元 (MMU) 控制。在系统级程序中需要仔细使用多级缓存，从而平衡数据的访问延迟和数据容量。

内存一般为动态RAM (DRAM)，内存的容量比缓存大得多，但数据访问速度比缓存低。采用命中率来表示相邻级别的存储器之间的数据传输性能：在某个存储器中找到信息的概率。存储器的访问频率和有效访问时间取决于程序行为和对存储器设计的选择。通常，对程序跟踪的全面分析可以产生优化的机会。

优化磁盘访问

视频编码过程会处理大量的数据。因此，与处理过程本身相比，磁盘的I/O速度、内存延迟、内存带宽等因素经常会成为性能瓶颈。很多文献提供了许多关于磁盘访问的优化技术。RAID (*redundant arrays of inexpensive disks*, 廉价磁盘冗余阵列) 是一种常见的数据存储虚拟化技术。RAID可以控制数据的访问，并可权衡磁盘的可靠性，可用性，性能和容量。

指令流水线

目前，CPU的架构有：复杂指令计算集（CISC）处理器，精简指令计算集（RISC）处理器，超长指令字集（VLIW）处理器，矢量超级计算机等。对于不同的CPU架构，每个指令的执行周期会因其对应的时钟频率的不同不同。但是，仍然需要认真对待指令流水线和流水线同步，从而要实现最少的无操作（NOP）和流水线停顿，进而优化资源利用率。

专用资源

除了最大化资源利用率之外，还可以使用专用资源来提高性能。该领域的具体改进具体如下。

专用媒体指令集

现代CPU都具有增强指令集（*enhanced instruction sets*），其中就包括具有并行能力的专用媒体指令²。例如，可以使用128位寄存器的单指令多数据流（*SIMD, single instruction multiple data*）¹计算像素矢量（该矢量具有8个16位的像素数据）的绝对误差和（*SAD, sum of absolute difference*）。SIMD仅需要加载一次数据以及一个并行操作就能完成该SAD的计算。而传统的顺序计算方法则需要加载16次数据，并进行8次减法运算，8次绝对值运算以及8次加法运算。对于诸如运动估计之类的编码任务，这种媒体指令在提升计算密集型任务的运算速度中发挥着重要作用。

GPU加速

视频编码任务已经可以在多核CPU上执行。视频编码等计算密集型任务的运行通常会使得所有CPU核的较高的利用率。对于更高分辨率的视频，CPU可能会出现负载过载的情况，进而导致无法实时完成这些任务。很多研究工作在各种共享内存和分布式内存平台上采用并行化技术来解决实时编码的问题，我们将在下一节中对其讨论。很容易发现，仅通过CPU的方法，根本无法获得理想的可扩展的编码解决方案。

最近的处理器，如Intel Core和Atom处理器，通过使用集成的图形硬件处理器为视频编码和处理任务提供硬件加速。专用硬件单元针对某些任务进行了特殊优化。而通用计算单元可以针对各种任务进行编程，因此通用该计算单元更灵活。英特尔的图形硬件处理器是固定功能和可编程单元的组合，从而可以实现速度、灵活性、可扩展性之间的平衡。对于运行这些图形硬件的系统，还需要优化系统的电量消耗，从而在降低耗电量的情况下提供高性能。因此，只要确保输入视频数据的实时处理，使用硬件加速来处理视频编码和任务就实现了性能和功率的完美平衡。

图5-1显示了典型的编码过程中使用和不使用GPU加速时的CPU利用率信息。从图中可以明显看出，采用GPU加速不仅使CPU可用于处理其他任务，而且还提高了编码本身性能。在此示例中，编码速度从小于1 FPS上升到超过86 FPS。

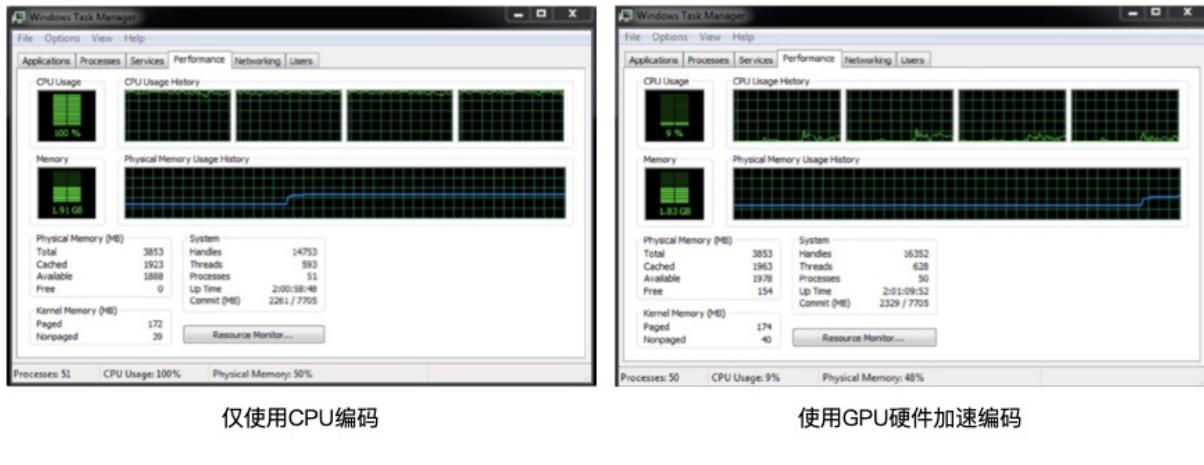


图5-1. 使用GPU加速前后，CPU利用率的对比

1. **SSE指令**: 包括70条指令，其中包含单指令多数据浮点计算、以及额外的SIMD整数和高速缓存控制指令。其优势包括：更高分辨率的图像浏览和处理、高质量音频、MPEG2视频、MPEG2加解密，语音识别占用更少CPU资源，更高精度和更快响应速度。 ↪
2. **MMX指令**: MMX是MultiMedia eXtensions（多媒体扩展）的缩写，是第六代CPU芯片的重要特点。MMX技术是在CPU中加入了特地为视频信号(Video Signal)，音频信号(Audio Signal)以及图像处理(Graphical Manipulation)而设计的57条指令，因此，MMX CPU极大地提高了电脑的多媒体（如立体声、视频、三维动画等）处理功能。 ↪

调整视频参数

为了获得最佳的编码器性能，需要调整视频参数。了解影响性能的主要因素，定位并解决典型的性能瓶颈非常重要。

决定编码速度的因素

有很多因素会影响视频的编码速度，例如：系统的硬件配置、网络配置、存储设备的类型、编码任务的性质、可并行化的能力、视频的复杂度和格式、以及是否具备硬件加速能力。如上所述的这些因素之间的相互作用会使性能调整变得复杂。

系统配置

有一些可配置的系统参数会在不同程度上影响工作负载的性能，例如视频编码速度。接下来会对这些参数进行介绍。

Number of cores (CPU/GPU的核数)

CPU和GPU的内核的数量对于提高工作负载的性能有直接的影响。将工作负载分配到不同的核处理可以提高计算速度。通常，所有的核应处于相同的性能状态，以实现最佳的资源利用率。[6.4.1](#)节中会详细介绍CPU的性能状态。

CPU and GPU frequencies (CPU/GPU的频率)

CPU/GPU的核 (CPU/GPU core) 以及CPU/GPU (CPU/GPU package¹) 的时钟频率是决定编码任务执行速度的主要因素。视频编码任务可以利用全硬件加速（利用CPU和GPU共同完成）来实现。此时，CPU和GPU的资源利用率，CPU和GPU时钟频率的差异，任务之间的依赖性，相应的数据的访问延迟等因素是性能优化的关键因素。

Memory size and memory speed (内存容量和速度)

对于视频编码和处理任务而言，内存越大越好。对于越来越高的视频分辨率，更大容量的内存会避免产生过多的内存分页。并且，更高的内存速度也有助于加速视频编码任务。

Cache configurations (缓存配置)

高速缓存是内置在CPU或其他硬件单元中的快速存储器，或者是位于单独芯片上的快速存储器。高速缓存用于存放经常重复的指令和数据，从而避免CPU利用较慢的系统总线从内存加载和存储数据，进而提升整个系统的速度。

L1级缓存一般内置于CPU且为CPU专有，不在多个CPU中共享。而L2级缓存则驻留在CPU旁边的单独芯片上。某些CPU内置了L1和L2缓存，并将缓存芯片指定为3级（L3）缓存。

使用L3级缓存可显著提高视频编码和处理任务的性能。类似地，集成GPU也具有多层缓存。此外，具有嵌入式动态随机存取存储器（eDRAMs, *embedded dynamic random access memories*）的最新处理器通常为视频编码任务产生高出10%~12%的性能。

Data access speed (数据访问速度)

除了调度延迟之外，固定存储器（NVM, *non-volatile memory*）²的存储速度和存储类型也会影响数据的可用性。例如，SSD硬盘比传统的机械硬盘的数据访问速度要快。硬盘中的磁盘缓存与CPU中的缓存是相同的原理。在单独的RAM段中存储经常访问的硬盘数据，从而避免硬盘的频繁检索。视频编码应用程序经常需要访问重复数据，因此，磁盘缓存可以显著提高编码器的性能。

Chipset and I/O throughput (I/O吞吐量)

编码任务经常需要处理未压缩的视频，某些处理任务也会输出以非压缩格式的视频。对于这些任务，I/O操作经常成为其性能瓶颈，特别是对于具备更高分辨率的视频。对于I/O密集型（I/O-bound）³任务，适当优化的芯片组可以解决读写I/O的瓶颈，从而提高整体性能。提高I/O操作效率和降低I/O延迟的技术有：视频数据在并行磁盘阵列上的智能存储，磁盘搜索优化，磁盘调度以及自适应磁盘预取。

System clock resolution (系统时钟频率)

Windows系统的默认计时器精度（timer resolution）为15.625ms，因此每秒会产生64个计时器中断。对于视频编码之类的任务，与视频帧相关的所有操作必须在指定的时间内完成（例如，对于30 fps的视频处理每帧的时间为33ms），此时默认的定时器精度是不够的。对于15.625ms的定时器精度，33ms的处理时间可能会跨越多个时钟周期，这可能需要等到下一个可用的CPU周期才能继续执行任务。编码任务之间经常存在依赖关系（例如DCT变换必须在可变长编码之前执行），因此在调度这些任务时，必须仔细考虑定时器的精度和功耗，从而获得最佳的性能。在许多应用中，1ms的定时器精度通常是更好的选择。

BIOS

可以通过BIOS调整与性能相关的参数，其中包括：

- PCIe (*peripheral component interconnect express*) 延迟和时钟门控 (*clock gating*)⁴。
- ACPI⁵设置。例如，禁用休眠。
- CPU配置。例如，启用预读取邻近的缓存数据的功能⁶。
- CPU和图形的功耗 (*power*) 管理控制。例如，允许支持两个以上的频率范围，允许turbo模式，允许CPU在未充分利用时进入C状态 (*C-states*)⁷，配置C状态延迟，设置中断响应时间限制，启用图形渲染。第6章会讨论C状态的详细信息。
- 启用超频。例如，设置图形超频频率。

Graphics driver (显卡驱动)

显卡驱动包含各种性能优化，尤其是针对硬件加速的视频编码和处理任务。适当的或者更新的显卡驱动有利于获得最佳性能。

Operating system (操作系统)

操作系统 (OS) 会进行许多优化，以便提高运行时 (*run-time*) 环境的性能。OS还控制进程和线程的优先级。例如，ART和Dalvik分别是运行Android应用程序的新、旧运行时 (*run time*)。虽然Dalvik是一个即时 (*JIT*) 运行时，其仅在需要时才执行代码。但是在Android 4.4引入的ART运行时则在代码需要执行之前就已经执行。Android 4.4中的Dalvik和ART之间的比较表明，ART模式带来了更强大的性能和电池效率。2014年10月16日，Google发Android 5.0，ART全面取代Dalvik成为Android虚拟机运行环境。至此，Dalvik退出历史舞台，AOT (*Ahead-of-time*) 也成为唯一的编译模式。

Power settings (电源设置)

除了TDP (*thermal design power*, 散热设计功耗)⁸之外，英特尔在Y-系列移动处理器中又提出了SDP (*Scenario Design Power*, 场景设计功耗) 的概念。虽然TDP规定了在最坏情况下的实际工作负载，但SDP规定了特定使用场景下的功耗。SDP可用于对特定目的的设计需求和系统冷却功能进行基准测试和功耗特性评估。通常，具有较高的TDP (或SDP) 的处理器可以提供更高的性能。因此，根据需要，用户可以选择具有更高TDP的系统。但是，在某些平台上，操作系统通常提供不同的功耗模式，例如高性能模式，正常模式或省电模式。不同的模式控制系统进入各种级别的空闲状态。不同的功耗模式对性能有着显著影响，特别是对于视频编码和处理应用程序。

1. package是封装好了的，肉眼看到的CPU/GPU处理器。CPU最小级别的就是超线程处理器的一个smt核，次小的一级就是一个多核cpu的核，然后就是一个物理cpu封装，再往后就是cpu阵列。 ↵
2. 指当电流关掉后,所存储的数据不会消失的计算机存储器。 ↵
3. I/O-bound：指的是系统的CPU效能相对硬盘/内存的效能要好很多。此时，系统大部分的状况是CPU在等I/O(硬盘/内存)的读/写，此时CPU Loading不高。CPU-bound(计算密集型)指的是系统的硬盘/内存效能相对CPU的效能要好很多。此时，系统大部分的状况是CPU Loading 100%，CPU要读/写(硬盘/内存)，I/O在很短的时间就可以完成，而CPU还有许多运算要处理，CPU Loading很高。 ↵
4. 时钟门控(Clock-Gating)一直以来都是降低微处理器功耗的重要手段,主要针对寄存器翻转带来的动态功耗。 ↵
5. ACPI (Advanced Configuration and Power Management Interface) 表示高级配置和电源管理接口。对于Windows2000，ACPI定义了Windows 2000、BIOS和系统硬件之间的新型工作接口。这些新接口包括允许Windows 2000控制电源管理和设备配置的机制。 ↵
6. 计算机在读取数据时，会智能的认为要读取的数据旁边或邻近的数据也是需要的，那么其在处理的时候就会将这些邻近的数据预先读取出来，这样会大大加快读取速度。BIOS里的 adjacent cache line prefetch 就是这项命令的开关，如果不需要预读取功能，可以将其关闭。 ↵
7. ACPI定义系统处理器的电源状态，要么为活跃状态(正在执行)，要么为睡眠状态(未执行)。处理器电源状态被设计为C0,C1,C2,C3...Cn。C0电源状态是活跃状态，即CPU执行指令。C1到Cn都是处理器睡眠状态，即和C0状态相比，处理器消耗更少的能源并且释放更少的热量。C0是CPU的正常工作模式，CPU处于100%运行状态。C后的数越高，CPU睡眠得越深，CPU的功耗被降低得越多，同时需要更多的时间回到C0模式。 ↵
8. TDP主要是提供给计算机系统厂商，散热片/风扇厂商，以及机箱厂商等等进行系统设计时使用的。一般TDP主要应用于CPU，CPU TDP值对应系列CPU的最终版本在满负荷时可能会达到的最高散热热量，散热器必须保证在处理器TDP最大的时候，处理器的温度仍然在设计范围之内。 ↵

工作负载的性质

工作负载的性质会影响性能，也有助于查明可能的性能瓶颈。例如，对于视频编码应用，应考虑以下常见的影响因素。

Compute-bound tasks (计算密集型任务)

如果是计算密集型任务，那么任务会在更快的CPU上更快的完成。如果任务是可并行化的，并且处理器数量增加可以使任务更快的完成，则任务也可以认为是计算密集型任务。计算密集型任务意味着任务的大部分时间都在使用CPU进行计算，而不是处理I/O或内存操作。取决于所使用的参数，许多视频编码任务（例如运动估计和预测，模式决策，变换和量化）都是计算密集型的任务。使用固定功能的硬件执行某些计算密集型任务的集成显卡极大提高了计算密集型任务的性能。

I/O-bound tasks (I/O密集型任务)

I/O密集型任务会随着I/O子系统的速度或I/O吞吐量的增加而更快的完成。通常，磁盘速度限制了I/O密集型任务的性能。从文件中读取原始视频数据并输入到视频编码器，尤其是读取高分辨率的未压缩视频数据，一般是I/O密集型任务。

Memory-bound tasks (内存密集型任务)

内存密集型任务的速度受可用内存容量和内存访问速度的限制。例如，将多个参考帧存储在内存中以用于视频编码的任务可能是内存密集型任务。更快的CPU可能会使得相同的任务从计算密集型转换为内存密集型。

Inter-process communication (进程间通信)

由于进程间的依赖性，并行执行的不同进程通常需要彼此通信。进程间通信在视频编码的并行任务中非常常见。可以使用消息传递，共享内存，或其他技术实现进程间通信。过多的进程间通信会降低性能。并且随着进程数量的增加，需要权衡进程执行和进程通信之间的平衡。实际上，即使要以增加计算或存储器操作为代价，并行视频编码器也需要最小化进程间的通信成本，从而获得更大的可扩展性。

Task scheduling (任务调度)

并行执行的任务的调度会影响整体性能，异构计算平台上更为明显。具有相同指令集架构 (*ISA, instruction set architecture*) 的异构多核处理器通常由小功率高效核和高性能核组成。通常，如果工作负载具有高级别的指令级并行性 (*ILP, instruction level parallelism*)，则小功率高效核可以实现良好的性能。另一方面，如果工作负载具备较高的内存级并行性 (*MLP, memory-level parallelism*) 或需要动态提取ILP，则高性能核可提供良好的性能。因此，通过分析不同类型的CPU核会如何利用工作负载的ILP和MLP特性，可以显著改善此类平台上的任务调度。另一方面，错误的调度决策可能导致次优的性能和过多的功耗。文献¹中提供了一些技术用以了解哪些工作负载到核心的映射可能提供最佳性能。

Latency (延迟)

远程内存访问的通信延迟 (*delay*) 会导致延迟 (*latency*)。延迟 (*latency*) 还会涉及到网络延迟 (*delay*)，缓存未命中以及由拆分事务中的竞争引起的延迟 (*delay*)。延迟隐藏可以通过四种互补的方法来完成：

- 使用预取技术，在实际需要之前将指令或数据带到处理器附近。
- 使用硬件支持的一致缓存 (*coherent caches*)² 减少高速缓存未命中。
- 使用宽松内存一致性模型 (*relaxed memory consistency models*)。
- 使用多上下文支持，从而使得CPU在遇到长延迟操作时可以从一个上下文切换到另一个上下文。

延迟影响系统响应。对于视频会议等实时视频通信应用，由于延迟会影响用户体验，因此延迟是一个重要的性能因素。

Throughput (吞吐量)

吞吐量用来衡量系统在单位时间内可以执行的任务量。吞吐量也称为系统吞吐量。CPU在单位时间内可以处理的任务数是CPU吞吐量。由于系统吞吐量来自CPU（和其他资源）吞吐量，当多个任务交叉执行时，CPU吞吐量会高于系统吞吐量。由于I/O，编译器和操作系统等带来的系统开销，因此CPU在很小的时间内会保持空闲状态。在实时视频通信应用中，视频的平滑度取决于系统吞吐量。因此，优化系统中的所有阶段的性能非常重要，只有这样，某个阶段的低效率才不会影响系统的整体性能。

¹. Scheduling Heterogeneous Multi-Cores through Performance Impact Estimation.

²

2. 在计算机科学中，缓存一致性（英语：Cache coherence，或cache coherency），又译为缓存连贯性、缓存同调，是指保留在高速缓存中的共享资源，保持数据一致性的机制。其有多个译名，如：缓存一致性，高速缓冲存储器一致性，等建议译为：高速缓存间一致性。 ↵

编码工具和参数

不仅是编码算法会影响编码器的性能，视频编码工具和参数同样会影响编码器的性能。这些编码工具中的大多数都是用于质量改进工具或作为优化传输错误的工具。但幸运的是，这些工具通常会通过并行化技术来提供性能优化能力。无法提供并行能力的工具可以利用算法优化和代码优化技术来优化性能。算法优化和代码优化会在接下来的章节介绍。以下是一些重要的工具和参数。

独立数据单元

为了促进并行能力和性能优化，视频编码算法通常会利用帧级（*frame-level*）独立性或帧级组（*group of frame-level*）独立性，或者将视频帧分解成独立的数据单元（切片（*slices*），切片组（*slice groups*），瓦片（*tiles*）¹ 或波前（*wavefronts*））。

由于运动补偿的依赖性，因此对于帧级而言，几乎没有并行性。并且即使并行化，由于帧复杂度的变化，编码和解码时间的波动也会很大，从而导致资源利用率的不平衡。另外，依赖结构（*dependency structure*）还可能会导致总体延迟随着帧级并行化而增加。

视频帧由一个或多个切片组成。切片是通常以光栅（*raster*）扫描顺序处理的一组宏块（*macroblocks*）。图5-2显示了划分为多个切片或切片组的视频帧。

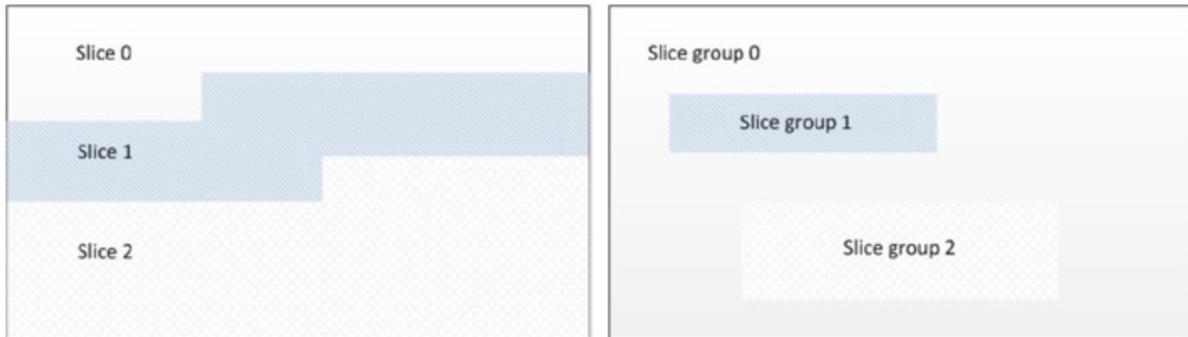


图5-2. 视频帧划分为切片或切片组

切片主要是为了防传输错误带来的质量损失。由于切片被定义为独立的数据单元，除非有其它切片会用丢失的切片作为参考，否则切片的丢失是局部的，不会影响到其他切片。利用切片的独立属性，切片可以并行计算从而提高性能。在使用AVC编码器的实验中发现，与每帧一个切片相比，每帧四个切片可以提升5%~15%的性能，其中性能提升的幅度取决于编码参数。为了保持数据单元的独立，可能会增加每帧数据的空间冗余。因此，切片并行可能会导致明显的编码效率损失。这种编码效率损失可能会表现为视觉质量的损失。例如，在先前的AVC编码器实验中，与每帧单个切片相比，每帧四个切片导致视觉质量损失约为0.2dB~0.4dB，其中，质量损失的多少取决于编码参数。并且，如果编码器接收到每帧只有一个切片的视频序列，依赖于多切片并行处理的编码器则无法获得应有的性能提升。

H.264之后的标准中，可以将图像分成矩形瓦片：即由垂直和水平边界划分的编码树块组。与切片边界类似，瓦片边界会打破解析和预测的依赖性，从而可以独立处理瓦片。但是，诸如去块滤波器（*deblocking filtering*）的环路滤波器（*in-loop filters*）仍然可以跨越瓦片边界。瓦片允许图片分区形状，从而使得瓦片的像素点比切片具有更高的相关性，并

且瓦片没有切片的头部开销。因此，与切片相比，瓦片具有更好的编码效率。但是，由于沿着瓦片边界的依赖性的破坏以及在每个瓦片开始时CABAC编码的概率性重置，与切片类似，编码效率会随着瓦片的数量增加而降低。

H.265标准中，引入波前（wavefront）来并行处理编码树块的行，每行在处理上一行的第二块之后再开始用可用的CABAC概率执行编码。这中处理过程产生了不同类型的依赖关系，但与切片和瓦片相比仍然具备独特的优势——在行边界处没有打破编码依赖性。图5-3展示了波前技术的例子。

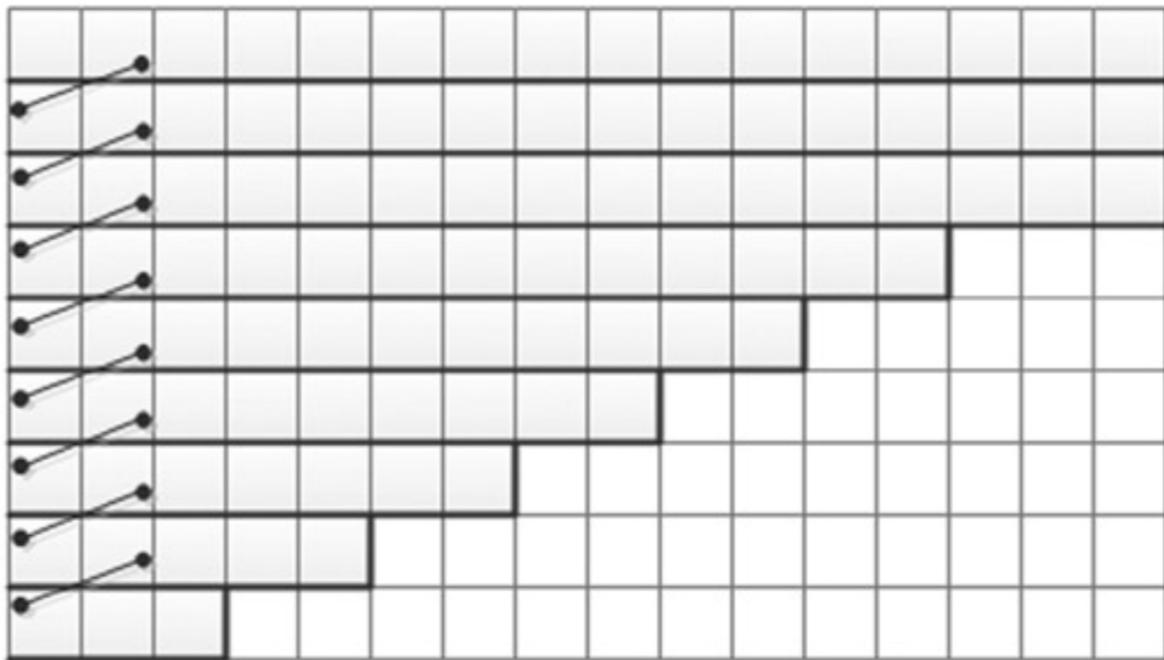


图5-3.适合并行处理的波前技术：对于每一行的起始宏块，CABAC的概率从上一行的第二块宏块获取

CABAC概率由前一行的第二个块而来，因此不需要改变光栅扫描顺序。与非并行比特流相比，波前提升了编码效率并且仅导致非常小的率失真（*rate-distortion*）差异。但是，波前依赖性意味着无法同时开始处理所有行。这导致了并行化的低效率，并且这种效率折损在多CPU中更为突出。

然而，通过重叠连续图像的执行可以降低波前并行处理的效率损失²。Chi等人的实验结果表明：在3.33 GHz的12核CPU系统上解码3840×2160视频序列，重叠波前提供了近11倍的加速，而常规波前和瓦片分别提供了9.3倍和8.7倍的加速。

¹. 视频编码方案支持将图片划分为被称为瓦片单元的较小矩形单元。能够分别通过单独的编码器和解码器对每一个瓦片单元独立地进行编码和解码。瓦片单元的主要目的是允许图片的并行处理以便降低实现成本和复杂性。 ↪

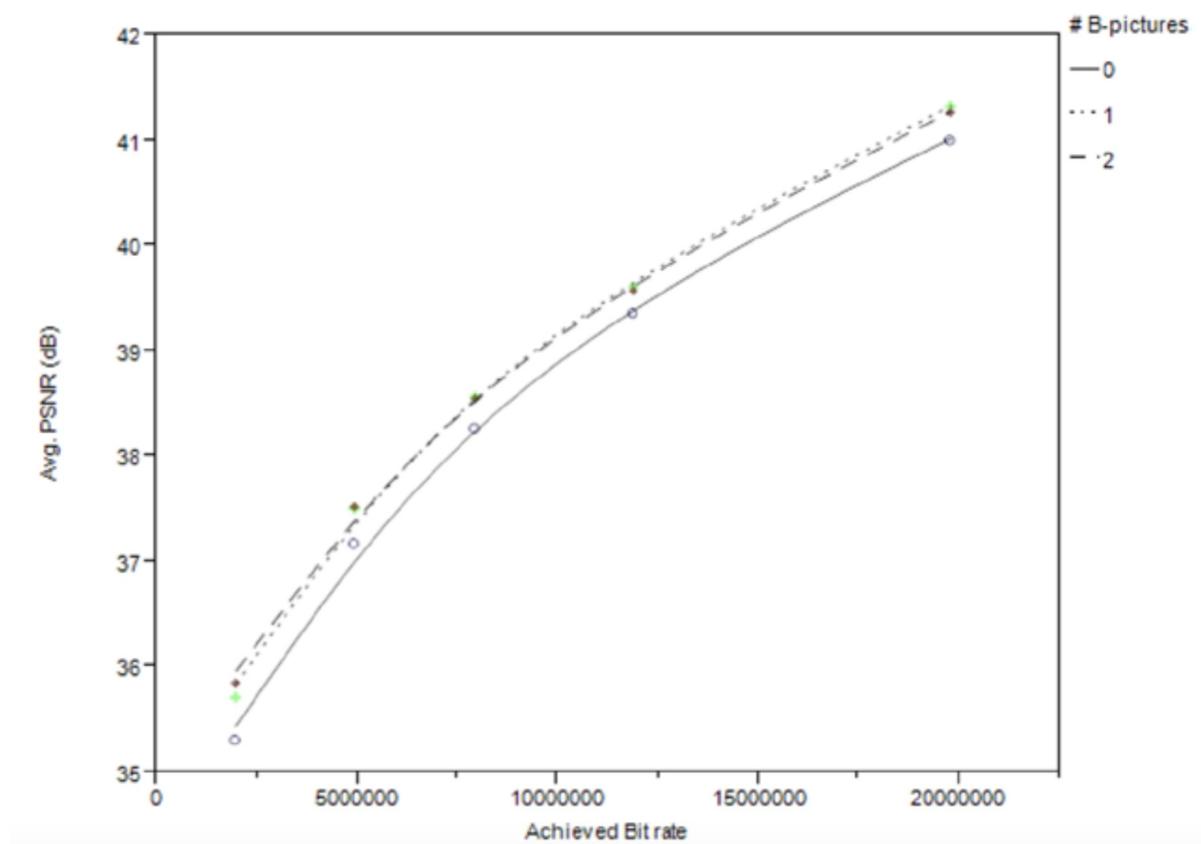
². C. C. Chi, M. Alvarez-Mesa, B. Juurlink, G. Clare, F. Henry, et al.: Parallel

Scalability and Efficiency of HEVC Parallelization Approaches. ↪

GOP结构

编码I帧、P帧、B帧的计算量不同，因此编码不同类型帧的编码时间不同。因此，帧的组合模式——通常称为图像组（GOP，group of pictures）结构，是影响编码速度的重要因素。在H.264之前的标准中，由于运动估计及其相关的复杂性的原因，I帧的编码速度最快，B帧的编码速度最慢。然而，在H.264以及更新的标准中，由于帧内预测的原因，编码I帧可能也需要花费很长时间。

根据视频内容的差异，使用H.264标准中的B帧可以在相同质量的情况下降低多达10%的码率。使用B帧时，内存访问频率的变化在-16%~+12%之间，因此B帧对性能的影响因视频序列而异。图5-4显示了另一个实验的结果，该实验对不使用B帧，使用一个B帧，使用两个B帧时获得的质量进行对比。由实验可知，使用更多的B帧会获得更好的质量。根据经验，B帧可能使单个处理单元的编码过程更慢。但B帧具备有效地并行化能力，因为除非用B帧作为参考帧，否则B帧通常不依赖于另一个B帧。



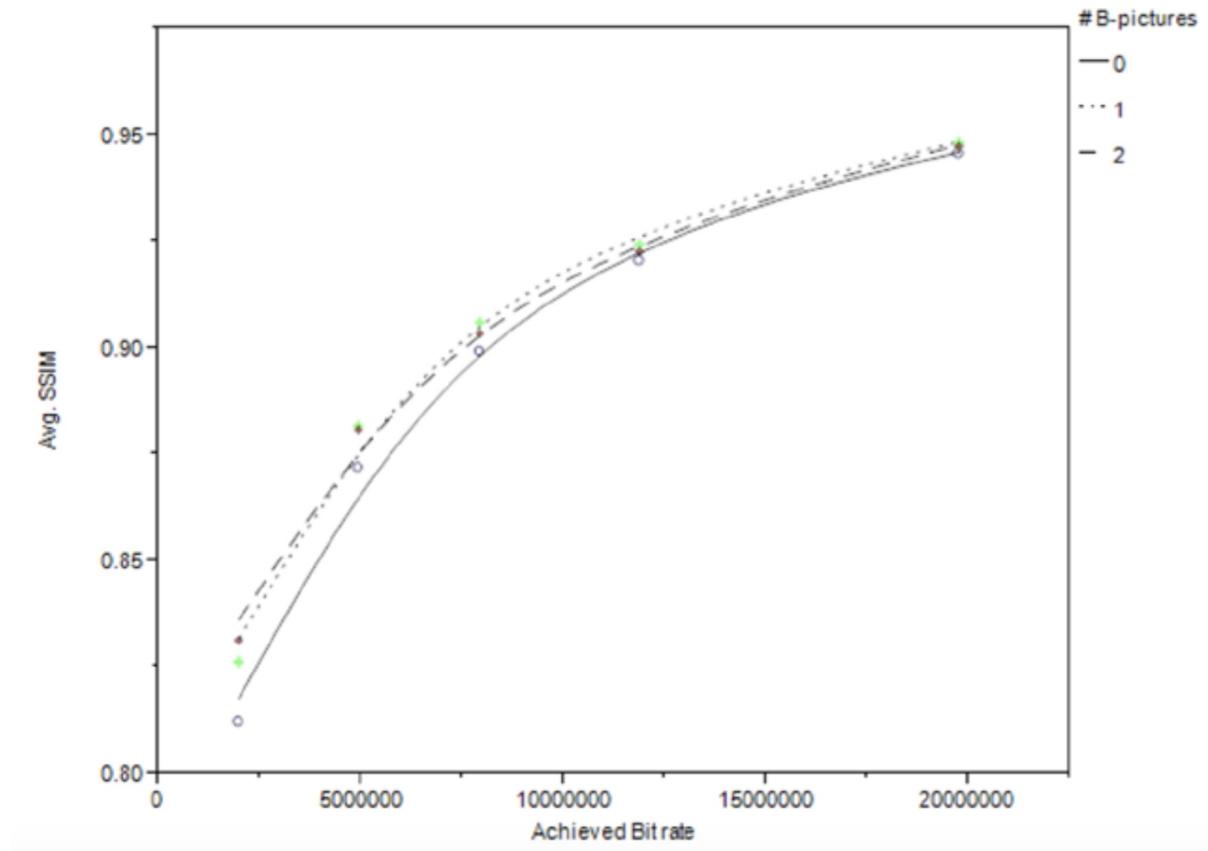


图5-4. B帧对1280×720的H.264编码的视频质量的影响

码率控制

相对于基于可用的码率水平和图片复杂度来控制每帧图像的量化参数，对GOP中的每帧图像使用固定的量化参数会使得编码速度更快。量化参数的控制需要额外计算。同时，因为编码器试图保持码率并尽量避免解码器缓冲器上溢或下溢，因此视频编码器中的码率控制机制需要确定所选择的量化参数对码率的影响。这会涉及从熵编码单元到码率控制单元的反馈路径，在反馈路径中会利用比特使用的更新信息重新计算码率控制模型的参数。通常，该过程会经历多次熵编码或模型参数计算。尽管该过程本质上是顺序的，但是可以优化码率控制算法以改善在有限的视频传输带宽上的应用性能。例如，在多次 (*multi-pass*) 码率控制算法中，可以通过尝试减少通过次数改善性能。算法还可以尝试收集统计数据并分析第一遍中的复杂度，然后在后续通过中执行实际熵编码，直到达到码率约束的要求。

多帧参考

很容易找到一个参考帧，从而可以产生比另一个参考帧更好的块匹配并因此降低运动预测成本。例如，在涉及遮挡区域的运动预测中，使用紧邻的先前或紧接的未来图像的规则模式对于某些宏块而言可能不会产生其最佳匹配。此时，可能需要在不同的参考帧中搜索该宏块的可见位置。有时，与单个参考帧相比，多参考帧会给出更好的运动预测——在与参考帧的特定网格不对齐的不规则对象的运动期间就是这种情况。图5-5显示了多参考帧的示例。

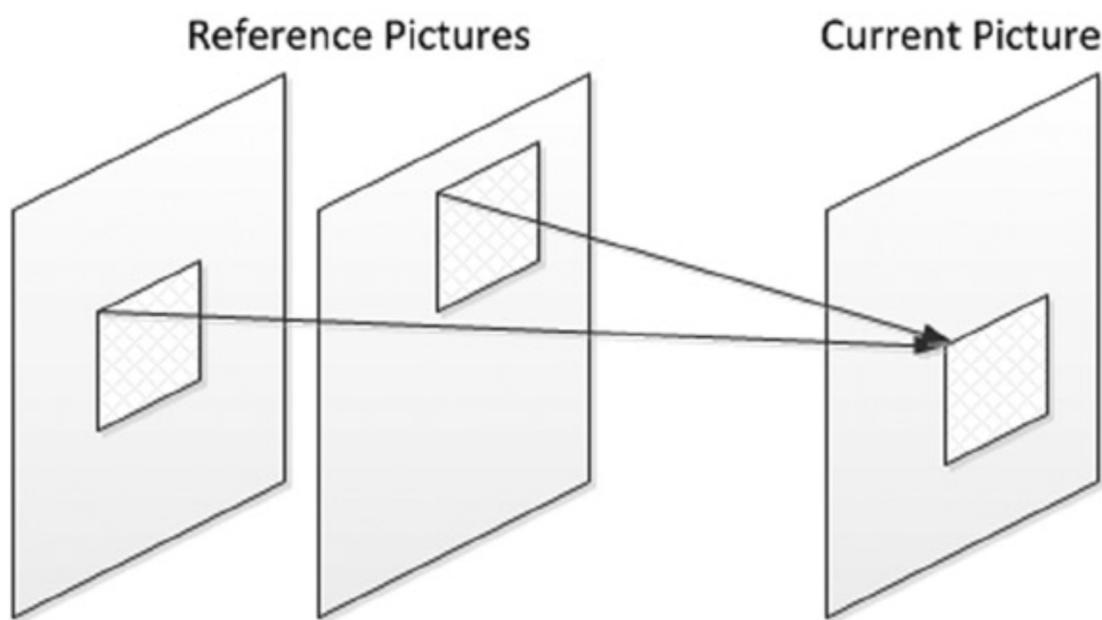


图5-5. 多参考帧的运动补偿预测

在H.264及更新的标准中，为了满足多种预测的需要，引入了多参考帧特征，从而提高视频的视觉质量。但是，在多个参考帧中执行搜索时会产生明显的性能成本。如果多参考帧中的搜索可以并行完成，则与单参考帧的运动预测相比，多参考帧可以在降低一定程度的性能的同时提供更高的视觉质量。

率失真的拉格朗日优化

对于使用H.264或者更新标准编码的视频，通常会使用拉格朗日优化技术 (*Lagrangian optimization techniques*) 来选择宏块模式 (*macroblock mode*) 和估计运动矢量 (*estimation of motion vectors*)。通过最小化R-D (*rate-distortion*) 代价函数，从所有可能的模式中选择每个宏块的模式。在选择的过程中，失真 (*distortion*) 由同一宏块的原始信号和重建信号之间的平方差之和表示，码率为用熵编码器对宏块进行编码所需的码率。类似地，可以通过最小化R-D代价函数来有效地估计运动矢量，其中失真通常由当前宏块和运动补偿宏块之间的平方差之和表示，码率为传输运动信息（运动信息包括运动矢量和其对应的参考帧序号）所需的信息的码率。上述两个最小化问题中的拉格朗日参数取决于量化参数，而量化参数又取决于目标码率。

显然，这两种最小化都需要大量的计算。虽然可以利用循环并行化，矢量化，和其它技术优化性能，但是算法也可以选择提前对出循环计算过程。提前退出循环也会带来可能的风险：

- 得到非最佳宏块模式
- 特定码率下的运动矢量可能会影响视觉质量

并行化方法将在下一节中讨论。

隔行扫描的帧/场模式

对于隔行扫描 (*interlaced*) 的视频，在宏块或图像级别选择帧/场模式会显着影响性能。另一方面，通常使用诸如宏块自适应或图像自适应的帧/场编码之类的工具来改善隔行扫描的视频质量。仅使用帧/场编码的某种特定模式可以增强性能，但是这可能会损伤视觉质量。

自适应去块滤波器

在重建的图像上使用环路去块滤波器 (*in-loop deblocking filter*)¹ 可以减少块状伪像 (*blocky artifacts*)²。因此，去块后的图像可以作为质量更好的参考帧用于帧内和帧间预测，并且在相同码率下具备更好的视觉质量。

去块滤波器³ 的强度可变并且可以做到在三个级别上自适应：

- 切片级别：基于视频序列的个体特性的切片级别 (*slice*)
- 块边缘级别：基于帧内模式与帧间模式决策、运动差异、以及两个相邻宏块残差的块边缘级别 (*block-edge*)
- 像素级别：基于分析来区分真实边缘和由块状伪像创建的边缘的像素级别 (*pixel*)。

去块滤波器应保持真边 (*true edges*) 不被过滤，而使量化边平滑。

一般来说，对于中等质量而言，去块会带来6%~9%的码率节省⁴，也就是说在相同的码率下，图像的主观质量会得到明显的改善。去块滤波器会在每帧图像上增加大量操作，因此会大大降低编码过程。此外，因为去块滤波计算不局限于独立的数据单元（例如切片），因此去块过程很难并行化。去块滤波是视觉质量和性能之间权衡的另一个例子。

¹. 在H264标准的8.7小节中规定了去块滤波的内容，这部分被称为环路滤波器 (*loop filter*)。环路滤波器是被放置在编解码的图像重建环路当中。在启用了环路滤波的编解码环境中，无论是编码器还是解码器，都是在图像被重建后才进行滤波。在编码器中，滤波后的图像会作为后续编码运动补偿的参考图像；在解码器中，滤波后的图像会被输出显示并且作为后续图像解码重建的参考图像。[←](#)

². H.264在编码过程中对像素残差进行了DCT变换，变换后得到的DCT系数是与每个像素都相关的，这些系数代表了被变换数据的基础色调与细节。h.264在DCT变换后对DCT系数进行了量化，量化能有效去除相邻像素间的空间冗余，也就是说会抹去元素数据的部分细节。比较理想的情况是量化抹去人眼无法识别的细节部分，但是在低码率的情况下就会导致原始数据的细节丢失过多。而且，DCT变换时基于块的，即将8x8或者4x4的像素残差进行变换后得到8x8或者4x4DCT系数，此时如果进行了低码率的量化，就会使得相邻两个块的相关性变差，从而出现块效应。h.264的运动补偿加剧了由变换量化导致的块效应。由于运动补偿块的匹配不可能绝对准确，各个块的残差大小程度存在差异，尤其是当相邻两个块所用参考帧不同、运动矢量或参考块的差距过大时，块边界上产生的数据不连续就更加明显。块效应主要有两种形式：一种是由于DCT高频系数被量化为0，使得强边缘在跨边界处出现锯齿状，称为梯形噪声；另一种经常出现在平坦区域，由于量化导致本来平缓变换的亮度块DC系数发生跳跃，造成变换块的基础色调改变，这种称为格形噪声。[←](#)

³. 为了减轻和消除视频图像中的块效应，通常会使用滤波器对块边界处的像素进行滤波以平滑像素值的突变，这种滤波被称为去块滤波器（Deblocking Filter）。[↪](#)

⁴. Ostermann et al., “Video Coding.” [↪](#)

视频复杂度和格式

视频复杂度是影响编码速度的重要因素。视频中越复杂的场景需要越长的编码时间，这主要是因为量化之后仍然有很多信息需要编码。复杂场景包括：具有精细纹理细节，任意形状，较高运动速度，随机不可预测运动，遮挡区域等的场景。例如，具有树木，火焰，烟雾等的场景通常是复杂场景，这些场景压缩效率通常较低，并且会影响编码速度。另一方面，简单的场景包含单色背景和一两个前景对象，例如头部和肩部类型的场景。简单场景通常具备更好的预测性，可以提前找到匹配的预测单元，并且简单场景有利于加速编码速度。可视电话，视频会议，新闻广播等应用程序一般会产生简单场景。频繁的场景变化需要对很多帧进行独立编码，导致帧数据的预测很难用于编码过程。如果要达到相同的视频质量，则只能实现较低的压缩。较低的压缩意味着需要处理更多的数据，这将不可避免的影响性能。

视频源格式和目标格式也是需要考虑的重要因素。除了电影和电视工作室生成的专业视频内容外，典型的视频来源包括：智能手机，傻瓜相机，消费类便携式摄像机和DVR/PVR。在视频内容消费领域，视频内容通常需要转换为特定的目标格式，以适配不同的设备（例如Apple iPad, Microsoft XBox, Sony PSx控制台等）或者用于上传到互联网。这种转换可以使用也可以不使用诸如缩放、去燥等类型的视频处理操作。因此，根据视频的目标格式的使用场景，转码操作的复杂性也不尽相同。并且还需要根据目标视频的使用场景在速度和性能之间做好权衡。

基于GPU加速的优化

应用程序和系统级软件可以利用硬件加速（特别是基于GPU的加速）来加速视频编码和任务处理。可以选择使用部分硬件加速或者全部硬件加速。例如，在转码应用中，解码、编码、或编解码，以及必要的视频处理任务都可以利用硬件加速来获得更好的性能。基于GPU的硬件加速，通常比实时性能要快一个数量级。即使对于复杂的视频，基于GPU的硬件加速也能将速度提升一个数量级。

此外，可以使用基于硬件的安全解决方案来实现硬件加速编码和任务处理的无缝集成，以全面提高优质视频内容的编码速度。在传统的安全解决方案中，安全软件偶尔会产生中断并降低运行在CPU上的长（时间）编码会话的速度。但是，基于硬件的安全性解决方案可以同时优化性能和安全性。

性能优化方法

主要的视频编码任务都需要进行性能优化，而通常会以牺牲视觉质量或功耗为代价进行性能优化。有些性能优化技术在提高性能的同时不会对功耗产生巨大的影响，有些也可能不会对质量带来很大影响。其它的性能优化技术则可能在提高性能的同时对质量或功率产生影响。

算法优化有助于加速视频编解码中涉及到的处理速度。如果算法在多核或多CPU环境中运行¹，则存在很多的并行化方法以供选择。此外，编译器优化和代码优化也会带来额外的性能提升。除了如上的技术之外，发现并消除性能瓶颈也是一种优化性能的重要方式。接下来将介绍视频编码上下文中的常见的性能优化技术。

¹. 多核和多CPU是两个完全不同的概念，多核CPU是单颗CPU里边有多个核心，可以多线程工作。多CPU是物理上就有多个CPU，此时的工作模式也不同，需要看操作系统与应用软件如何分配。 ↵

算法优化

视频编码算法多会采用以牺牲性能为代价来提高视频质量，例如在编码过程中使用B帧，在运动估计中使用多帧参考，两遍（two-pass）码率控制，R-D朗格朗日优化，自适应去块滤波器等。

另一方面，算法的性能优化会尝试以两种方式来提升性能：

- 第一种方法是使用快速算法，快速算法通常以更高的复杂性，更高的功耗或更低的质量为代价进行性能优化。文献¹中也提供了性能和复杂性的联合优化方法¹。
- 第二种方法是算法的并行化优化，这种优化方式不会对视频质量造成严重损失²。

¹. J. Zhang, Y. He, S. Yang, and Y. Zhong: Performance and Complexity Joint Optimization for H.264 Video Coding. [↔](#)

². S. M. Akramullah, I. Ahmad, and M. L. Liou: Optimization of H.263 Video Encoding Using a Single Processor Computer, Performance Tradeoffs and Benchmarking. [↔](#)

快速算法

很多文献已经提出了用于各种视频编码任务的快速算法，特别是对于那些需要更长时间才能完成的任务。例如，与存在质量损失隐患的全搜索 (*full-search*) 算法相比，许多快速运动估计算法试图实现高一个数量级的速度。最近的快速运动估计算法利用运动矢量的统计分布，仅在最可能的运动矢量候选集上进行搜索，从而达到在几乎没有质量损失的情况下优化算法性能。

类似地，快速DCT算法¹依赖于智能因子分解和智能代码优化技术。很多算法都会利用如下的理论事实：量化过程中存在的固有的舍入和截断不会影响DCT和逆DCT的整体精度²。用于其它视频编码任务的快速算法则试图通过降低搜索空间规模，提前退出循环，利用视频的固有属性等技术来实现更好的性能。

接下来我们会介绍几种使用算法优化来提高编码速度的方法。

¹. E. Feig and S. Winograd: Fast Algorithms for the Discrete Cosine Transform. ↵

². L. Kasperovich: Multiplication-free Scaled 8x8 DCT Algorithm with 530 Additions.

↵

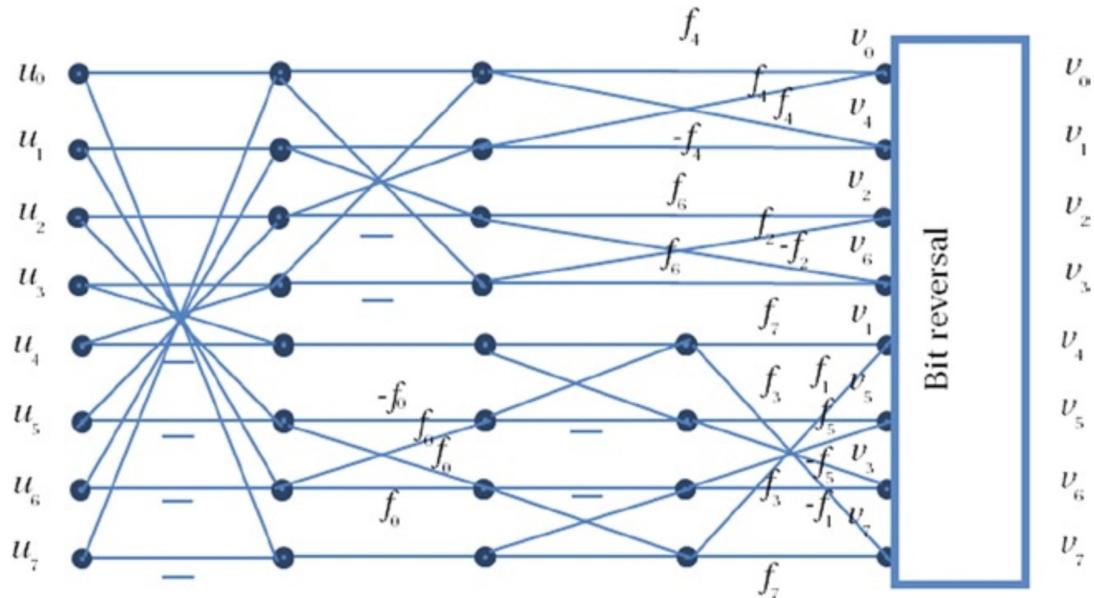
快速变换算法

快速变换使用因子分解和其他算法操作来降低所需的算术运算的计算复杂度。快速傅里叶变换 (*FFT, Fast Fourier Transform*) 就是其中之一，FFT可以以 $O(N \log N)$ 的时间复杂度完成离散傅立叶变换 (*DFT*) 算法的复杂度为 $O(N^2)$ 的运算。规模越大的数据集，FFT带来的节省时间会越明显。实际上，FFT使得实时计算傅里叶变换成为现实，从而使得许多实际应用得以实现。此外，快速变换倾向于使用整数运算来替代浮点数运算，从而实现更有效地优化。一般而言，类似DCT的快速变换不会引入误差，因此快速DCT不会对视觉质量产生额外影响。然而，由于算术运算量的减少，功耗增加通常也不再是一个重要问题。

快速DCT或其变体普遍应用于视频编码标准。H.264以及更新的编码标准中，变换与量化通常一起执行以避免算术精度的损失。尽管如此，由于视频数据量较大，因此可以采用数据并行方法来并行化变换并提升计算性能。下面的例子用以说明数据并行方法。

考虑如下图所示的DCT第一阶段的蝶形运算¹，可以用5-1来表示：

$$(5-1) \quad \begin{aligned} (u'_0, u'_1, u'_3, u'_2) &= (u_0, u_1, u_3, u_2) + (u_7, u_6, u_4, u_5) \\ (u'_4, u'_7, u'_5, u'_6) &= (u_3, u_0, u_2, u_1) + (u_4, u_7, u_5, u_6) \end{aligned}$$



Here, $f_0 = \cos(\pi/4)$, and $f_k = \frac{1}{2} \cos(k\pi/16)$, $k = 1, \dots, 7$.

假设每个输入 u_k 为16位整数，可以将类似的四个输入数据重新排列成64位宽的向量寄存器，如图5-6所示。对于需要执行运算的数据元素而言，必须要执行重排列操作。这样就可以并行提供64位宽的加法和减法操作，从而有效地将这部分操作的速度提升4倍。同样，可以利用更宽位数的向量寄存器来进一步提高性能。

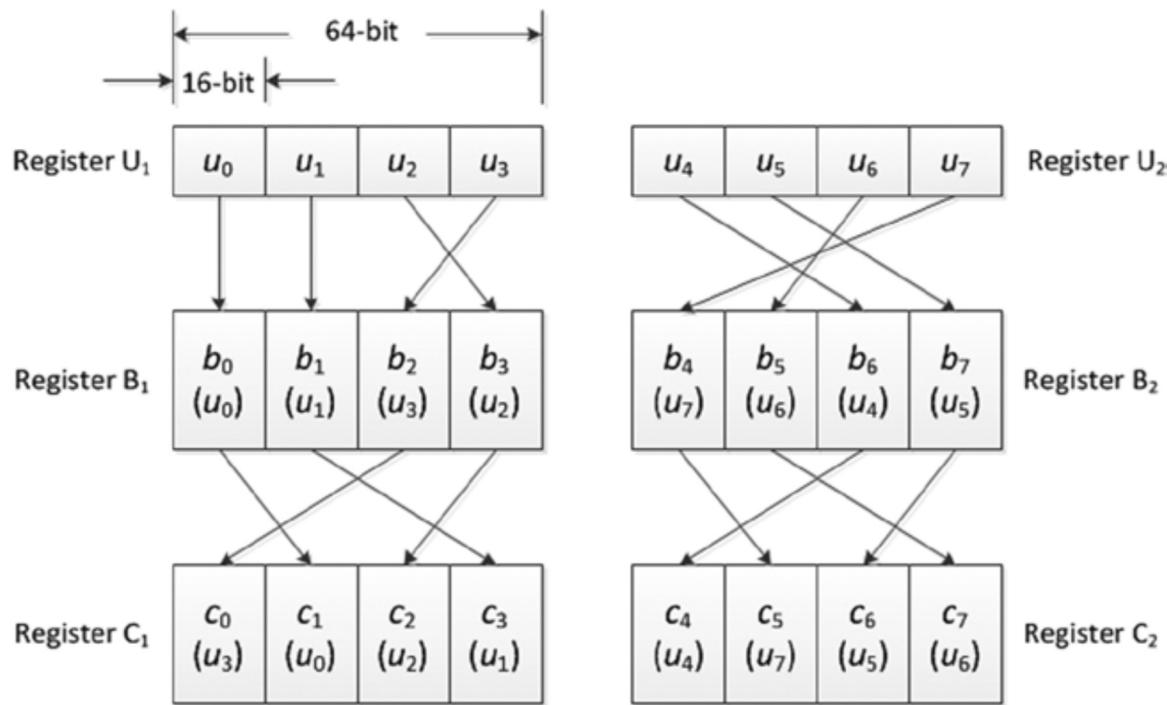


图5-6. 8点DCT中用于数据并行的数据重组

¹ 蝶形运算，2点DFT运算称为蝶形运算，而整个FFT就是由若干级迭代的蝶形运算组成，而且这种算法采用原位运算，故只需N个存储单元。 ↪

快速帧内预测算法

H.264以及更新的标准中，除了变换之外，还利用帧内预测来减少空间冗余。然而，帧内编码处理存在很多数据依赖并且是计算密集型的编码方法，这限制了整体的编码速度。帧内编码的特性不仅导致很高的计算复杂度，而且还会导致很大的延迟（特别是对于实时视频应用而言）。为了解决这些问题，基于DCT属性和空间活动分析，Elarabi和Bayoumi¹提出了一种高吞吐量、快速、精确的帧内模式选择算法和方向预测算法，该算法降低了计算复杂性和处理所需的时间。与标准AVC相比，该算法把帧内预测运行时间优化了56%。与其他快速帧内预测技术相比，帧内预测运行时间优化了35%~39%。同时，该算法的PSNR比JM 18.2算法优化了1.8%，比其他快速帧内预测算法优化了18%~22%。在另一个实验中，和标准算法相比，Alam等人²使用Z字形模式计算 4×4 的DC预测同时优化了PSNR（最高优化1.2dB）和运行时间（最高优化25%）。

¹. T. Elarabi and M. Bayoumi, “Full-search-free Intra Prediction Algorithm for Real-Time H.264/ AVC Decoder. [↪](#)

². T. Alam, J. Ikbal, and T. Alam, “Fast DC Mode Prediction Scheme for Intra 4×4 Block in H.264/AVC Video Coding Standard. [↪](#)

快速运动估计算法

块匹配运动估计是用于帧间运动预测和减少时间冗余的常用技术。运动估计执行在当前图片中的当前块中进行搜索以找到参考图片中的最佳匹配块。

估计过程一般分为两步：

1. 整数像素级精度的估计。
2. 分数像素级精度的估计。

通常，在最佳整数像素位置周围以半像素和四分之一像素精度完成分数像素级运动搜索，并适当缩放所得到的运动矢量以保持精度。

运动估计是编码框架中最耗时的过程。运动估计的计算过程通常占整个编码过程的60%~90%，具体占比多少取决于配置和算法。因此，运动估计的快速执行对于实时视频应用非常重要。

有很多方法可以用来加速运动估计，例如：

- 可以采用搜索较少位置的方式来找到匹配的块。然而，如何确定待搜索的位置已经成为二十多年来的一个活跃的研究领域，并且产生了很多快速运动估计算法。如果搜索位置不正确，则很容易陷入局部最小值，进而错过搜索空间中的全局最小值，从而导致无法获取理想的运动矢量。与将块简单地编码为帧内的情况相比，如果利用运动矢量从参考块来预测，则编码效率更高。因此，最终被编码为帧内块的块则无法利用现有的时间冗余。最近的算法通常在运动矢量的最可能候选者附近搜索以找到匹配块。基于相邻宏块的运动矢量，画面间的物体的运动趋势，或运动统计来形成预测的运动矢量。有的搜索算法使用重要性不同的多个搜索区域。例如，算法可以围绕预测的运动矢量开始搜索，并且如果需要，可以继续围绕参考图片中位于同一位置的宏块进行搜索。实验确定的阈值通常用于控制搜索流程。采用H.264和更高标准的相关软件使用了描述这些特征的快速搜索算法。
- 在每次搜索位置的过程中，可以利用部分信息而不是全部信息来匹配块。例如，从当前块中每隔一个像素选择一个像素，然后用选择出的像素与参考块中的对应像素匹配。
- 可以根据某些条件和实验确定的阈值来提前终止搜索。这种提前终止搜索的例子可以在Zhang等人提出的自适应运动估计技术中找到¹。该技术仅检查5个位置，从而将运动中的宏块的速度提高了约25%；即使对于静止的宏块而言，其性能也提高了约3%。并且该技术的平均PSNR损耗微不足道，大约为0.1 dB。
- 无需等到重建图片可用，可以采用源图像作为参考，从而节省了编码器所需要的重建过程。尽管此技术可明显提高性能，但它的缺点是预测误差会从一帧传播到下一帧，

从而导致视觉质量的显著下降。

- 运动估计很容易以数据并行方式并行化。因为相同的块匹配操作（例如，SAD）将用于所有的匹配候选者，并且匹配的候选者之间彼此独立，因此可以轻松地使用SIMD并行化运动估计。此外，只要可以从参考图片中获得每个块的合适的搜索窗口，就可以并行执行每个块的运动估计。结合这两种方法，可以对每张图片使用SPMD (*single program multiple data*) 类型的并行化。
- 使用按比例缩放的参考图片的层次结构，可以分别并行处理小数和整数像素部分，然后对结果进行组合。
- 在双向运动估计中，可以并行进行前向和后向估计。

¹. D. Zhang, G. Cao, and X. Gu, Improved Motion Estimation Based on Motion Region Identification. ↪

快速模式决策算法

H.264和之后的标准允许使用可变的块大小，这为提高编码效率提供了可能性。但是，由于选择块大小的计算复杂度非常高，这也导致块大小模式选择成为另一重要且耗时的过程。为了提高模式决策性能，Wu等人¹提出了一种基于空间均匀性和视频对象的时间平稳特性的快速帧间模式决策算法，该算法只需要从很少的模式中选择候选模式。宏块的空间均匀性取决于其边缘强度，而时间平稳性则取决于当前宏块与其在参考帧中位于同一位置的对应块之间的差异。该算法可减少30%的编码时间，而PSNR损失可忽略不计（仅为0.03 dB），同时码率增加了0.6%。

¹. D. Wu, F. Pan, K. P. Lim, S. Wu, Z. G. Li, et al., “Fast Intermode Decision in H.264/AVC Video Coding.” ↪

快速熵编码算法

诸如CABAC之类的熵编码本质上是一个顺序的任务，不适合并行化。因此，熵编码的性能通常是视频编码性能的瓶颈。CABAC引擎的性能优化可以提高整体编码的吞吐量。

Zhou等人¹在论文中提供了一个例子：通过预规范化（*pre-normalization*）、混合路径覆盖、和旁路二值化（*bypass bin*）拆分，可提高34%的吞吐量。通过使用状态双转换方案减少关键路径从而盖上上下文的建模性能，实现在330 MHz的65 nm的视频编码器芯片上进行实时超高清电视视频编码。

熵编码内容补充

熵编码是视频编码的最后一步和解码的第一步所使用的一种无损编码。熵编码所处理的对象，是在前期的预测、变换阶段所产生的一系列语法元素(Syntax Elements)，包括预测模式和残差数据等。这些语法元素描述了CU，PU，TU和LF等多种语法元素的特性。对CU，有块结构信息以及帧内/帧间预测模式；对PU，描述了帧内预测模式和运动信息等；对TU，主要包含残差信息的变换系数等；LF语法元素在每一个最大编码单元LCU中传输一次，描述了环路滤波中SAO的类型和偏移量。CABAC主要包括三大步骤，即二值化、上下文建模和算数编码。

- 二值化：二值化的作用是将语法元素映射为二进制符号(bin)。HEVC中的二值化采用几种不同的方式，与H.264类似，主要有一元(Unary)，截断一元(Truncated Unary)，k阶指数哥伦布编码(EGK)和定长(Fixed Length)。这几种不同模式的区别体现在将某个无符号整数N二值化的结果的不同，具体该表中的例子表述了大部分情况所采用的二值化方法，另外可能存在多种方法的组合，以及一些特定化的二值化方法。在实际应用中，具体采用哪一种二值化模式取决于语法元素的类型，或者之前处理过的语法元素的值以及条带参数中的设置。
- 上下文模型：CABAC之所以在压缩比率上可以取得巨大提高，关键就是因为上下文模型的引入为编码过程提供了精确的概率估计。CABAC采用的上下文模型是高度自适应的，不同二进制码元采用的模型不同，而且可以依据之前处理的二值化码流进行模型更新。每一个bin的上下文模型的选择依据包括语法元素类型、bin的位置、亮度/色度和相邻块信息等。CABAC的概率模型采用7bit结构，其中6bit的概率状态位和1bit的最大概率模型位，在HEVC中其概率模型更新方法与H.264的类似，而改进了上下文选择的逻辑以提高数据处理效率。
- 算数编码：算数编码是一种基于区间的递归划分的熵编码方法。一个初始化为[0,1]的区间根据bin的概率分布划分为两个子区间，并且依照bin的取值选取两个区间之一。该区间更新为选择的子区间，并进行下一次分割，依此循环往复。为防止下溢出，当

区间长度小于某个值时，停止递归并重新进行区间归一化。在编码的过程中，可以使用概率估计（上下文编码）和等概率模式（旁路编码）。旁路编码中，区间划分由某个偏移量实现，而上下文编码的bin需查表。HEVC的编码过程与H.264类似。

¹. J. Zhou, D. Zhou, W. Fei, and S. Goto, “A High Performance CABAC Encoder Architecture for HEVC and H.264/AVC. ↪

并行化方法

并行化对于启用适用于当今多核体系结构的多线程编、解码应用程序至关重要。独立数据单元可以利用并行单元轻松实现扩展，而数据之间的相关性则限制了数据的伸缩性和并行化效率。在视频数据结构中可以找到独立的数据单元，因此其并行化非常简单。但是，并非所有的数据单元和任务都是独立的。

当数据单元或任务之间存在依赖关系时，有两种方法可以处理依赖关系：

- 将适当的数据单元传达给正确的处理器
- 使用冗余数据结构

必须注意，与顺序处理相比，并行处理需要增加处理器间的通信开销。因此，并行化方法需要关注进程间的通信成本，有时也需要关注冗余数据的存储。需要在计算、通信、存储要求和资源利用率之间做好权衡以实现高效的并行化。

数据分区

H.264标准将语法元素分为最多三个不同的分区，以进行基于优先级的传输。例如，通常头信息、运动矢量和其它预测信息的传输优先级高于表示视频内容的语法元素。这种数据分区主要是为了提高传输错误的鲁棒性而设计的，并非用于并行化。实际上，并行处理头部的几个字节和详细视频数据的很多字节将是无效的。但是，视频数据可以以多种不同的方式进行分区，使其适合于并行化并提高性能。未压缩和压缩的视频数据都可以划分为独立的部分，因此视频编码和解码操作都可以从数据分区中受益。

在视频编码并行化中，数据分区起到重要作用。将视频序列按照时间维度分割为多个独立的子序列，并以流水线的方式同时处理分割之后的多个独立的子序列。至少存在几个子序列必须可用来填充流水线阶段。因此，这种类型的分区适用于脱机视频编码¹。空间分区则将视频帧分为多个同时进行编码的不同部分。由于一次只能输入一帧，因此这种分区适用于在线和低延迟编码应用程序，这些应用程序逐帧处理视频。显然，视频子序列的并行编码处理的数据粒度较粗，可以将其进一步划分为较细粒度的数据：例如单个帧的一部分，例如切片（*slices*）、切片组（*slices group*）、图块（*tiles*）或波前（*wavefronts*）。

¹. I. Ahmad, S. M. Akramullah, M. L. Liou, and M. Kafil, “A Scalable Off-line MPEG-2 Video Encoding Scheme Using a Multiprocessor System. ↪

任务并行化

视频编码任务的并行化方法最早是在1991年针对光盘交互式应用程序而引入的¹。该方法利用了多指令多数据（MIMD, *multiple instruction multiple data*）的优势。视频编码器划分为多个任务，每个任务分配给计算机的一个或多个处理器处理。该计算机具有100个节点的消息传递并行能力，并且节点由数据处理器、内存、通信处理器和I/O接口组成。这种任务并行化方法较为松散，在给定时间内，某些处理器运行不同算法的任务，而其它处理器则执行相同算法的任务。在更高的级别上，任务被分为两个阶段：用于预测和内插的运动估计阶段，其中在每个帧中搜索运动矢量；以及视频压缩，确定要使用这些运动矢量中的哪一个。

运动估计阶段的并行化本身并不是任务并行的。运动估计阶段的并行化涉及到为每个处理器分配帧以及相关的参考帧。此过程不可避免地需要将参考帧复制到几个合适的处理器上，从而产生性能开销。同样，在所有处理器执行某些任务之前，必须先读取许多帧。视频压缩阶段并没有独立的帧，因此需要并行处理帧的几个部分。由一组处理器组成的压缩单元重复接收要编码的连续块集。压缩单元的任务是模式决策，DCT，量化和可变长度编码。压缩单元产生的比特流发送到运行在单独处理器上的输出管理器，该输出管理器将来自所有压缩单元的片段组合在一起，并将结果发送到主机。压缩单元重建自己产生的结果比特流的一部分以获得参考帧。

请注意，量化参数取决于先前处理的所有块中的数据缩减量，仅依靠一个处理器无法计算量化参数。因此，必须使用专用处理器来计算量化参数，并将参数发送到适当的压缩单元，然后从每个压缩单元收集压缩数据的大小以进行进一步的计算。通常会基于先前的运动矢量对运动矢量进行差分编码，因此会引入额外的复杂度。但是，独立工作的压缩单元无法访问先前的运动矢量。因此，为了解决这个问题，压缩单元必须将比特流中使用的最后一个运动矢量发送给压缩下一个数据块的压缩单元。图5-7显示了任务并行化方法的通信结构。

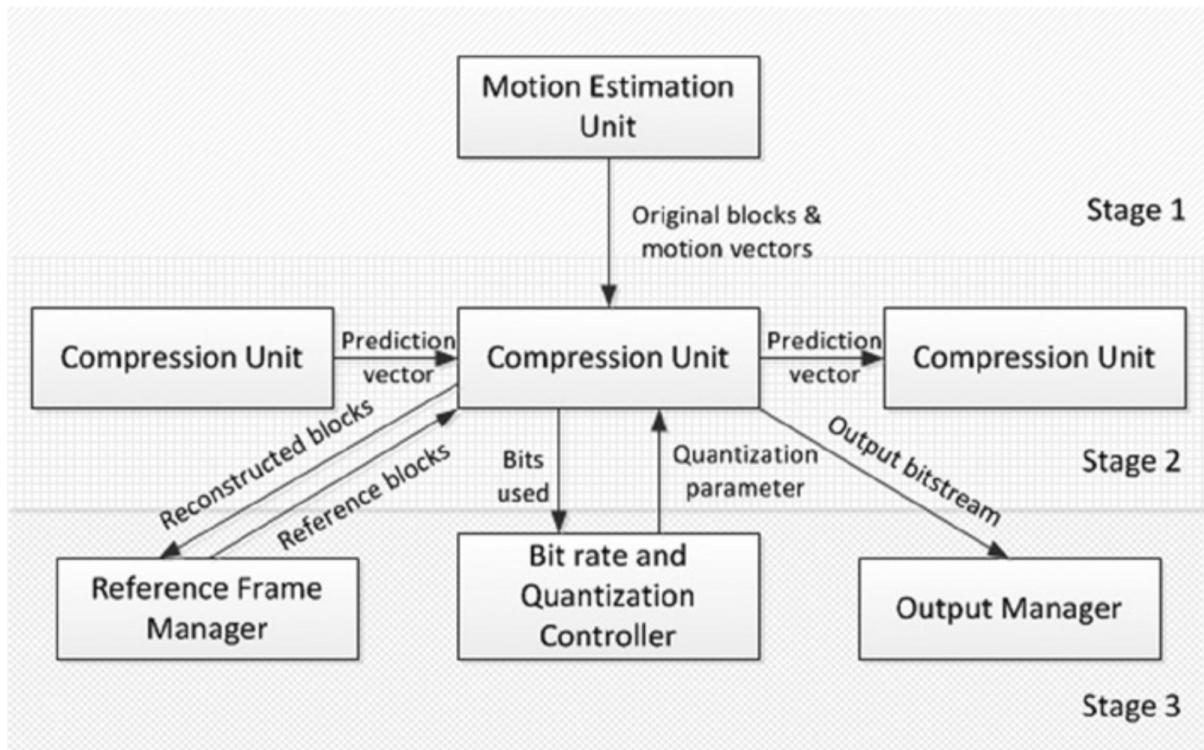


图5-7. 并行任务的通信结构

不管使用哪种视频编码标准或算法，该想法都可以用于视频编码。但是，可以进一步改进该思想以减少通信开销。例如，在系统中，处理器可以在环境中标识自己，并且可以将其处理器的编号作为标签附加到它们所处理的数据上。这些标签随后可以由合适的目的处理器删除，并且目的处理器可以根据需要轻松地重新排列数据。重要的是要了解：因为由于帧级别的依赖性，许多任务都依赖于其他任务，因此在任务并行化方法中必须进行适当的任务调度。

¹. F. Sijstermans and J. Meer, “CD-I Full-motion Video Encoding on a Parallel Computer. ↵

流水线技术

流水线是处理阶段的级联，其中每个阶段都对从一端流到另一端的数据流执行某些固定的功能。流水线可以是线性的也可以是动态的（非线性）。线性流水线是具有流线型连接的简单级联阶段，而在动态流水线中，反馈和前向反馈连接路径可能存在于两个不同的阶段。线性流水线可以进一步分为同步流水线和异步流水线。在异步流水线中，相邻阶段之间的数据流由握手协议控制，其中阶段 S_i 在准备好传输数据时将就绪信号发送到下一个阶段 S_{i+1} 。一旦阶段 S_{i+1} 接收到数据，它就将确认信号发送回 S_i 。在同步流水线中，时钟锁用于连接各个阶段。当时钟脉冲到达时，所有锁同时将数据传输到下一级。对于 k 级线性流水线，需要 k 个时钟周期的倍数才能使数据流过流水线¹。流水线两次启动之间的时钟周期数称为流水线等待时间。流水线效率取决于每个流水线所用时间的百分比，这称为级利用率（stage utilization）。

如图5-7所示，视频编码任务可以形成一个三阶段的动态流水线。第一阶段包括运动估计单元。第二阶段有几个并行的压缩单元。第三阶段是输出管理器。可以将码率和量化控制单元以及参考帧管理器视为具有与第二级组件的反馈连接的两个延迟级。

¹. Hwang, Advanced Computer Architecture. ↪

数据并行化

如果可以将数据划分为独立的单元，则可以以最小的通信开销并行处理。视频数据具有这样的特征。有几种常见的数据并行化执行模式，包括单指令多数据（SIMD），单程序多数据（SPMD），多指令多数据（MIMD）等。

SIMD是处理器支持的技术， SIMD允许同时操作多个数据点。 SIMD提供了数据级的并行性，并且比标量处理更有效。例如，某些循环操作在连续的迭代中是独立的，因此一组指令处理不同的数据集。在开始执行下一条指令之前，通常需要在多个数据集上执行同一指令的执行单元之间进行同步。

SIMD特别适用于在大量数据点上执行相同操作的图像和视频应用程序。例如，在亮度调整中，将相同的值添加到一帧中的所有像素（或从中减去）。实际上，这些操作非常普遍，以至于大多数现代CPU设计都包含针对SIMD的特殊指令集，以提高多媒体应用的性能。图5-8显示了SIMD技术的例子：对两个16位短整数的源数组A和B的每个元素同时做加法运算，并将对应的结果写入目标数组C。使用SIMD技术，可在在一个时钟周期内完成对128位宽的数据的加法操作。

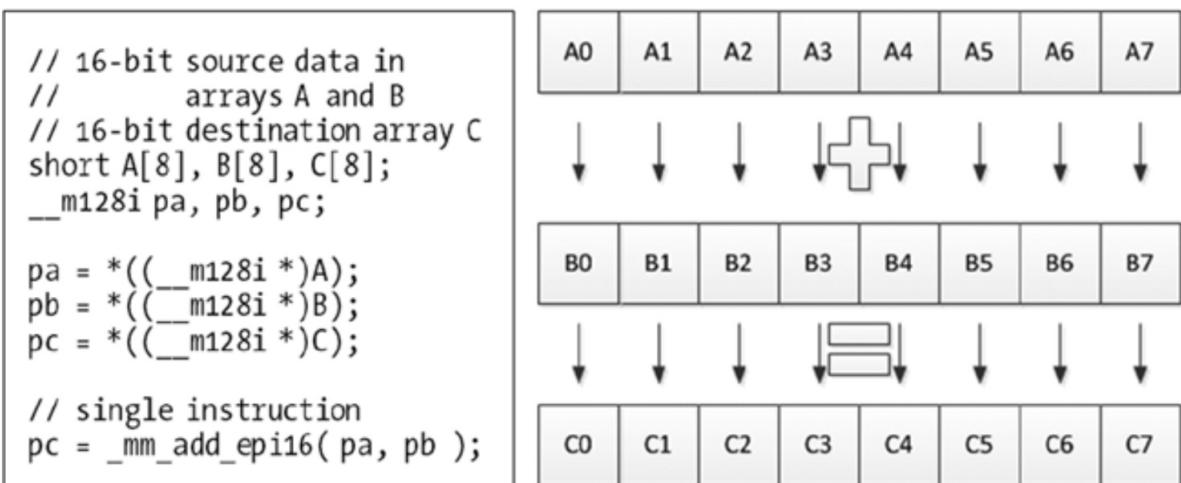


图5-8. SIMD示例

过程或任务级并行化通常在MIMD执行模式下执行，其中SPMD是一种特殊情况。在SPMD中，程序被分成较小的独立过程或任务，并且这些任务在具有潜在的不同输入数据的多个处理器上同时运行。与任务中的指令级相反，通常需要在任务级进行同步。通常可以在利用消息传递实现同步的分布式内存计算机体系结构中找到SPMD模式的实现。对于视频编码应用，Akramullah等人提出了一种类似的SPMD方法¹。

¹. S. M. Akramullah, I. Ahmad, and M. L. Liou, "A Data-parallel Approach for Real-time MPEG-2 Video Encoding. ↪

指令并行化

编译器将视频算法的高级实现转换为低级机器指令。但是，有些指令并不依赖于先前的指令，因此，可以安排无依赖相关性的指令同时执行。因为可以并行评估指令，因此指令之间的潜在重叠构成了指令并行化的基础。例如，考虑如下的代码：

```
R4 = R1 + R2
R5 = R1 - R3
R6 = R4 + R5
R7 = R4 - R5
```

在如上的例子中，指令1和2之间或指令3和4之间没有依赖性，但是指令3和4取决于指令1和2的完成。因此，指令1和2以及指令3和4可以并行执行。通常基于编译器的优化和硬件技术来实现指令的并行化。但是，不确定的指令（*indefinite instruction*）的并行化是不可能的。并行化通常受数据依赖性，过程依赖性和资源冲突的限制。

精简指令集计算机（RISC）处理器中的指令有四个阶段可以重叠，以实现每个周期接近一条指令的平均性能。这些阶段是：指令获取，解码，执行和结果写回。通常同时获取和解码两个指令A和B，但是如果指令B对指令A具有写后读取依赖性，则B的执行阶段必须等待直到对A的写入完成。因此，在一次执行一条指令的标量处理器中，每个周期不能实现一条以上的指令。但是，超标量处理器利用指令并行化来一次执行多个不相关的指令。例如， $z = x + y$ 和 $c = a * b$ 可以一起执行。在这些处理器中，硬件用于检测并并行执行独立指令。

作为超标量处理器的替代方法，超长指令字（VLIW）处理器体系结构利用了指令并行化的优势，并允许程序显式指定要并行执行的指令。这些架构采用积极的编译器，可以在每个周期的一个VLIW中调度多个操作。在这样的平台上，编译器负责查找和调度并行指令。在实用的VLIW处理器（如Equator BSP-15）中，集成的缓存很小——32KB的数据缓存和32 KB的指令缓存通常作为处理器内核和内存之间的桥接（bridges）。连续传输数据非常重要，这样就可以避免等待时间。

为了更好地理解如何在视频编码中利用指令并行性，考虑在VLIW平台上实现视频编码器的示例¹。图5-9给出了编码系统的通用结构的框图。

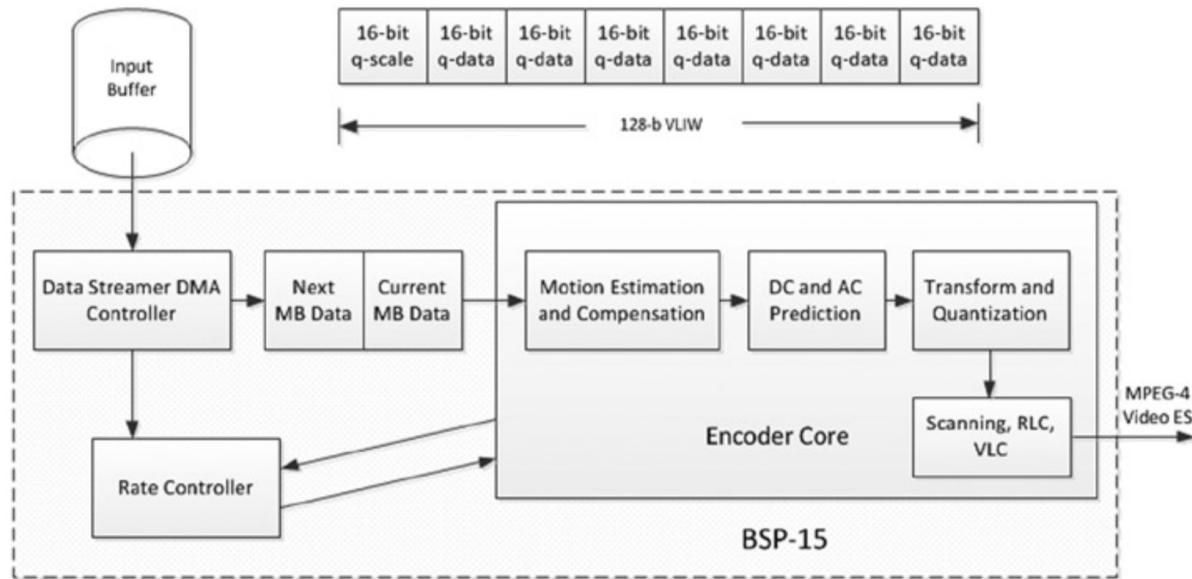


图5-9. VLIW平台上的视频编码器的框图

图5-9中，宏块以流水线方式处理，同时它们在编码器核心的各个流水线级中经历不同的编码任务。DMA (*direct memory access*) 控制器（通常称为数据流处理器）有助于预取必要的数据。使用双重缓冲技术来连续提供流水线级。该技术以交替方式使用两个缓冲区——当主动使用一个缓冲区中的数据时，下一组数据将加载到第二个缓冲区中。处理活动缓冲区的数据后，第二个缓冲区将成为新的活动缓冲区，并开始对其进行处理，而数据处理完毕的缓冲区则将被新数据重新填充。这种设计有助于避免可能存在的性能瓶颈。

需要特别注意，把适当的数据存储到缓存中非常重要，以便可以最大化地利用数据高速缓存和指令高速缓存。为了最大程度地避免缓存未命中的情况，流水线中每个阶段的指令必须满足指令高速缓存的要求，而数据必须满足数据高速缓存的要求。可以重新安排程序，以便编译器可以生成满足指令高速缓存要求的指令。同样，仔细考虑数据预取将保持数据高速缓存处于满额的状态。例如，可以以某种方式存储量化的DCT系数，以帮助某些帧内预测模式下的数据预取，此时，在给定时间内仅需要七个系数（从上一行或从左列）。系数具有 (-2048, 2047) 的动态范围，每个系数需要13 bits，但一般用16位有符号实体表示。七个这样的系数将适合于两个64位寄存器，其中一个16位插槽不能占用。注意，可以将与该流水线级相关的16位元素（例如量化器缩放比例或DC缩放比例）与量化系数打包在一起，以填充寄存器中的空闲插槽，从而实现更好的缓存利用率。

¹ S. M. Akramullah, R. Giduthuri, and G. Rajan, “MPEG-4 Advanced Simple Profile Video Encoding on an Embedded Multimedia System.” ↪

多线程技术

线程由程序上下文表示，该程序上下文包括程序计数器，寄存器集和上下文状态。在多线程并行计算模型中，无论是在SIMD，多处理器或多计算机上运行还是在具有分布式或共享内存，基本单元都由同时运行的多个计算线程组成，每个线程都处理基于上下文切换中的不同的上下文。多线程并行的基本结构如下¹：计算从顺序线程开始，然后是监督调度，在此调度中计算线程开始并行工作。分布式存储器体系结构通常在每个处理器上运行一个或多个线程，需要根据需要执行处理器间的通信。最后，在开始下一个并行工作单元之前进行多个线程的同步。

即使一个线程停滞了，它也不会阻止其它线程使用可用资源，并且处理同一数据的多个线程可以共享高速缓存以更好地使用高速缓存。因此，多线程提高了整体执行性能。但是，线程通常在独立的数据集上工作，并且在使用共享资源时经常会相互干扰。这通常会导致高速缓存未命中。另外，多线程的同步，优先级和抢占处理要求等方面增加了计算复杂性。

从多个线程同时执行指令通常被称为并行多线程，或称为Intel处理器上的Intel超线程技术。为了减少流水线中相关指令的数量，超线程利用了虚拟或逻辑处理器内核。对于每个物理核心，操作系统都寻址两个逻辑处理器，并在可能时共享工作负载和执行资源。

由于仅使用媒体专用指令的性能优化不足以实现实时编码性能，因此利用线程级并行提高视频编码器的性能已变得非常流行。如今，多线程通常用于视频编码器速度优化。异步运行的线程可以将帧数据分派给基于CPU的软件和GPU加速的实现中的多个执行单元。在CPU和GPU之间分配各种执行线程也成为可能。

多线程通常与任务并行化，数据并行化或其组合一起使用，其中每个线程对不同的任务或数据集进行操作。关于视频编码中使用的多线程的讨论可以在Gerber等人的文章中找到²，该论文使用多线程技术实现帧级（*frame-level*）和切片级（*slice-level*）并行性。

¹. G. Bell, “Ultracomputers: A Teraflop before Its Time.” ↪

². R. Gerber, A. J. C. Bik, K. Smith, and X. Tian, “Optimizing Video Encoding Using Threads and Parallelism.” ↪

向量化技术

向量由具有相同标量数据类型的多个元素组成。向量长度是指一起处理的向量元素数，通常为2、4、8或16个元素。

$$\text{Vector length (in number of elements)} = \frac{\text{size of vector registers (in bits)}}{\text{size of the data type (in bits)}}$$

例如，128位宽的向量寄存器可以处理八个16位的短整数。在这种情况下，向量长度为8。理想情况下，向量长度由开发人员或编译器选择，以匹配基础向量寄存器的宽度。

向量化是一种转换循环的过程，该循环遍历多对数据项并为每对数据分配一个单独的处理单元。每个处理单元都属于一个向量通道。向量通道的数量与向量长度相同，因此，可以使用尽可能多的向量通道同时处理2、4、8或16个数据项。例如，考虑将大小为1024个元素的数组A和数组B相加，并将结果写入数组C，其中B和C与A的大小相同。要实现此加法，使用标量代码需要1024次循环迭代。但是，如果在处理单元中有8个向量通道，则可以一起处理数组的8个元素的向量，因此仅需要128（1024/8）次迭代。向量化与线程级并行性不同。向量化试图通过尽可能多地使用更多矢量通道来提高性能。向量通道在单个处理器内核上运行的每个线程之上提供了额外的并行性。向量化的目的是最大程度地利用每个内核的可用向量寄存器。

从技术上讲，历史上的向量处理体系结构被认为与SIMD体系结构是分开的。如上的观点基于如下的事实：向量机用于通过流水线处理器一次处理一个单词（尽管仍然基于单个指令），而现代SIMD机却同时处理向量的所有元素。但是，如今，具有SIMD处理功能的许多计算单元在硬件级别可用，并且向量处理器实质上是SIMD处理器的同义词。在过去的几十年中，每个处理器内核中都有越来越广泛的向量寄存器可用于向量化：例如，奔腾中的64位MMX寄存器支持MMX扩展，奔腾IV中的128位XMM寄存器支持SSE和SSE2扩展，第二代Core处理器中的256位YMM寄存器以支持AVX和AVX2扩展，Xeon Phi协处理器中的512位ZMM寄存器支持MIC扩展。对于数据并行的应用（例如视频编码）而言，这些宽向量寄存器非常有用。

常规编程语言因其固有的串行特性而受到限制，并且不支持SIMD处理器提供的计算功能。因此，需要扩展常规编程语言才能利用这些功能。为此，开发了串行代码的向量化和向量编程模型。例如，OpenMP 4.0支持C/C++和FORTRAN的向量编程模型，并提供语言扩展以简化向量编程，从而使开发人员能够从SIMD处理器中优化更多性能。英特尔Click Plus是另一个支持类似语言扩展的示例。

自动向量化过程会在给定其串行约束的情况下尝试对程序进行向量化处理，但却未充分利用可用的计算能力。随着向量宽度和核数的增加，英特尔开发了明确的方法来应对这一趋势。随着现代CPU中集成显卡和协处理器的应用，具有显式向量编程功能的通用编程模型被添加到诸如Intel编译器，GCC和LLVM的编译器中，以及OpenMP 4.0等标准中。该方法类似于多线程，解决了多个内核的可用性并使这些内核上的程序并行化。向量化还通过显式向量编程解决了增加向量宽度的可用性。

向量化在视频编码性能优化中很有用，尤其是对于基于CPU的软件实现而言。像素数据长度为16个元素的向量可以在关键循环内提供高达16倍的速度提升，例如，用于运动估计，预测，变换和量化操作。在诸如视频转码的应用中，某些视频处理任务（例如降噪）可以利用视频数据的易于向量化的结构来并提高速度。

编译器和代码优化

有很多编译器代码优化或者人工代码优化技术可以提高程序的性能。几乎所有这些技术都可以在不影响视觉质量的前提下提高编、解码器的性能。但是，通常需要根据应用程序的需求优化程序的关键路径。在本节中，简要介绍了一些常见的编译器优化和代码优化技术。这些技术对于通过GPU加速视频编码器的性能一般是有限的，并且仅限于应用程序和SDK级别。在GPU加速技术中（硬解），主要的编码任务实际上是由硬件单元完成的。但是，其中一些技术已成功用于基于CPU的速度优化中（软解）¹，并显著提高了视频编码的性能。

¹. Akramullah et al., “Optimization. ↪

编译器优化

大多数编译器带有可选的优化标志，编译器优化可以权衡编译后的代码大小和执行速度。为了提高速度，编译器通常执行以下操作。

- **将变量存储在寄存器中：**编译器会将常用的变量和子表达式存储在寄存器中，编译器还将自动为这些变量分配寄存器。
- **采用循环优化：**编译器可以自动执行各种循环优化，包括完整或部分循环展开，循环分段等。循环优化可以显著提高视频应用的性能。
- **忽略调用堆栈上的帧指针：**通常，帧指针（*frame pointers*）在调用堆栈上不是严格必需的，可以安全地省略。一般而言，忽略调用堆栈上的帧指针可以或多或少的提高程序的性能。
- **改善浮点数的一致性：**例如，可以通过禁用可能改变浮点数精度的优化来提高一致性。这是不同类型的性能优化之间的权衡。
- **减少函数调用的开销：**例如，可以将某些函数调用替换为编译器的固有函数来实现。
- **权衡寄存器空间的节省与内存事务：**实现这种折衷的一种方法是在每次函数调用之后从内存中重新加载指针变量。这是在不同类型的性能优化之间进行选择的另一个示例。

代码优化

软件中的所有的代码并不是都值得去优化。我们需要将优化的重点放在那些可以最大程度地减少执行时间的代码。因此，通常需要分析应用程序中各种任务的执行时间。

以下的技术通常可以显著提高程序的性能，尤其是在编译器无法有效使用系统资源的情况下。

减少冗余操作

仔细的编码是产生紧凑代码的关键。在不影响功能的情况下，可以通过仔细检查代码来减少或消除代码中的多余操作。

数据类型优化

为程序的关键路径选择合适的数据类型对程序的性能优化至关重要。直接从任务定义中得出的数据类型可能无法为各种功能单元带来最佳性能。例如，在大多数的DCT和IDCT 算法中，使用比例缩放的浮点常量并将预算算的常量分配给寄存器的方式比直接使用整数和浮点变量的混合模式（定义好的）具有更好的性能。在某些情况下——例如量化或引入存储在寄存器中的临时变量，可以显著提高程序的性能。

循环展开

循环展开是循环的变换，循环展开会使得循环体的代码量变得更大，但是循环的迭代次数也因此而变的更少。除了编译器自动优化之外，还经常需要手动循环展开以确保正确的展开量，因为过度展开可能会对性能产生不利影响。通过更有效地使用CPU寄存器，循环展开过程可以最大化地减少加载/存储指令的数量、以及编译器效率低下的指令调度所引起的数据危害。循环展开有两种类型：内部和外部。内部展开包括将最内层循环的某些迭代折叠成更大，更复杂的语句。这些语句需要更多的机器指令，但是可以由编译器的优化器更有效地调度。外部循环展开包括使用更多的寄存器把迭代从外部循环移动到内部循环中，以最大程度地减少访问存储器的次数。在视频编码应用中，运动估计和运动补偿预测就是循环展开的良好选择。

算术运算优化

除法和乘法通常被认为是最耗时的运算。但是，在大多数RISC处理器中，就指令执行延迟和指令吞吐量而言，基于32位的乘法比基于64位的乘法需要更多的时钟周期。此外，与混合整数和浮点除法相比，浮点除法的循环开销会更小。因此，减少使用这些算术运算——尤其是在循环内部对于提升程序性能而言至关重要。

超频

尽管不建议这样做，但通过修改系统参数可以使处理器的运行速度超过其额定的时钟频率，一般称该操作为超频。尽管可以提高速度，但出于稳定性目的，超频可能还必须在更高的电压下运行。因此，大多数超频技术会导致功耗增加，并会产生更多的热量。如果处理器要保持功能正常，则必须予以散热。这又增加了风扇噪声和冷却的复杂度。相反，部分制造商会选择降低电池供电设备的处理器的频率，以延长电池寿命。超频技术也可以应用于芯片组，独立显卡或内存。

超频会让系统超出当前系统组件的能力。由于增加了散热要求，系统的可靠性会降低，并且组件存在潜在损坏的风险。超频的受众主要是发烧友和业余爱好者，而不是专业用户。

成功的超频需要对电源管理有充分的了解。正如我们将在第6章中看到的那样，现代处理器中的电源管理非常复杂。电源管理需要处理器硬件和操作系统协才能实现。在此过程中，会根据当前工作负载动态调整处理器核的频率。在这种情况下，将某个内核推至100%的频率可能会对功耗产生不利影响。图5-10的例子说明了这一概念，其中工作负载运行在四核（八个逻辑核）的英特尔第二代酷睿处理器。

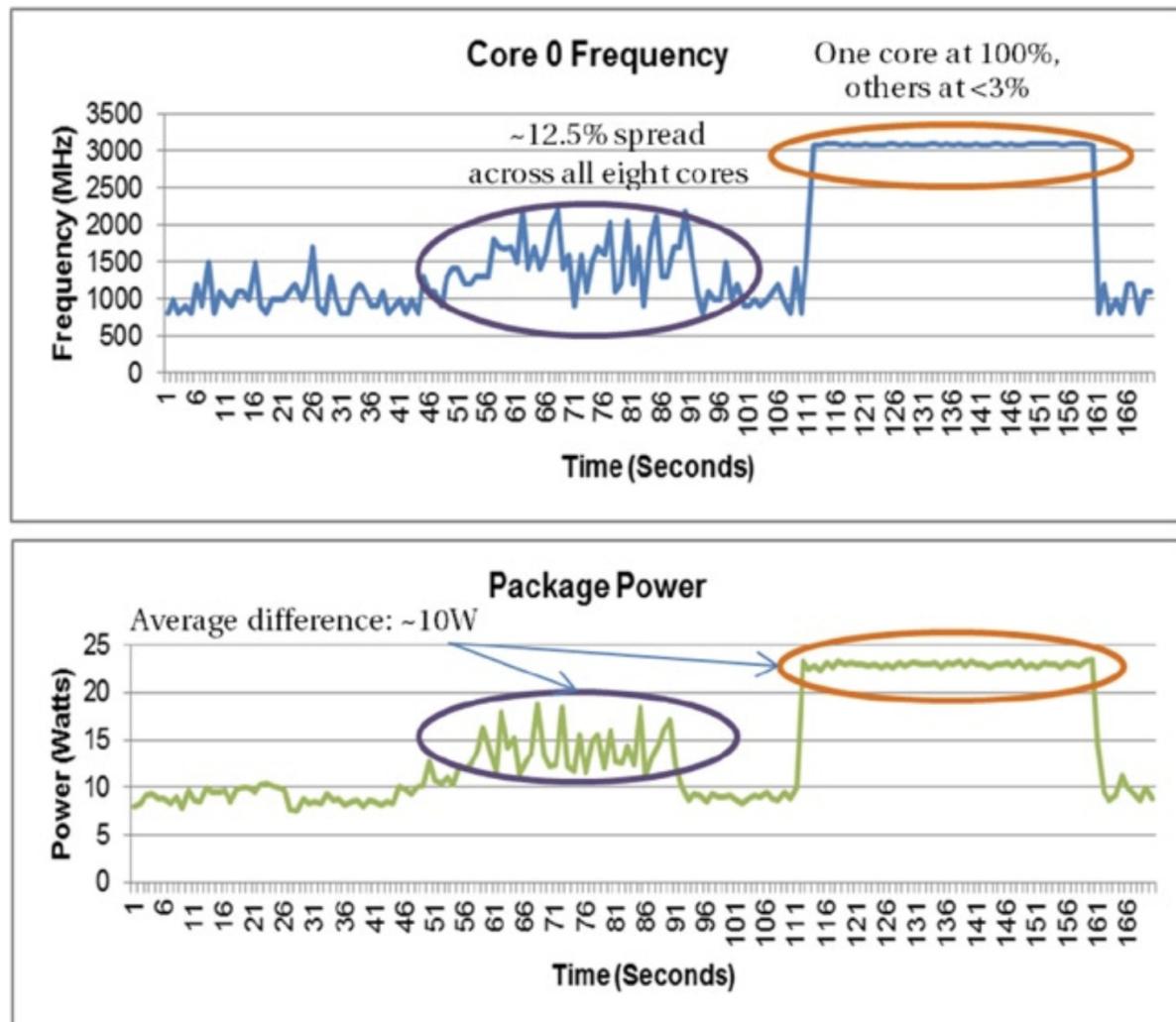


图5-10. 将某个内核推至100%的频率对功耗造成的影响

在多核处理器中，如果将一个CPU内核的频率提高到100%，而将其他CPU的内核空闲，则会导致更高的功耗。在图5-10的例子中，与使用全部八个CPU核相比，单个CPU核以100%的频率运行会消耗多达~10瓦的功率，并且所有CPU核的平均频率分布为~12.5%。

最新的具有集成显卡的Intel处理器允许采用硬件加速的视频编码器可以在必要时自动达到处理器的最大频率，然后在完成任务后将其保持在空闲状态。第6章将讨论该机制的详细信息。在现代处理器的功耗受限的环境中，最好将频率调整留给硬件和操作系统。

性能瓶颈

当系统性能受到系统的一个或多个组件（阶段）的限制时，就会出现性能瓶颈。通常，单个阶段的性能下降会导致整个系统变慢。性能瓶颈可能是由于硬件限制或低效率的软件配置，或两者兼而有之。尽管系统可能会具有短时峰值性能，但是对于可持续的吞吐量，系统只能实现与其性能最差的组件一样快的性能。理想情况下，系统应该没有性能瓶颈，以便达到可用资源的最佳利用。

需要仔细检查资源利用率以确定性能瓶颈。当一个或多个资源未被充分利用时，通常表明系统中存在瓶颈。瓶颈识别是一个增量过程，修复一个瓶颈可能导致另一个瓶颈的出现。应当以顺序的方式识别瓶颈，在此期间，一次只能识别和更改一个参数，并且可以获得单个更改的影响。一次更改多个参数可能会掩盖更改的影响。消除瓶颈后，必须再次测量性能以确保没有引入新的瓶颈。

通过仔细检查和分析各种执行期间的文件（*execution profiles*），可以发现并解决与性能相关的问题，其中包括：

- 执行历史记录，例如性能调用图
- 各个级别的执行统计信息，包括包、类、和方法
- 执行流程图，例如方法调用统计信息

可能有必要使用性能指标对代码进行性能分析，以进行如上类类型的性能分析。但是，大多数现代操作系统都提供用于运行时和静态性能分析的性能分析工具。

可以使用英特尔性能瓶颈分析器框架¹来识别、分析、缓解应用程序的性能瓶颈。该框架会自动查找Intel Core和Atom处理器的架构瓶颈并确定其优先级。框架结合了最新的性能监视技术和静态汇编代码知识，以识别性能瓶颈。框架还对一些困难和模棱两可的情况进行优先排序并标记以进行进一步分析。该工具通过二进制文件重新创建指令执行的最关键路径，并分析这些路径，然后基于历史的性能监视事件来搜索众所周知的代码问题。

¹. E. Niemeyer, “Intel Performance Bottleneck Analyzer,” Intel Corporation, August 2011. Retrieved from www.software.intel.com/en-us/articles/intel-performance-bottleneck-analyzer. ↵

性能度量和调整

需要测量性能以验证当前的性能是否满足设计需求。此外，这种测量允许确定任务的实际执行速度，识别和缓解性能瓶颈以及性能调整和优化。性能测量还允许比较两个任务：例如，就性能而言比较两个视频编码解决方案。因此，性能测量在确定各种视频应用程序的性能、质量、功耗、压缩量等指标之间的折衷上具有重要作用。

有很多方法可以用来调整应用程序的系统性能。例如，编译时方法（*compile-time*）包括：将编译器指令插入代码以引导代码优化，使用程序分析器在多次编译过程中修改目标代码，等等。运行时方法（*run-time*）包括收集程序跟踪信息和事件监视信息。

性能思考

由于可配置的系统参数会影响整体性能，因此有必要将这些参数固定为某些值，以获得稳定、可靠、可重复的性能测量结果。例如，在进行性能测量之前，必须设置BIOS、操作系统的性能优化选项、英特尔图形通用用户界面（CUI）等¹。在BIOS设置时，应考虑以下因素：PCIe延迟，时钟门控，ACPI设置，CPU配置，CPU和图形电源管理控制，C-状态延迟，中断响应时间限制，图形渲染待机状态，超频状态，等等。

正如我们在前面的讨论中所述：工作负载特征可能会影响性能。因此，另一个重要的考虑因素是工作负载参数。但是，通常很难收集并分析所有可能的编译时和运行时的性能指标。此外，工作负载和用于性能测量的相关参数的选择，通常由特定用法以及应用程序如何使用这些工作负载来确定。因此，重要的是要考虑实际的使用模型，以便选择合适的测试用例作为关键性能指标。例如，这种选择在比较两个视频编码解决方案时（这两种编码方案具有性能差异，但在其它方面却各具特点）非常有用。

¹. This graphics user interface works on a system with genuine Intel CPUs along with Intel integrated graphics. There are several options available—for example, display scaling, rotation, brightness, contrast, hue and saturation adjustments, color correction, color enhancement, and so on. Some of these options entail extra processing, incurring performance and power costs. ↵

性能指标

很多运行时性能指标在不同的应用程序中很有用。例如，对处理器和内存使用模式的了解可以用来指导代码优化。对程序进行关键路径分析可以发现性能瓶颈。消除瓶颈或缩短关键路径可以显著提高整体系统的性能。在文献中，经常以每条指令的周期（CPI）、每秒数百万条指令（MIPS）、每秒数百万条浮点运算（Mflops）的形式来报告系统的性能。此外，还会根据内存周期或完成一个内存引用所需的时间来报告内存性能，而该指标通常是处理器周期的倍数。

但是，实际上，诸如视频编码之类的应用程序的性能调整通常还需要测量其它的指标：例如CPU和GPU利用率，以FPS为单位的处理或编码速度，以MB/s为单位的内存带宽。在硬件加速的视频应用程序中，以CPM (*clocks per macroblock*, 每宏块时钟) 表示持续硬件性能，并可以用CPM表示图形驱动程序和视频应用程序带来的性能差异。因此，可以利用CPM在合适的级别上进行适当的调整以获得最佳性能。用于调试目的的其它指标包括：高速缓存命中率，页面错误率，负载索引，同步频率，内存访问模式，内存读写频率，操作系统和编译器开销，进程间通信开销等。

性能分析工具

大量的可用于性能测量的工具可以说明性能测量的重要性。有的性能分析工具支持采样和基于编译器的应用程序性能分析，有时还具有上下文相关的调用图功能。有的工具则支持基于硬件事件的非侵入式且开销较低的采样和分析。还有的工具会利用现代微处理器提供的硬件性能计数器。部分工具可以发现与数据局部性、缓存利用率、线程交互等特性有关的性能问题。在本节中，我们简要讨论了适用于视频应用程序（尤其是GPU加速的应用程序）的性能测量的流行工具。第6章简要介绍其它的流行工具，例如Windows Perfmon, Windows Xperf和Intel Graphics Performance Analyzer。

VTune Amplifier

VTune Amplifier XE 2013是Intel开发的流行的性能分析器¹。VTune支持各种编程语言的性能分析，包括C, C++, FORTRAN, Assembly, Java, OpenCL和OpenMP 4.0。VTune为热点（hotspot），调用树，线程，锁，等待，DirectX，内存带宽等信息采集了丰富的性能数据，并为满足各种性能调整需求提供所需要的数据。

热点分析提供了一份使用CPU时间较高的函数的有序列表，用以指示可带来最大收益的性能调整的位置。热点分析还可以利用锁和等待分析来对多线程进行调整。这使用户能够快速查找信息来确定并行程序的性能降低的原因，这些可查找的信息包括：线程在锁等待上的时间，而线程在等待期间未充分利用CPU。诸如热点、锁定、等待之类的分析文件（profiles）都使用了可在Intel和兼容处理器上运行的软件数据收集器。该工具还提供高级热点分析。英特尔处理器的片上（on-chip）性能监视单元（PMU, Performance Monitoring Unit）通过采样硬件事件实现以极低的开销提高数据收集能力。因此，高级热点分析可以用PMU来识别很小的性能瓶颈并发现快速函数的性能瓶颈。此外，该工具还支持高级硬件事件分析文件，例如内存带宽分析、内存访问和分支错误预测，从而可以帮助发现调优机会。最新的版本还支持可选的堆栈样本集合，以标识程序的调用顺序。此外，该工具还支持在不重启应用的情况下分析远程系统。

GPUView

GPUView由Matthew Fisher和Steve Pronovost开发，GPUView是一种用于确定GPU和CPU性能的工具。后来，将GPUView合并到Windows Performance Toolkit中，并作为Windows SDK的一部分下载²。GPUView用于DMA（直接内存访问）缓冲区处理以及视频硬件上所有其它视频处理过程的性能分析。对于GPU加速的DirectX应用程序，

GPUView是一种功能强大的工具。可以用GPUView了解CPU上完成的工作与GPU上完成的工作之间的关系。GPUView使用ETW（Windows事件跟踪，*Event Tracing for Windows*）机制对系统和应用程序的性能以及资源使用情况进行详细测量并分析。数据收集过程包括：启用跟踪捕获，运行需要性能分析的测试应用程序的场景，停止数据捕获。数据收集最终会将数据保存在ETL（事件跟踪日志，*event trace log*）文件中。GPUView可以在同一台或不同计算机上分析ETL文件并以图5-11所示的图形格式显示ETL信息。

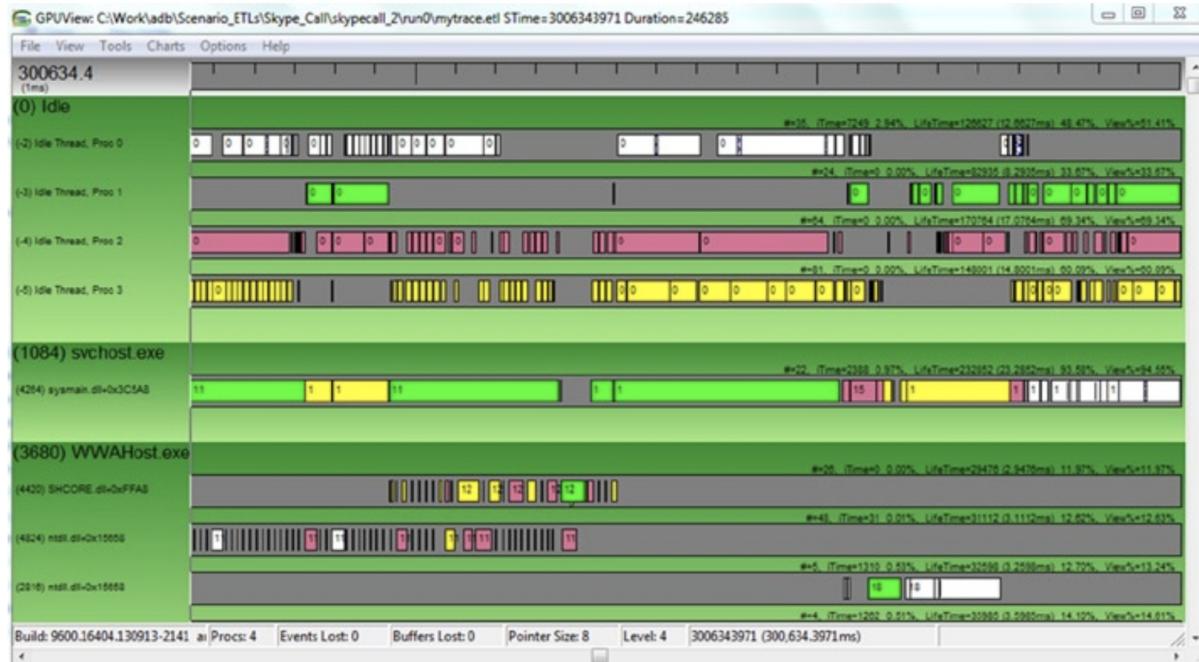


图5-11. 使用GPUView分析不同线程的活动

GPUView在硬件加速视频应用程序的分析和调试中非常有用。例如，如果视频播放应用程序存在掉帧现象，则会对用户体验造成负面影响。在这种情况下，使用GPUView检查事件跟踪可以帮助我们确定问题的原因所在。图5-12给出了正常播放视频的事件跟踪的例子，其中工作负载按固定间隔均匀分布。图5-12中的蓝色垂直线显示正常的vsync，红色垂直线显示当前事件。

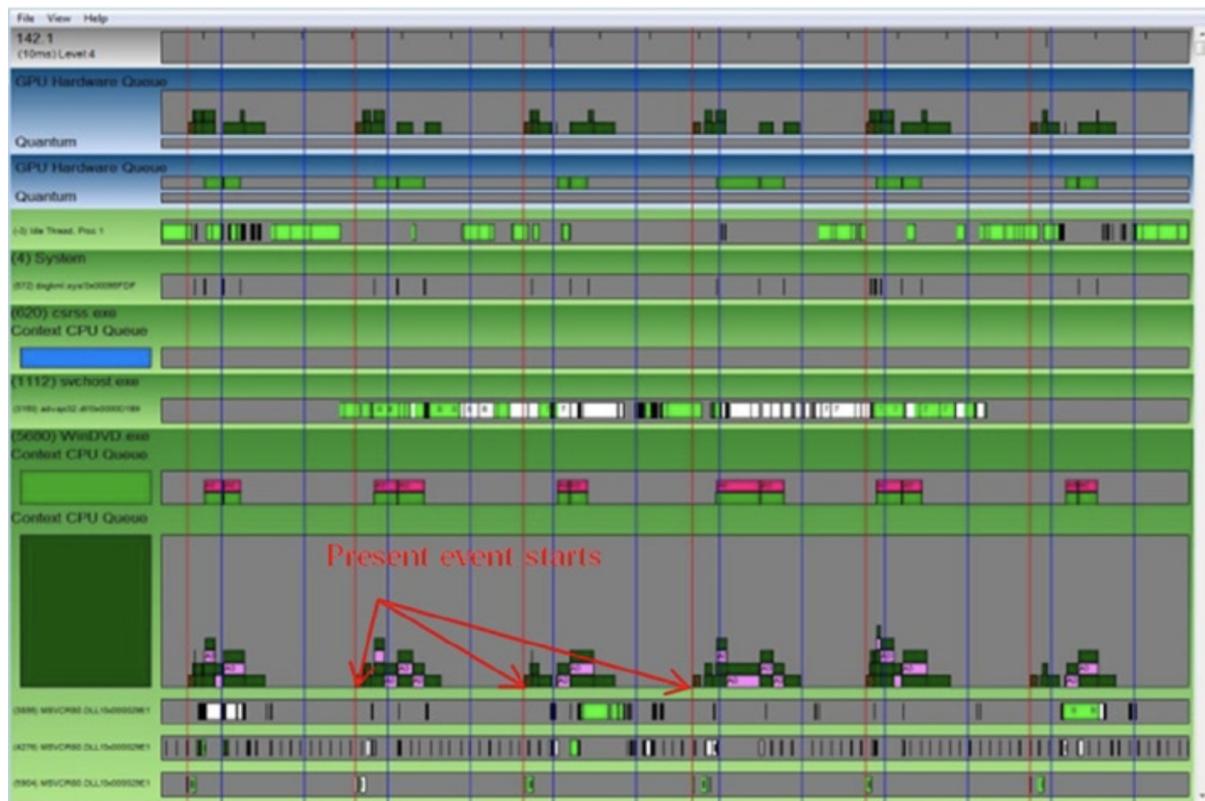
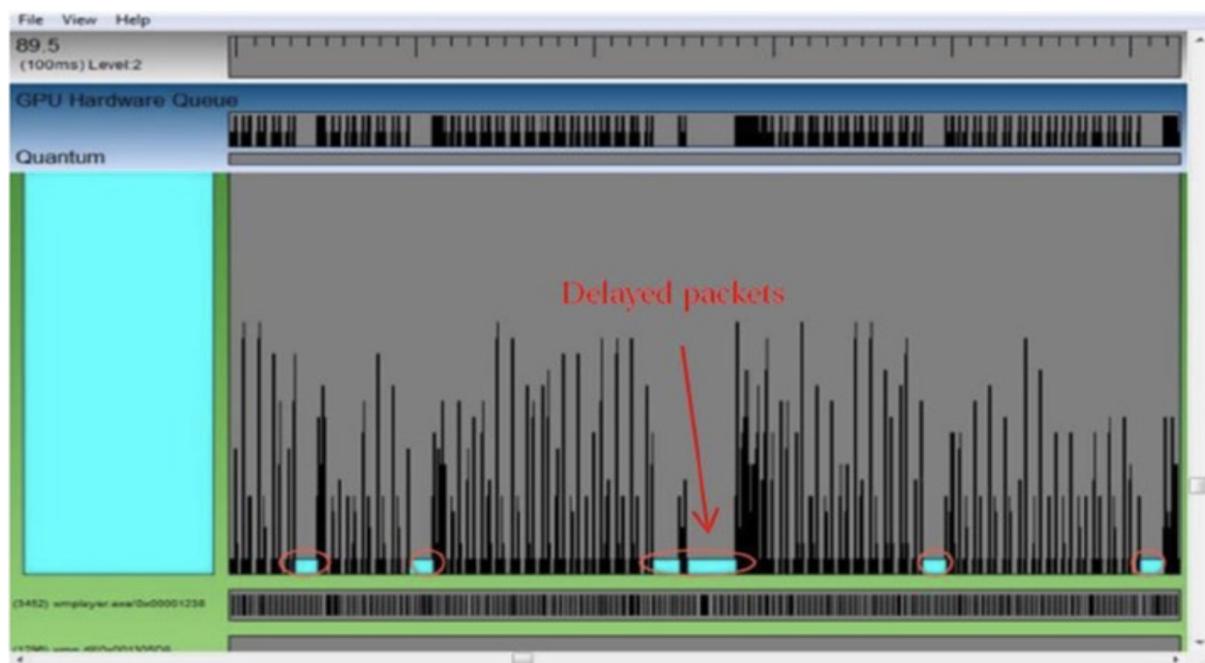
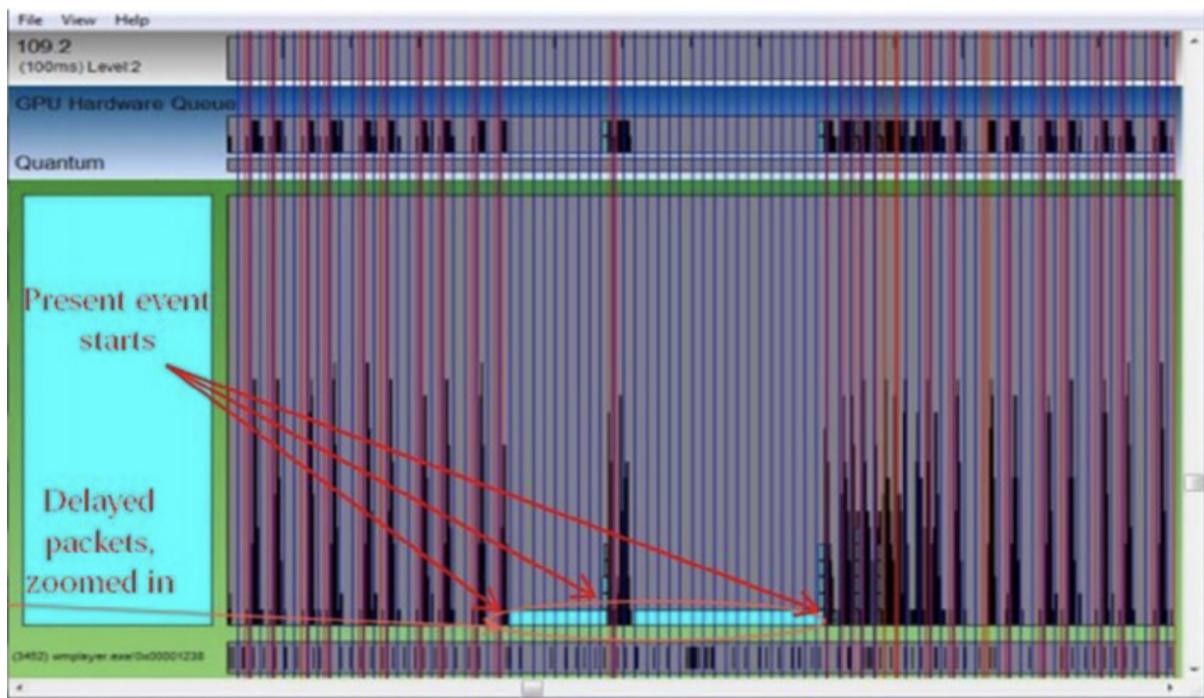


图5-12. 正常播放视频的事件跟踪

图5-13展示了同一视频播放应用程序的事件轨迹，但是当帧显示截止时间到期时，播放程序会丢弃视频过期的视频帧。与图5-12中所示的正常模式相比，此轮廓看上去大不相同。在放大版本中，当前事件行是可见的，从中可以不难理解，当应用程序将视频数据包发送到GPU进行解码时，时常会出现长时间的延迟。因此，使用GPUView可以轻松识别并解决问题的根本原因。



缩小的播放分析文件



放大的播放分析文件

图5-13. 掉帧情况下的事件跟踪信息

- ¹. The latest version of VTune Amplifier for Systems (2014) is now part of Intel System Studio tool suite at: <https://software.intel.com/en-us/intel-system-studio>. ↵
- ². Available from <http://msdn.microsoft.com/en-us/windows/desktop/aa904949.aspx>. ↵

总结

在本章中，我们讨论了CPU时钟速度以及时钟速度可能增加的程度。我们注意到，现代处理器设计的重心已经从单纯提高时钟速度转向了功率和性能的有效的结合。然后，我们重点介绍了实现高性能视频编码应用程序的动机，以及实现此类性能所需的权衡。

接下来，我们深入讨论了资源利用率以及影响编码速度的因素。然后讨论了各种性能优化方法，包括算法优化，编译器和代码优化以及几种并行化技术。可以组合使用所讨论的并行化技术以获得更高的性能，尤其是在视频编码应用中。我们还讨论了视频编码应用中的超频和常见性能瓶颈。最后，我们介绍了各种性能评估注意事项，性能评估的工具和应用程序，性能评估的方法和指标。

视频应用的功耗

在前面的章节中，我们讨论了视频压缩，视频质量与视频性能，本章中，将聚焦视频应用调优的另一维度：功耗。功耗往往需要和其他因素一起考虑，如根据应用程序的供电限制来维持设备的续航能力(低耗能场景)，或是提供最佳用户体验（高耗能场景），需要在不同场景、因素中权衡利弊，做出调整以较大限度的支持其中某一个维度的目标。因此，本章中先介绍了功耗的基本概念，从日常使用的现代视频设备，了解常规的设备功耗限制，然后再讨论消费者使用的设备中通用媒体应用的功耗。在此之后，简要介绍了功耗感知平台设计的各种标准。

按照讨论功耗的常规步骤，本章主要涉及三方面主题：1 功耗管理，功耗调优和功耗测量注意事项，在功耗管理方面，介绍了操作系统和处理器所使用的标准和管理方法。2 关于功耗调优，提出了架构，算法和系统集成优化的方法。3 功率测量，将从测量方法上，引发对功耗的整体思考。

在本章的这三个议题外，还将简要介绍几种功耗测量工具及其应用，列举其优点和局限性。

功耗及其限制

按照当今移动世界的发展秩序，我们需要面对日益增长的用户“刚性”需求：可穿戴界面，无线连接，全天候计算，高性能，但是同时，移动设备在“刚性”需求下，却需要拥有更小巧的外形，尺寸，更轻的重量，更小的充电设备。这种看似矛盾的需求，却提出了“史无前例”的挑战：1 新型移动设备需要更长的电池使用时间 2 移动设备上设计笨重的风扇是无法容忍的，小型风扇散热能力又有限，导致的散热难题；

视频应用需要在有限的功率范围内工作。从消费者的角度来看，更关注的是设备是否拥有更长的电池使用时间。电池系统在设备的整个生命周期内都存在成本和效益的问题，更大的电池容量意味着更高昂的价格，因此，在有限的电池容量下，视频应用的功率限制是当今移动设备设计中的一大基本考虑因素。

功率极限通常以热设计功率（TDP）表示，TDP在设计中考虑了冷却条件下设备的最大发热量，因此，可以认为TDP是设备允许的最大功耗。TDP通常分为单个组件的功耗，例如CPU，GPU等。表6-1列出了各种处理器型号的典型TDP。

Type	TDP(Watts)	Comment
桌面服务器/工作站	47W–120W	高性能工作站和服务器
一站式	47W–65W	常用
笔记本	35W	较差
超极本	17W	-
平板电脑	8W	TDP降低的重点突破领域
智能手机	<4W	-

对于台式机工作站和服务器而言，降低电源消耗，保存其电量同样重要，但是通常情况下，它们基本上不受电源可用性限制。另外一方面，消费类设备（如便携式平板电脑，平板手机和智能手机）使用尺寸大小有限的电池供电，意味着充电结束后，设备可使用的电量是有限的。

平板电脑，平板手机和智能手机，这样的设备主要缺点是耗电快，平板电脑的电池通常会在跨大西洋飞行结束前的几个小时就耗尽了，智能手机则几乎每天都需要充电。因此，现在的市场要求平板电脑具有超过10个小时的电池续航时间，让用户能够享受长途飞行的乐趣，而智能手机的续航时间要在24小时以上。此外，平板和智能手机都会有一些高耗能的应用，我们需要了解这些消费者平台上的应用程序的能量使用情况。功率是单位时间消耗

的能量，通常以每秒多少焦耳或者瓦特标识。消耗的开关功率使用静态CMOS门的芯片（例如，以频率 f 和电压 V 运行的，具有电容 C 的移动计算设备的CPU消耗的功率）大致如下：

$$(6 - 1) P = A C_{dyn} V^2 f + P_s$$

上式中， P_s 是主要由于CMOS泄露功率等引入的静态功率成分， A 是处理器是否处于活动/睡眠状态，或者在时钟门控条件下有关的活动常数。对于特定的处理器， C_{dyn} 是固定值；其中， V 和 f 可能相差较大。该公式并不完美，因为实际设备中的CPU不一定100%由CMOS组成，可能还涉及特殊电路。同样，静态泄露电流也并不总是相同，从而导致方程的后半部分出现变化，这种变化对于低功率器件而言变得尤为重要。尽管方程不精确，但对于显示如何改变系统设计会影响功率。以更高的时钟频率运行设备的处理器可获得更好的性能。如公式6-1所示，在较低的频率下，其散热量较小，因此功耗较低。换句话说，功耗不仅决定性能，还影响电池寿命。

在当今的技术中，各种电子设备和平台的电源或能源供应通常来自以下三种主要来源之一：

- 电源插座，通常称为“交流电源”
- SMPS单元，通常称为“直流电源”
- 充电电池

开关电源(SMPS)单元对交流市电输入进行整流和滤波，以获得直流电压，然后以数百KHz到1 MHz的频率(高频率)打开和关闭直流电压。

高频开关可使用廉价，轻便的变压器，电感器和电容器电路，以进行后续的降压，整流和滤波，以输出干净稳定的直流电源。通常，SMPS用作计算机电源。

对于移动应用，可充电电池为越来越多的电子设备（包括几乎所有的多媒体设备和平台）提供能量。然而，由于在充电和放电期间内阻的变化，可再充电电池随着时间而退化。可充电电池的寿命（也称为“电池寿命”）取决于充电/放电的循环次数，直到最终电池无法再保持有效充电为止。

电池的额定单位为瓦特每小时（或电流乘以电压，单位为A/h乘以V/h）。以瓦特为单位的系统功耗测量可以很好地了解电池在需要充电之前可以工作几个小时。这种电池寿命的测量通常对于当今电子设备的消费者来说很重要。

用户设备上的媒体负载

易用性是设计消费类电子设备时重要考虑目标之一。如今，多媒体功能在现代移动设备中的高度集成已变得非常普遍。

例如，理想的智能手机不仅需要作为无线电话和通信设备，而且还应该能包含纳日历，时钟和计算器等应用程序。同时，还应该带有罗盘，GPS和地图的导航设备；而且它应该可以作为娱乐设备使用，带有游戏，多媒体应用程序。除此之外，还应包含教育类应用，如给讲故事应用，虚拟课堂应用等。

在输入方面，用户在与这些设备的交互时，期望设备拥有高分辨率的摄像头，与互联网的高速无线连接以及语音，触摸和手势输入。在输出方面，又期望高保真扬声器，高分辨率显示器，快速处理和低功耗。但是，设备在支持多种高端功能时，通常与节省功率的需求相冲突。电池容量的增加只能缓解该问题，多媒体集成的扩展和增强的用户体验需要更大的电池容量。分析和理解这些多媒体应用负载的性质，将有助于在上述限制内实现性能和功率优化。

按流行的消费类电子设备评测方法：通常使用非多媒体基准工作负载（例如Kraken，Sunspider和Octane Javascript基准）来测量和分析功率数据。这些基准测试程序是致力于测量设备作通用计算时的功耗，而未检查设备作为其他用途时的功耗。这些用于"其它用途"的应用程序，通常并未针电源进行优化，或者可能仅针对某些平台和操作系统进行了优化，目标却是为了实现更高的性能（意味着更高的消耗）。基准测试对于系统任务迁移和处理单元之间的资源共享的影响并未作出分析。随着集成处理器图形平台可用性的提高，在此类分析中必须包括媒体使用情况和应用程序。

在后续章节中，我们将讨论一些常见的媒体设备的用途及其应用场景。

媒体应用

多媒体应用通常属于高耗电类型应用。随着对越来越多的功能的需求，对功耗的要求日益提高。某些场景下，应用程序可能需要同时开启多个进程。用作娱乐平台的移动设备，通常运行游戏和视频这两种主要类型的应用程序。2D和3D视频游戏是最受欢迎的，但这些设备也需要运行许多其他应用程序支持。如：

- 静态图像拍摄
- 静态图像预览/取景器
- 无线投屏或Miracast：克隆模式或扩展模式
- 基于浏览器的视频流
- 录像和双机码流录制
- 视频回放
- 音频播放
- 浏览互联网
- 可视电话和视频聊天
- 视频会议
- 视频转码
- 视频电子邮件和彩信
- 视频上传到互联网
- 视频编辑
- 增强现实
- 生产类应用

以上应用大多数是通过特殊的软件程序实现的，如果硬件平台支持，则应用可能会受益于硬件加速，以此获得更好的用户体验。

硬件平台支持如英特尔处理器，通过集成图像处理器，在应用程序进行图像处理时可进行硬件加速。这种集成处理器既可以用作通用计算平台，又可以满足特殊应用的需求。将图形单元集成到中央处理器后，移动设备可以省去体积更大的视频硬件卡，和定制的视频处理器，从而保持适合移动应用的小尺寸。

在许多移动设备上，多媒体应用程序的某些组合被用户同时使用。例如，当用户正在浏览 Internet时，音频播放可能会继续，或者视频播放可能会同时进行视频录制。其中一些应用程序使用通用的硬件模块来实现视频编解码器和处理任务的硬件加速。从系统资源调度和利用的观点来看，这些硬件模块的并行操作是令人感兴趣的。

这种复杂系统行为，如多方视频会议，以及通过Miracast Wireless Display传输视频。经过Wi-Fi认证的Miracast是一种行业标准的解决方案，无需电缆或网络连接即可在设备之间无缝显示多媒体。它使用户可以通过大屏幕电视上的智能手机查看图片，或通过平板电脑上的家用有线电视盒观看直播节目。连接功能是使用Wi-Fi Direct的设备内部功能，因此不需要单独的Wi-Fi访问¹。

图6-1显示了Miracast应用程序的使用模型。

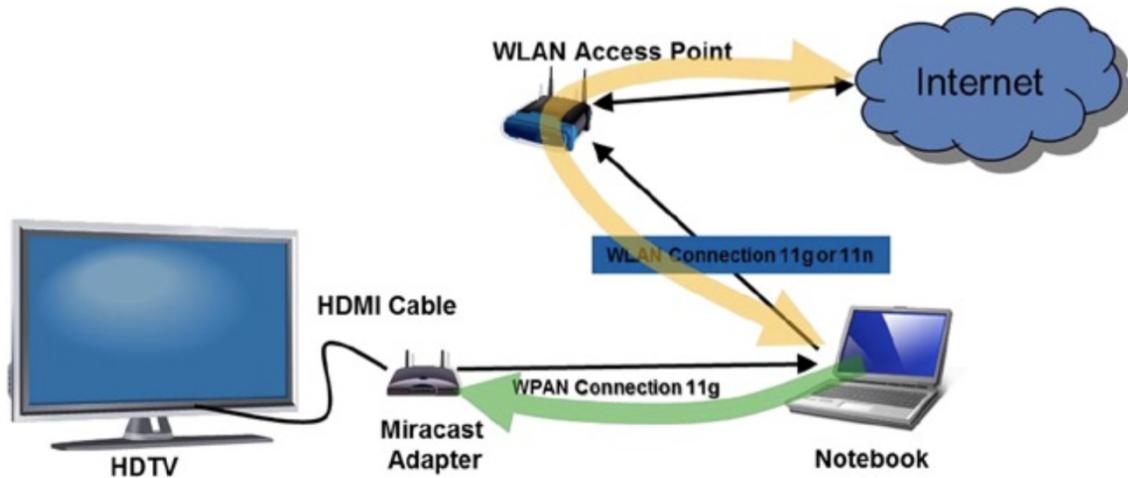


图6-1.通过Miracast传送视频

为了更好地了解视频应用程序的功耗，让我们详细分析一种多媒体的使用：Miracast Wireless Display²上的视频传输。

Wireless Display (WiDi) 是最初为实现以下目的而开发的英特尔技术：与Miracast相同的目标。现在Miracast已成为行业标准，自3.5版以来，它已在英特尔(R)无线显示(TM)中得到支持。此应用程序的目标是提供一种具有高级内容功能的无线显示解决方案，该解决方案允许PC用户或手持设备用户通过与远程显示器的无线链接来远程显示视听内容。换句话说，笔记本电脑或智能手机会通过无线局域网从Internet接收视频，或者使用本地摄像头捕获视频。使用本地播放应用程序在设备上播放视频。然后，用于Miracast无线显示的特殊固件将捕获设备的屏幕并执行硬件加速的视频编码，以便通过Wi-Fi数据交换技术将压缩的比特流发送到Miracast适配器。适配器将位流解码为HDMI格式，然后通过HDMI电缆连接将其发送到显示设备。

端到端框图如图6-2所示。

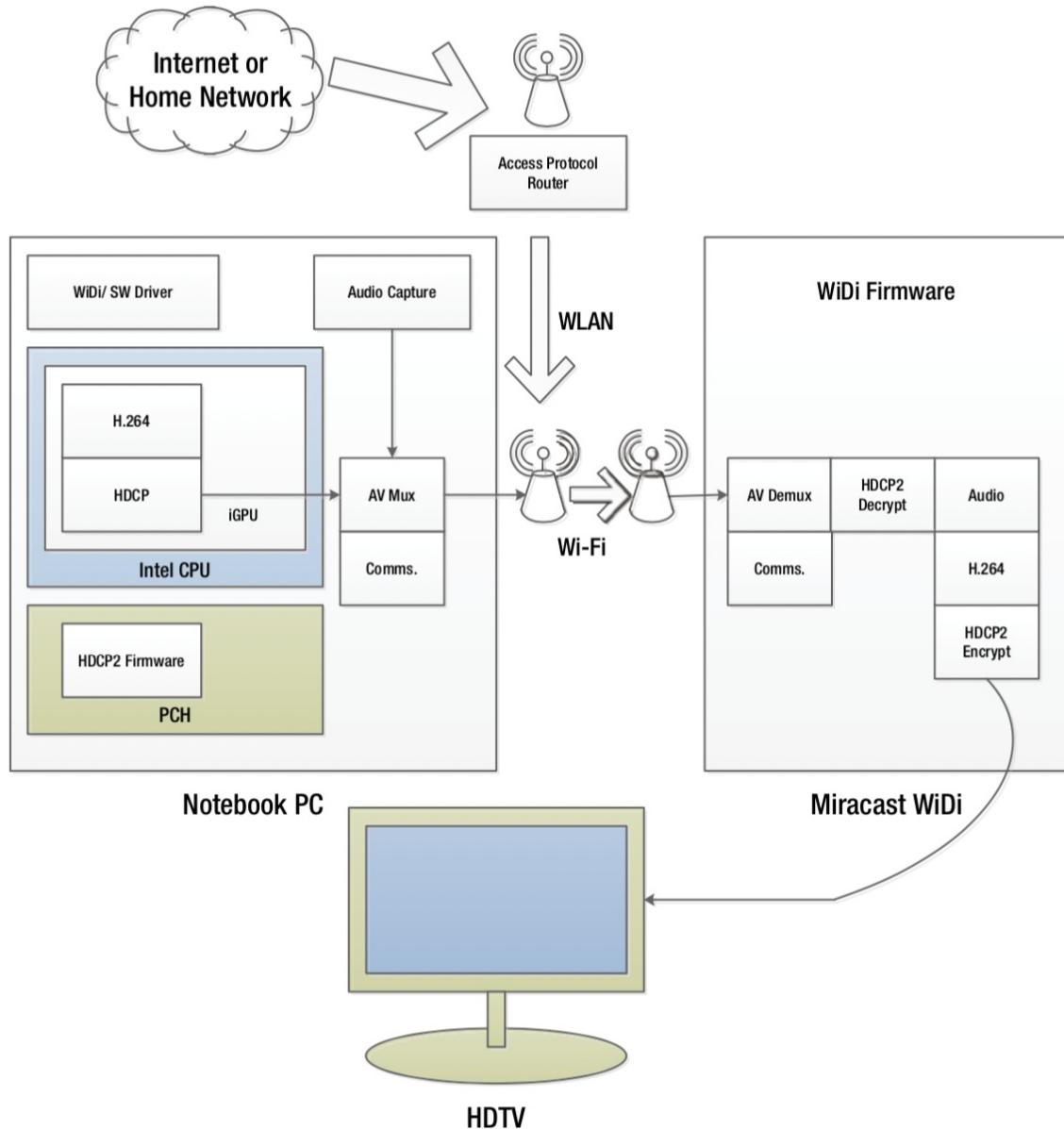


图6-2.Miracast无线显示端到端框图

在图6-2中，主要的功耗硬件模块是CPU，PCH，集成GPU中的视频编解码器，硬件加速的内容保护模块，内存，笔记本的本地显示器以及远程HDTV显示。Miracast无线显示适配器主要运行无线显示固件，并且消耗较少的电量。本示例中的典型功耗分布如图6-3所示。

%Power consumption

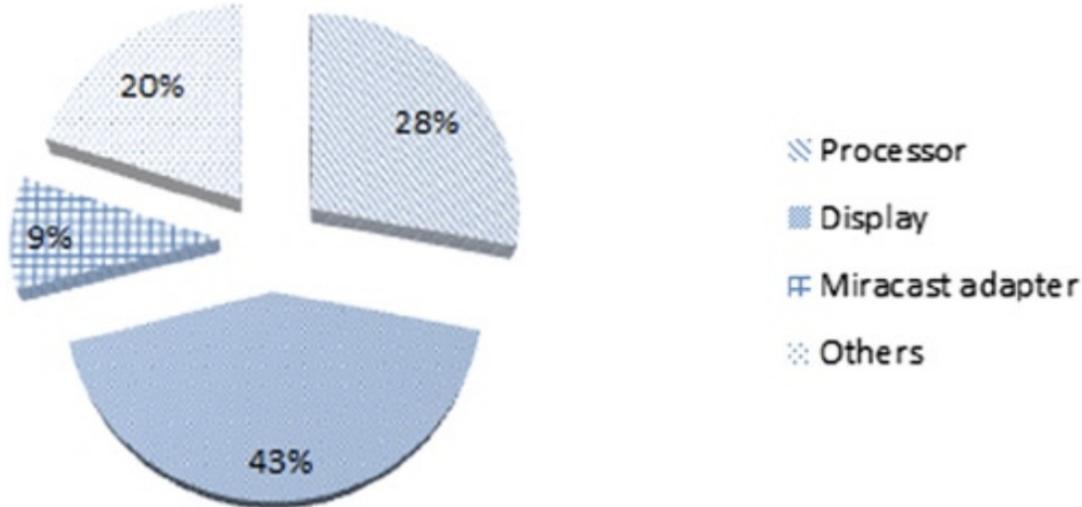


图6-3.Miracast无线显示应用程序中按组件的典型功耗分布

如图6-3所示，显示器通常消耗大部分功率，在此示例中，显示器消耗的功率约为处理器的1.5倍。Miracast适配器本身消耗的电量适中-在此示例中，大约占应用程序总功耗的9%。由于此应用程序的复杂性，应在性能需求和功耗之间保持谨慎的平衡，以便可以进行适当的优化和折衷，以获得令人满意的用户体验。

另一个常见的多媒体应用是视频播放以及关联的音频。Agrawal等人³提供了对该应用程序的详细分析³。在这里，我们仅注意到通过执行媒体播放的硬件加速，总体功耗从~20W降低至~5W，同时功耗曲线也发生了变化显着。媒体回放管道中的各种任务都可以通过硬件加速来提高性能，因为这些任务从CPU转移到了具有更好功率性能特性的专用固定功能硬件上。

通过对这些应用程序的分析，可以确定哪些模块是电源优化的主要候选对象。例如，可以通过将诸如色彩空间转换之类的任务从显示单元迁移到GPU来实现一些功耗优化。某些英特尔平台具有这些功能，可以实现高水平的功耗优化。本章稍后将详细讨论各种功率优化技术。

¹. For details, see (www.wi-fi.org/discover-wi-fi/wi-fi-certified-miracast). ↪

². For details, see (www.intel.com/content/www/us/en/architecture-and-technology/intel-wireless-display.html). ↪

³. A. Agrawal, T. Huff, S. Potluri, W. Cheung, A. Thakur, J. Holland, and V.

Degalalah, [Power Efficient Multimedia Playback on Mobile Platform,] Intel Technology Journal 15, no. 2 (2011): 82–100. ↪

电源感知设计

功耗是硬件和软件效率的函数。因此，提高性能或节省功率越来越取决于提高效率。这通常是根据每瓦设备性能来完成的，最终转化为每美元性能。每瓦性能是计算系统可以为每瓦消耗的功率提供的计算量。当今的平台旨在通过在设计过程中加入“功耗意识”，从而在这一指标上取得高分，因为人们在很大程度上考虑了计算环境中的能源成本。

在具有功耗意识的设计中，通常通过采用以下方式来实现功耗节省：分而治之的策略：将系统划分为几个独立的电源域，并且仅向活动域供电。根据每个域的活动状态，智能的电源管理可以实现最佳的电源解决方案。另外，为了在系统的节能和价值主张方面获得优势，在每个领域内都进行了优化。

功耗意识不仅在硬件设计中很重要，而且在应用程序中也很重要，这样才能最大化每美元的性能。为此，大多数操作系统都提供电源管理功能。由于应用程序了解各种硬件资源和任务的使用模式，因此，如果应用程序可以为电源管理单元提供适当的“提示”，则可以实现更好的电源管理。此外，这些应用程序的功耗意识可以产生软件级别的优化，例如上下文感知的功耗优化，降低复杂性和减少内存传输。这些技术将在本章的后面部分中针对功率优化进行讨论。

电源管理注意事项

在计算机和计算机外围设备（例如显示器和打印机）中进行电源管理的目的是在系统不活动时关闭电源或将系统切换到低功耗状态。计算平台中的电源管理具有许多优势，包括延长电池寿命，降低热量排放，降低碳足迹以及延长诸如显示面板和硬盘驱动器之类的设备的寿命⁴。

电源管理发生在计算机系统（也称为“系统”）中可用的各种组成硬件设备上；其中包括 BIOS，中央处理器（CPU），硬盘驱动器（HDD），图形控制器，通用串行总线（USB），网络和显示器。也可以监视和管理电源使用情况内存的各个部分，例如动态随机存取存储器（DRAM）和非易失性闪存，但这更加复杂且不那么普遍。表6-2中列出了一些电源管理示例；其中一些将在本章的后续部分中讨论。

表6-2，电源管理功能

设备/组件	电源管理功能
BIOS	CPU设置(例如，启用的CPU状态，CPU风扇的节流),平台设置（例如，高/低水印热，机箱风扇的节流等）
CPU	HLT（在x86中暂停指令，直到发生下一个外部中断时，CPU才停止），停止时钟，Intel SpeedStep(又称动态频率缩放)
显示器	消隐，调光，省电模式，能源之星国际标准规定的高效能源使用
Graphics Controller	断电至中间状态，电源关闭
硬驱动/CD-ROM	光驱
网络/网卡	局域网唤醒
USB	鼠标，USB驱动器等

可能有特殊的电源管理硬件或软件可用。通常，处理器中的硬件电源管理涉及各种CPU状态（也称为C状态）的管理，例如核心C状态，模块C状态，程序包C状态等。（有关C状态的详细信息，请参见下一节。）另一方面，操作系统或驱动程序中的软件电源管理涉及诸如CPU核心离线，CPU核心屏蔽，CPU负载平衡，中断负载平衡，CPU频率控制等。随着主要在Intel平台中引入集成图形处理单元（iGPU）的出现，各种GPU状态也是电源管理的重要考虑因素。

⁴. M. Vats and I. Verma, Linux Power Management: IEGD Considerations (Intel Corporation, 2010). Available at

www.intel.com/content/dam/www/public/us/en/documents/white-papers/linux-power-mgmt-paper.pdf. ↪

ACPI和电源管理

高级配置和电源接口（ACPI）规范是业界采用的开放标准，用于操作系统对I/O设备和资源的系统级配置和管理，包括电源管理。最初由Intel, Microsoft和Toshiba于1996年提出，后来惠普和Phoenix共同加入了规范工作。最新的ACPI规范版本5已于2011年发布。随着行业的广泛采用，有必要支持许多操作系统和处理器体系结构。为此，标准机构在2013年同意将未来的发展与统一可扩展固件接口（UEFI）论坛合并，该论坛是领先技术公司的联盟，包括最初的ACPI参与者和主要行业参与者，如AMD, Apple, Dell, IBM和Lenovo。最近所有的计算机和便携式计算设备都具有ACPI支持。

ACPI 电源状态

对于用户，计算机系统显示为“开”或“关”。但是，系统可以支持ACPI规范中定义的多种电源状态。ACPI符合性表明系统支持定义的电源管理状态，但是这种符合性并不能保证实现最节能的设计。此外，系统可以具有电源管理功能和调整功能，而无需遵守ACPI。根据ACPI规范，计算机系统的设备以一致的方式公开给操作系统。这样，对于系统级电源管理，ACPI定义单个设备的功耗状态（称为设备状态）以及整个计算机系统（称为全局状态或系统睡眠状态）。操作系统和设备可以查询和设置这些状态。重要的系统总线（例如外围组件互连（PCI）总线）可能会利用这些状态。

全局状态

ACPI规范为符合ACPI的计算机系统定义了四个可能的全局“Gx”状态和六个可能的睡眠“Sx”状态（表6-3）。但是，某些系统或设备可能无法处于所有状态。

Table 6-3. ACPI 全局状态

状态	Gx	Sx	Description
活跃	G0	S0	该系统完全可用。CPU处于活动状态。设备可能处于活动状态，也可能不处于活动状态，并且可能进入低功耗状态。S0有一个子集，称为“离开模式”，其中监视器已关闭，但后台任务正在运行。
沉睡	G1	S1	系统似乎已关闭。降低了功耗。所有处理器高速缓存均被刷新，并且CPU停止执行指令。维持CPU和RAM的电源。不必要的设备可能已关闭电源。这种状态很少使用。
沉睡	G1	S2	系统似乎已关闭。CPU上电 高速缓存被刷新到RAM。与S1类似，也很少使用S2状态。

沉睡	G1	S3	通常称为待机，睡眠或Suspend-to-RAM (STR)。系统似乎已关闭。系统上下文保存在系统DRAM上。所有电源都关闭了非关键电路，但保留了RAM电源。过渡到S0所需的时间分别长于S2和S1。
休眠	G1	S4	称为休眠或磁盘挂起 (STD)。系统似乎已关闭。降低功耗到最低水平。系统上下文保存在磁盘上，以保留OS，应用程序和打开的文档的状态。主存储器的内容保存到非易失性存储器（例如硬盘驱动器）中，并且除了恢复逻辑外，系统都已断电。 软关机 G2 S5 系统似乎已关闭。系统上下文未维护。某些组件可能仍保持供电，因此计算机可能会从键盘，鼠标，LAN或USB设备的输入中唤醒。如果工作上下文存储在非易失性存储器中，则可以将其恢复。除重新启动所需的逻辑外，所有电源均已关闭。需要完全引导才能重新启动。 硬关机 G3 -- 系统完全关闭，不消耗电源。需要完全重新引导才能使系统返回到活动状态。

设备状态

所有设备硬件的电源功能都不相同。例如，LAN适配器将具有唤醒系统的功能；当系统处于待机模式时，音频硬件可能允许流式传输，依此类推。某些设备可以细分为具有独立电源控制的功能单元。此外，某些设备（例如键盘，鼠标，调制解调器和LAN适配器）具有在设备本身处于睡眠状态时将系统从睡眠状态唤醒的功能。这些设备的硬件必须汲取较小的泄漏电流，并配备检测外部唤醒事件的能力，因此有可能实现这种功能。ACPI规范定义了设备相关的D状态，如表6-4所示。

Dx	Subset of Dx	Description
D0	---	完全打开并运行
D1	---	中间电源状态。定义因设备而异
D2	Hot	设备已关闭并且对总线无响应，但是系统开启。设备仍连接到电源。D3 Hot具有辅助电源，可启用更高的电源状态。从D0到D3的过渡意味着D3 Hot
D3	---	设备无电-设备和系统均关闭。该设备可能会消耗trickle流功率，但是需要唤醒事件才能将设备和系统移回D0和/或S0状态

ACPI定义了状态D0到D3，并将D3细分为D3热和D3冷。某些设备或操作系统无法区分D3是热还是冷，并且将D3视为没有设备电源。但是，Windows 8明确跟踪D3热和D3冷。D1和D2状态是可选的，但如果使用得当，它们将在设备空闲时提供更好的清洁度。

操作系统的电源管理

现代操作系统通常提供许多电源管理功能。Linux, Windows, OS X和Android均使用较新的ACPI标准而不是旧的BIOS控制的高级电源管理（APM）支持更智能的电源管理。但是，所有主要操作系统都一直在为基本的电源管理功能提供稳定的支持，例如向用户空间发出电源事件通知（例如，电池状态指示，空闲时挂起CPU等）。

在以下各节中，我们将讨论Linux和Linux的电源管理。Windows操作系统。在Linux电源管理的上下文中，提到了三个重要组件：X窗口，窗口管理器和英特尔嵌入式图形驱动程序（IEGD）。Windows电源管理课程包括Windows电源要求，电源策略，Windows驱动程序模型和Windows驱动程序框架。还简要介绍了Windows 8下的设备电源管理，然后讨论了如何处理电源请求。

Linux 电源管理

Linux支持较旧的APM和较新的ACPI电源管理实现。APM专注于基本系统和OS电源管理，其中许多电源管理策略都在BIOS级别上进行控制；当电源管理事件在BIOS和OS之间传递时，APM驱动程序充当BIOS和Linux OS之间的接口。通知设备这些事件，以便它们可以适当响应。

ACPI在电源管理和平台配置方面提供了更大的灵活性，并允许平台独立性和OS对电源管理事件的控制。除了电源管理策略外，ACPI还支持以下策略：响应热事件（例如风扇），物理运动事件（例如按钮或盖子），CPU状态，电源（例如电池，交流电源）等此类策略。

电源管理软件管理状态转换以及设备驱动程序和应用程序。设备驱动程序负责在将设备状态置于低功耗状态之前保存设备状态，然后在系统处于活动状态时还原设备状态。通常，应用程序不参与电源管理状态转换。一些专用软件（例如Linux的IEGD）直接处理某些设备，以处理状态转换。除了IEGD外，Linux中还有一些常用的软件技术，包括X Window系统，Window管理器以及一些开源过程，例如/sys/power/state和/proc/acpi/event，它们也提供了一些Linux电源管理的一部分功能。

X Window

X Window系统受许多操作系统支持，包括Linux，Solaris和HP-UX。它为OS提供了图形功能，并支持用户级别，系统级别和/或关键的待机，挂起和恢复。在APM实现中，X服务器控制电源管理事件。在ACPI实现中，X Window系统将图形消息作为用户进程进行处理，但是系统范围内的电源管理事件（如暂停/恢复）由内核模式驱动程序处理。

X Window系统受许多操作系统支持，包括Linux，Solaris和HP-UX。它为OS提供了图形功能，并支持用户级别，系统级别和/或关键的待机，挂起和恢复。在APM实施中，X服务器控制电源管理事件。在ACPI实现中，X Window系统将图形消息作为用户进程进行处理，但是系统范围内的电源管理事件（如暂停/恢复）由内核模式驱动程序处理。

窗口管理器

Linux上的窗口管理器是用户级进程，可提供图形用户界面并提供可靠的操作系统电源管理。在Linux中受支持的多个窗口管理器中，有两个流行的窗口管理器GNOME和KDE。在GNOME中，电源管理使用硬件抽象层（HAL）并涉及在称为DBUS的开源消息传递接

口上构建的开源平台电源管理，而KDE3提供名为KPowerSave的专有电源管理解决方案。

英特尔嵌入式显卡驱动程序

在英特尔嵌入式图形驱动程序（IEGD）电源管理中，内核模式驱动程序可帮助Linux内核管理电源。它还负责图形设备的初始化和资源分配。为了让您清楚地了解电源事件的流程，以下是“挂起至RAM”示例的主要部分⁵。

当平台中发生电源管理事件时（例如，当按下按钮或触发窗口管理器选项时，就会将挂起RAM启动），并通知操作系统该事件。通常，Linux操作系统采用协议在软件组件和Linux内核之间传递事件。使用此类协议，操作系统（通常通过窗口管理器）命令内核进入低功耗状态，这时Linux内核通过通知X-Server驱动程序来启动挂起过程。进入低功耗状态之前，X显示屏必须切换到控制台模式。通过在Linux内核中实现ACPI，此切换由X-Server进行。IEGD进程保存图形状态并注册信息时，驱动程序将调用“离开虚拟终端”功能。然后，Linux内核冻结所有用户进程，包括X Window进程。现在内核准备好检查哪些设备已准备好进行挂起操作，并调用每个设备驱动程序的挂起功能（如果已实现），以便将设备时钟置于D3模式-有效地将所有设备置于较低功耗州。在这一点上，只有Linux内核代码正在运行，这会冻结除代码运行所在的处理器之外的所有其他活动处理器。

在执行内核侧挂起代码之后，将执行两种ACPI方法，即PTS和GTS，其结果对于Linux内核可能并不明显。但是，在实际进入睡眠之前，内核将内核唤醒代码的地址写入固定ACPI描述表（FADT）中的某个位置。这使内核能够在接收到还原命令后正确唤醒。恢复命令通常是由用户事件产生的，例如按键，鼠标移动或按下电源按钮，这会打开系统电源。开启后，系统跳至BIOS起始地址，执行诸如设置内存控制器之类的内务处理任务，然后扫描ACPI状态寄存器以向RAM指示系统先前已挂起。如果支持视频重新发布，则在恢复操作期间，BIOS还将调用此函数以重新执行视频BIOS（vBIOS）代码，从而完全重启vBIOS。

然后，系统跳到先前编程的地址，如ACPI寄存器的状态和FADT所示。唤醒地址导致内核代码执行，从而使CPU返回保护模式并恢复寄存器状态。从这一点出发，其余的唤醒过程将遍历挂起过程的相反路径。调用ACPI WAK方法，恢复所有驱动程序，并重新启动用户空间。如果正在运行，则X服务器驱动程序将调用Enter虚拟终端功能，并且IEGD将恢复图形设备状态和注册信息。保存控制台模式后，X服务器驱动程序重新进入GUI，从而成功完成唤醒。

⁵ Ibid

Windows 电源管理

Windows操作系统（尤其是Windows 8），相比于以往的Windows系统中的电源管理在此方面有了显着改进。

电源需求

在Windows 8之前的版本中，电源要求涉及主要在移动个人计算机平台上支持常见的ACPI状态，例如S3和S4状态。但是，Windows 8旨在在包括台式机，服务器，便携式笔记本电脑，平板电脑和电话在内的所有平台上标准化单一电源需求模型。目标之一是提高便携式平台的电池寿命，而Windows 8将智能手机电源模型应用于所有平台，以实现从待机到就绪的快速过渡，并确保隐藏的应用程序消耗最少的资源或不消耗任何资源。

为此，Windows 8定义了表6-5.6中列出的需求⁶。

需求类型	需求
系统电源	1. 最大电池寿命应该以最小的能源消耗来实现
需求	2. 启动和关闭的延迟应最小
需求	3. 应该以智能方式制定电源决策，例如，处于无法更改系统电源状态的最佳位置的设备不应这样做
需求	4. 应具有按需调整风扇或驱动器电动机以实现安静运行的功能
需求	5. 应以独立于平台的方式满足所有要求
设备电源	6. 设备（尤其是用于便携式系统的设备）必须非常注重功耗
需求	7. 设备应积极节省功耗：
需求	a. 应该提供即时功能
需求	b. 低过渡到较高状态的延迟
需求	c. 在可能的情况下，应将设备逻辑划分为单独的电源总线，以便可以根据需要关闭设备的某些部分
需求	d. 应支持适当的已连接待机，以便快速连接
Windows 硬件	8. Windows HCK测试要求所有设备必须支持S3和S4，而不会拒绝系统睡眠请求
认证	9. 待机和连接的待机必须持续数天
需求	10. 在D1-D3状态下，设备必须排队并且不能丢失I/O请求

电源策略

电源策略（也称为电源计划或电源方案）是操作系统定义的首选项，用于选择影响能耗的系统和BIOS设置。对于每种电源策略，操作系统通常会设置两个不同的设置。默认情况下，一个使用电池电源，另一个使用交流电源。为了尽可能节省电池，电池电源的设置旨在更积极地节省电量。Windows默认情况下定义了三个电源策略：

- 性能模式：在性能模式下，系统尝试在不考虑功耗的情况下提供最佳性能。
- 平衡模式：在此模式下，操作系统尝试在性能和功耗之间达到平衡。
- 省电模式：在此模式下，操作系统尝试节省最大电量，以保留电池寿命，甚至牺牲一些性能。

用户可以创建或修改默认计划，但是电源策略受访问控制列表的保护。系统管理员可以覆盖用户对电源策略的选择。在Windows上，用户可以使用名为"powercfg.cpl"的小程序来查看和编辑权限策略。还提供了名为"powercfg.exe"的小程序的控制台版本，该版本允许更改访问控制列表权限。应用软件可以通过注册电源计划来获取电源策略的通知，并可以通过多种方式使用电源策略：

- 根据用户当前的电源策略调整应用程序行为。
- 修改应用程序行为以响应电源策略的更改。
- 根据应用程序的要求移动到其他电源策略。

Windows驱动程序模型

在Windows驱动器模型（WDM）中，操作系统将请求发送到驱动程序，以将设备定为更高或更低的电源状态。收到此类请求后，驱动程序仅保存或恢复状态，跟踪设备的当前和下一个电源状态，而称为物理设备对象（PDO）的结构执行实际上增加或降低设备电源的工作。但是，这种安排是有问题的，因为模型需要驱动程序实现状态机以处理电源IRP（I / O请求数据包），以及由于执行电源状态转换所需的时间，可能会导致不必要的复杂性。例如，对于断电请求，驱动程序将设备的状态保存在内存中，然后将请求传递给PDO， PDO随后将断电。只有这样才能将请求标记为已完成。但是，在随后可能发生的加电请求期间，驱动程序必须首先将该请求传递给PDO，然后PDO在恢复设备状态之前恢复供电，并通知驱动程序将请求标记为已完成。为了克服此困难，提出了Windows驱动程序框架（WDF）。

Windows驱动程序框架

为了简化驱动程序中的电源管理，Windows在最新的Windows Driver Framework (WDF) 驱动程序模型中引入了事件的概念。在此模型中，驱动程序中有可选的事件处理程序功能，由此框架在适当的时间调用事件处理程序以处理电源转换，从而消除了对复杂状态机的需求。Windows 8以称为PoFx的电源框架的形式，提供了一种新的，更精细的方式来满足多功能设备上功能的电源需求。另外，它引入了连接待机的概念，允许关闭电源的设备偶尔连接到外部世界并刷新状态或数据以用于各种应用程序。主要好处是可以从待机状态快速恢复到ON状态，就像系统一直处于唤醒状态一样。同时，功耗成本很低，足以使系统数天处于待机状态。

Windows驱动程序框架

为了简化驱动程序中的电源管理，Windows在最新的Windows Driver Framework (WDF) 驱动程序模型中引入了事件的概念。在此模型中，驱动程序中有可选的事件处理程序功能，由此框架在适当的时间调用事件处理程序以处理电源转换，从而消除了对复杂状态机的需求。Windows 8以称为PoFx的电源框架的形式，提供了一种新的，更精细的方式来满足多功能设备上功能的电源需求。另外，它引入了连接待机的概念，允许关闭电源的设备偶尔连接到外部世界并刷新状态或数据以用于各种应用程序。这样做主要好处是可以从待机状态快速恢复到ON状态，就像系统一直处于唤醒状态一样。同时，功耗成本很低，足以使系统数天处于待机状态。

Windows 8中的设备电源管理

在Windows 8中，为响应即插即用 (PnP) 管理器的查询，设备驱动程序宣布其设备的电源功能。驱动程序对称为DEVICE_CAPABILITIES的数据结构进行了编程，指示该信息如表6-6所示。

Field Function	设备D1和D2 指示设备是否支持D1或D2，或两者都支持	从Dx唤醒 指示设备是否支持从Dx状态唤醒	设备状态 定义与每个Sx状态相对应的Dx状态	DxLatency 到D0的标称过渡时间
------------------	--------------------------------	-------------------------	--------------------------	------------------------

从Dx迁移到D0时，设备存在延迟，这是因为设备需要一小段时间后它们才能再次运行。对于较高的Dx状态，等待时间较长，因此从D3到D0的过渡将花费最长的时间。此外，设备在响应新请求之前必须处于运行状态，例如，硬盘必须旋转后才能再次降低速度。延迟由驱动程序通过DEVICE_CAPABILITIES数据结构宣布。

请注意，如果在低功耗状态下没有花费足够的时间，则设备状态转换可能不值得，因为与设备处于特定状态时相比，转换本身可能会消耗更多的功率。

为了管理设备电源，Windows Power Manager需要知道每个设备的过渡延迟，即使对于同一设备，不同调用的延迟也可能不同。因此，使用者仅指示标称值。Windows操作系统控制查询和设置的电源请求之间的时间间隔，在此期间设备进入睡眠状态。较高的时间间隔将增加睡眠时间，而较低的值将导致操作系统放弃关闭设备电源。处理电源请求。Windows驱动程序模型（WDM）和Windows NT设备驱动程序使用称为I/O请求数据包（IRP）的内核模式数据结构与操作系统以及彼此之间进行通信。IRP通常是由I/O管理器根据用户模式下的I/O请求创建的。在Windows 2000中，添加了两个新的管理器：即插即用（PnP）和电源管理器，它们也创建IRP。此外，IRP可以由驱动程序创建，然后传递给其他驱动程序。

处理电源请求

在Windows 8中，电源管理器将请求（即电源IRP）发送到设备驱动程序，命令它们更改相关设备的电源状态。电源IRP使用主要的IRP数据结构IRP_MJ_POWER，并具有以下四个可能的次要代码：

- IRP_MN_QUERY_POWER：一种查询，用于确定设备安全地进入新请求的Dx或Sx状态或关闭或重新启动设备的能力。如果设备能够在给定时间进行转换，则驱动程序应在宣布该功能之前将与转换相反的任何其他请求排队，因为SET请求通常在QUERY请求之后。
- IRP_MN_SET_POWER：将设备移至新的Dx状态或响应新的Sx状态的命令。通常，设备驱动程序会执行SET请求，而不会失败。总线驱动程序（如USB驱动程序）除外，如果设备正在卸下中，则总线驱动程序可能会返回故障。驱动程序通过请求适当更改设备电源状态，在移至较低电源状态时保存上下文以及在转换至较高电源状态时恢复上下文来满足SET请求。
- IRP_MN_WAIT_WAKE：请求设备驱动程序启用设备硬件，以便外部唤醒事件可以唤醒整个系统。一个这样的请求可以在任何给定的时间保持待定状态，直到外部事件发生为止。事件发生时，驱动程序返回成功。如果设备无法再唤醒系统，则驱动程序将返回故障，然后电源管理器将取消该请求。
- IRP_MN_POWER_SEQUENCE：查询D1-D3计数器，即设备实际处于低功耗状态的次数。睡眠请求之前的计数与睡眠请求之后的计数之间的差值将告诉电源管理器设备是否确实有机会进入低功耗状态，或者是否被长时间延迟禁止，以便电源管理器可以采取适当的措施，并且可能不会发出该设备的睡眠请求。

对于驱动程序开发人员而言，调用各种电源IRP的困难之一是确定何时调用它们。

Windows 8中的内核模式驱动程序框架（KMDF）实现了许多状态机和事件处理程序，包括用于电源管理的状态机和事件处理程序。通过调用事件处理程序，它简化了电源管理的任务在适当的时间。典型的电源事件处理程序包括：D0进入，D0退出，设备电源状态更改，设备从S0/Sx唤醒以及设备电源策略状态更改。

6. J. Lozano, Windows 8 Power Management. (StarJourney Training and Seminars, 2013.) ↵

处理器的电源管理

为了实现精细的电源管理，英特尔处理器目前支持通过片上电源开关创建的电压岛（一项日渐普遍的技术是将设计分割成许多的“电压岛”— 将各个功能块与不同的供电电压相关联）的多个分区。英特尔智能电源技术（Intel SPT）和英特尔智能空闲技术（Intel SIT）软件确定平台最省电的状态，并提供指导以在任何给定时间打开或关闭处理器上的不同电压岛。收到指示进入低功耗状态的指令后，处理器将等待所有共享电压的分区到达安全点，然后再进行请求的状态更改。

CPU状态（C状态）

CPU并非始终处于活动状态。某些应用程序需要来自系统或用户的输入，在此期间，CPU有机会等待并变为空闲状态。当CPU处于空闲状态或运行低强度应用程序时，无需保持CPU的所有内核通电。CPU工作状态（C状态）是空闲处理器关闭未使用的组件以节省功率的能力。对于多核处理器，C状态可以应用于程序包级别或核心级别。例如，当一个单线程应用程序在四核处理器上运行时，只有一个核心处于繁忙状态，而其他三个核心可以处于低功耗，更深的C状态。任务完成后，没有内核处于繁忙状态，整个封装可以进入低功耗状态。ACPI规范为处理器内核定义了几种低功耗空闲状态。当处理器以C0状态运行时，它正在工作。在任何其他C状态下运行的处理器都处于空闲状态。较高的C状态数字表示较深的CPU睡眠状态。在较高的C状态下，将采取更多的节能措施，例如停止处理器时钟，停止中断等。但是，较高的C状态也具有较长的退出和进入等待时间的缺点，从而导致较慢的唤醒时间。对于如果需要更深入的了解，请参阅表6-7中对Intel Atom处理器的各种C状态的简要说明 [Source](#)。

| 状态 | 功能 | 描述 || C0 | 全开 | 这是运行软件的唯一状态。所有时钟都在运行，并且处理器内核处于活动状态。处理器可以为侦听服务，并在此状态下保持缓存一致性。接口，时钟门控等的所有电源管理，在单位级别进行控制 || C1 | 自动停止 | 当核心处理器执行自动暂停指令时，会发生第一级功耗降低。这将停止执行指令流，并大大降低了核心处理器的功耗。核心处理器可以为监听提供服务，并在其中保持缓存一致性这种状态。处理器的North Complex逻辑未明确区分C1和C0 || C2 | 停止补助 | 当核心处理器进入“停止授权”状态时，会发生下一个级别的功耗降低。核心处理器可以在此状态下侦听并保持高速缓存一致性。北部综合大楼仅支持接收单个停车授权。在核心处理器请求C2（或更深层的请求）之后，将进入C2状态。检测到中断事件后，将退出C2状态，进入C0状态。处理器必须确保PLL处于唤醒状态，此时存储器将无法自我刷新 || C4 | 深度睡眠 | 在这种状态下，核心处理器将关闭其PLL(锁相环)，并且无法处理监听请求。还告诉核心处理器稳压器降低处理器的电压。在C4状态期间，North Complex继续处理到内存

的流量，只要此流量不需要监听（即，不为任何一致的流量请求提供服务）。通过从核心处理器/ OS接收C4请求进入C4状态。当North Complex检测到可监听的事件或Break事件时，会退出C4，这将导致它唤醒核心处理器并启动序列以返回C0状态 || C6 | 深度掉电 | 在进入C6状态之前，核心处理器会刷新其缓存并将其核心上下文保存到不同电源板上的特殊片上SRAM中。一旦完成C6进入顺序，就可以完全关闭核心处理器的电压。C4状态和C6状态之间的North Complex逻辑的主要区别在于，由于核心处理器的缓存为空，因此无需在内部前端总线（FSB）上执行监听。这意味着可以在C6状态期间不受阻碍地流动总线主控事件（这将导致从C4状态弹出到C2状态）。但是，核心处理器仍必须返回到C0状态以处理中断。驻留计数器由核心处理器读取 基于对过渡的能量意识和驻留/过渡的历史进行智能的提升/降级 |

性能状态（P状态）

处理器设计者已经意识到，以固定的频率和电压设置运行CPU并非对所有应用都有效。实际上，某些应用不需要在最高额定频率和电压设置所定义的工作点上运行。对于此类应用，可以通过降低工作点来节省功率。处理器性能状态（P状态）是处理器执行以下操作的能力：在支持的不同工作频率和电压之间切换以调节功耗。ACPI为电源管理定义了几个特定于处理器的P状态，以便将系统配置为以节能的方式对系统工作负载做出反应。数字上较高的P状态表示较慢的处理器速度以及较低的功耗。例如，与在P1状态下运行的处理器相比，处于P3状态的处理器将运行得更慢并且消耗的功率更少。当设备或处理器处于运行状态且未空闲（分别为D0和C0）时，它可以处于几种P状态之一。P0始终是性能最高的状态，而P1至Pn则依次为性能较低的状态，根据实现的不同，n最多可以为16。P状态根据动态电压或频率缩放的原理使用，并且在市场上可以作为Intel处理器的SpeedStep PowerNow！获得。适用于AMD处理器，以及适用于VIA处理器的PowerSaver。P状态与C状态的不同之处在于C状态是空闲状态，而P状态是操作状态。这意味着，除了C0（CPU处于活动状态并且正在忙于执行某事）之外，C状态是空闲状态。并在较高的C状态下关闭CPU是有意义的，因为CPU没有执行任何操作。另一方面，P状态是操作状态，这意味着CPU可以在任何P状态下进行有用的工作。C状态和P状态也是正交的-也就是说，每个状态可以彼此独立地变化。当系统恢复到C0时，它将返回到该P状态定义的工作频率和电压。尽管P状态与CPU时钟频率有关，但它们并不相同。CPU时钟频率是衡量CPU主时钟信号上升或下降的速度的指标，这可能是性能的指标，因为更高的性能通常意味着使用更高的时钟频率（内存绑定任务除外）。但是，这是向后看的，因为只有在经过某些时钟周期后才能测量平均时钟频率。另一方面，P状态是OS希望在特定CPU上看到的性能状态，因此P状态是前瞻性的。通常，较高的时钟频率会导致较高的功耗。当CPU空闲时（即处于较高的C状态），无论OS请求的P状态如何，频率都应为零（或非常低）。但是，请注意，出于实际原因，当前CPU封装上的所有内核都共享相同的电压。由于在保持相同电压的情况下以不同的频率运行不同的内核效率不高，因此所有活动内核将在任何给定时间共享相同的时钟频率。但是，某些核心可能处于空闲状态，并

且应具有零频率。由于OS向逻辑处理器请求某个P状态，因此只有在没有一个内核处于繁忙状态并且所有内核都保持在相同的零频率时，才可能将一个内核保持在零频率。虽然为所有核使用全局电压电源会导致无法满足某些P状态要求的情况，但为每个核提供单独的局部电压电源将使成本过高。关于全球和本地电压平台的折衷方案是采用具有不同电压岛的多核架构，其中多个核电压岛上的电压共享相同但可调节的电源电压。英特尔的多核平台称为单芯片云计算就是一个例子。另一个示例是使用某些处理器中可用的全集成稳压器（FIVR），该稳压器允许每个内核处于P状态，以便内核可以在彼此独立的频率下运行。正如集成GPU的行为类似于CPU一样，已为GPU定义了C状态和P状态，例如CPU C状态和P状态。这些称为渲染C状态（RC状态）和渲染P状态（RP状态）。

超频

可以对CPU超频，这意味着以高于制造商指定频率的频率运行内核。实际上，由于存在多个内核以及功率预算导致的所谓的超频功能，这种行为是可能的。超频是增加频率和电压的独特情况，因此超频状态与电压和频率降低的各种P状态相反。在低于某些参数规格的情况下运行时，CPU进入加速状态，这使其有一定的余量来提高性能，而不会违反其他设计规格。这些参数包括活动核心的数量，处理器温度以及估计的电流和功耗。如果处理器的操作低于此类参数的限制，并且工作负载需要其他性能，则处理器频率将自动动态增加，直到达到上限为止。在超频状态下，复杂的算法同时管理电流，功率和温度，以实现最大的性能和功率效率。为了更好地理解超频场景，让我们考虑一个例子。假设四核CPU的TDP限制为35W，那么每个核的最大功率预算为8.75W。如果四个内核中的三个处于空闲状态，则唯一的操作内核可以利用整个35W的功率预算，并且可以“超频”达到比使用频率可能高得多的频率不到9W的功率预算。同样，如果两个内核处于活动状态，则它们使用相同的更高频率并共享功率预算，而空闲内核则不消耗任何能量。当条件允许处理器进入超频模式时，最大超频频率是可获得的最高可能频率。英特尔睿频加速技术的频率取决于工作量，硬件，软件和整体系统配置。由于功率特性的变化，对于涡轮范围频率的P状态的请求可能会或可能不会得到满足。因此，折中的选择取决于许多因素，例如其他内核和GPU的性能，处理器所处的热状态等等。此行为也随时间变化。因此，由于工作频率随时间变化并且取决于C状态选择策略和图形子系统，因此选择P状态值不能保证特定的性能状态。通常，P1状态对应于OS可以请求的最高保证性能状态。但是，OS可以请求具有较高性能的机会状态，即P0。当功率和热预算可用时，此状态允许处理器配置，其中一个或多个内核可以以比保证的P1频率更高的频率运行。处理器可以包括高于P1频率的多个所谓的Turbo模式频率。

一些处理器具有较大的超频范围，并且通常在内核寻求加速时为所有内核授予最大可能的超频频率。但是，并非所有应用程序都能有效地使用增加的核心频率。不同的内存访问模式，可能的缓存争用，或类似来源。因此，让所有内核处于超频模式的最高级别可能

会不必要地消耗功率。为了克服这种低效率，技术已经提出了有效地使一个或多个内核独立运行的建议⁷。其中一项技术会定期分析所有授予超频模式的内核，以根据在观察间隔内内核是否被分类为停转来确定是否应增加、降低或保持其频率不变。

热态 (T态)

为了防止潜在的过热损坏，处理器通常具有热保护机制，在这种情况下，为了减少能量消耗，处理器会通过关闭处理器时钟然后根据预定的占空比重新开启来进行节流。定义热状态或T状态以控制这种节流以便降低功率，并且可以将其应用于各个处理器内核。T状态可能会忽略性能影响，因为它们的主要原因是出于散热原因而降低了功率。T状态共有8个，从0到7，而活动状态为T0。这些状态通常不用于电源管理。

Source . Data Sheet, Intel Corporation, October 2012.

www.intel.com/content/dam/www/public/us/en/documents/product-briefs/atom-z2760-datasheet.pdf. ↪

⁷ M. K. Bhandaru and E. J. Dehaemer, U. S. Patent No. US 20130346774, A1, 2013. Providing energy efficient turbo operation of a processor. Available at www.google.com/patents/WO2013137859A1?cl=en. ↪

电压-频率曲线

了解处理器电压和频率之间的关系至关重要，因为CPU和集成GPU都遵循这种关系以便扩展。操作系统请求的P状态实际上是V-F曲线上的特定操作点。如图6-4所示，电压与频率的关系曲线倾向于有一个拐点，在该拐点处，电压开始随频率成比例变化。

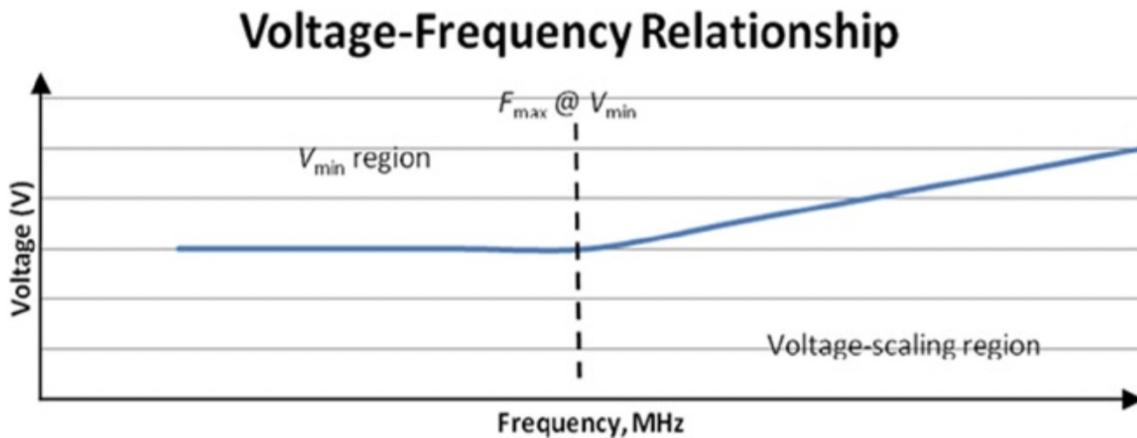


图6-4. 典型处理器的电压-频率关系

到目前为止，无论频率如何变化，都需要一个最小电压 V_{\min} 来使电路工作。最小电压 V_{\min} 时的最大频率 F_{\max} 是处理器部分可以以 V_{\min} 工作的最高频率。如图6-5所示，这是功率效率最佳的地方。将频率增加到超过 F_{\max} 要求增加电压供应，以使电路能够工作。在电压标度区域，所需的电压标度与频率呈合理的线性关系。该区域提供了降低功率的机会，如稍后所述。

图6-5显示了典型处理器中的功耗与频率的关系以及功耗优化的方向。

Power-Frequency Relationship

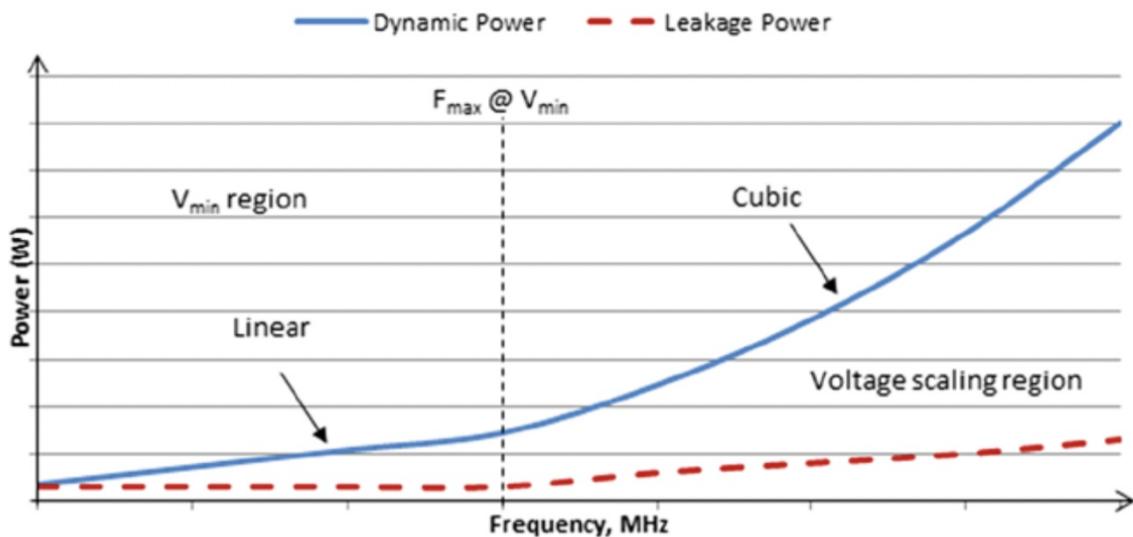


图6-5. 频率功率关系和优化

在 V_{min} 区域，功率下降的速度不如频率快。当动态功率随频率下降时，泄漏功率保持恒定。另一方面，在电压定标范围内，功率的增加速度远快于频率，因为电压定标大致呈线性随着频率 动态功率随 V^2f 上升而泄漏功率随 V^3 大致上升。泄漏功率仅取决于少量的泄漏电流，并且相对于频率遵循与电压曲线几乎相同的模式。

电源优化

让我们回想一下功率方程：

$$(6-2) \text{ Total power} = \text{leakage power} + A C_{dyn} V^2 f$$

此处， A 是活动常数（处理器是否处于活动睡眠状态，或者在时钟门控条件下有关的活动常数）， C_{dyn} 是动态电容， V 是电压， f 是工作频率。从这个方程式可以很容易地看出，为了降低功耗，可以采用以下方法：

- 降低电压和频率
- 减少活动量和 C_{dyn} 当电压在电压定标范围内随频率近似线性增加时，术语V2f表示功率相对于频率的三次方关系（见图6-5）。因此，降低电压和/或频率导致功率的显著降低。可以将以这种方式节省的功率提供给系统的其他部分，以使整个系统的运行受益。

但是，频率降低不能帮助在 V_{min} 处低于 F_{max} （低于该电压不能降低电压，因为存在电路工作所需的最小电压）。在 V_{min} 区域中，电压保持恒定，因此降低频率可以节省很少的功率。

在这一点上，只有活动和减少 C_{dyn} 才能提供进一步的功耗优化。这需要更有效的算法和微体系结构设计，以及动态关闭电路中未使用的部分。上述降低功耗的考虑催生了功耗优化的新思想和方法，包括各种门控优化和特殊用途的异构硬件组件的使用，例如集成了能够进行多媒体处理的GPU，相机图像处理等。总的来说，这不是硬件问题。它也需要在软件微体系结构级别进行仔细考虑。

通常，良好的电源优化需要结合许多方法，所有这些方法都在朝着节省功率和延长电池寿命的目标协调工作。从系统工程的角度来看，可以通过有选择地关闭系统某些空闲部分的电源，在系统内的各个单独电源域中进行优化。可以在系统的各个级别上进行功耗优化。通常，功率优化是在以下级别完成的：

- 架构优化
- 算法优化
- 系统集成优化
- 应用程序级别优化

通常，功率优化结合了所有这些方法。架构优化在处理器硬件级别处理优化机会，并尝试获得合适的硬件-软件分区。算法优化在系统和应用算法中寻找节电机会。例如，在硬件和硬件抽象层之上，图形执行堆栈包括应用程序，中间件，操作系统和图形驱动程序的分层。节省功耗的机会存在于每个层中，并且可以通过算法优化加以利用。

但是，层间优化机会更加复杂，并且通过在系统集成级别采用优化来解决效率低下的问题。例如，可以通过选择使用较少的层并重新定义层的边界以找到最省电的优化位置来提高效率。此外，在应用程序级别，可以通过在最节能的设备上运行任务来考虑在CPU和集成GPU之间进行负载分担，以便在一个单元中节省功率，而另一个单元又可以节省功率。这些优化技术的讨论将在后面详细介绍。

架构优化

在处理器体系结构级别优化电源效率的技术包括：

- 硬件-软件分区
- 动态电压和频率缩放
- 电源门控
- 时钟门控
- 切片门
- 使用低级缓存

软硬件分区

优化硬件和软件交互的方法已经发生了范式转变。较早的方式是通过消除执行瓶颈来获得性能。例如，如果CPU是图形应用程序的瓶颈，那么主要的性能和性能调整方法就是使用更好的CPU或调整CPU代码以最大化图形性能。但是，处理器架构师很快意识到，仅消除执行瓶颈是不够的。从功耗的角度来看，也禁止以最大的性能同时运行系统的所有子部分，因为各种组件都在争夺它们在功率分配中的占比。

这种实现带来了两个优化方向：(a) 可以将一个子部分中节省的功率应用于另一子部分；(b) 未使用的功率可以应用于Turbo行为。因此，要考虑整个系统的电源管理以及在CPU，图形和其他子系统之间切换电源的考虑。因此，软硬件交互的新理念不仅旨在消除性能瓶颈，但还要继续进行调整以提高效率并节省成本力量。例如，现在将重点放在设计目标上，包括：

- 减少CPU处理
- 优化驱动程序代码以使用最少的CPU指令来完成任务
- 简化设备驱动程序接口以匹配硬件接口，以最大程度地减少命令转换成本
- 对某些任务使用专用硬件，并采用平衡的方法执行任务 固定用途的硬件通常使用最少量的门电路来实现，这些门电路用于切换状态或在状态之间切换以执行某些特定任务。由于动态功耗是要切换的门数量的函数，并且切换次数越少意味着动态功耗就越小，因此与通用硬件相比，在专用固定功能硬件上执行相同任务是有益的可能无法为该特定任务使用最佳数量的开关门电路。显然，如果任务的性质发生变化，则无法使用专用硬件，因为它通常不够灵活，无法适应使用方式的变化。在这种情况下，可以通过牺牲任务的灵活性以及通常通过迁移工作负载来实现节电 从通用硬件到固定功能硬件。硬件-软件分区的仔细设计对于以这种方式节省功率是必要的，并且非可编程任务可以从通用执行单元迁移到固定目的硬件。例如，使用为该任务明确设计的

GPU硬件运行的视频处理算法通常比运行与CPU上运行的软件相同的算法消耗更少的功率。

动态电压和频率缩放

为了降低功耗，可以使用动态电压和/或频率缩放功能以可能降低性能的价格来更改CPU内核电压，时钟速率或两者。可替代地，可以以更高的功耗为代价来实现更高的性能。但是，正如前面的P状态讨论中所提到的，随着CPU技术的发展，此过程变得越来越复杂，并且有许多促成因素，例如 多个CPU内核和GPU，热状态等。另一方面，动态电压和频率缩放之外的新技术正在涌现以应对挑战。

电源门控

处理器可以通过不向未使用的电路部分供应电流来有选择地关闭内部电路的电源，从而降低功耗。这可以通过硬件或软件来完成。此技术的示例包括Intel Core和AMD CoolCore，在多处理器环境中，在给定的时间仅某些核心处理器（或这些处理器中的部分电路）处于活动状态。电源门控通常比时钟门控对设计的影响更大，并且可能导致门控状态进入和退出等待时间更长。通常在节省的功率和所涉及的延迟之间考虑架构上的折衷。另一个重要的考虑因素是用于电源门控电路的面积（如果实现）在硬件上。例如，在细粒度的功率门控中，可以将开关晶体管合并到标准单元逻辑中，但是它仍然具有较大的面积损失，并且每个单元难以独立进行电压控制。另一方面，在粗粒度电源门控中，网格型睡眠晶体管通过共享的虚拟电源网络在本地驱动单元，并以牺牲灵敏度为代价来节省面积。为了从功率门控状态快速唤醒，有时保留寄存器可用于关键应用。这些寄存器总是上电，但是它们具有特殊的低泄漏电路，因为它们保存了功率门控模块主寄存器的数据，从而可以快速重新激活。

时钟门控

时钟门控是一种流行的技术，它通过使用较少的开关逻辑并通过关闭不必要的时钟电路来减少动态功耗，从而节省了在给定时间切换无用状态所需的功率。时钟门控可以用RTL代码实现，也可以手动插入设计中。时钟门控有几种形式，从手动到全自动，可以结合使用，也可以分开使用，具体取决于优化情况。一方面，由驱动程序软件执行手动时钟门控，其中驱动程序根据需要管理和启用空闲控制器使用的各种时钟。另一方面，在自动时钟门控中，硬件可以检测到空闲或无工作状态，并在不需要时关闭给定时钟。例如，在特

定板上，内部总线可能使用自动时钟门控，因此它会暂时关闭，直到处理器或DMA引擎需要使用它时，而该总线上的其他外围设备如果未使用则可能会永久关闭。或在该板上不受支持。

切片门控

当前的英特尔处理器，例如第四代核心处理器架构 或更高版本具有集成的图形处理单元，该图形处理单元除了具有用于特定任务的固定功能硬件外，还具有可编程执行单元（EU）的阵列。欧盟与媒体采样器一起按切片排列。例如，某些第四代核心SKU具有40个EU，分布在两个等效片之间，每个切片包含20个EU，并且位于两个不同的电源域中。图6-6显示了典型的英特尔第四代核心处理器图形执行单元的切片结构。

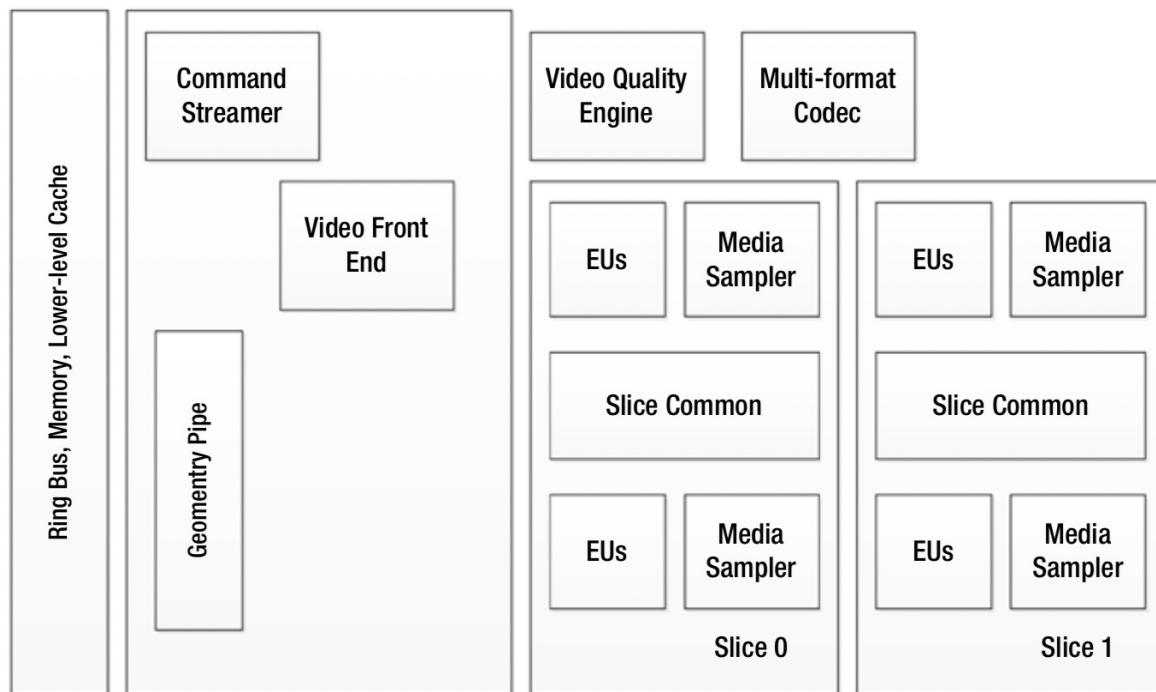


图6-6. 英特尔第四代核心处理器图形执行单元分片结构

由于硬件加速的多媒体任务需要图形硬件中的许多不同硬件资源，因此某些媒体任务可能会对切片内部的硬件资源（例如EU或媒体采样器）以及切片外部的硬件提出不同的要求，例如作为视频前端或视频质量引擎。对于某些需要基于切片的资产进行相对较少工作的媒体工作负载，处理器可以关闭一个切片以节省泄漏功率。例如，对于某些媒体工作负载，需要的EU少于20个，因此驱动程序软件可以关闭一个片的电源，而不会影响性能。这是切片门控，也称为切片关闭。切片门控的优势在于，它可以在各种任务中最大化功率效率。

使用低级缓存

通过使用低级缓存和设计算法以有效方式利用这些缓存，可以显着降低内存能力。除非使用内存受限的系统，否则视频应用程序通常是计算绑定的，而不是内存绑定的。因此，算法可以利用存储器带宽减少方法的优势，从而降低功耗。例如，在英特尔核心架构中，缓存按层次结构排列，其中同时使用了低级缓存和三级缓存。由于较低的内存访问成本，因此可以实现功耗优化。

算法优化

算法优化的目标是通过快速运行任务并在不需要时关闭处理单元来减少执行时间。这可以通过多种方式实现，包括：

- 由于功耗与执行驻留时间成正比，因此在CPU中运行较少的代码可以减少功耗。因此，执行关键软件模块的代码优化有助于算法优化。
- 平台支持将处理任务卸载到专用的省电固定功能媒体硬件块中。
- 为了在给定用法的任务管道中执行各个阶段，通常需要将数据扩展为阶段中的某些中间表示形式。存储此类数据需要更大的内存和缓存带宽。通过最小化存储器带宽可以降低功耗方面的存储器事务处理成本。因此，带宽减少技术是算法优化的重要考虑因素。
- 可以探索管道的各个阶段或子阶段之间可用的并发性，并可以采取适当的并行化方法来减少执行时间。
- 可以通过适当的缓冲来优化I/O操作，以打包大量数据，然后进行更长的空闲时间，因为频繁的短传输不会使模块有机会在空闲时间掉电。此外，I/O优化应考虑磁盘访问延迟和文件中的碎片，因为它们可能对功耗产生重大影响。
- 适当的中断调度和合并为最大化空闲时间提供了机会。
- 所有活动任务可以在平台的所有部分（例如，CPU，GPU，I/O通信和存储）重叠。在进行算法优化时应考虑到功能，性能和质量之间的权衡。根据应用程序的要求，在尝试节省功率时，应注意保持性能和/或视觉质量。以下各节介绍了一些常见的算法优化技术。

降低计算复杂度

当不主动进行计算时，计算设备或系统仅消耗很少的功率，因为仅显示引擎需要唤醒。其他计算引擎可能暂时处于睡眠状态。降低计算复杂度的思想是，仅在必要时将系统保持在高功率或繁忙状态，并尽可能使系统返回空闲状态。改善应用程序的性能可以轻松实现节能，因为它可以使系统更早地返回空闲状态，因为工作可以更快地完成。有几种降低计算复杂性的方法，包括算法效率，活动占空比降低，最小化诸如忙等待锁和同步之类的开销，减少在特权模式下花费的时间以及提高I/O处理的效率。接下来，我们将讨论其中一些方法，但是要对它们进行彻底的处理，请参阅能源感知计算。

选择有效的数据类型

通过使用整数算术，可以优化浮点计算中比较繁琐的算法。例如，使用提升方案的离散小波变换计算通常涉及许多浮点运算。但是提升系数可以用2的幂的有理数来实现，因此数据路径中的浮点单元可以用整数算术单元代替⁹。这样可以节省功耗，因为降低了硬件复杂性。

类似地，以适合于利用编译器优化的方式来重新布置代码，或者以某种数据相关性允许在进入循环之前而不是在循环内部之前进行计算的方式来重新布置代码，可以显着提高性能，从而节省功率。在一个音频应用示例中¹⁰，显示了一些正弦和余弦函数在忙碌循环中的固定值上反复调用；由于值是固定的，因此可以在进入循环之前进行计算。此优化可提高大约30%的性能，同时还可以节省功耗。

在另一个示例中，运动矢量和离散余弦变换计算是在像素矢量上完成的¹¹，而不是分别使用每个像素¹¹，这不仅使纯软件H.263视频编码器的整体性能提高了5%，而且还提供了通过两种方式来节省电力：通过更快地进行计算，以及通过使用改进的内存访问和缓存一致性。

代码并行化和优化

消除代码运行时，处理能力的效率低下问题，也是代码并行化和优化的目标。多线程，流水线，向量化，减少在特权模式下花费的时间以及避免轮询构造是代码并行化和优化的常用技术。使用所有可用资源的适当线程的应用程序通常比单线程对应的应用程序更早完成，并且更有可能提供性能和功耗优势。在这种情况下，选择正确的同步原语是很重要一些应用程序，尤其是媒体应用程序，特别适合在多核或多处理器平台中使用多线程进行改进。在Steigerwald等人提到的多线程媒体播放示例中[12]，虽然实现了几乎线性的性能扩展，但同时四核处理器的功耗也降低了一半，因为所有四个核都忙于运行平衡的工作负载。类似地，可以通过使用矢量操作（例如单指令多数据（SIMD））在相同时钟周期上对数据矢量有效地执行对不同数据的相同操作。大多数现代处理器都支持SIMD操作。英特尔自动矢量化扩展（AVX）在单个处理器时钟周期内支持八个32位浮点同时操作。正如Steigerwald等人所声称的¹³，对于媒体播放应用程序，使用此类SIMD操作可能会导致大约功耗降低30%。在清单6-1中，请注意以下Direct3D查询结构和轮询构造只会消耗CPU周期，导致功率浪费。程序如6-1 Direct3D查询结构的轮询示例（低功耗）

```

while (S_OK != pDeviceContext-> GetData (pQuery, &queryData,
sizeof (UINT64) , 0) )
{
    sleep (0); // wait until data is available
}

```

睡眠(0); //等待数据可用 最好使用阻塞结构来挂起CPU线程。然而，Windows 7 DirectX处于非阻塞状态。尽管使用OS原语的阻塞解决方案可以避免繁忙等待循环，但是这种方法还会增加延迟和性能损失，并且可能不适用于某些应用程序。相反，可以使用软件解决方法，其中启发式算法在循环中检测到GetData()调用。在这种解决方法的示例中[14]，功率降低至3.7W，而性能不会下降。清单6-2显示了解决方法的概念：

程序如6-2轮询构造的替代示例 // ...

```

INT32 numClocksBetweenCalls = 0;
INT32 averageClocks = 0;
INT32 count = 0;
// Begin Detect Application Spin-Loop
// ...
UINT64 clocksBefore = GetClocks();
if ( S_OK != pDeviceContext->GetData( pQuery, &queryData, sizeof(UINT64), 0 ) ) {
    numClocksBetweenCalls = GetClocks() - clocksBefore;
    averageClocks += numClocksBetweenCalls;
    count++;
    if ( numClocksBetweenCalls < CLOCK_THRESHOLD )
    {
        averageClocks /= count;
        if ( averageClocks < AVERAGE_THRESHOLD )
        {
            WaitOnDMAEvent( pQuery, &queryData, sizeof(UINT64) );
            return queryData;
        } else {
            return queryBusy;
        }
    }
} else {
    return queryData;
}
// End Detect Application Spin-Loop

```

减少内存传输 限制数据移动和有效的数据处理可带来更好的性能和更低的功耗。在这种连接方式下，通过使用内存和缓存层次结构将数据保持在尽可能靠近处理元素的位置，并最大程度地减少了从主内存的数据传输，效率更高。

由于减少了存储器访问次数，因此减少了存储器传输可以减少功耗，即使是以适度增加计算复杂性为代价。Bourge和Jung提出了通过对图像中的预测图片使用嵌入式压缩来减少存储器的传输¹⁵。编码反馈回路。可以使用嵌入式编码方案，该方案将参考帧以压缩格式保留在帧存储器中，以便与常规未压缩编码方法相比使用大约三分之一的存储器。如果使用无损压缩，则所需的内存将减半。

但是，通过使用基于块的内存访问并通过仔细管理 嵌入式编码方案的计算复杂度Bourge 和Jung显示出总体上可以节省功率¹⁶。他们通过对编码方案施加一些限制来实现这一目标，这是一种有损方案，能够获得更好的压缩率和相应的功率。比无损计划更省钱。这些限制包括独立地编码每个块，固定每个块的压缩率以及将亮度和色度块共同存储在存储器中。最终结果是，即使计算复杂度增加，在功耗方面占主导地位的内存传输也可以节省 55%。尽管此特定方案会导致较高比特率下的视觉质量下降，但使用适当的无损方案可能会节省总体功耗 由于较少的内存传输。最重要的是，由于减少了内存传输，因此可以使用更靠近CPU的嵌入式较小内存，从而减少访问期间的电缆耗散。在某些硬件实现中，可以使用低成本的片上存储器代替片外SDRAM。

- ⁸. B. Steigerwald, C. D. Lucero, C. Akella, and A. R. Agrawal, *Energy Aware Computing*(Intel Press, 2012). ↪
- ⁹. P. P. Dang and P. M. Chau, “Design of Low-Power Lifting Based Co-processor for Mobile Multimedia Applications,” *Proceedings of SPIE 5022* (2003): 733–44. ↪
- ¹⁰. Steigerwald et al., *Energy Aware Computing*. ↪
- ¹¹. S. M. Akramullah, I. Ahmad, and M. L. Liou, “Optimization of H.263 Video Encoding Using a Single Processor Computer: Performance Tradeoffs and Benchmarking,” *IEEE Transactions on Circuits and Systems for Video Technology* 11, no. 8 (August 2001): 901–15. ↪
- ¹². Steigerwald et al., *Energy Aware Computing*. ↪
- ¹³. Ibid. ↪
- ¹⁴. D. Blythe, “Technology Insight: Building Power Efficient Graphics Software,” *Intel Developer Forum*, 2012. ↪
- ¹⁵. A. Bourge and J. Jung, “Low-Power H.264 Video Decoder with Graceful Degradation,” *Proceedings of SPIE 5308* (2004): 372–83. ↪
- ¹⁶. Ibid. ↪

系统集成优化

可以在系统集成期间优化软件堆栈中各个层之间的交互，以产生更节能的解决方案。操作系统，图形驱动程序，诸如英特尔媒体软件开发套件（SDK）之类的中间件以及应用程序可以在这种优化中相互配合。由于这些层通常是由不同的公司开发的，因此很自然地期望这种交互会导致效率低下。为了提高层间效率，可以考虑采用以下方法进行系统集成优化：

- 减少层数。
- 增进对各层作者的理解关于彼此的能力和局限性。
- 重新定义图层的边界。但是，由于缺少此类根本方法，并且在这些方法可用之前，仍可以在各个级别上进行系统集成优化，其中一些方法如下。

P-F曲线上的系统工作点 图6-7在最小的工作点（在Vmin时为Fmax）和最大的工作点（以Turbo频率运行）相比，在工频曲线上显示了典型的系统工作点。

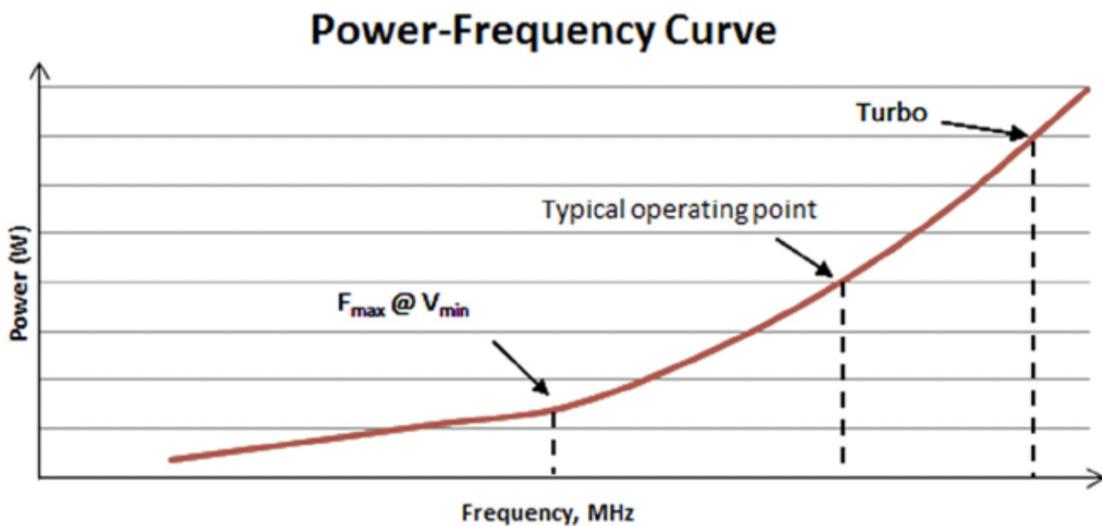


图6-7. 典型系统工作点

如图6-7所示，在功率曲线的电压缩放范围内，调整系统的工作频率对于节省功率很重要。只要满足性能要求，就有可能偶尔以较低的频率运行系统并节省功率。从功耗的角度来看，最佳工作点是Fmax (Vmin)；但是，此频率可能不足以用于某些应用。在另一方面，从性能的角度来看，最佳工作点是在Turbo频率范围内。根据资源利用配置文件，可以知名的图形驱动程序来确定如何调整处理器的频率，并且有可能在Turbo和常规工作频率之间动态移动。随着操作系统管理电源，某些系统提供了各种电源策略，范围从低功

耗低性能到高性能高功率。另外，BIOS提供了一些灵活性来设置系统频率。最终用户可以利用这些电源策略将系统工作点调整到适当的水平；例如，使用节电策略可以降低工作频率，从而节省功耗。

智能调度

硬件-软件分区的级别通常在体系结构优化的范围内。但是，系统级优化也应仔细考虑仅在架构设计中未涵盖的节能机会。例如，在软件层和专用硬件单元之间调度和迁移任务是可以提供这种节能机会的方式。

操作系统为CPU执行任务调度，而图形驱动程序可以为GPU调度和管理任务。智能调度和CPU和GPU之间的负载共享是一个活跃的研究领域，中间件和应用程序层也可能对此做出重要贡献。因此，重要的是找到最有效的处理位置。例如，仅对CPU工作进行多线程操作可能不够，并且就每次操作的焦耳效率而言，每秒的操作效率可能较低。要完成从CPU到更省电的专用硬件模块的此类任务迁移，需要执行堆栈各层的合作。为了便于调度，有时需要将系统的一部分划分为几个较小的块。例如，可以与三个运行时环境（例如OpenGL运行时，Direct3D 11运行时和Direct3D 9运行时）进行交互的共享用户模式驱动程序（UMD）可以重新定义并分为三个组件：OpenGL UMD，D3D 11 UMD和D3D 9 UMD。这将有助于特定的硬件访问以及与运行时环境的交互；它将使系统更适合于电源门控。类似地，内核模式驱动程序针对每次调用重复执行的一些固定工作可能会移至硬件本身。可以在英特尔第四代核心处理器体系结构中找到此类系统级优化的示例，其中使用此类系统级优化可使流行的3D游戏应用程序的CPU功耗降低2.25W¹⁷。

减少占空比

通过并行化系统中的基本活动任务（例如，CPU中的任务），GPU，内存和I/O子系统整体非核心工作周期可以最小化。这将使相关功率子域仅在需要时才用于所需操作，并且仅在最短时间内保持活动状态，否则将其关闭。功率子域包括各种传感器，PLL，存储器接口互连总线等，可以对其进行单独控制以最大程度地降低功耗。此外，为了在更高效的工作点上运行，可以通过沿电压-频率曲线移动并使用来减少处理器的占空比。在较长时间进入空闲状态之前，请在较短的时间内提供较高的频率和较高的功耗。对于整个持续时间，这通常会导致较低的功耗。相反，对于相同的频率，可以使用较低的电压设置来节省功率，因为功率与电压的平方成正比。占空比减少通常是通过图形内核模式驱动程序在系统集成优化级别完成的。图6-8描绘了占空比减少算法的效果，该算法的重点是在较短的时间内使用较高的频率来完成视频应用程序的任务，而CPU则在较长时间处于空闲状态。在此示例中，CPU利用率降低了约20%。

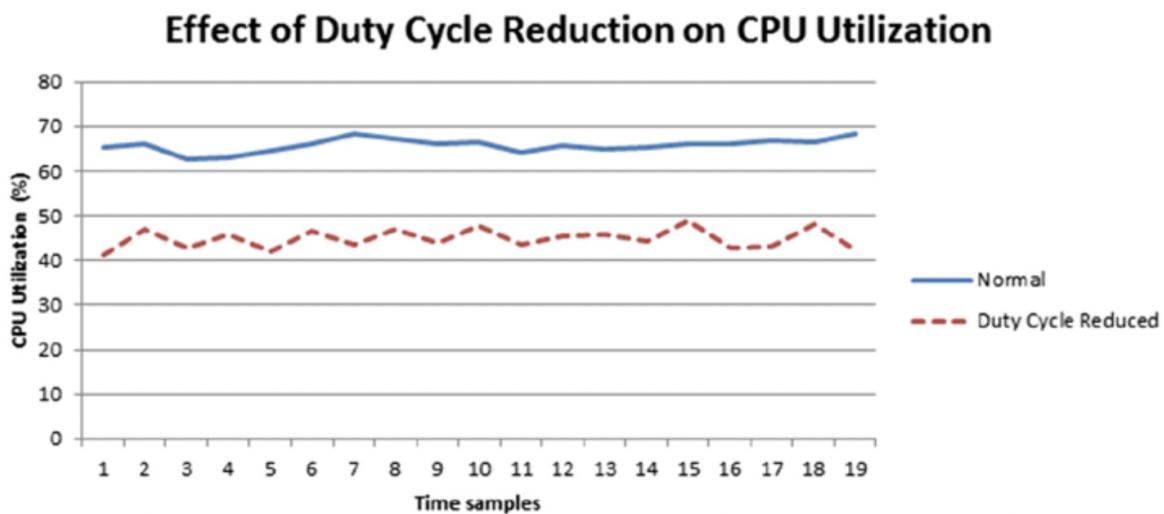


图6-8. 占空比减少对CPU利用率的影响

¹⁷. Blythe, "Technology Insight." ↪

应用级优化

为了支持移动计算设备中的众多功能，使用了多个传感器。因此，现代的平台包括光传感器，陀螺仪，加速度计，GPS接收器和近场通信。通过了解在给定的时间可能有多个传感器处于活动状态的可用系统资源和用户环境，应用程序可以帮助避免电源滥用，并可以帮助用户确定电源不足情况下传感器和功能部件的优先级。

应用程序的上下文意识

编写错误的应用程序可能会不必要地消耗功率，而这些功率否则可以节省。另一方面，如果应用程序知道其运行的系统资源，并且可以感知系统资源可用性的变化，则该应用程序可能以友好的方式对总体功耗做出反应。例如，在检测到电池电量低并随后通知用户时，应用程序可以在进入低功耗状态之前等待用户的干预。或者，在更主动的响应中，它可能会在检测到较暗的环境光条件后默认使显示器变暗。操作系统有责任按照应用程序的要求为每个应用程序分配系统资源。该应用程序注册的电源相关事件使操作系统可以将电源事件通知应用程序，从而使应用程序能够做出适当的响应。应用程序还可以使用操作系统提供的API（应用程序编程接口）来查询系统状态信息。例如，取决于系统是否由电池供电或连接到交流壁式电源，应用程序可以使各种 节电决策：

- 病毒检查程序可能会开始使用电池电源对系统进行部分扫描，而不是使用交流电源进行完整的系统扫描。
- 媒体播放器可能会决定权衡视频质量，以延长蓝光电影的播放时间。
- 游戏应用程序可以选择牺牲一些特殊效果来容纳游戏的更多部分。在Windows中，应用程序可以使用称为GUID_ACDC_POWER_SOURCE的唯一GUID（全局唯一标识符）查询操作系统，以获取电源设置信息，并在发生电源事件时使用此知识。同样，要确定电池的剩余容量，可以使用GUID_BATTERY_CAPACITY_REMAINING。要了解当前的电源策略，可以使用GUID_POWERSCHEME_PERSONALITY。也可以使用GUID_BACKGROUND_TASK_NOTIFICATION来确定是否适合在当前状态下运行后台任务，或者最好等待活动状态以免干扰空闲状态。在Linux中，也存在类似的方法，其中CCBatteryInfo结构可用于确定电池状态。此外，如果应用程序切换了上下文，则可以降低不再运行的应用程序上下文的功耗。

寻求用户干预的应用

应用程序可能会邀请用户进行干预以节省电量。例如：

- 应用程序可以监视电池容量，并且当电池电量下降到其容量的某个分数时-例如，

50%或25%-应用程序可能会在用户界面上显示警告，以警告用户剩余电池电量。

- 应用程序可以通过将更改通知用户并提供使显示屏变暗的选项来响应电源从交流到直流的变化。
- 应用程序可以响应环境光水平，并要求用户调整显示器的亮度。这些动作中的某些动作也可以由系统自动执行，但是取决于应用程序，有些动作可能需要用户干预。通常，用户可配置的选项允许用户个性化系统，应用程序和体验。系统和应用程序设计人员在决定要给用户的选择以及默认实现的选择时可能需要考虑各种折衷。例如，Windows为用户提供了三种电源策略供您选择或定义自己的设置。这些选项和设置会驱动系统级行为，从而严重影响平台的电源效率。

电源功率测量

既然我们已经涵盖了功率优化的不同领域，那么让我们考虑如何实际测量功率。在本节中，我们介绍了测量方法和各种功率测量注意事项。测量和考虑系统各个级别的电源的能力使系统设计人员或用户可以了解现有的电源管理策略，或根据需要部署优化的电源管理策略。测量功率可以发现与功率有关的问题，从而导致系统成本更高。测量功率的主要动机包括：

- 了解应用程序对系统功耗的影响，并可能通过调整应用程序找到优化机会。
- 确定软件更改在用户级别，驱动程序级别或内核级别的影响；并了解由于代码更改而导致的性能或功耗下降。
- 验证是否已从软件中删除调试代码。
- 确定电源管理节省的电量功能，并验证这些功能已打开。
- 确定每瓦性能以驱动性能和功率调整，从而在实际的热约束环境中获得最佳折衷。

但是，很少有工具和说明可用来测量平台中的功耗。同样，根据对精度的要求，可以使用不同的功率测量方法，从简单且便宜的设备到专用数据采集系统（DAQ）。我们提出了各种功率测量方法。

功率测量方法

在计算系统中，功率是在各种系统级别和母板级别进行测量的。特别是，这适用于CPU封装功率，内存功率和显示功率测量。根据电源的类型，这种测量有两种类型：交流电源和直流电源。

交流功率测量

对于系统交流功率或壁式功率测量，通常将交流功率计连接在电源和被测系统之间。价格此测量设备的精度可能会有所不同，具体取决于精度和精度要求。简单，低成本的设备通常具有几个缺点，包括范围小，采样率低和不精确，无法与其他设备（如AC / DC转换器或数据采集系统）一起使用，分辨率低以及无法测量小功率变化。另一方面，它们易于使用，几乎不需要设置时间。对于系统级和母板测量而言，交流功率测量是不合适的，因为这些方法无法深入了解系统的功耗。

直流功率测量

尽管可以使用示波器和万用表测量直流电，但是最简单，最准确，最精确的直流电测量方法是使用自动数据采集系统（DAQ）。DAQ将模拟信号作为输入，并使用专用软件将其转换为数字数据序列，以进行进一步的处理和分析。通常，DAQ可以支持多个输入通道，并且可以通过标准的串行或USB端口与数据分析计算机接口。它们能够处理非常高的数据速率，并且能够测量微小的电压差，因此非常适合自动功率测量。电阻上的功耗可以表示为：

$$P = V^2/R$$

其中V是以伏特为单位的电压，R是以欧姆为单位的电阻，P是以瓦特为单位的功率。通过电路的电流由V与R之比确定。黑盒电路的最大功率，通常做法是与黑盒串联增加一个具有低电阻r的非常小的检测电阻，该电阻具有较大的电阻R，因此电路的总电阻约为等于R。在这种情况下，黑盒所需的功率可以在公式6-3的修改版本中进行近似估算：

$$P = \frac{v \times \Delta v}{R}$$

其中DV是检测电阻两端的压降，V是通道输入相对于地的电位。由于电压是两点之间的电势差，因此对于每个电压测量，都需要两个输入：一个代表接地或参考电势，另一个代表非零电压。在单端测量中，参考是由DAQ自己的接地提供的，并且输入通道电压仅测量非零电压。与为每个通道使用单独的接地相比，单端测量的精度可能较低，但是它们具有使用更快的采样或更多输入通道的优势。DAQ可以将电压形式的通用模拟信号作为输入。模拟信号在转换为电压形式之前可能最初是使用传感器捕获的，或者它们可能已经以电压形式存在。在后一种情况下，一个简单的低电阻检测电阻可以用作传感器。为了测量某个系统或母板组件的功率，通常会安装适当的电源轨，以使检测电阻器串联在电源轨上。如图6-9所示，当电流流过检测电阻器时，会产生压降DV，可以通过DAQ对其进行测量，其中使用了非常小的检测电阻器（例如2毫欧的电阻）。

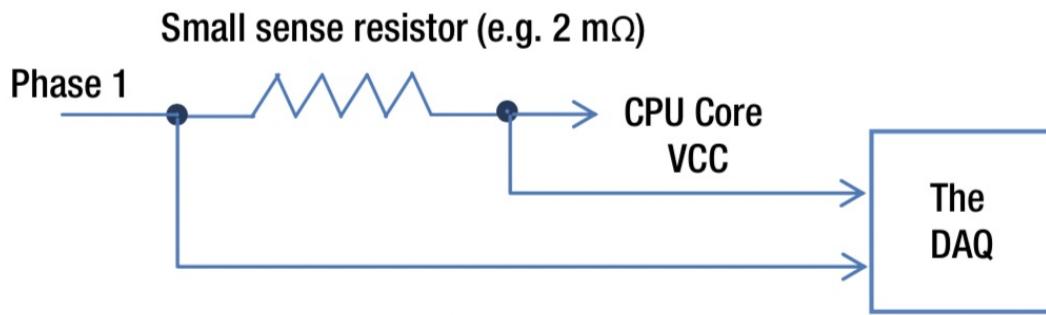


图6-9. 电源track中的功率测量设置

使用DAQ随附的专用软件（例如National Lab LabView），可以对数据进行分析和汇总以提供一段时间内的测量功率。

功率测量注意事项

在测量功率时，通常会考虑以下因素：

- 被测处理器部件的TDP。
- 数据采集系统的准确性和准确性；DAQ和相关软件用于将模拟电压信号实时转换为数字数据序列以及进行后续处理和分析的功能。
- 从一组测量到另一组测量的环境温度，散热和冷却变化；为了避免因环境因素而导致的每次运行之间的差异，通常会进行三轮测量，并考虑中间测量值。
- 单独注释适当的电源轨以节省电能，同时以1KHz的典型采样率（即每1毫秒采样一次）记录所有电源轨上的功耗，并且具有与热相关的测量窗口在1-5秒之间作为移动平均线。
- 识别操作系统后台任务和电源策略；例如，当没有媒体工作负载在运行并且处理器显然处于空闲状态时，CPU可能仍在忙于运行后台任务；另外，操作系统的节能策略可能已经调整了CPU的高频限制，需要仔细考虑。

- 考虑一段时间内的平均功率为了消除电源瞬变中的突然尖峰，并且仅考虑稳态功耗行为。
- 包括综合设置和常用场景的基准；为高端用途考虑的适当工作负载，以便系统的各个部分有机会达到其潜在极限。
- 考虑使用最新的可用图形驱动程序和媒体SDK版本，因为在驱动程序和中间件级别可能进行了功耗优化；此外，使用新的图形驱动程序可能会导致功耗或性能下降，例如对GPU内核，内存，PLL，电压调节器设置和超频（turbo）设置的潜在更改。

工具与应用

功率测量工具包括专用和精确的测量系统（例如DAQ）以及不太精确的基于软件的工具和应用程序。我们考虑一个专门的DAQ系统，并介绍几种功能各异的软件工具。

直流功率测量系统示例

直流功率测量系统的一个示例是基于National Instruments * 基于PXIe 6363 PCI-express的DAQ和相关的LabView Signal Express软件应用程序，用于信号分析。PXIe 6363的信号捕获带宽为每秒125万个样本，每个电压输入通道的A / D转换分辨率均为16位。该输入电压可编程至低至 $\pm 1V$ ，因此很容易放大低压信号。同样，对于当今的低功耗设备，还提供具有更高输入电压的较新版本的PCIe DAQ。通常，将2毫欧的电流检测电阻与所有感兴趣的电源轨串联使用，例如，CPU封装，内存DIMM和显示器，其峰值，平均值和最小直流功耗为测量。而且，将监视运行时CPU和GPU的频率以确定适当的涡轮运行。每次运行时都会自动校准电源设置，以检测环境温度可能导致的检测电阻和测试线束变化。为了捕获并计算以瓦特为单位的功率，必须测量每个电源轨的电压和电流。这是通过在每个电压轨上使用与输入电源串联的电流检测电阻器来实现的。电流检测电阻两端的电压降是一个小幅度信号，与流经检测电阻的电流量直接相关。每个电源轨的电压（正极和负极）和电流检测电阻的输出（正极和负极）通过可拆卸的TB-2706接线盒模拟输入模块直接连接到PXIe 6363。使用LabView Signal Express记录并绘制测得的功率，以生成详细而全面的功率性能曲线。该应用程序可捕获并处理来自PXIe 6363 DAQ模块的电压和电流测量结果，并只需将测得的电压和感测电流相乘即可计算以瓦特为单位的功率。

该应用程序还支持各种统计测量，例如用于详细信号分析的移动平均值，峰值，平均值和最小功率。图6-10描绘了用于功率测量系统的LabView配置界面的示例。在该界面中，可以选择感兴趣的电压通道。

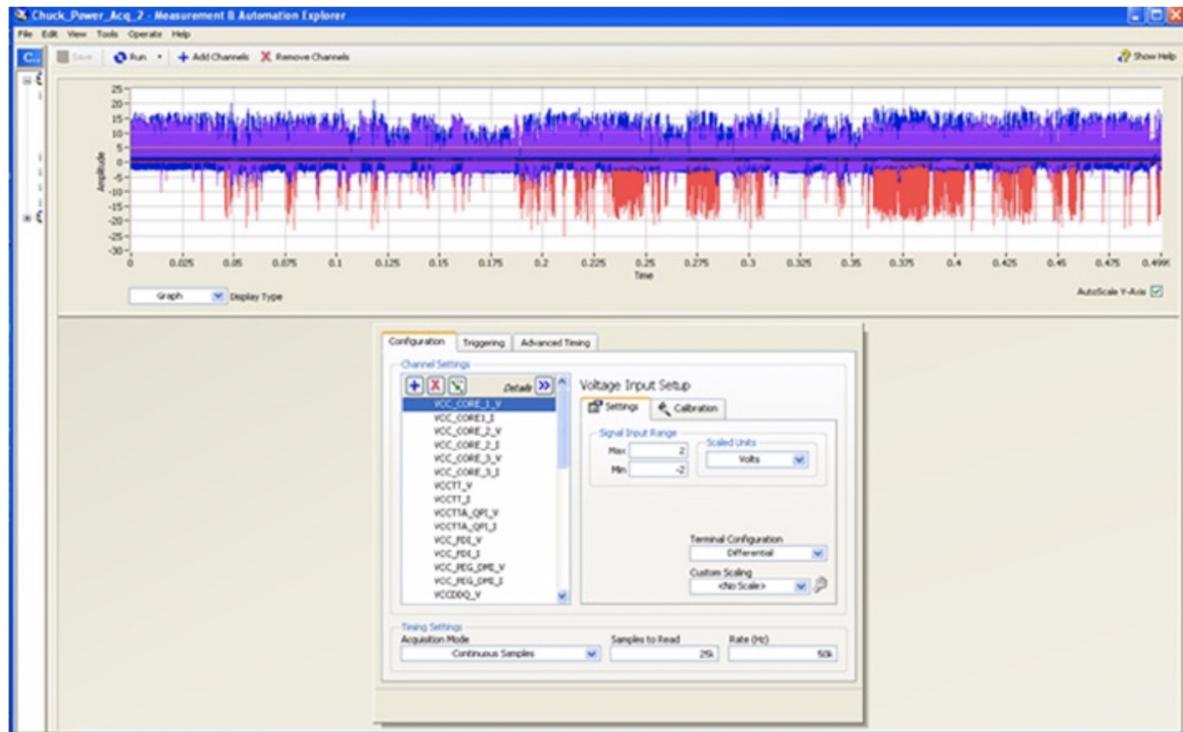


图6-10. 用于功率测量的LabView数据采集设置

然后，图6-11显示了进行功率测量时LabView界面的示例。顶部和底部窗口分别显示来自所有输入通道和单个通道（通道0）的电压，电流或功率信号。

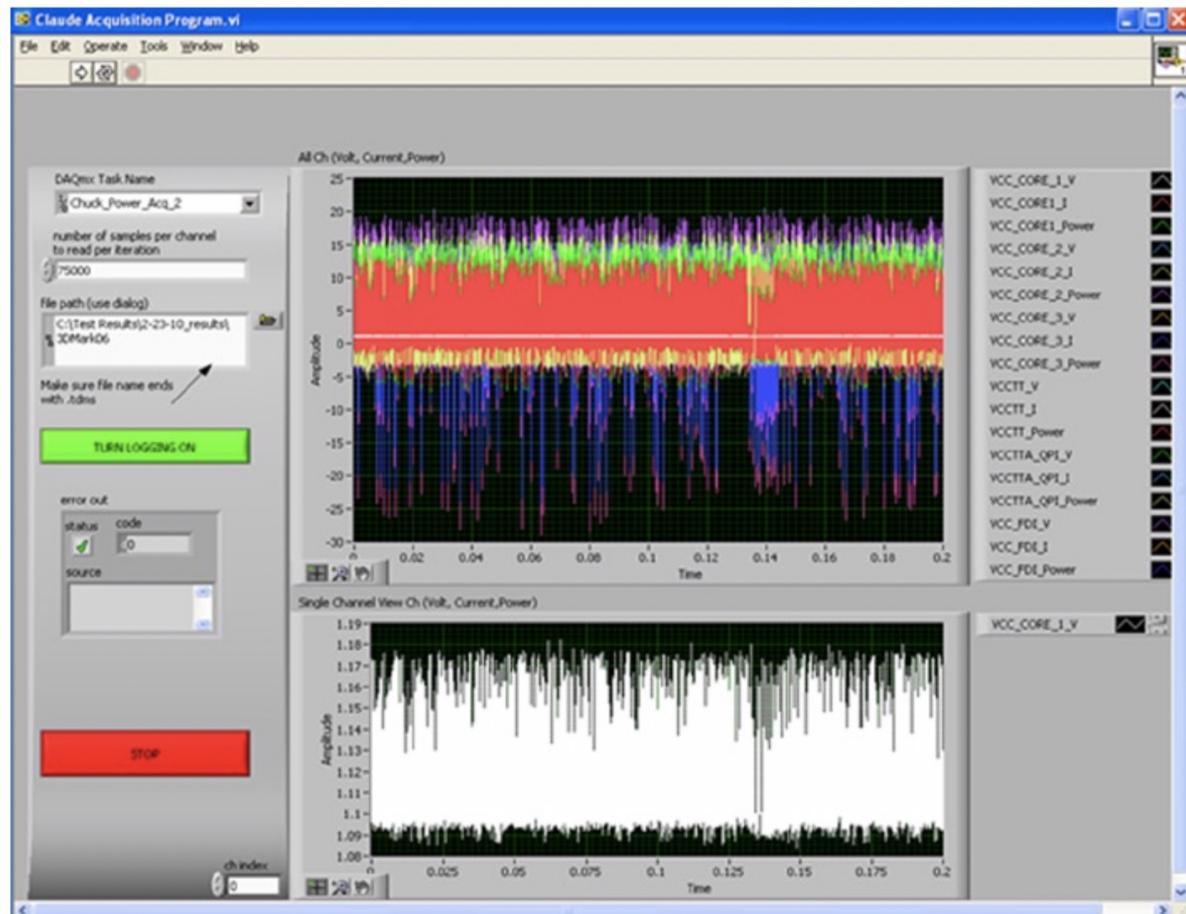


图6-11. 电源数据采集正在进行中

电源测量需要考虑的因素

在测量功率时，通常会考虑以下因素：

- 被测处理器部件的TDP。
- 数据采集系统的准确性和准确性；DAQ和相关软件用于将模拟电压信号实时转换为数字数据序列以及进行后续处理和分析的功能。
- 从一组测量到另一组测量的环境温度，散热和冷却变化；为了避免因环境因素而导致的每次运行之间的差异，通常会进行三轮测量，并考虑中间测量值。
- 单独注释适当的电源轨以节省电能，同时以1 kHz的典型采样率（即每1毫秒采样一次）记录所有电源轨上的功耗，并且具有与热相关的测量窗口在1-5秒之间作为移动平均线。
- 识别操作系统后台任务和电源策略；例如，当没有媒体工作负载在运行并且处理器显然处于空闲状态时，CPU可能仍在忙于运行后台任务；另外，操作系统的节能策略可能已经调整了CPU的高频限制，需要仔细考虑。
- 考虑一段时间内的平均功率为了消除电源瞬变中的突然尖峰，并且仅考虑稳态功耗行为。
- 包括综合设置和常用场景的基准；为高端用途考虑的适当工作负载，以便系统的各个部分有机会达到其潜在极限。
- 考虑使用最新的可用图形驱动程序和媒体SDK版本，因为在驱动程序和中间件级别可能进行了功耗优化；此外，使用新的图形驱动程序可能会导致功耗或性能下降，例如对GPU内核，内存，PLL，电压调节器设置和超频（turbo）设置的潜在更改。

测量工具与应用

功率测量工具包括专用和精密的测量系统（例如DAQ）以及基于软件的工具和应用程序（精确度比DAQ等精密测量系统较低）。我们考虑一个专门的DAQ系统，并介绍几种功能各异的软件工具。

直流功率测量系统实例

直流功率测量系统的一个示例是基于National Instruments * 基于PXIe 6363 PCI-express的DAQ和相关的LabView Signal Express软件应用程序，用于信号分析。 PXIe 6363的信号捕获带宽为每秒125万个样本，每个电压输入通道的A/D转换分辨率均为16位。该输入电压可编程至低至 $\pm 1V$ ，因此很容易放大低压信号。同样，对于当今的低功耗设备，还提供具有更高输入电压的较新版本的PCIe DAQ。通常，将2毫欧的电流检测电阻与所有感兴趣的电源轨串联使用，例如，CPU封装，内存DIMM和显示器，其峰值，平均值和最小直流功耗为测量。而且，将监视运行时CPU和GPU的频率以确定适当的涡轮运行。每次运行时都会自动校准电源设置，以检测环境温度可能导致的检测电阻和测试线束变化。为了捕获并计算以瓦特为单位的功率，必须测量每个电源轨的电压和电流。这是通过在每个电压轨上使用与输入电源串联的电流检测电阻器来实现的。电流检测电阻两端的电压降是一个小幅度信号，与流经检测电阻的电流量直接相关。每个电源轨的电压（正极和负极）和电流检测电阻的输出（正极和负极）通过可拆卸的TB-2706接线盒模拟输入模块直接连接到PXIe 6363。使用LabView Signal Express记录并绘制测得的功率，以生成详细而全面的功率性能曲线。该应用程序可捕获并处理来自PXIe 6363 DAQ模块的电压和电流测量结果，并只需将测得的电压和感测电流相乘即可计算以瓦特为单位的功率。

该应用程序还支持各种统计测量，例如用于详细信号分析的移动平均值，峰值，平均值和最小功率。图6-10描绘了用于功率测量系统的LabView配置界面的示例。在该界面中，可以选择感兴趣的电压通道。然后，图6-11显示了进行功率测量时LabView界面的示例。顶部和底部窗口分别显示来自所有输入通道和单个通道（通道0）的电压，电流或功率信号。

软件测量工具与应用

为了获得有关计算机平台在运行期间消耗了多少能量的最佳和最准确的数据，需要一个硬件功率计。 Fluke的网络数据采集单元（NetDAQ）， National Instrument DAQ和 Yokogawa WT210是此类采集系统的示例。但是，硬件功率计通常非常昂贵，对于仅感兴趣的普通消费者或应用程序开发人员来说，这些成本就显得过高了，一次左右的功率测量。对于这些用户，选择测量功耗的软件工具或应用程序更有意义。

此类测量工具和应用程序主要用于在运行和不运行工作负载的情况下识别电源问题，以优化系统的功耗。通常遇到的问题包括：

- CPU /芯片组电源：通过检查CPU C状态驻留时间来确定是否以最佳方式管理CPU和芯片组电源，并深入了解导致平台功耗增加的原因，可以识别出此类问题。例如，由于设备中断或软件活动，在深C状态（例如C3）的高驻留状态可能指示频繁的C状态转换。
- CPU利用率：通常采用CPU利用率示例 在每个计时器滴答中断时-即每15.6毫秒 大多数媒体应用程序和某些后台应用程序。但是，可以将计时器分辨率从默认的15.6毫秒缩短为试图捕获其中的活动较短的时间。对于多核CPU，CPU利用率和功耗取决于活动持续时间，而每个核可能仅在平台处于活动状态的总持续时间的一部分中处于活动状态。因此，CPU核心利用率和平台利用率应分开计算。逻辑上，当两个内核的活动重叠时，大多数电源测量工具会将CPU利用率显示为两个利用率的总和。实际上只有极少数工具（其中包括Intel Battery Life Analyzer）可以使用细粒度的过程信息来确定平台程序包和逻辑CPU的总活动时间。通过调查CPU利用率，可以识别出效率低下的软件组件及其热点，并可以确定软件组件及其热点的影响来寻找优化机会。
- CPU活动频率：电动工具可帮助识别导致CPU状态频繁转换的软件组件。确定每个组件的活动频率以及每个滴答周期中发生的活动数量非常有价值。了解为什么频繁发生过渡可能有助于指出与电源相关的问题或改进前景。
- GPU功能：在现代处理器上，由于大多数媒体应用程序都在GPU上运行，因此了解GPU C状态转换和GPU利用率的影响也很重要。GPU利用率在很大程度上控制了媒体应用程序的功耗。但是，只有很少的工具能够报告GPU利用率。英特尔GPA就是这样一种工具。通常，有几种工具和应用程序可用于测量和分析计算设备各个组件的功耗。有些与分析空闲系统行为特别相关，而另一些则适合于媒体应用程序。在一节中，我们将讨论其中一些工具，首先是基于Linux / Android的PowerTop，然后介绍几种基于Windows的工具。然后，我们讨论监视和分析电池寿命的特定工具。强力病毒还提到了它，主要用于热测试。但是，随着新工具的不断开发，显然没有涵盖某些工具。

PowerTop

PowerTop是由Intel开发并获得GPL许可发布的软件实用程序，旨在测量和分析运行在Android, Linux或Solaris操作系统上的应用程序，设备驱动程序和内核的功耗。这对识别有电源问题的程序，并查明导致过度用电的软件。这对于延长电池寿命的移动设备特别有用。

PowerCfg

PowerCfg是Windows中的命令行工具，允许用户控制系统的电源管理设置以及查看或修改电源策略。它通常用于检测电源效率，处理器利用率，计时器分辨率，USB设备选择性挂起，电源请求和电池容量中的常见问题。

PwrTest

PwrTest是Windows驱动程序工具包中提供的电源管理测试工具，它使应用程序开发人员和系统集成商能够从电源管理信息中获取电源管理信息，例如各种睡眠状态信息（例如C状态和P状态信息）和电池信息。系统并记录一段时间。

Perfmon和Xperf

Windows Perfmon提供了监视Windows中可用的性能计数器（包括C状态和P状态驻留状态）的功能，这些功能有助于理解CPU利用率和与活动有关的问题。Xperf是一种命令行工具，可通过监视系统和内核事件（如一段时间内的上下文切换，中断服务例程和延迟的过程调用）并生成报告进行图形审查来帮助开发人员进行系统范围的性能分析。在系统处于空闲状态，正在运行Web浏览或媒体应用程序期间，将事件与系统状态关联起来很有用。Xperf生成事件跟踪日志，可以使用Xperfview查看它。Windows Performance Toolkit中提供了这两个工具。

Joulemeter

Joulemeter由Microsoft Research开发，是一种建模工具，用于测量虚拟机（VM），各种外形尺寸和功率容量的计算机甚至计算机上运行的各个软件应用程序的能耗。它可以测量CPU，屏幕，内存和存储等组件对其总功耗的影响采用。它的优点之一是它可以测量不具有硬件接口，因此不适合使用硬件功率计进行测量的软件组件（例如VM）的影响。

可从Joulemeter获得的数据包括每个组件的当前能源使用情况，例如基本或闲置能源使用情况，高于基准闲置状态的CPU使用情况，监视器和硬盘。输出数据以瓦特表示，并且每秒更新一次。可以在焦耳计中找到详细信息：计算能量的测量和优化¹⁸。

英特尔Power Gadget

为了帮助最终用户，独立软件供应商，原始设备制造商和应用程序开发人员在不使用系统任何硬件仪器的情况下准确估算功耗，英特尔开发了一种名为英特尔Power Gadget的软件工具，该工具可用于第二代英特尔核心处理器。该工具的其他功能包括估算多路插座系统上的电源以及可在应用程序代码的各个部分中提取电源信息的外部可调用API。该小工具包括Microsoft Windows侧栏小工具，驱动程序和库，可使用处理器中的电能计数器来监视和估计以瓦特为单位的实时处理器封装功率信息。安装后，可以在运行工作负载或系统空闲时简单地启动小工具以监视处理器的电源使用情况。“选项”弹出窗口允许以毫秒为单位设置采样分辨率，以瓦特为单位设置最大功率。输出数据（特别是处理器封装的功率和频率）是实时生成的，可以用逗号分隔值（CSV）格式记录在文件中。可以从Intel网站下载该小工具¹⁹。

英特尔Power Checker

英特尔功率或能量检查器工具根据所做的有用工作来确定系统的电源效率，该工作涉及工作期间消耗的能量。对于媒体或游戏应用程序开发人员来说，这是一种简便的方法，可以在使用Intel Core或Atom处理器的移动平台上检查其应用程序的电源效率。该工具不需要外部功率计，可用于为英特尔处理器或Java框架应用程序编译的任何应用程序的功率分析。默认情况下，此工具检查系统功能以提供功耗数据以及是否安装了称为EzPwr.sys的特定驱动程序（英特尔Power Gadget的一部分），如果使用外部功率计设备，则很有必要。通常，该工具首先在不运行目标应用程序的情况下测量基准功率，同时关闭不必要的进程，例如操作系统更新，Windows索引服务，病毒扫描，Internet浏览器等。在下一步中，将运行目标应用程序，并从目标应用程序执行的期望点开始再次测量功率。最后，当目标应用程序完成并返回到空闲状态时，它将再次测量电源。该工具提供对经过时间，能耗和平均C3状态驻留时间的分析，并提供平台计时器持续时间（以毫秒为单位）。该工具现在是英特尔软件开发助手的一部分²⁰。

英特尔电池寿命分析仪

英特尔电池寿命分析器（BLA）是在Microsoft Windows上运行的软件工具，主要用于监视硬件和软件平台组件的活动并确定其对电池寿命的影响。它可以识别阻止平台进入低功耗状态的驱动程序，进程或硬件组件。BLA具有许多支持功耗分析的模块，包括CPU C状态和软件活动分析。系统在深C状态花费的时间越长，消耗的功率就越少。BLA建议使用C状态驻留程序的阈值，尤其是对于包含多个处理器内核的处理器封装（即插槽），空闲时最深的C状态驻留率应大于95%，每个内核98%。同样，闲置时，包装的C0和C1状态应小于5%。BLA工具中有一些选项可以设置适当的C状态阈值。可以通过Intel的电子邮件请求BLA工具的副本²¹。

英特尔图形性能分析器

英特尔图形性能分析器2013（英特尔GPA）是一套由三个图形分析和优化工具组成的套件，即系统分析器，帧分析器和平台分析器，可帮助游戏和媒体应用程序开发人员优化游戏和其他图形密集的应用程序。英特尔GPA支持运行Microsoft Windows 7、8、8.1或Android操作系统的最新一代基于Intel Core和基于Intel Atom处理器的平台。系统分析仪实时提供CPU和GPU的性能和功耗指标，并允许用户快速确定工作负载是CPU约束还是GPU约束，因此用户可以专注于特定的优化工作。帧分析器能够分析性能并降低功耗到帧级别。平台分析器通过上下文中的任务，线程，Microsoft DirectX和GPU加速的媒体应用程序的时间线视图，提供CPU和GPU指标以及工作负载的离线分析。该工具也可以从Intel获得²²。

GPU-Z和HWiNFO

GPU-Z是TechPowerUp的轻型系统实用程序，旨在提供有关视频卡和/或集成图形处理器的重要信息；它支持nVIDIA，ATI和Intel图形设备。HWiNFO是可从Internet获得的免费软件，它结合了CPU-Z和GPU-Z的功能，并提供CPU，GPU和内存使用情况以及其他系统信息。

Power Virus

Power Virus执行特定的机器代码以达到最大CPU功耗限制，即CPU的最大热能输出。该应用程序通常用于在产品的设计阶段对计算机组件进行集成测试和热测试，或用于使用综合基准进行产品基准测试。

¹⁸. Available from Microsoft Research at research.microsoft.com/en-us/projects/joulemeter/default.aspx. ↵

¹⁹. Available from software.intel.com/en-us/articles/intel-power-gadget. ↵

²⁰. Available from software.intel.com/en-us/isda. ↵

²¹. Request for BLA tool can be made at batterylifeanalyzer@intel.com.

²²Available from software.intel.com/en-us/vcsource/tools/intel-gpa. ↵

小结

在现代处理器中，功耗考虑已超出电池寿命，并试图决定性能。我们回顾了运行在主流计算设备上的媒体应用程序的功耗行为。首先，我们讨论了典型系统的功耗要求和限制，功率方程式以及各种电源的方面。然后，我们介绍了如何将移动设备用作计算，通信，生产力，导航，娱乐和教育的平台。我们还调查了三个主要主题：电源管理，电源优化和电源测量注意事项。最后，我们了解了几种功率测量工具和应用，以及它们的优缺点。特别是，我们以一个特定的示例为例，使用DAQ的直流功率测量系统，以及几种基于软件的功率测量工具。尽管没有适合所有类型功率测量场景的单个工具或应用程序，但是某些工具完全能够提供对视频应用程序功率曲线的重要见解，并且对于此目的非常有用。

低功耗平台上的视频应用的功耗

某些移动应用，尤其是在设备低电量情况下运行时，出现耗电量不合理增加，造成设备电池不必要的老化。某些应用程序在用户停止使用后，仍会自动重启，或无法进入休眠状态，在大多数情况下，这些应用仍在后台执行非关键任务。还有一些应用，在执行核心任务时耗电较少，但是在用户行为跟踪、上传用户信息或下载广告上耗电更多。

为了节约电池电量，系统经常会根据耗电需求，进入深度休眠或唤醒；因此，某些应用为了执行不必要的任务，滥用唤醒和休眠特性，比如检查服务更新、内容更新、接受邮件或消息、报告用户行为和地理位置等。因此，合理使用休眠或唤醒能力，可以节省电量，延长电池寿命。

然而，不同移动应用为了省电有不同的优化重点。应用通常情况下没有足够的性能空间来执行辅助任务，因此需要考虑媒体应用特征的复杂性，以及设备的资源限制。

本章主要从媒体应用的角度出发，针对第6章提出的耗电、优化方案扩展到低电量的领域。本章首先低电量设备上的耗电优先级。然后阐述低电量设备的典型媒体应用场景，即分析如何通过Miracast无线显示器进行视频播放，视频录制，视频会议和视频传输。同时提供了每种应用场景下的电量优化方案实例。接着阐述系统低耗电方案中存在的机遇和挑战，对ACPI电量方案在第6章的基础上进行细化。

下一节将阐述低电量场景下的电源管理技术，特别是用于节省电量的显示器管理方案。然后，通过调研电源优化的软硬件因素，提出耗电优化方案。在最后一节中将简要介绍低耗电测试的注意事项和指标。

低功耗设备的重要事项

在过去几十年中，根据众所周知的摩尔定律，计算机的计算速度和密度经历了指数级增长。由于增长最终会受到电量和设备物理极限的限制，这种增长趋势将会结束。但就目前而言，个别计算机元件的耗电尚未达到极限。移动计算设备，如智能手机、笔记本电脑和平板电脑，或者家用娱乐电子产品，如机顶盒、数码相机、带调制解调器，基本上都遵循同样的趋势，因此仍然存在耗电优化的空间。

另一方面，无论是家用电器的电子控制器，还是家庭能源管理系统，或者是车载信息娱乐设备，还是复杂的医疗设备，对于许多低功率嵌入式系统而言，移动设备的耗电增加正变得普遍。这就要求，设备不仅做到尽量省电，而且必须做到“始终开启，始终可用”。

因此，在消费者永不满足的需求和激烈竞争的推动下，移动设备制造商持续提供的功能越来越多、越来越快，这些都需要更加省电以便扩展电池寿命。此外，小型的可穿戴计算设备的出现（如智能耳机和智能手表）需要极其低耗电的处理器。例如，一块表面积小至1600平方毫米的智能手表，需要超过一周的电池续航时间，这些都引领着物联网（IoT）的发展。

设计人员的目标是在降低成本的同时降低包装、制造、操作和可靠性的成本，同时还需要支持日益复杂的设计。这种雄心勃勃的目标通常会随着每两年引入更精细的几何形状的新硅工艺技术的出现而实现。然而，由于栅极二极管和结二极管泄漏的增加，每一代工艺都会导致更高的功耗泄漏（*power leakage*）。尽管动态功耗随几何尺寸而减小，但不断增长的导线密度抑制了这种降低。因此，仅动态功耗的简单缩减不足以用于下一代应用。对于越来越复杂的应用程序，越来越多的性能要求对电池电量提出了苛刻的要求。这要求更积极的管理泄漏功耗和有功功耗。

通常，低功耗设备从一种工艺几何过渡到另一种较小几何的优先级包括：

- 降低动态功率。这是可能的，因为对于较小的几何形状，动态电容和电压通常都会降低。
- 保持总静态泄漏功耗。对于硬件架构师来说，这是一个重点关注的领域。因为由于工艺技术的原因，对于较小的几何形状，泄漏功耗往往会上升。因此有必要在该区域进行优化以维持泄漏功耗。
- 将有功泄漏和动态功率占比保持在一个较小的比例范围（例如10%~15%）。同样，决不允许出现泄漏功率决定功耗的情况。

让我们回顾一下第6章介绍的功耗-频率关系。图7-1显示了相对于频率的动态功耗和泄漏功耗。

Power-Frequency Curve

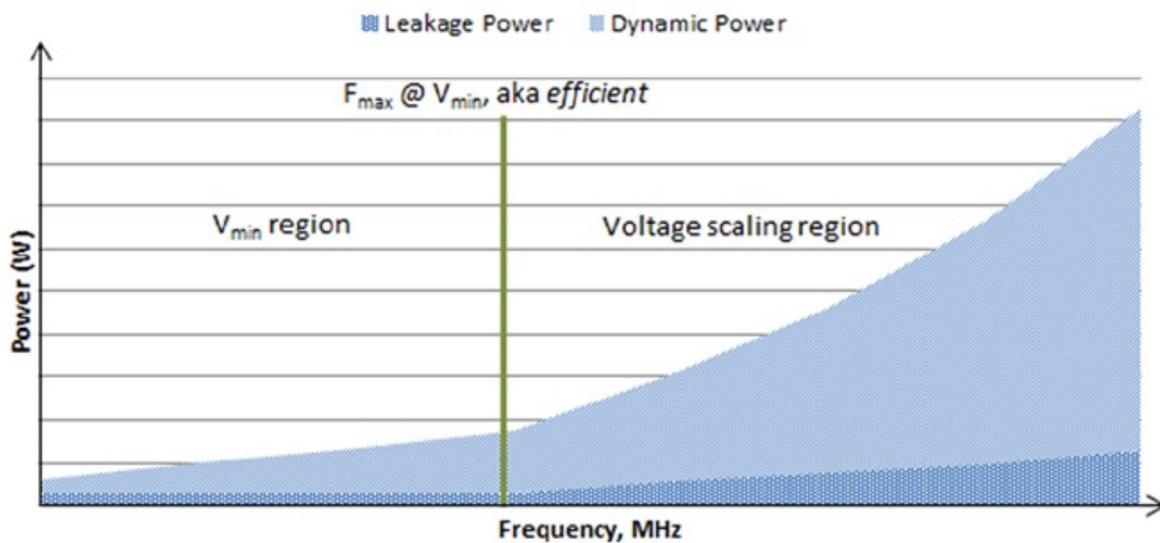


图7-1. 功耗-频率关系图。低功耗设计的目标是将泄漏功耗保持在动态功耗的10%~15%
从图7-1可以明显看出，存在一个有效频率。当低于该频率时，电压和频率缩放无法实现
良好的功耗降低。在此 V_{min} 区域中，电压仅随频率线性缩放，因此泄漏电流以及因此带来的
泄漏功耗成为进一步降低功耗的决定因素。还必须指出，电压不能任意降低，因为将电
路驱动到活动状态存在最小电压。但是，在有效频率点之上，由于电压随频率呈三次方关
系缩放，因此可以实现良好的电压缩放。在该功耗相对较高的区域中，可以通过简单的电
压-频率折衷来降低功耗。

图7-2给出了移动、低功耗平台架构的例子。电源管理和优化对于架构的每个模块：系统
级芯片（SOC, *system-on-a-chip*），存储模块，输入和输出（I/O）模块，传感器和摄像头，
控制器和通信模块，都是必不可少的。在典型的现代SoC中，存在专用电路用于热控
制和电源管理，例如电源管理控制器（PMC），后面的内容会对其进行讨论。

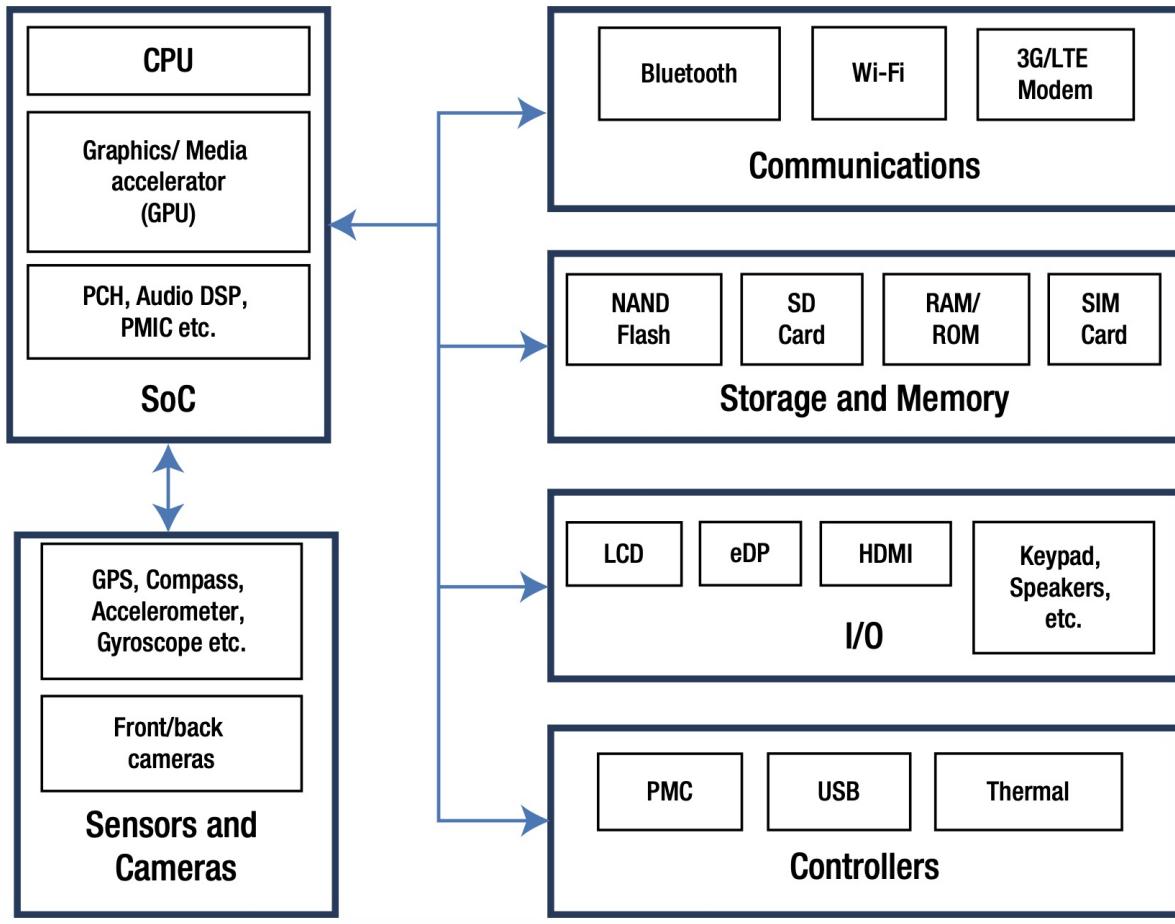


图7-2. 移动平台架构的例子

低功耗平台上典型的媒体应用

低功耗计算平台涵盖了广泛的设备和应用程序。无论是否涉及到多媒体，所有这些设备和应用程序在没有使用或仅部分使用时都需要节省电量。表7-1列举了一些低功耗平台的示例。

表7-1. 低功耗平台举例

领域	举例	特殊需求
Computing on the go	Smartphones, tablets, netbooks, wearable devices	Small form factor and limited storage capabilities; minimum heat dissipation and special cooling requirements; reduced memory bandwidth and footprint; multiple concurrent sessions; hardware acceleration for media and graphics; sensors; real-time performance; extended battery life
Medical equipment	Imaging systems, diagnostic devices, point of care terminals and kiosks, patient monitoring system	Amenable to sterilization; secure, stable and safe to health; resistant to vibration or shock; lightweight and portable; high-quality image capture and display capability; fanless cooling support, etc.
Industrial control systems	Operator-controlled centralized controller of field devices	Special I/O requirements, vibration and shock withstanding capabilities, variety of thermal and cooling requirements, ranging from fanless passive heat sink to forced air, etc.
Retail equipment	Point of sale terminals, self-checkout kiosks, automatic teller machines (ATMs)	Ability to withstand extreme ambient temperature, good air-flow design, security from virus attacks, non-susceptibility to environmental conditions (dust, rain etc.)
	Centralized monitor of a	Internet connectivity;

Home energy management systems	home's usage of utilities such as electricity, gas and water; data mining of energy usage for reporting, analysis and customization (e.g., warning users when washing clothes is attempted during peak electrical rates or when a light is left on without anyone present, etc.)	various sensors; ability to control various smartphone apps; wireless push notification capability; fanless operation; ability to wake from a low-power or power-down state by sensors or signals from Internet; low power consumption while working as an energy usage monitor
In-vehicle infotainment systems	Standalone navigation systems, personal video game player, Google maps, real-time traffic reporting, web access, communication between car, home and office	Ability to withstand extreme ambient temperature; special cooling mechanisms in the proximity of heating and air-conditioning system; small form factor to fit behind dashboard; very low power level (below 5W); fast return from deep sleep (S3) state
Digital signage	Flight information display system, outdoor advertising, way-finding, exhibitions, public installations	Rich multimedia content playback; display of a single static image in low-power state; auto display of selected contents upon sensor feedback (e.g., motion detection); intelligent data gathering; analysis of data such as video analytics; real-time performance
Special equipment for military and aerospace	Air traffic control, special devices for space stations and space missions, wearable military gears	Security; real-time performance; fast response time; extreme altitude and pressure; wearable rugged devices with Internet and wireless connectivity; auto backup and/or continuous operation
Embedded gaming	Video gaming devices, lottery, slot machines	High-end graphics and video playback; keeping attractive image on the screen in low-power state; various human interfaces and sensors; high security requirements; proper ventilation
	Scalable hardware and	Compliance with guidelines requirements for

Satellite and telecommunications	software in data-centers and base stations, throughput and power management and control	environmental condition such as National Equipment Building Systems (NEBS); Continuous low-power monitoring
Internet of Things	Smart appliances, smart watches, smart earphones, smart bowl	Very low power for control and monitoring; Internet connectivity

尽管各种低功耗平台的要求各异，但实际需要考虑因素包括：处理器面积，功耗，性能，视觉质量和设计复杂度之间的权衡。在维持低功耗并保证可用性和优雅体验的同时，需要减少不必要的功能或牺牲视觉质量。如上的方法优先考虑简单性而不是性能，以便支持特定设备或应用程序的能效要求。因此，在低功耗平台中，只有极少数的媒体使用是一件司空见惯的事情。对于通过本地和远程无线设备播放并浏览视频、视频录制、以及视频会议而言尤其重要。接下来会讨论这些用法的一些细节。

视频播放

无论是播放本地视频还是播放互联网的视频流，视频播放都是低功耗设备上最流行和最具知名度的媒体使用模式。可以用单独的应用程序或浏览器播放来自互联网的视频内容。数字视频一般以各种压缩、编码格式存在，因此视频播放涉及到视频解压缩操作。视频在显示设备上播放之前，需要先解码视频。如果视频的分辨率与显示设备的分辨率不同，则需要调整解码视频的大小以匹配显示设备的分辨率。

仅使用CPU进行视频解码和播放是一种非常复杂且耗时的操作。一般会用硬件加速获得更高的性能和更低的功耗，因此优化的硬件单元会专用于类似视频解码这种复杂且耗时的操作。在现代处理器中，使用硬件加速播放视频已成为常态。对于低功耗平台而言，硬件加速至关重要。

图7-3展示了Android系统的视频播放模型的软件堆栈。Media Extractor（媒体提取器）分流（demultiplexes）视频媒体文件中的视频数据和音频数据。分流之后的数据输出到相应的Open MAX（OMX）解码器，OMX将视频码流输出到LibMix库开始解码操作。媒体驱动器驱动硬件解码器进行实际的视频解码。解码后的视频缓冲区将发送到Android的音视频同步（AVsync）模块。AVSync比较视频缓冲区的PTS和音频时钟，并将缓冲区排队到Surfaceflinger，以便在适当的时间将缓冲区数据显示在显示器。

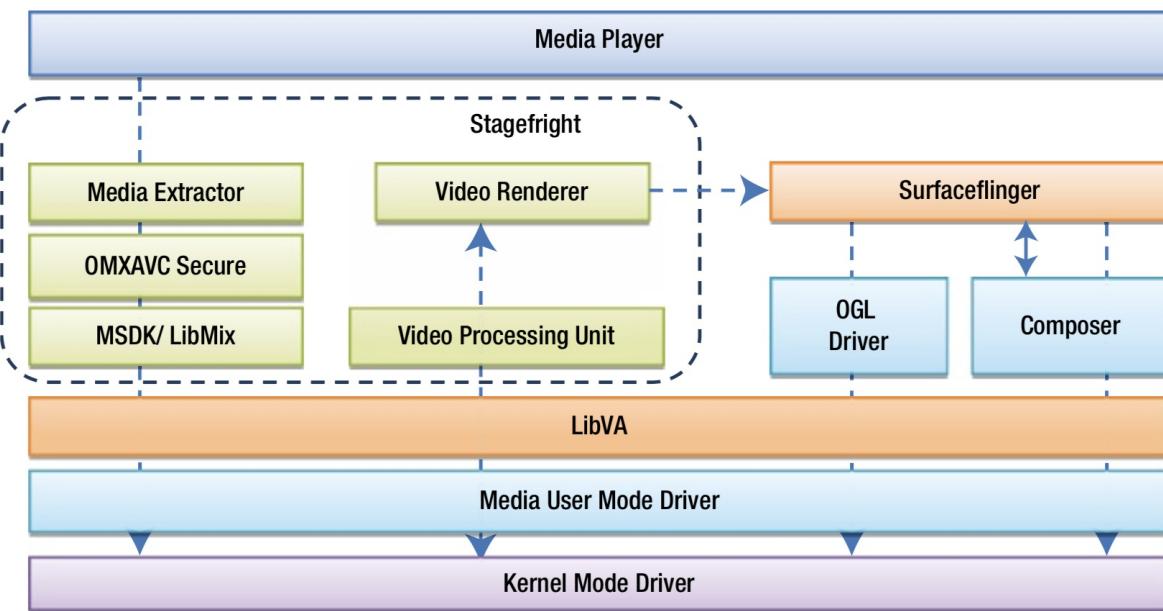


图7-3. Android系统上视频播放的软件栈

图7-4给出了视频播放中的解码过程的流程图。解码过程通常由国际标准研究小组定义，所有兼容的解码器都必须实现特定的编解码器格式。为了获得一定的性能和功耗要求，必须在各种硬件和软件级别上进行仔细的优化。尤其是在低功耗平台上，更需要进行仔细的

软件、硬件优化。优化方法包括：在专用硬件上执行重复操作，优化内存加载/存储/复制，高速缓存一致性，调度，各种硬件功能单元中的任务负载平衡，优化后处理，选择性地进入节能状态等等。

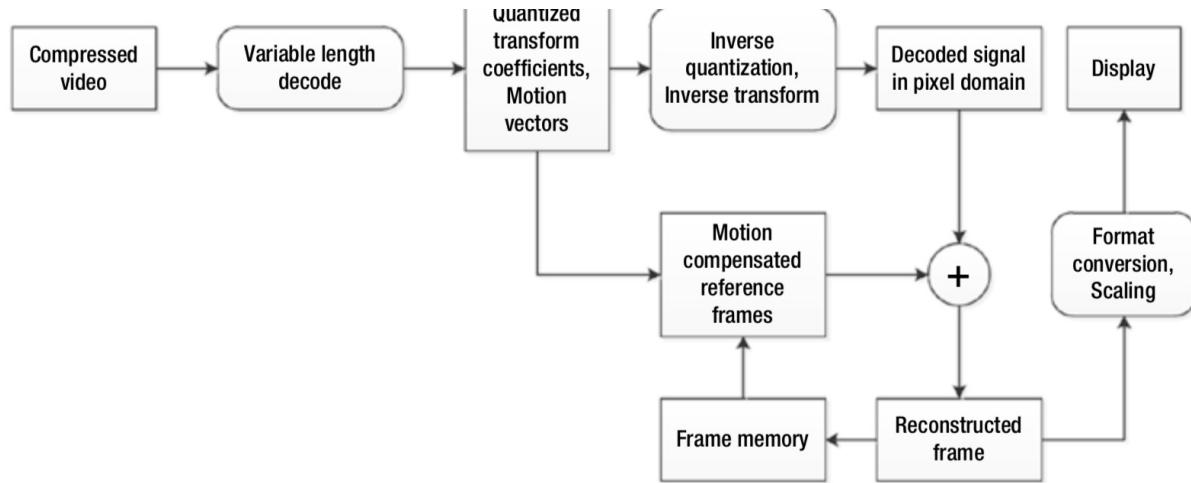
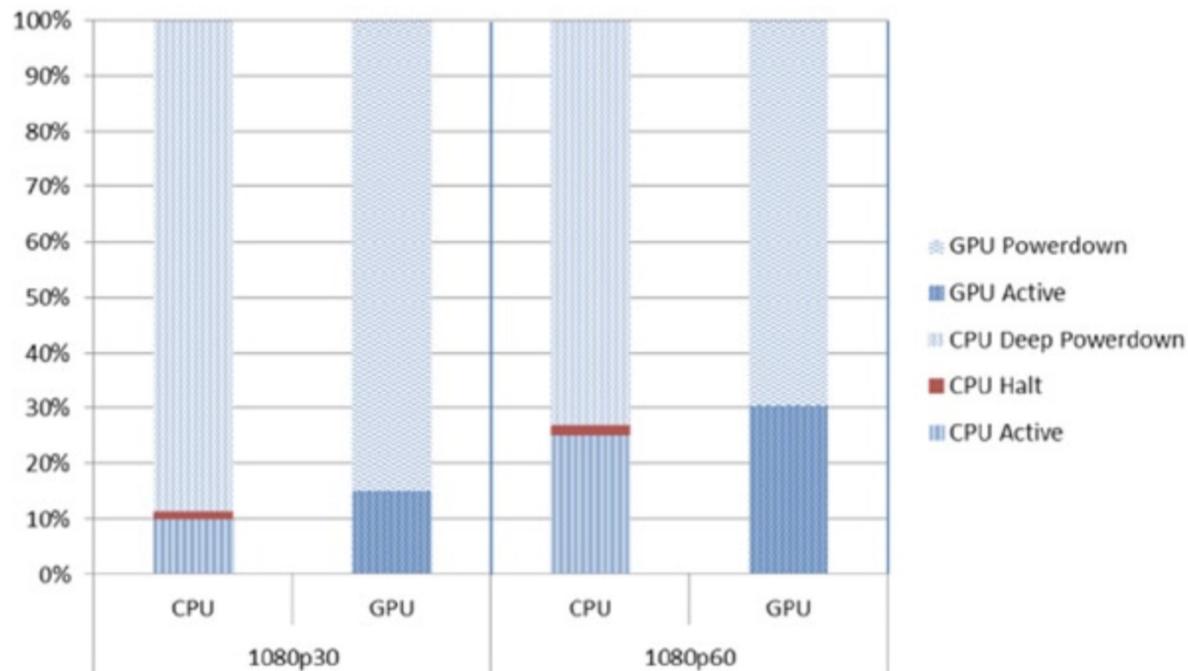


图7-4. 视频解码的流程图

图7-5给出了AVC播放1080p30和1080p60视频的性能结果文件。在基于英特尔架构的低功耗平台中，根据帧率的不同，播放1080p的视频通常需要10%~25%的CPU消耗以及15%~30%的GPU消耗。虽然显示单元消耗了平台50%的功率，但SoC的功耗为20%~30%。明显的功耗损有：稳压器 (~15%)，内存 (~5%)，平台的其它部分。

Video Playback Activity Profile



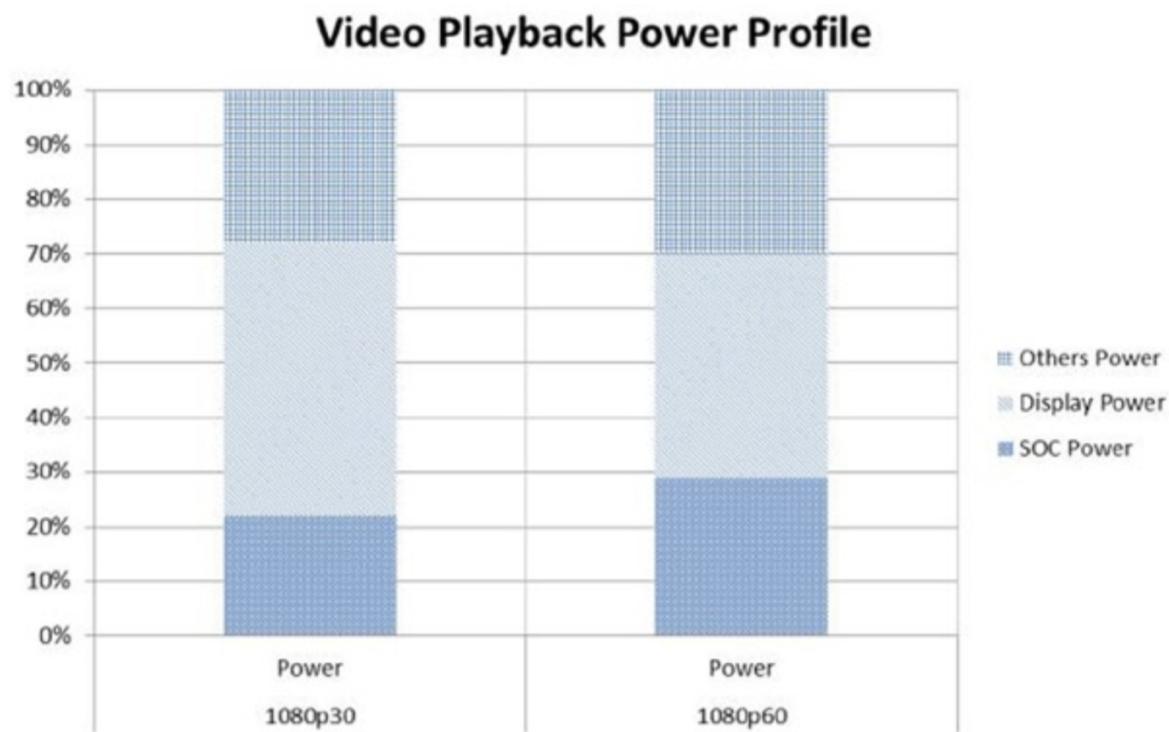


图7-5. 播放不同帧率的视频的功耗图

视频录制

在视频录制中，用户使用设备的主摄像头捕获1080p分辨率的视频，使用集成麦克风捕获音频。摄像头的输出（通常是压缩的）数据经过解码和预处理，并根据需要进行降噪和缩放。使用硬件加速编码生成的未压缩的数据。编码并合流（*multiplexed*）视频和音频流，然后将其存储在本地文件中。虽然预览模式对于视频录制而言相当普遍，但为简单起见，图7-6所示的软件堆栈中未考虑视频预览。

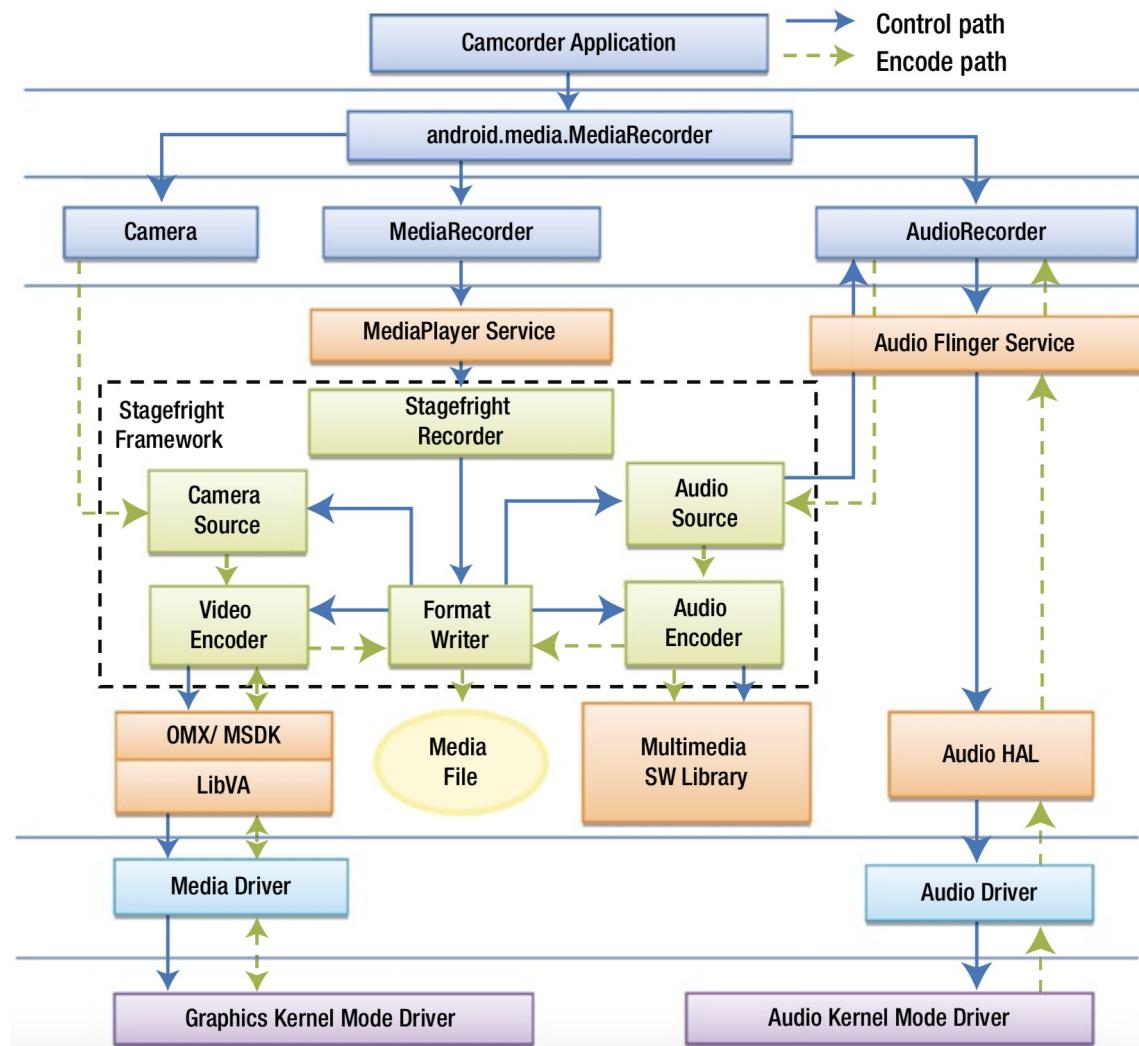
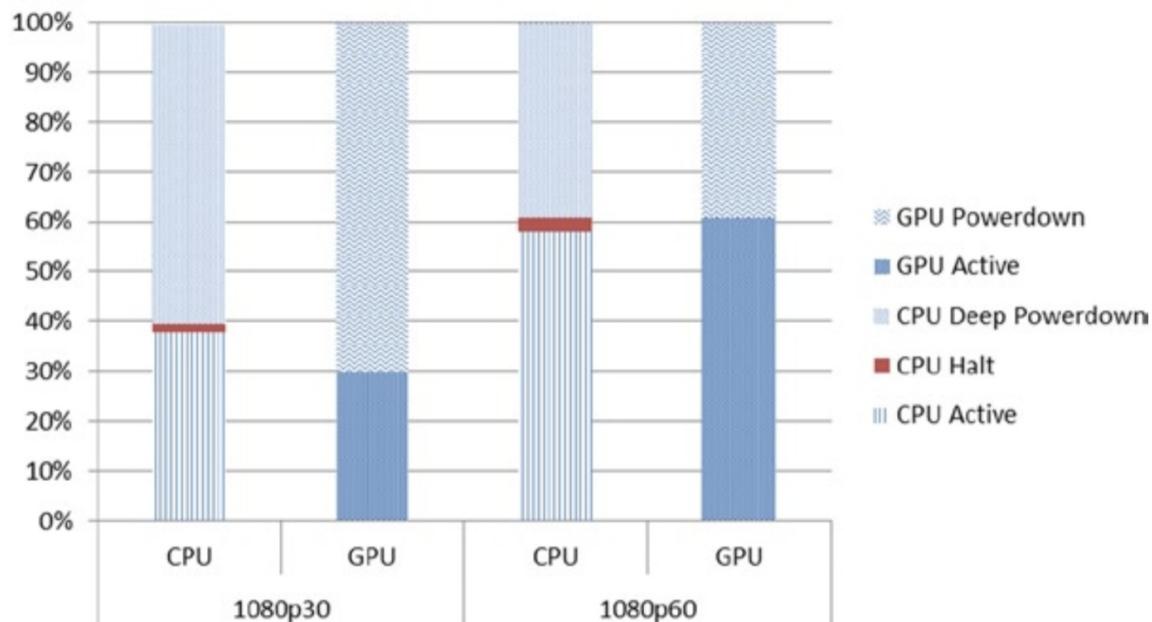


图7-6. Android系统视频录制的软件堆栈

图7-7展示了英特尔架构平台上利用AVC录制1080p30和1080p60视频的功耗信息。录制1080p视频需要40%~60%的CPU消耗以及30%~60%的GPU消耗。GPU的消耗数量取决于视频帧率。SoC和显示单元各自均消耗约1/3的功耗。和视频播放类似，已下的部分也会产生大量功耗：稳压器（~15%），内存（~7%），平台的其它部分。请注意，由于编码的工作量较大，因此视频录制对GPU的消耗要高于视频播放。

Video Recording Activity Profile



Video Recording Power Profile

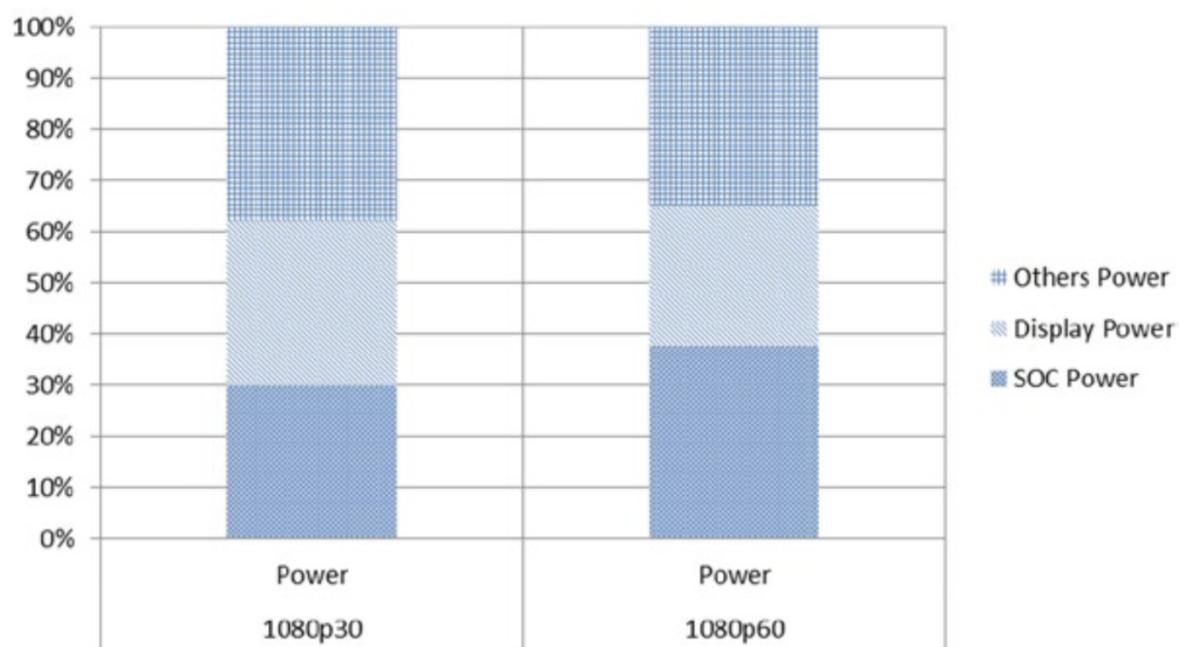


图7-7. 视频录制的功耗图

视频分发

在使用无线显示（WiDi）和Miracast进行视频传输的过程中，用设备的Wi-Fi把从本地显示器捕获的编码的屏幕内容流式的传输到远程HDTV或监视器。WiDi也支持无线传输所关联的音频内容。可以利用视频和音频编码的硬件加速功能为远程显示设备生成封装在Wi-Fi数据包中的音频-视频流，该远程设备会连接一个WiDi适配器，并且该视频流可在通过peer-to-peer链接传输。由于无线局域网（WLAN）的多角色支持，因此多个连接可能同时在无线访问点上可用，从而为WiDi对等连接以及与Internet的专用连接（同时共享该连接）提供服务频率。如下所述，多角色允许以克隆或多任务模式浏览Internet。利用WiDi的数字版权管理（DRM）协议保护视频播放或浏览Internet。第6章概述了WiDi的完整解决方案和Miracast的行业标准。

WiDi支持两种主要的使用模式：

- 克隆模式，在本地和远程显示器上都显示相同的内容。可以修改本地显示器的分辨率以匹配远程显示器的最大分辨率。另外，如果WiDi性能不足，则远程显示器的帧率可能会降低。
- 扩展模式，其中虚拟显示器可以远程传输数据流，并且内容不会显示在本地显示器上。扩展显示有两种方案：
 - 在远程显示器上显示内容，而本地显示器仅显示UI控件。
 - 允许多任务处理，其中视频显示在远程显示器上，而独立的应用程序（例如浏览器）也可以运行并在本地显示器上显示内容。

理想情况下，平台的设计应确保激活无线显示时不会降低平台的性能。

图7-8展示了英特尔架构平台的WiDi流程图。一个或多个屏幕内容被捕获、合成、缩放并转换为指定的格式以适合硬件加速的视频编码器，同时还会独立捕获并编码音频。然后，根据HDCP2协议对编码的视频和音频码流加密，并合流并打包加密的码流，进而生成MPEG-2数据包，用以准备通过Wi-Fi信道发送。

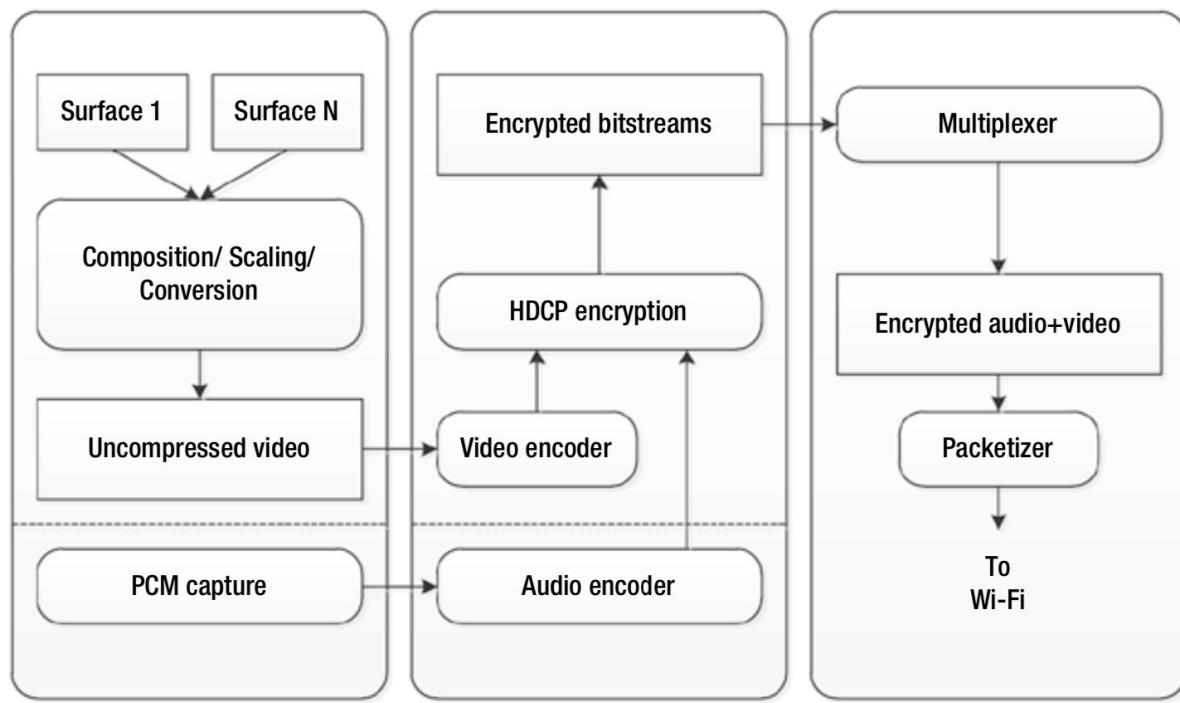


图7-8. Intel架构的无线显示流程图

图7-9展示了三种WiDi方案的活动详情（activity profile）：

- 设备空闲时的克隆模式——即本地显示屏显示的大部分是静态屏幕。
- 在扩展模式下播放视频，其中视频在远程显示器中播放。
- 扩展模式下的多任务，其中在远程显示器上播放视频时，在本地显示器上独立显示浏览器（或其他窗口）。

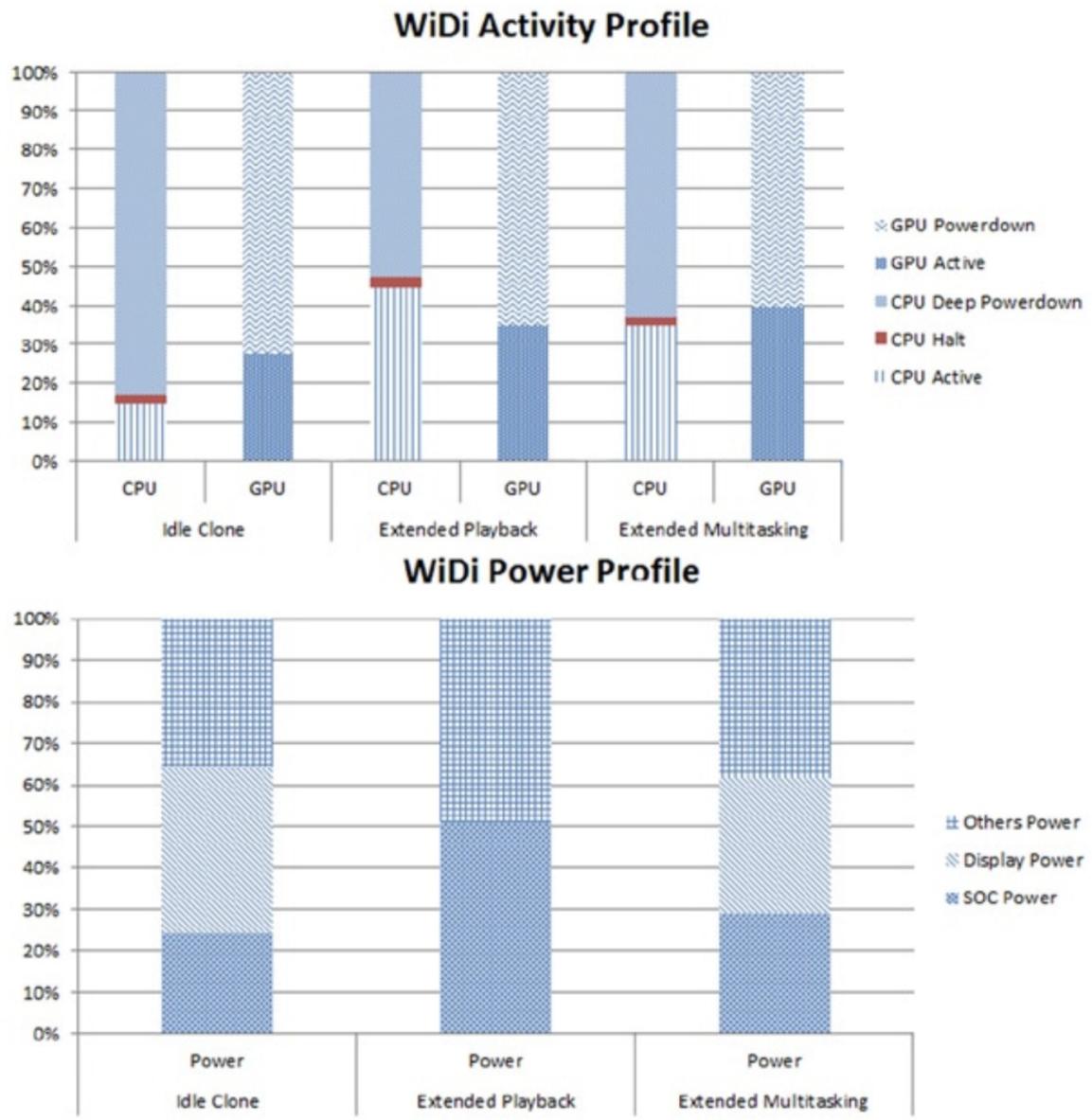


图7-9. 各种WiDi模式的活动详情

空闲克隆方案取决于显示器的刷新率，可以使用硬件加速以720p60或1080p30捕获并编码屏幕内容。编码的视频码流经过加密、与音频流合流、并在使用Wi-Fi协议传输之前分成传输数据包。硬件加速的视频也可以在合成、缩放、转换为未压缩的视频之前进行预处理。在克隆模式下播放视频时，使用硬件加速会同时进行解码，捕获和编码操作。对于扩展模式，本地显示器不显示视频，内容仅在扩展无线显示器中显示；此时没有进行合成，缩放或格式转换，仅执行解码和编码。在所有的情况下，音频通常都使用CPU编码。

在Intel体系结构，根据WiDi的情况，WiDi通常需要消耗15%~45%的CPU和30%~40%的GPU。尽管本地显示器在扩展视频模式下没有功耗，但SoC消耗了大约一半的平台功率，另一半则分配给了平台的其余部分。对于其他WiDi模式，显示器，SoC和平台的其余部分基本上均分了平台的功率。

视频电话（会议）

多方视频会议是两方视频电话或视频聊天的扩展，多方视频会议使用设备集成的摄像头来捕获视频，并基于MJPEG或其他压缩格式进行后续处理。利用设备的硬件加速功能，可以对摄像头的输出进行解压缩（如果摄像头的输出已经压缩），然后以VBR码率将其重新编码为低延迟的具有一定错误恢复功能的视频格式。这基本上是同时应用视频播放和视频录制的使用模型，并且具有以下规定：

- 编码和解码操作必须实时同步，通常会采用硬件加速。
- 摄像头的输出也应实时输入到编码单元。摄像头的输出帧率应与编码帧率保持一致。
- 从摄像头捕获到输出打包的码流的端到端的延迟应该是恒定的。
- 视频基本流应与相应的音频基本流有适当同步。

对于多方视频会议，通常在本地显示器中有一个主视频窗口和多个涉及多方的缩略图视频。图7-10给出了典型的视频会议的流程图。假设摄像头的捕获格式为压缩格式，则在对捕获的视频进行编码、解码、缩放、预处理之前，需要将其编码为合适的格式（例如具有适当的码率，帧率和其他参数的AVC格式），以便在视频的质量、延迟、功耗和压缩量之间获得良好的折中效果。同样，在显示之前，将输入的视频码流解码并组合在一起。通常都会使用硬件加速完成如上的操作。合流/分流和打包/解包通常在CPU中完成，而音频可以通过特殊的硬件单元、音频驱动程序、内核进行处理。

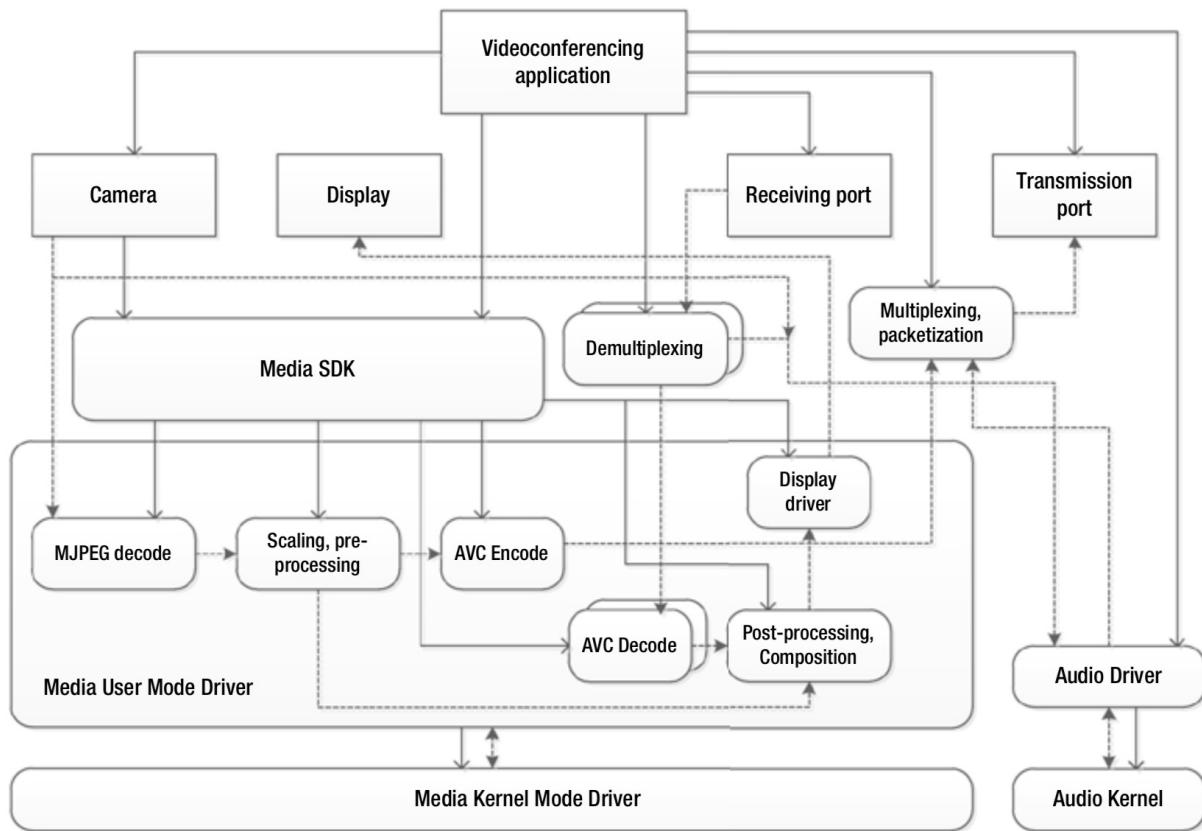


图7-10. 视频会议流程图

视频会议的活动详情类似于WiDi。在这两种情况下，都涉及同时编码和多次解码。编码是在低延迟和可变码率的情况下进行的，以适应网络带宽的变化。视频会议和WiDI也都具有实时性的要求。

低功耗系统状态

从硬件和软件的角度来看，低功耗平台设计的目标是成功应对电池的有限电量、散热、时钟速度、媒体质量、用户体验等带来的挑战，同时还需要满足日益增长的需求。为此，如第6章所述，ACPI定义了几种处理器状态，以在处理器未完全处于活动状态时来降低功耗。但是，电源管理的ACPI模型不能完全用于现代移动应用程序，尤其是在移动应用要求“始终在线”的情况下。定义新的低功耗状态来解决这些问题，但是新的定义也带来了一些问题。

简单ACPI模型的缺点

ACPI电源管理的简单模型存在如下的局限性：

- 该模型假定操作系统或电源管理器管理电源。然而，实际上却并非总是如此。某些设备（例如磁盘驱动器，CPU和监视器）可以管理自己的电源并实施超出ACPI模型可实现的电源策略。
- 设备状态只有4个是不够的。例如，D3具有两个子集。同样，如下一节所述，处于D0状态的处理器具有其他CPU（Cx）状态和性能（Px）状态。此外，设备可以执行多个独立的功能，并且每个功能可能具有不同的电源状态。
- 操作系统的发展对电源管理提出了新要求。例如，Connected Standby的概念要求系统关闭除侦听传入事件（例如电话）以外的所有活动。

如上的缺陷使得Windows在S0状态内出现了特殊的S0iX状态。这些特殊的状态用于微调待机状态。整数X越大表示延迟越高，但功耗会越低。

待机状态

Connected Standby (CS) 模仿智能手机的电源模型，以在PC上提供即时开/关的用户体验。在Windows 8以及更高版本中，Connected Standby是低功耗状态，CS在保持与Internet连接的同时使用极低的功耗。CS使得系统可以连接到适当的可用网络，从而使得应用程序无需用户干预也能够保持更新或获取通知。传统睡眠状态 (*Sleep state*) (S3) 的唤醒的等待时间在2s+，而休眠 (*Hibernate*) (S4) 可能需要无限长的时间，但是支持CS的移动设备可以在500ms以内恢复系统。

当系统进入CS时，操作系统会关闭除了保留DRAM内容所需的最低要求外的大多数硬件。但是，网络接口控制器 (*NIC, Network Interface Controller*) 仍然可以获取一点点的电量，使其可以扫描接受到的数据包并匹配特殊的唤醒模式。为了保留网络连接，NIC将根据需要唤醒操作系统。操作系统还会每隔几个小时自主唤醒以更新其DHCP。因此，即使NIC在大部分时间都处于关机状态，系统仍可以保持其第2层和第3层的连接性。

另外，可以实时唤醒系统。例如，当有Skype呼叫到达时，系统需要快速启动铃声。通过特殊的唤醒模式数据包实现实时唤醒。Skype服务器在长时间运行的TCP套接字上发送数据包。NIC硬件经过编程以匹配该数据包并唤醒OS。操作系统接收并识别Skype数据包，然后开始播放铃声。

Connected Standby具有以下四点：

- 大多数硬件处于低功耗状态。
- 当NIC需要OS干预以维持第2层连接时，NIC会唤醒OS。
- 操作系统会定期唤醒NIC以刷新其第三层连接。
- 当发生实时事件（例如，Skype来电）时，NIC会唤醒操作系统。

为Connected Standby平台设计的SoC和DRAM一般具有如下的特征：

- 空闲模式和活动模式之间的切换时间小于100ms。活动模式允许在CPU或GPU上运行代码，但可能不允许访问存储设备或其他主机控制器或外围设备。空闲模式可以是时钟门控状态或电源门控状态，但应为SoC和DRAM的最低功耗状态。
- 支持DRAM的自刷新模式，以最大程度地降低功耗。通常使用移动DRAM (LP-DDR) 或低压PC DRAM (PC-DDR3L, PC-DDR3L-RS)。
- 支持Power Engine Plug-in (PEP) 的轻量级驱动程序，PEP对SoC特定的电源依存关系进行抽象并协调设备状态和处理器空闲状态的依存关系。所有支持CS的平台都必须包含一个PEP，当SoC准备以最低功耗的空闲模式工作时，PEP应与操作系统进行通信。

如图7-11所示，Connected Standby状态下为低功耗目的标准备硬件的过程可以显示为上下颠倒的金字塔。整个SoC关机时，功耗最低，但这仅在SoC上的每个设备均关机时才会发生。因此，此状态的等待时间在Connected Standby中最高。

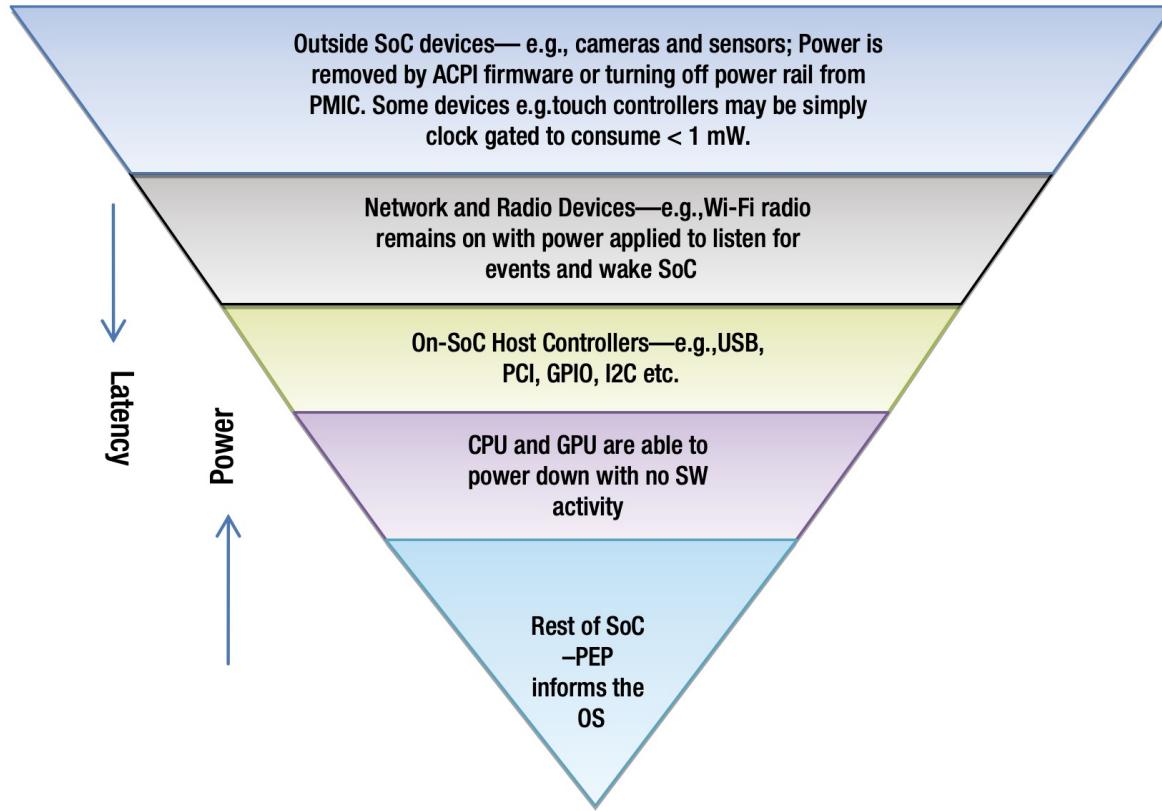


图7-11. 在Connected Standby期间准备过渡到低功耗状态

S0iX状态是最新的待机（standby）状态，包括S0i1, S0i1-Low Power Audio, S0i2和S0i3。可以在最新的英特尔平台上使用这些状态，并且所有操作系统均以启用这些状态。S0iX是特殊的待机状态，可让系统消耗很少的电量，因此它们对于Connected Standby至关重要。当按下“开/关”按钮或空闲超时后，设备进入CS模式。在CS模式下，设备消耗的功率非常低。例如，在CS模式下，基于Intel Z2760的设备消耗的功率小于100 mW，并且设备可以在此模式下停留15天以上而无需充电。图7-12显示了CS模式下的操作流程。

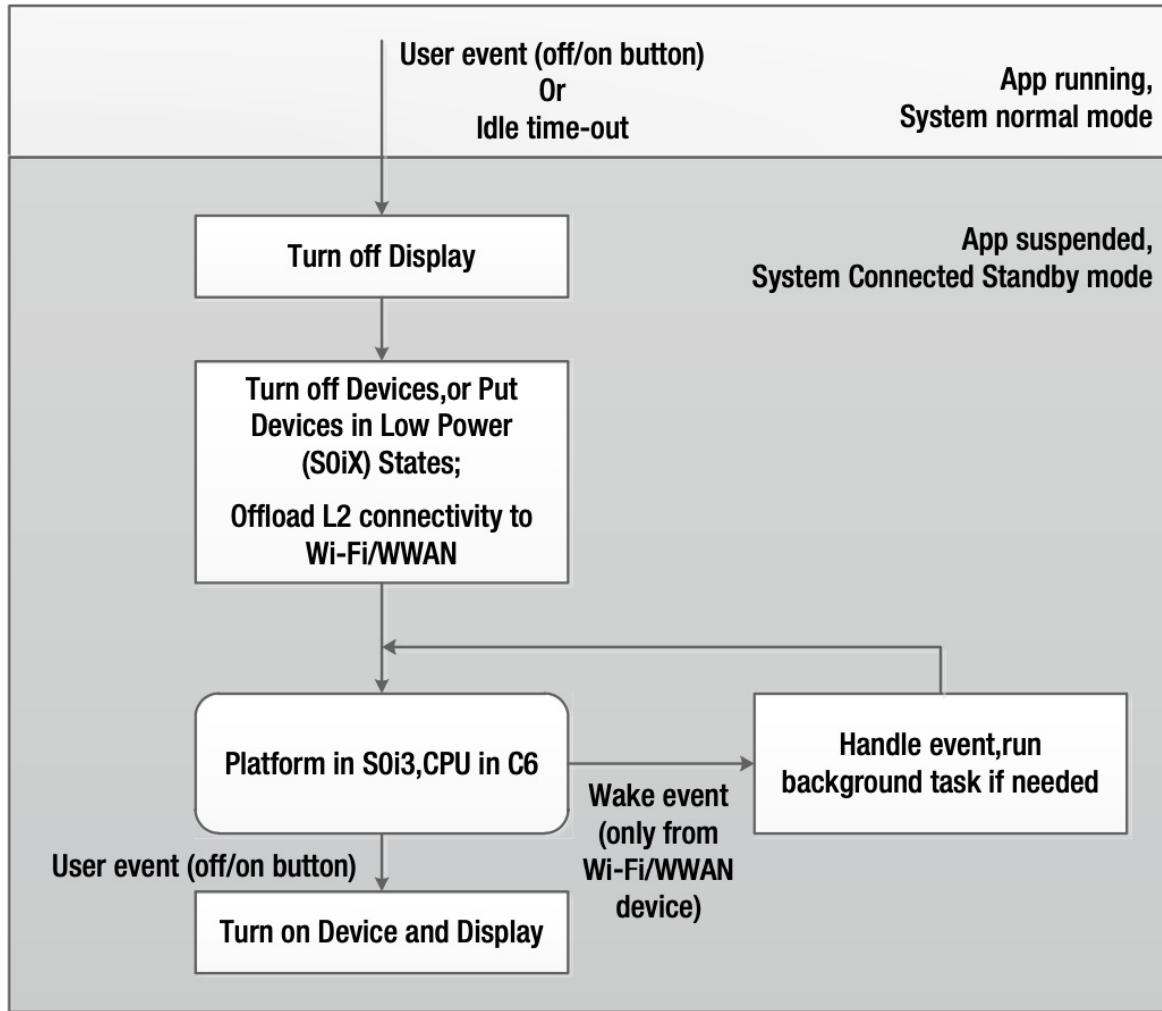


图7-12. Connected Standby的操作流程

与此形成鲜明对比的是ACPI S3状态，在该状态下，处理器在系统处于睡眠状态时将暂停所有活动，并且仅根据来自键盘、鼠标、触摸板或其他I/O设备的信号才能恢复活动。Connected Standby使用各种S0iX状态，以严格控制的方式自动暂停和恢复系统的活动，以确保低功耗并延长电池使用时间。具有45瓦时电池的典型系统在S3状态下可以达到100小时的电池寿命，而在S0i3中，电池寿命可以达到300小时。

图7-13显示了S0i3状态相对于S0和S3状态的比较。S0i3本质上兼顾了功耗和延迟的最佳性能：它比S0（活动）状态的功耗要少，所消耗的功率几乎与S3（休眠）状态的功耗相同，而唤醒延迟则类似于S0状态，以毫秒为单位。

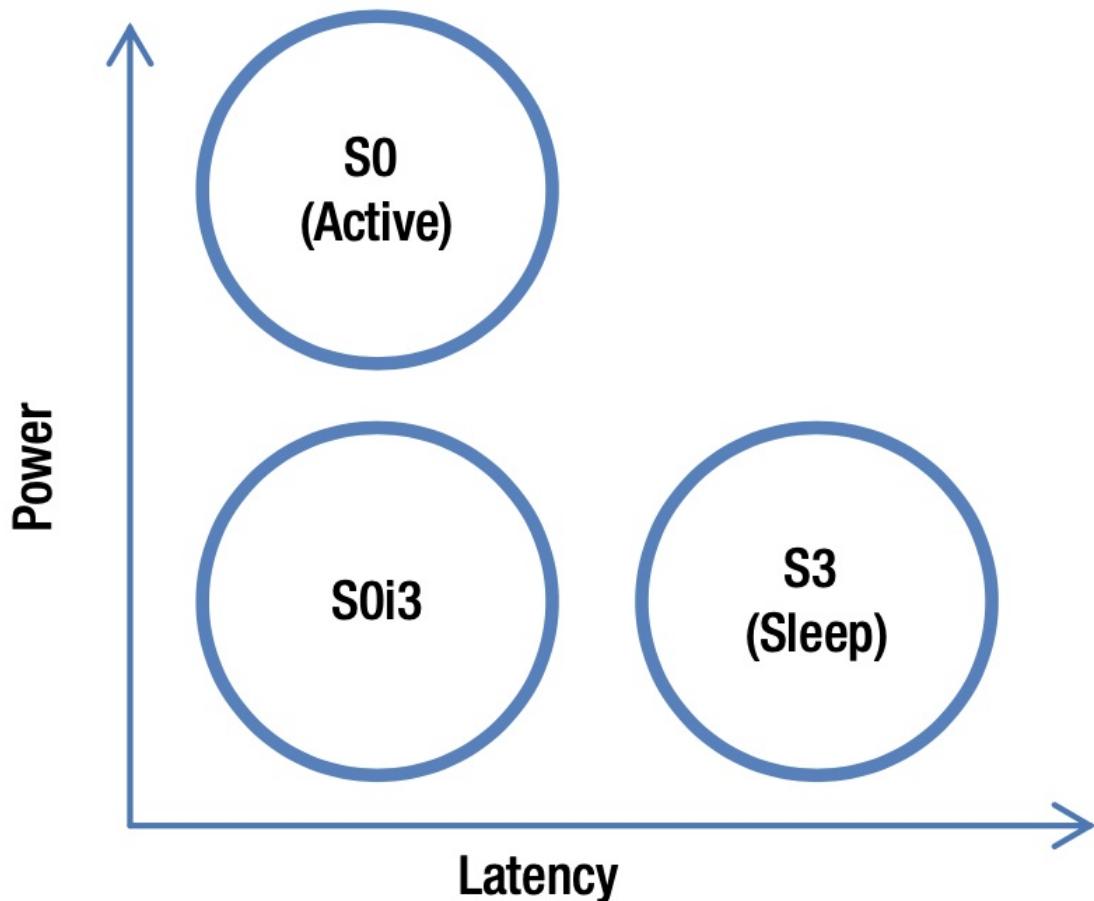


图7-13. S0i3与传统的ACPI Sleep（S3）状态的相对功率和延迟的对比关系

表7-2提供了英特尔架构平台上进入/退出每个低功耗待机状态的预估等待时间。特别是在Windows 8和Windows 8.1操作系统上，从用户体验的角度来看，这些快速的延迟可转化为在节省功耗的前提下实现的异常快的唤醒速度。

表7-2. 英特尔架构平台上进入/退出待机状态的预估时间

状态	进入延迟	退出延迟
S0i1	~200 usec	~400 usec
S0i2	~200 usec	~420 usec
S0i3	~200 usec	~3.4 msec

低功耗状态的组合

表7-3列出了典型的将系统的低功耗状态和ACPI功耗状态组合而形成的典型的低功耗处理器。第6章介绍了ACPI功耗状态，因此此处仅添加S0iX的效果，以便能够解释的更加清楚。

表7-3. 系统低功耗状态的定义

状态	描述
G0/S0/PC0	Full on. CPUs are active and are in package C0 state.
G0/S0/PC7	CPUs are in C7 state and are not executing with caches flushed; controllers can continue to access DRAM and generate interrupts; DDR can dynamically enter deep self-refresh with small wake-up latency.
G0/S0	Standby ready. CPU part of the SoC is not accessing DDR or generating interrupts, but is ready to go standby if the PMC wants to start the entry process to S0iX states.
G0/S0i1	Low-latency standby state. All DRAM and IOSF traffic is halted. PLLs are configured to be off.
G0/S0i1 with audio	Allows low-power audio playback using the low-power engine (LPE), but data transfer happens only through specific interfaces. Interrupt or DDR access request goes through the PMC. The micro-architectural state of the processor and the DRAM content are preserved.
G0/S0i2	Extended low-latency standby state. S0i2 is an extension on S0i1—it parks the last stages of the crystal oscillator and its clocks. The DRAM content is preserved.
G0/S0i3	Longer latency standby state. S0i3 is an extension on S0i2—it completely stops the crystal oscillator (which typically generates a 25 MHz clock). The micro-architectural state of the processor and the DRAM content are preserved.
G1/S3	Suspend-to-RAM (STR) state. System context is maintained on the system DRAM. All power is shut to the noncritical circuits. Memory is retained, and external clocks are shut off. However, internal clocks are operating.
G1/S4	Suspend-to-Disk (STD) state. System context is maintained on the disk. All power is shut off, except for the logic required to resume. Appears similar to S5, but may have different wake events.
G2/S5	Soft off. System context is not maintained. All power is shut off, except for the logic required to restart. Full boot is required to restart.

G3

Mechanical off.

如上的数据来源于：[Source: Data Sheet, Intel Corporation, April 2014.](#)

低功耗平台的电源管理

电源管理的主要任务由操作系统完成，例如：使空闲的CPU核停止工作；将任务、线程和进程迁移并整合到最少数量的内核，以允许其他内核空闲；对于可变的工作负载，限制在不同的CPU核之间频繁切换任务；管理功率增益与硬件或软件延迟之间的权衡；活跃CPU核之间的负载平衡……。除了操作系统根据ACPI标准提供的电源管理功能外，还有其它的用于电源管理的硬件和软件方法，它们的唯一目标就是节能最大化。接下来，我们将对其中的一些方法进行讨论。

电源管理的专用硬件

由于电源管理对低功耗平台而言至关重要，因此在大多数现代设备中，都有专用硬件单元用于电源管理任务。接下来会介绍类似的两个这样的硬件单元。

电源管理电路

随着系统复杂性的增加，系统需要各种电压水平的多种电源。仅靠电压调节器无法满足此类需求。为了获取更大的灵活性，引入了电源管理集成电路（*PMIC, power management ICs*）。PMIC是专用集成电路（或SoC设备中的系统模块），用于管理主机系统的电源。PMIC通常位于电池供电设备中，例如：手机，便携式媒体播放器。根据应用程序的空闲或活动状态，典型的PMIC的效率为80%至85%。PMIC可能具有以下一项或多项功能：

- 电池管理
- DC/DC转换
- 电压调节
- 电源选择
- 功耗排序（*power sequencing*）
- 其它功能

电源管理通常根据以下假设而工作：

- 所有平台的设备均符合ACPI。
- SOC设备是除GPIO之外的PCI设备。
- PCI设备驱动程序直接写入PMCSR寄存器以关闭/打开设备电源。PMCSR寄存器会触发PMC的中断。
- 设备驱动程序使用ACPI控制进入/退出D0/D3。

电源管理控制器

仅驱动程序和操作系统还不足以执行电源管理任务，例如保存和还原上下文或处理特殊唤醒事件。通常使用一种特殊的微控制器——电源管理控制器（*PMC, Power Management Controller*）来为处理器中的电源域加电和断电。PMC还支持传统的电源管理功能集以及一些扩展功能。例如，在英特尔第四代Atom SoC中，PMC执行各种功能，包括：

- 启动处理器并还原上下文。

- 关闭处理器电源并保存上下文。
- 处理各种睡眠状态的唤醒事件。
- 安全启动。
- 执行低功耗的音频编码（使用LPE）和通用输入输出（GPIO）控制，以便在LPE处于活动状态时能够进入S0i1。

PMC还可用于ARM11处理器的省电模式中。在该模式下，PMC确定何时将处理器置于休眠或关机模式；声明并保持内核复位引脚（*pin*）；在保持复位状态时切断处理器内核的电源，等等。

显示器电源管理

显示器是大多数低功耗平台的主要功耗组件。多种电源管理功能可以管理和优化各种平台的显示功耗。接下来将介绍英特尔平台的显示器电源管理功能。

面板自刷新

面板自刷新（*PSR, panel self-refresh*）功能通常用于嵌入式显示端口（*eDP, embedded display port*）显示器，当系统空闲但显示器打开时，SOC可以进入较低的待机状态（S0iX）。在显示器的帧缓冲区不变的前提下，PSR通过完全忽略DDR存储器的显示刷新请求来实现此目的。在这种情况下，PSR会阻止显示引擎从外部存储器获取数据并关闭显示引擎。

显示节能技术

显示节能技术（*DPST, display power-saving technology*）是一种Intel背光控制技术。在最新版本中，主机侧显示引擎可以减少27%+的面板背光功率，这也会对功耗产生线性影响。英特尔DPST子系统分析要显示的帧，并在分析的基础上更改像素的色度值，同时降低背光的亮度，以最小化可见的视觉劣化。如果当前显示的帧和下一个要显示的帧之间存在相当大的差异，则将计算新的色度值和亮度级别。

需要利用显示驱动程序才能启用DPST。DPST尚不能与PSR并行工作。如果MIPI3面板没有本地帧缓冲区，则应启用DPST并禁用PSR。通常，应该为没有本地帧缓冲器的桥接芯片启用DPST，以进一步节省功耗。

内容自适应的背光控制

液晶显示器（*LCD*）包含通过滤镜发光的背光。滤光器调制背光，并由要显示的像素值控制：像素越亮，通过的光越多。通过控制LCD滤波器中的像素阵列，可以在显示器上显示不同亮度的图像。这种控制通常基于功耗-亮度的折衷。

内容自适应背光控制（*CABC*）利用了以下事实：感知的亮度取决滤镜调制的背光。一个黑色的帧暗看起来很暗，因为滤镜不允许太多的光通过。但是，通过使用调光器背光并控制滤镜以允许更多的光通过，可以达到相同的效果。换句话说，背光变暗，并且提高帧的亮度，也可以达到相同的感知亮度。

降低背光灯的亮度会降低可用的动态范围。CABC可能会裁剪需要更高动态范围的像素，从而产生冲洗效果。同样，目标背光亮度可能会在帧与帧之间变化很大；如此大量地改变背光可能会导致闪烁。因此，必须在基于CABC的节能与视觉质量之间进行权衡。为了获得最佳性能，通常对每个颜色分量进行单独的直方图分析，以确定所需的亮度。然后，选择其中的最大值作为目标背光。这样可以确保准确显示所有颜色。

CABC引擎位于显示桥或面板内部。CABC可以随内容变化动态地控制背光，此功能通过降低背光功率来节省功耗。CABC算法处理并分析当前帧，以更新下一帧的背光。CABC可以与PSR并行工作。当播放更高FPS的视频时，CABC的省电效果会更明显。

环境光传感器

环境光传感器（ALS, *Ambient Light Sensor*）的背光节能功能可根据周围的光线调节背光。当环境光线较少时，可节省更多电量（最多30%）；而当环境光线较多时，则节省的电量会少一点（节省20%）。显示电源管理通常在面板或桥接器内部使用ALS。

低功耗平台的思考

从图7-1的功耗-频率关系图很容易理解，对于给定的功耗预算，存在一个最佳频率。使用此基本概念，需要牢记一些软件和体系结构方面的考虑，尤其是从电量受限的平台的角度来看。

软件设计

对于资源稀缺的可穿戴平台和低功耗的移动平台，很多重要的想法需要从应用程序设计的角度进行评估。毫无疑问，新的应用程序会考虑这些因素。但是，也可以修改现有的应用程序从而使其从中受益。

鉴于大多数的低功耗设备提供了各种省电机制，因此应用程序应适当利用这些省电机制。例如，允许网络设备在较长时间内处于较低功耗的状态，从而避免不必要的网络流量。但是，接下来会讲一些额外的省电建议。

智能电量意识

低功耗设计通常使用诸如电压缩放或频率缩放之类的技术来权衡性能以节省功耗，从而降低电压或频率以降低动态功耗。但是，必须提供最低电压电平才能使电路工作。此外，降低电源电压会增加电路延迟，从而限制了处理器的工作频率；延迟不能太大以至于无法满足特定工作频率下的内部时序。因此，如Intel (R) Speed Step (TM) 电压缩放技术所证明的那样，必须采取一种谨慎的方法来确定适当的电压缩放量。

不幸的是，现代处理器对系统性能的要求太高，无法使电压缩放成为一种有吸引力的节能解决方案。而且，处理器频率并不是影响复杂多媒体应用的功耗-性能 (*power-performance*) 的唯一因素。内存访问延迟和内存带宽也是重要因素。此外，多媒体应用具有实时的截止期限，这进一步限制了电压缩放的用处。因此，在实际系统中，需要沿电压-频率曲线动态调节电压和频率。

电压和频率的动态调整通常应由操作系统来完成。但是，操作系统不了解预期的工作量，尤其是在涉及突发多媒体应用程序时。而且，这样的工作负载不容易以合理的准确性进行预测，并且基于时间间隔的调度程序是不够的。操作系统充其量可以使用一些试探法更改处理器频率并尝试控制功耗。但是，这种试探的方案不适用于突发工作负载，例如视频解码，编码或处理。

但是，如果应用程序关注电量，就可以动态地、更好地预测实际的工作负载，进而将典型的视频解码器的功耗降低30%以上。应用程序的功耗在很大程度上取决于指令和存储器所需的时钟周期数。如果应用程序可以将未来需要的信息提供给OS，则OS可以执行更高效的任务调度，并且不需要进行不置信的预测。

如上的应用程序和OS联合起来降低功耗的方式对于资源匮乏的移动设备而言特别有用。应用程序必须了解其处理的任务的要求，尤其是应用程序处理的当前任务所需的时钟数，并且必须将这些信息通知给OS。然后，OS可以利用周期计数以及功耗-频率曲线来调节

并达到可以满足应用截止期限的最小频率，从而实现最佳功耗。

例如，对于H.263解码，编码帧中使用的位数与解码时间成正比。根据Pouwelse等人的报告，利用每个帧中使用的位数的提示，应用程序可以确定视频的预期频率要求，并节省大约25%的功耗。

通常，会有省电的固件逻辑来控制主机系统的功耗。固件必须依靠测量来确定功耗节省的策略。然而，固件又不了解应用程序正在尝试做的工作，因此无法优化功耗。另一方面，具备功耗意识的智能应用通常可以利用对任务的了解以及内存使用模式将相关信息发送给固件，以帮助固件进一步节省功耗。

在一个类似的性能-省电的例子中，Steigerwald等人提出了一个在线事务处理应用程序，该应用程序监视事务处理的性能计数器以确定性能影响。然后将性能信息提供给中间件，中间件学习并更新系统上的最佳功耗限制策略。最后，电源控制器固件来限制系统的功耗。

对质量的要求

尽管会为了节省功耗而牺牲性能和视觉质量，但是大多数现代游戏和媒体应用程序的用户对质量的期望都很高。虽然CPU可以以更低频率运行并降低功耗，但对于各种游戏和视频编解码器应用是不可接受的，因为降低CPU频率就意味着无法在规定的时间内处理完响应的任务，进而导致出现卡顿，丢帧等影响体验的行为。但是，如果在设计和优化软件时仔细考虑这些因素的话就可以在满足质量要求的同时节省功耗。

硬件加速模

最新的移动手持设备的低功耗处理器带有各种能够执行特定任务的硬件模块，例如：相机图像信号处理，视频解码。与通用CPU执行任务相比，这些硬件模块会对任务进行优化以降低功耗。应用程序应利用这些硬件模块的优势，并尽可能减少工作量，以减轻计算负担并节省功耗。

高效能的用户接口

从前面描述的各种使用模型中可以明显看出，显示器是最耗电的单元。但是，可以以节能的方式设计图形用户界面（*GUI*），特别是对于LCD或OLED的显示器。对于这些显示器而言，不同的颜色、颜色模型、颜色序列，其消耗的电量也不同。因此，可以使用低功耗的配色方案降低功耗。例如，通常在PDA中使用的TFT（*thin film transistor*）LCD在黑色时比白色消耗更少的电量。

高效节能GUI设计的另一种方法是通过使用更少和更大的按钮改善用户的交互的延迟，以提供更好的用户便利性。

代码密度和内存占用量

在智能手机等低功耗的移动平台上，较差的代码密度意味着在处理器上运行所需的固件和驱动程序的代码太大。与优化固件相比，设备制造商必须使用更多的RAM和闪存来存储固件。因此，使用更多的存储器会导致更高的成本并会增大电池的耗电量。

由于内存成本在整个设计的成本和功耗中占主导地位，因此固件优化和代码密度可以发挥很大的作用。编译器级别的优化以及可能的指令集体系结构（*ISA, instruction set architecture*）可以实现密度更高的固件二进制文件，从而有助于降低功耗。

减少二进制文件的内存占用量的另一种方法是在写入内存时执行无损压缩，并在从内存读回时进行相应的解压缩。但是，节电还取决于工作负载的性质。由于媒体工作负载具有良好的局部性，因此将内存压缩用于某些媒体使用可能会有所帮助。

优化数据传输

专为低功耗设备设计的现代移动应用程序通常会利用Internet来处理云端的大量数据，在设备上则仅维护一个瘦客户端。尽管这种方法可以提高性能，但大量的数据传输意味着较高的功耗。利用要传输的数据的模式，可以减少数据带宽的使用，从而降低数据传输带来的功耗。此外，应用程序可以利用新的S0iX状态进行通知，从而允许处理器更频繁地进入到低功耗状态。

并行化和批处理

任务调度是多任务环境的一个挑战。长时间的任务序列化不会为处理器提供进入低功耗状态的机会。任务的并行化，流水线处理和批处理可以使得处理器有机会进入到低功耗状态，从而降低功耗。如果任务没有特定的处理速度，则以最高频率运行处理器以尽快完成任务，然后让处理器在尽可能长的时间内处于闲置状态，这样可能会更有效。

体系结构的思考

即使对于完全不同的硬件架构，例如Qualcomm 45nm Serra SoC和Intel 22nm第四代Atom SoC，在设计低功耗架构时所考虑的因素也非常相似。尽管设计各种系统组件以降低功耗很重要，但优化系统设计并对其进行管理以降低功耗也很重要。特别地，用于节能的架构思想包括本节要讲到的以下的内容。

组合系统组件

将处理器和平台控制器中枢（*PCH, platform controller hub*）集成在同一芯片上可降低功耗、占用面积和成本，还可以简化固件接口。利用由内核启动的、在适当的上下文中进行优化的运行时设备电源管理，可以实现低功耗。

优化硬件和软件的交互

因为专业化时代更要求降低计算规模并降低功耗。因此在专业化时代，驱动程序与操作系统或者硬件版本与操作系统之间的代码共享已变得不那么重要。在专业化时代，硬件命令更类似于API命令，并且命令转换会更少。以下的想法体现了这一概念。

负载从通用硬件转移到专用硬件

将任务从通用功能硬件转移到固定功能硬件可以节省功耗，因为可以将更少数量的门电路用于特殊用途，从而减少了切换门电路的动态功耗。这种技术对非可编程任务特别有用，例如设置、剪辑、坐标计算、某些媒体处理等等。

CPU和GPU的电源共享

为了在平台给定的能力内最大化其性能，诸如Intel Burst Technology 2.0之类的技术会自动允许处理器内核在电量低于某个特定水平、特定温度下以低于指定的基本频率而运行。CPU和GPU均支持这种Turbo模式和电量共享。可以动态调整高性能突发操作点。

使用低功耗的CPU核

处理器的所有部分（核心，非核心和图形）都需要优化以降低功耗。GPU是重要的动力源，导航栏、状态栏、应用程序UI、视频等组成的任务应尽可能多地交给显示控制器，以保证最低的设备活动和内存带宽。

另外，功耗很大程度上取决于实际的使用情况，因此应该在考虑实际使用的情况下优化处理器功耗。也就是说，低功耗的解决方案会受到最差应用的限制，应该考虑有功功率（*active power*）和泄漏功率（*leakage power*）的优化和管理。

为了降低泄漏功率，要知道尽管PMIC可以调节功耗并使睡眠期间的全芯片功耗关闭，但这种功耗关闭需要将数据保存到内存中并需要重新启动处理器的所有部分才能进行恢复。那么，理想的影响是节省了电量，即一段时间内节省的泄漏电量大于用于保存和恢复的电量开销。

降低RAM/ROM外设和核心阵列的功耗

核心阵列中的所有内存都需要进行泄漏控制（*leakage control*）。还应该考虑外围电路的功耗降低，因为可以从RAM/ROM外围设备来降低功耗。

休眠期间降低VDD来管理电流泄露

在睡眠期间使整个芯片的 V_{DD} 最小化是有益的，尤其是在较短的睡眠周期内。睡眠期保留了所有存储器和寄存器状态，因此具有软件开销小和恢复速度快的优点。利用电压最小化控制泄漏是比用软件进行复杂管理更有效的方法。但是，对于存在Connected Standby需求的情况时，则无法降低电压。

对大型高密度存储器采用独立的内存控制

一种常见的省电技术是在应用程序无法访问的存储体上使用电源门控。利用门控控制非必须的存储区的时钟/数据访问，可以降低待机/主动泄漏。但是，这需要在软件和固件中进行适当的电源管理，以便正确识别并适当控制潜在的空闲存储体。同样，电源门控并非适用于所有模块。需要对动态功耗和泄漏功耗所节省的成本进行仔细的分析比较。

时钟树优化

时钟树功耗通常是整个互连内的最大功耗。就高通Serra而言，时钟树消耗了30%~40%的整体功耗。由于时钟架构对功耗的影响很大，因此应仔细计划时钟树的使用和频率。尤其需要考虑锁相环（*PLL, phase-locked loops*）的数量，独立的时钟域控制，频率和门控，同步等。

时钟域分区

由于可以独立关闭时钟，因此将时钟划分为不同的钟域可以更好地管理各种时钟。例如，可以将控制I/O和CPU的时钟划分为各自独立的时钟域。

独立的时钟频率域

尽管异步域更灵活，但是跨时钟域会增加延迟。另一方面，同步时钟域可以在低性能模式下节省功耗，但在最坏的使用情况下，它们会消耗更多的功耗。

更细粒度的时钟门控

节省有功功率 (*activity power*) 的最有效的方法之一是在细粒度上自动和手动调整时钟门控。硬件或软件都可以控制时钟门控，但是每种方式都有成本。对时钟门控百分比和时钟门控效率的分析可以进一步发现优化点。

功率岛

功率岛 (*power islands*) 的概念可以用房屋的例子来说明。如果整个房屋中所有灯泡的电源开关只有一个，无论何时，即便房间中只有一个人，而他只需要一盏灯，打开电源开关时也必须消耗更多的能量。相反，如果有单独的开关来控制每一个灯泡，则仅打开所需要的灯泡的开关才能实现节能。类似地，在典型的计算系统中，定义20~25个独立的功率岛以实现节能。

不同模式采用独立的电压缩放

对于不同的功率岛，可以分别针对活动模式和睡眠模式进行电压缩放。这意味着降低了频率并因此节省了功耗。

覆盖多电源域的电源门控

PMIC可以控制多个电源域，从而产生更好的电源控制和效率。PMIC还具有独立的电压缩放和功率折叠 (*power-collapsing*) 功能。但是，电源域的数量会受到如下因素的限制：电源域网络 (*PDN, power domain network*) 阻抗增加，电源开关数量增加将导致IR降压 (*IR drop*) 增加，物料清单 (*BOM, bill of materials*) 增加，域边界必须的电平转换器和重新同步。使用小型、快速、高效的芯片调节器动态控制各个域的功耗是最佳选择。

当未对功率岛进行功耗门控时，应将所有未使用的外围设备作为功率岛的一部分进行时钟门控。另一方面，当对功率岛进行功耗门控时，所有相关外围设备都处于D0i3（电源电压可能仍处于打开状态）。

提供功耗感知的仿真和验证

要针对低功耗优化系统，请使用能感知功耗的仿真工具和验证方法来检查：进入和退出低功耗状态，验证钳位极性，验证电源域等。

在电量和时间方面做好权衡

定制设计流程和电路可以为使得某些任务更节能。但是，定制化的设计需要更多的时间和精力。需要仔细的选择定制和优化的区域，以便获得最大收益。良好的定制候选对象包括时钟树，时钟分频器，内存和知识产权单元（IP，这是创新设计的常用术语）等等。最好将自定义项移至IP块，并使用自动方法来插入，验证和优化该IP块。这样，可以将特定的改进和问题轻松归因于适当的功能块。

低功耗解决方案的比较分析

设计决策通常很复杂，尤其是在涉及许多折衷的情况下。要判断低功耗解决方案的性能如何，请使用类似的解决方案进行比较分析。通常，此方法会揭示每种解决方案的优点和缺点。

低功耗平台的电量优化

在低功耗平台上，多种功耗优化方式通常同时发挥作用。本书中，我们使用视频播放应用程序来说明各种电量优化方法。根据应用程序的要求和视频参数，每种功耗优化方式对总电量的节省具有不同作用。

快速执行然后关闭

在硬件加速的视频播放中，GPU通过硬件解码、缩放、后处理来完成繁重的工作。同时，CPU核执行OS进程，媒体框架，媒体应用程序，音频处理和内容保护任务。I/O子系统执行磁盘访问和通信，而Uncore¹则运行内存接口。在典型的平台上，这些子系统按顺序执行各种任务。一种明显的优化是利用任务的重叠并对任务并行化，以便硬件资源可以同时运行，从而在闲置之前在短时间内充分利用I/O。在这种情况下，优化重点在于减少活动的驻留时间并在资源空闲时实现节能。

¹. 新推出的Nehalem架构把处理器按CORE（直译过来是核心）和Uncore（直译是非核心）两部分。CORE部分当然就是负责主要的运算。Uncore部分包括了共享L3、内存控制器、QPI几个部分。 ↪

Activity调度

适当地调度软件和硬件的活动，并明智地利用内存带宽非常重要。在视频播放中，任务主要分为由CPU和GPU处理，任务可分为三大类：音频任务，视频任务和相关的中断。

音频活动是周期性的，其周期为1ms~10ms。音频由CPU线程进行音频解码和后处理任务，并且还涉及到用于DMA操作的相关音频缓冲区。除了音频DMA的周期行为外，DMA活动还涉及来自I/O设备的Wi-Fi和存储流量进入内存，这些内存本质上是突发性的。

视频任务的部分任务由CPU处理，CPU执行音频视频分流，而GPU执行解码和后处理任务。可以并行化CPU和GPU的任务调度以降低总体功耗。

CPU从低功耗状态唤醒到运行状态的次数以及这种状态转换所需的能量会影响CPU的功耗。因此一个有功耗意识的应用程序可以通过避免这种转换而节省电量。避免CPU状态转换使用维护执行序列的中断，而不是基于计时器的轮询结构，并适当地调度它们来实现此目的。通过将周期出现的音频DMA分开安排，甚至将音频转移到专用音频处理硬件，可以实现进一步的优化。

减少唤醒次数

由于传入网络数据包的突发性，播放视频流要求通信设备在访问内存时具有平滑缓冲区。智能通信设备可以通过组合中断，使用可编程刷新阈值来减少CPU唤醒次数。

突发模式

视频播放是逐帧进行的，而帧处理时间在15ms~30 ms，具体处理时间取决于帧率。视频播放的性质并不能使得CPU可以再如此短的时间内进入低功耗状态并再次返回到活动状态。为了克服此限制，可以创建多个帧的流水线，以使相关的音频正确的解耦并同步，从而使CPU能够积极利用其低功耗状态。尽管此方法需要大量的修改软件，但仍应针对低功耗设备进行探索。

完善CPU和GPU的并行化

当GPU完成解码和其它处理任务时，CPU剩下的主要任务是准备硬件加速请求，以进行解码，渲染和呈现视频帧。这些任务是独立的，因此可以在单独的硬件线程中实现，这些线程可以并行调度并执行。通过并行化执行线程，CPU可以长时间处于深度空闲状态，从而节省电量。

显存带宽优化

GPU执行视频解码和后期处理。在整个视频播放应用程序中，解码和后期处理需要最大的内存带宽。但是内存带宽随内容和显示分辨率以及每帧上完成的视频处理量而定。用blit方法显示帧（即直接将帧绘制到显示窗口上）需要将帧复制两次：一次复制到合成目标表面，一次复制到渲染表面。通过使用覆盖方法可以避免这些昂贵的复制操作，内核可以利用该方法将帧直接渲染到覆盖表面。

显示功耗优化

在视频播放应用中，显示会消耗三分之一以上的电量。有很多方法可以解决此问题，其中大多数方法已在前面介绍过。此外，媒体播放应用程序可以受益于帧缓冲区压缩和刷新率的降低。

帧缓冲区压缩不会直接影响视频窗口，但对于不需要更新主平面的全屏播放，则只需要更新覆盖平面，从而节省了内存带宽，并降低了功耗。另一方面，降低刷新率将直接减少内存带宽。例如，从60 Hz降低到40 Hz会使内存带宽减少33%。由于显示电路的利用率较低，这不仅会降低存储功率，而且还会降低整体显示功耗。但是，因视频内容而异，与完全刷新率相比，视频的质量和用户体验可能会较差。

存储功耗优化

频繁访问存储会导致过多的功耗，而减少这种访问则会使存储设备更快地进入低功耗状态，并有助于节省总体功耗。但是，与CPU不同，存储设备通常需要先闲置1s以上，然后才能转换为低功耗状态。

由于对媒体播放存储访问的要求约为10ms，因此存储设备在媒体播放期间不可能有机会进入睡眠状态。具有功耗意识的媒体播放应用程序可以预先缓冲10s左右的视频数据，从而使固态存储设备进入低功耗状态。但是，基于硬盘驱动器的存储设备速度较慢，因此需要数分钟的预缓冲才能实现有意义的节能。

低功耗的度量

通常，按照与第6章中所述相同的方法测量低功耗。但是，可能需要对几个功耗信号进行精确的测量和分析，才能确定对特定功耗硬件单元的影响。

电源的处理器信号

在典型的低功耗英特尔架构平台（如英特尔凌动Z2760）中，为电源接口定义的处理器信号如表7-4所示。可以适当地测量这些信号来完成功耗分析。

表7-4. 电源接口的重要处理器信号

信号	描述
V_{CC}	Processor core supply voltage: power supply is required for processor cycles.
V_{NN}	North Complex logic and graphics supply voltage.
V_{CCP}	Supply voltage for CMOS Direct Media Interface (cDMI), CMOS Digital Video Output (cDVO), legacy interface, JTAG, resistor compensation, and power gating. This is needed for most bus accesses, and cannot be connected to $V_{CCPAOAC}$ during Standby or Self-Refresh states.
V_{CCPDDR}	Double data rate (DDR) DLL and logic supply voltage. This is required for memory bus accesses. It needs a separate rail with noise isolation.
$V_{CCPAOAC}$	JTAG, C6 SRAM supply voltage. The processor needs to be in Active or Standby mode to support always on, always connected (AOAC) state.
LVD_VBG	LVDS band gap supply voltage: needed for Low Voltage Differential Signal (LVDS) display.
V_{CCA}	Host Phase Lock Loop (HPLL), analog PLL, and thermal sensor supply voltage.
V_{CCA180}	LVDS analog supply voltage: needed for LVDS display. Requires a separate rail with noise isolation.
V_{CCD180}	LVDS I/O supply voltage: needed for LVDS display.
$V_{CC180SR}$	Second generation double data rate (DDR2) self-refresh supply voltage. Powered during Active, Standby, and Self-Refresh states.
V_{CC180}	DDR2 I/O supply voltage. This is required for memory bus accesses, and cannot be connected to $V_{CC180SR}$ during Standby or Self-Refresh states.
V_{MM}	I/O supply voltage.

V_{SS}

Ground pin.

媒体应用的功耗指标

许多媒体应用程序都具有相同的特征，例如：实时需求，突发数据处理，数据独立性，可并行性。因此，在媒体应用程序中，重要的是测量和跟踪几个与电量有关的指标，以了解系统的行为并找到优化方案。这些指标包括SoC，显示器，稳压器和内存功耗（占全部平台功耗的百分比），CPU内核和CPU的C状态驻留时间，GPU活动和渲染缓存（RC）状态驻留时间，内存带宽等。

由于某些媒体应用程序可能受计算约束，内存约束，I/O约束或其他限制（例如实时期限），因此确定这些因素对功耗的影响可以更好地了解瓶颈和折衷方案。

了解视频的各种参数对功耗的影响非常重要，例如：视频分辨率，码率，帧率和其他参数。同时，了解系统的各种参数对功耗的影响也非常 important，例如：系统频率，散热设计功耗，内存大小，操作系统调度和功耗策略，显示分辨率和显示接口。分析和了解可用的选择可能会揭示如何优化功耗和性能。

总结

本章介绍了低功耗设备与应用程序性能的更高要求的结合以及实现这种低功耗使用的各种挑战。由于工艺技术的缩减以及电压和频率的缩减会降低功耗，因此这些技术无法实现最新的低功耗设计目标。从媒体使用的角度分析常见的低功耗方案表明，在考虑整个系统的同时，还需要采取更积极的节能措施。

为此，本章讨论了各种电源管理和优化方法。还介绍了低功耗测量技术。第6章和第7章在一起提供了一个很好的平台，可以理解为在增加动态范围以进行频率调谐与增加静态功耗之间的权衡，必须仔细平衡这些因素。在下一章中，将从媒体应用程序的角度来看一下功耗和性能之间的折衷。

性能，电量以及质量的权衡

在资源有限的情况下寻求解决方案时，必须要了解正确的权衡方法才能达到目标。通常为了达到某一方面目标往往需要舍弃另一方面效果。但是最终，根据解决方案的目标、优先级和容差，必然会在充分利用可用资源和获得最好的效果之间达到某个合适的平衡点。

以视频压缩为例，采取的措施往往是想要获取最好的视频质量，但同时也想使用最少字节数以获得最好的性能表现以及最低的电量消耗。对一个整体的解决方案而言，成功的标准还包括在硬件基础、软件基础和平衡了灵活性、可扩展性、可编程、易用以及成本的混合编码系统中的折中选择等。因此，对视频编码解决方案的用户来说，也需要在成本、适应性、可扩展性、编码效率、性能、耗电量及质量等指标中做出选择来获得一个最适合他们的视频编码解决方案。

权衡分析在现实生活的许多场景中都有应用。掌握这些权衡分析方法，尤其是就性能、功耗和质量而言的分析方法，对于建筑师、开发者、验证人员和技术营销人员来说是一种非常有价值的能力，同时，对于技术审核人员，采购人员以及编码解决方案的最终用户也会有所帮助。通过评估编码器的优缺点做出合适的产品决策、比较两个编码器的实际指标、以及通过合适的编码参数调整来优化编码器都是这种分析可提供的决策点。此外，当新功能添加到现有编码器时，此类分析可以揭示这些新功能在特定情况下的成本和收益。这有助于用户在各种条件约束和应用要求的情况下，决定是否启用某些可选编码功能。

在前面的章节中已经介绍了各种算法在比率失真方面的编码效率，这章我们将讨论转向一种考察权衡分析是如何生效的方法。我们聚焦于三个主要领域的优化方法及它们固有的权衡分析-即性能，耗电量和质量。这三个领域在现代的视频编码应用模型和编码解决方案中至关重要。

我们首先从这三项指标的常见权衡分析方法开始，同时也涉及可能出现的其他选择。接下来是讨论编码参数调整对这三个指标的影响。最后我们简要讨论一些常见的优化策略和方法。

我们将列举不同的权衡分析案例来研究性能和耗电量，性能和质量及耗电量和质量之间的关系。这些例子会从多个不同的角度来分析变量，揭示方法论常用于此类分析。

权衡分析的思考

权衡分析本质上是决策训练。尽管视频复杂多样、参数调整组合众多，权衡最主要还是以增强应用程序的视觉体验为基础。权衡的效果-即编码比特流的质量，决定了分析的价值。首先编码比特流必须是语法上有效的，其优劣通常根据其属性来判断，这些属性包括压缩量，解码时的感知质量，产出时间和耗电量。

权衡分析的一个重要应用是比较两种基于性能，耗电量和质量的编码解决方案。这种情况下，为了保证比较结果的公正性，就必须考虑各种系统参数，且使这些参数尽可能保持相同。这些参数包括：

- 通常是为了适应超频或降温而设置的可配置的TDP（cTDP）或场景设计功率（SDP）设置（在第四代Intel Core或更高版本的处理器中）如标准TDP，cTDP上升或cTDP下降。
- 电源模式，AC（插入式）或DC（电池）。
- 操作系统图形电源设置和电源计划，如最大电池寿命均衡，或获取最大性能。
- 显示面板接口，如嵌入式显示端口（eDP），或高清多媒体接口（HDMI）。
- 显示分辨率，刷新率，旋转和缩放选项，颜色深度和色彩增强设置。
- 连接到系统的显示单元的数量和类型（例如，单个或多个，主要或次要，本地或远程）。
- 开销和优化，应用程序的颜色设置校正和色彩增强，驱动程序设置，等等。
- 固件，中间件和驱动程序版本。
- 操作系统版本和版本。
- CPU和GPU的工作电压和频率。
- 内存配置和内存速度。
- 源视频内容格式和特征。

在进行比较时，还必须关闭不相关的应用程序或进程，使测试工作负载成为平台上唯一的运行负载。这样可确保将可用资源全部分配给工作负载并且没有资源争用或调度冲突。此外，通常会进行多次相同测量，并将多次结果中位数作为最终结果。这样可以减少测量公差内可能存在的任何噪声。同时保持系统温度稳定也是必要的，以减少测量中的可能的噪声数据；为了保持温度稳定，温度控制器会在温度高于设置的目标值时自动启动。

权衡分析的类型

对于移动设备，节电通常是高优先级。因此，如果某项措施只是适度地影响视觉质量，但节省了大量的电量，对于低功率移动设备，通常优先考虑有利于节电的质量权衡。通常，在保持视觉质量相同的情况下能提供更大压缩的技术是权衡分析的优秀候选。但是，这些技术往往具有更高的复杂性，对电量和性能的需求更大。例如，HEVC编码与AVC相比，提高效率的同时压缩会更复杂。因此，在功率受限的移动设备上，对高清视频的编码，HEVC编码代替AVC编码往往不是自动选择。由此，权衡分析必须考虑总体收益或净收益。

优先级主要由模型使用的要求驱动，因此根据用途也会管理权衡方法。例如，视频会议应用程序和视频转码应用程序，对于前者，在整个视频会话中，低延迟和实时性的需求会要求稳定的电量消耗，而对于后者，快速运行和睡眠的方法更有益。额外限制可以根据资源的可用性等来施加。例如，由于系统的限制，包括TDP，最大处理器频率限制等等，某些折损视觉质量以获得更好性能的技术可能并不总是可行的。类似地，低级缓存虽然可以提高许多性能视频应用程序，但无法在受限的系统中使用。

参数调整的效果

各种编码参数影响性能和电量消耗以及视觉质量之间的关系。调参的目的往往是发掘获得更高的性能，更好的质量，及节省电量的方法。此外，调参也可以暴露出那些未优化视频应用中的低效率问题，以及造成这些低效率的潜在原因，所有这些都可以带来更好的解决方案。

通常，相比于降低耗电量，这种调整的影响在改进视觉质量和性能中更容易看到。正如第4章和第5章所述，许多参数对性能和质量都有重大影响，包括视频空间分辨率，帧速率和比特率；图片结构的基团；参考图片数量；模式决策和运动矢量确定中的R-d优化；自适应去块滤波器；各种级别的独立数据单元，例如宏块，切片，帧或图像组；多次分析和处理，多代压缩以及预处理和后处理过滤器；和特效过滤器。

其中一些参数对视觉质量有较大影响，而其他参数则对性能和功率更有益；参数调优工作应该把这些相对的收益考虑在内。例如，使用B图片能够同时显著影响视觉质量和性能，但在模式决策和运动矢量确定中使用R-D优化，相比于它对视觉质量的改善，减慢编码速度的影响更显著。类似地，使用多个切片会略微降低视觉质量，但它可以提高可并行性和可扩展性，从而提供提升性能和节电机会。

此外，重要的是要考虑在编码视频保持合理视觉质量的同时可以节省多少电量或提高性能。视频内容的性质和比特分配策略是这里的重要考虑因素。

优化策略

有一些不会降低视觉质量的优化策略用于提高性能或节省功耗。这些策略通常用于视频编码应用的优化，并具有适当的权衡。

- **减少调度延迟：**批处理允许对各种宏块做一系列的函数调用（例如调用运动估计）。此外，批处理允许在合适的并行处理中采用宏块行级多线程（*macroblock-row-level multithreading*）。通常，批处理中的所有操作共享相同的内存，从而提高了数据获取速度和缓存命中率。然而，应用程序必须等待批处理中所有宏块完成写入，然后才能从内存读取数据。这会带来很小的延迟，但是该延迟仍然适用于视频流等视频应用。批处理实现的性能优势通常不会牺牲视觉质量，并且还为同时运行的其他工作负载留有足够的空间。
- **优化闲置时间：**通过workload shaping进行主动的电量优化是另一种功耗优化的方法。与最坏情况的设计原理相反，视频编解码器实现框架不仅知道硬件特定的细节，而且还主动调整实现策略以提供最佳的资源利用率。对于传统的无功优化方法，系统仅通过利用可用的空闲时间来调整其执行速度，以适应不断变化的工作负载。在主动优化方案中，可以在给定的时间内改变工作负载的情况，从而节省约50%~90%的电量。在这种情况下，空闲时间在多个数据单元上累积，以便底层处理器可以在更大的空闲时间段内使用更主动的节能方法，例如深度睡眠。此外，通过在可忍受的延迟内对视频帧进行重新排序，也可以节省功耗。使用高级的复杂模型调整工作负载，视频编解码器框架则在运行时解释模型并选择合适的操作频率和电压，从而在不损失质量的情况下，将功耗降至最低。
- **任务并行化：**第5章已经讨论了任务、数据和指令并行化的好处。把独立的任务并行化并将它们调度到多个处理器上可以充分利用处理器的能力。任务并行化可以加速执行速度，并使处理器进入长时间的睡眠状态，从而实现节能。任务管道还可以使资源在必要时保持繁忙，并最大程度地减少资源冲突。此外，通过适当设计并行应用程序，当处理器变得太慢或无响应时，可以将任务重新安排到其他处理器来消除性能瓶颈。对不同处理器的任务指定优先级也有助于提高整体性能。但是，并行化具有潜在的缺点，并行化会增加额外的开销，例如处理器间通信或同步成本。
- **优化I/O：**在选择功耗和处理延迟时，除了在最短的时间内最大程度地利用系统资源之外，提高数据访问速度和降低I/O瓶颈至关重要。在某些离线视频应用程序中（例如基于云的视频分发），当处理器在并行的编码视频片段时，可以使用专用的I/O处理器来获得更好的总体功耗分布。但是，这种技术会导致延迟增加。只有完成所有的处理器都完成编码后，才能将最终的视频码流拼接在一起。因此，由I/O处理器提供的服务的分组数量是一个需要折衷的参数。另外，应该尽可能的利用数据预取和流式传输，视频数据特别适合此类技术。
- **降低计算量：**算法优化允许使用诸如基于阈值的循环而提前退出循环，或利用SIMD

的并行性。这些技术有利于降低计算量。但是，这需要非常仔细的分析，才能确定和理解所涉及的各种权衡。在某些情况下，降低计算量有可能会影响到视觉质量。此外，减少执行的指令数量、手工优化代码或使用各种编译器优化技术都会直接影响性能和功耗。

- **优化成本与收益：**需要时刻认真考虑高性能在执行中的复杂性和随之而来的功耗。可能需要重新设计性能优化方法以解决功耗问题。在图像滤波实验中，发现滤波的负载行为和标量与向量的乘法加和的工作负载完全不同，尽管这两个工作负载都具有相似的可并行性。在图像滤波时，由于存在可用的数据复用，因此可能实现性能优化。对于许多视频编码和处理应用程序也是如此。然而，由于不均匀的数据复用和资源调度，图像滤波过程的功耗分布是不规则的。功耗最佳点对应于较大的展开因子，相对适中的阵列分区，流水线乘法器和非流水线循环。较大的展开系数可在合理的带宽需求下最大程度地利用已加载的数据。带宽需求通常在多个周期内摊销，以最大化给定资源的复用和效率。这导致复杂的控制流程和非直观的能源优化设计解决方案。对于这种复杂的工作负载，高性能的设计所带来的功耗增加未必总是合理的。

权衡性能和功耗

回想一下式6-1，式6-1表明功耗是频率的线性函数。在一定极限频率下，性能的提高直接取决于频率的增加，因此需要在保持功耗不变的情况下增加频率。为此，需要降低频率的辅助因子：即电压、电容和活性因子。此外，需要减少泄漏电流 (*leakage current*)。通常使用时钟门控来降低活动因子，而通过减小栅极尺寸来减小电容。如第6章所述，只能在电压缩放区域内降低电压，直到达到最低电压为止，该最低电压必须可以使晶体管工作。从图6-7可以看出，在低频 V_{min} 区域中的泄漏是恒定的，而在电压缩放区域 (*voltage-scaling region*) 中，泄漏在典型工作点处会持续增加。通过减小晶体管的宽度并使用较低的泄漏晶体管可以减小泄漏电流。但是，与漏电晶体管相比，低漏电晶体管的速度较慢。因此，硬件设计人员需要进行适当的优化，以在给定的功耗量下实现频率最大化。

90%的移动平台会受到功耗的限制。因此，每一点节能都等同于相应的频率增益。通常，在低功率平台的 V_{min} 区域，10%的功耗节省意味着15%的频率增益，而在电压缩放区域，20%的功耗节省仅相当于5%的频率增益。因此，不能过分夸大硬件设计对于最佳电压，电容，活动因子和泄漏的重要性，尤其是在给定功耗下获得最高频率时。对所有应用而言，更高的频率通常意味着更好的性能。

对于视频应用程序，更高的CPU或GPU工作频率可提供更快的编码速度，但同时也会消耗更多电量。因此，在系统设计和硬件体系结构级别，尤其是对于GPU加速的编码器，需要在功耗和编码速度之间进行权衡。编码的可编程部分还应该在性能和功耗之间保持适当的平衡。通常，通过并行执行编码任务、在CPU和GPU的多个线程之间安排适当的任务、在CPU和GPU之间动态迁移任务、调整任务的计划、优化单个任务的资源利用率实现性能和功耗之间的平衡，并且所有这些方案都不会对视觉质量造成明显的影响。例如，根据视频内容的复杂性，并行编码图片的两个切片 (*slices*) 可以提升10%的性能，而对质量的影响却可以忽略不计。某些硬件单元（例如码率控制单元，缩放单元等）可能会根据参数值而打开/关闭，因此编码参数的调整也会影响整体编码速度和功耗，再者，这种调整也可能会影响视觉质量。

考虑以下的有关视频转码应用程序的性能与功耗折衷的案例研究 (*case study*)。为了便于比较，在不同性能-功耗特性的两个平台上运行两个相同的测试。转码 (*transcoding*) 包括将压缩视频解码为未压缩格式，随后使用适当的编码参数将其编码为压缩格式的目标视频。对于相同的源视频内容，转码操作中的解码任务保持不变。通常，解码比编码快得多。因此，可以仅根据编码速度来测量转码的整体性能。在如上的描述下，性能和编码速度说的是一个事情。

权衡性能与功耗的案例研究

考虑两个转码的工作负载：视频长度约5分钟、分辨率为1920x1080p、帧率为30fps，将视频从高码率的H.264格式转码为低码率的H264格式。两次转码不同的地方在于，与转码任务2相比，转码任务1的视频内容复杂度较低，场景变化少，细节少，动作慢。

表8-1显示了运行转码任务的Intel平台的平台配置。

表8-1. 转码实验所用的平台的配置信息

指标	平台1	平台2
Processor	4th-gen. Core i7	4th-gen. Core i5
# Cores	4	2
CPU frequency	2 GHz	1.3 GHz
CPU turbo	3.2 GHz	2.6GHz
TDP	47 W	15 W
Cache size	6 MB	3 MB
Graphics	Intel (R) Iris Pro (TM) 5200	Intel (R) Iris Pro (TM) 5000
Max GPU frequency	1.2 GHz	1 GHz
Embedded DRAM	Yes	No
Memory	4 GB dual channel	4 GB dual channel
Memory speed	1333 MHz	1333 MHz

我们还考虑了三组编码参数，编号分别为1、2和3。表8-2显示了每个参数的重要区别。

表8-2. 参数配置的重要区别

参数	参数集1	参数集2	参数集3
Configuration	Fast	Faster	Fastest
Motion estimation algorithm	Algorithm 1	Algorithm 1	Algorithm 2
Motion estimation search range	48 pixels	48 pixels	28 pixels
Fractional pixel motion compensation	1/8 pixel	1/8 pixel	1/4 pixel
Adaptive search	Yes	No	No
Multiple reference pictures	Yes	No	No
Multiple predictions	Yes	No	No

Macroblock mode decision	Complex	Complex	Simplified
--------------------------	---------	---------	------------

图8-1显示了使用两种工作负载以及各种编码参数集在两个平台上的转码性能。值得注意的是，由于两个工作负载的复杂性不同，参数的调整对性能的影响也不同，并且影响的程度在两个不同的平台上也是不同的。

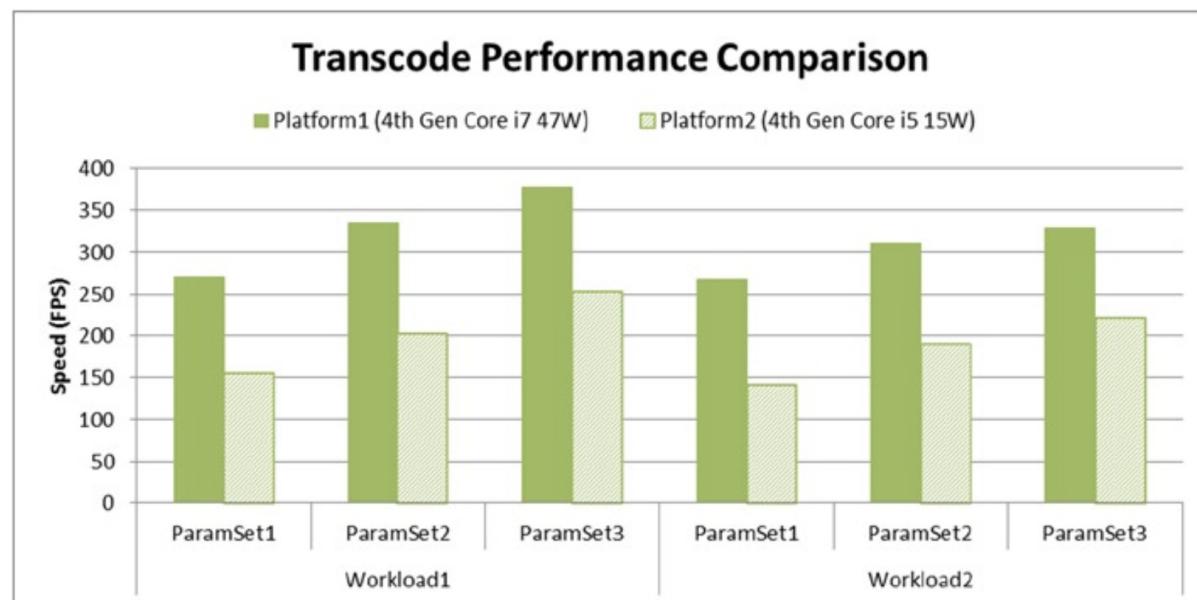


图8-1. 两个平台的转码性能比较

从图8-1可以看出，与平台2相比，平台1在编码速度方面的吞吐量平均提高了60%，其中嵌入式动态RAM占比10%，而GPU频率占比20%。其余30%的差异可归因于处理器图形硬件优化，GPU执行单元数量，缓存大小，CPU内核数量，CPU时钟速度，turbo容量等的组合。

当参数从快速过渡到最快时，Workload 2的性能不断提高。尤其是在平台1上，Workload 1的峰值性能比使用参数集3的实时速度快12倍以上。因此，很明显，工作负载特征以及参数调整会极大地影响转码性能。比较平台1上两个工作负载的最快参数集3，可以发现工作负载1的性能比工作负载2好13%。在平台2上也存在类似的趋势，其中工作负载1比工作负载2的性能高约12%。

由于工作负载2的特性以及平台2上资源的限制，参数集1在该平台上的性能明显降低，因为该参数集包括多个参考图片，多个预测以及对模式决策的详尽分析。

图8-2显示了两个平台在两个具有相同参数集的工作负载下功耗。

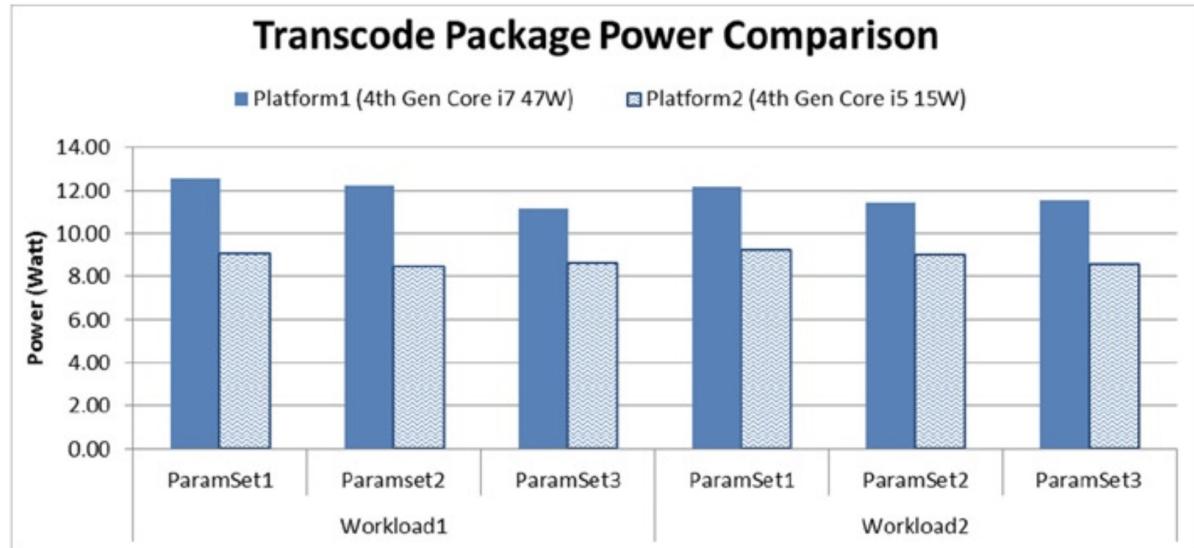


图8-2. 两个平台的转码功耗比较

从图8-2可以明显看出，与平台2相比，平台1平均要多消耗约34%的功耗。而对于工作负载，这两个平台都没有达到其最大TDP限制。但是，某些参数设置需要打开某些硬件单元从而消耗电量，而其他参数则不需要。从两个平台之间的功耗差异（14%~44%）可以明显看出这一点。

如果在每个平台上考虑绝对功耗，也可以得出有趣的观察结果。调整参数后，功耗通常会降低，尤其是在平台1上。此外，在平台1上，工作负载1具有28%的动态功耗范围，而工作负载2仅具有8%的动态功耗范围。但是，在平台2上，这些数字分别为7%和10%。这表明与平台2相比，在平台1上，工作负载1的功耗对更改的参数的反应更快。但是，在平台2上，这些工作负载不受计算的限制，因此对更改的参数没有反应。在这种情况下，缓存性能和GPU执行单元的数量成为主要因素，因而很少考虑编码参数。

图8-3显示了针对每个参数集的两个平台上两个工作负载的平台效率（以fps /瓦为单位）。平台1的效率通常比平台2更高，这主要是因为平台1的专用的嵌入式动态RAM、更高的GPU频率、更高的GPU执行单元数量以及更好的缓存性能。平台1的效率比平台2平均提高了约23%。

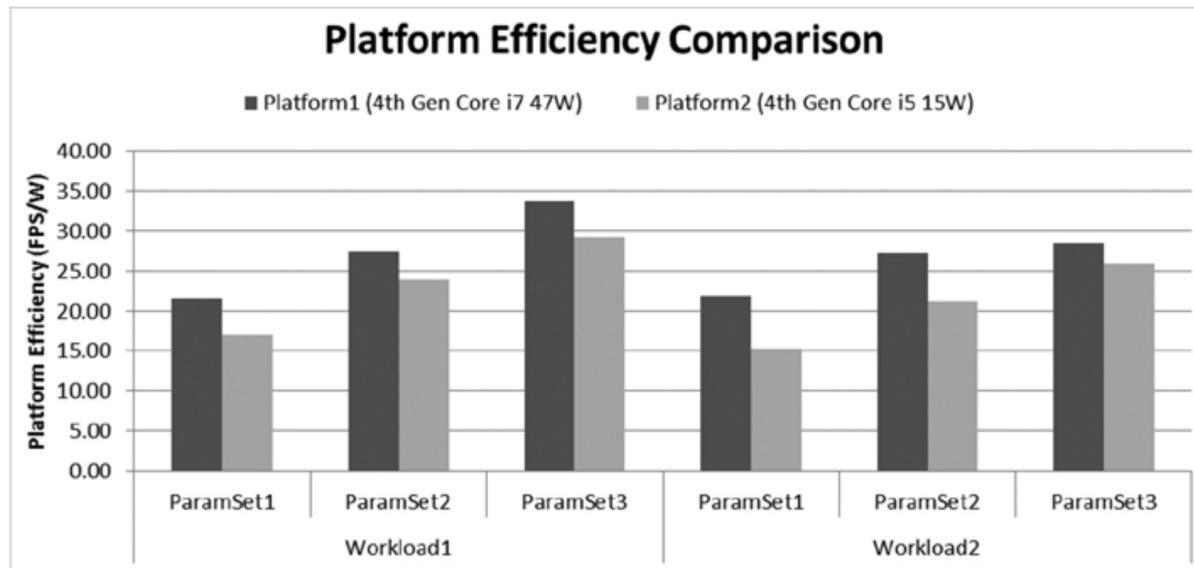


图8-3. 不同平台的转码效率

从图8-3中还可以看出，参数调整在两个平台上对工作负载1的影响都类似，但是对于工作负载2，平台2在平台效率方面显示出较大的差异。平台效率的这种行为不仅是由于参数变化，而且还由于工作负载的不同的特征。

从图8-3中还可以看出，参数调整在两个平台上对工作负载1的影响都类似，但是对于工作负载2，平台2在平台效率方面显示出较大的差异。平台效率的这种行为不仅是由于参数变化，而且还由于工作负载的不同的特征。

图8-4显示了性能与功耗分析的另一种观点。由于其可用资源的差异，这两个平台显然表现出不同的性能特征。在两个平台上，工作负载都存在聚集的现象。由于更大的缓存和嵌入式动态RAM的存在，与平台2相比，平台1通常消耗更多功耗，但是它也提供了更高的性能。

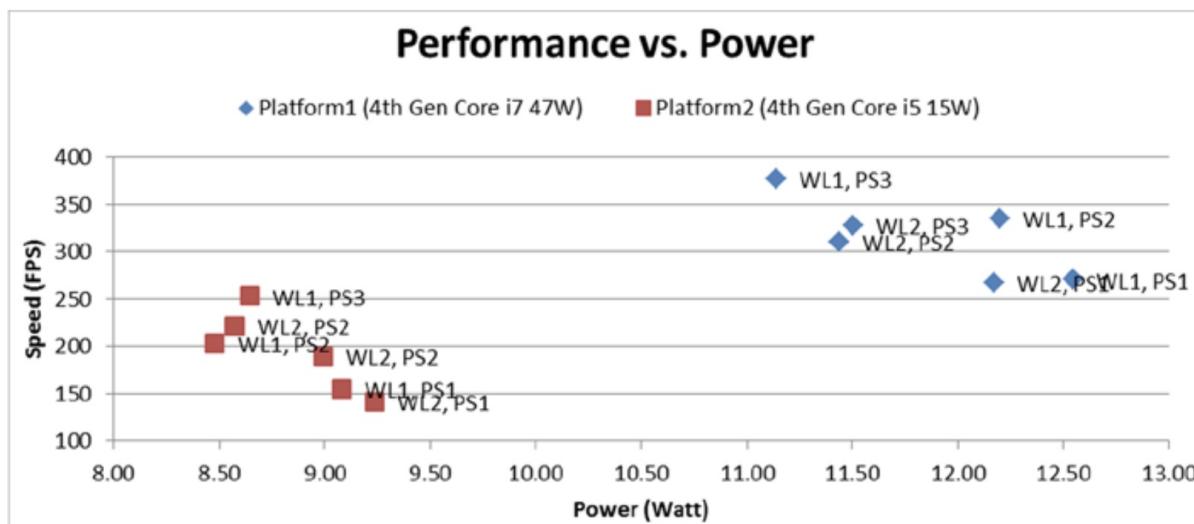


图8-4. 两个平台上的性能与功耗 (WL和PS分别是工作负载和参数集的缩写)

从图8-4可以看出，在给定的平台上，适当的参数选择可以提供良好的节能机会。例如，在平台2上，与参数集1相比，使用参数集3可以使工作负载1节省近1瓦的电力。

在进行性能和功耗的权衡分析时，并行化技术或优化资源利用率通常对视觉质量的影响很小。但是，调整编码参数的同时也会对质量造成影响。在这些情况下，功耗—性能折衷变成了功耗—性能—质量的权衡。如果码率控制算法试图以可变的码率保持相同的质量，会导整个视频存在不同的码率。此时，这种权衡就变成了功率—性能—编码效率的权衡。

权衡性能和质量

更高的编码速度可以通过调整视频编码参数（例如比特率或量化参数）获得。通过丢弃大部分高频细节、减少需要处理的信息，编码可以变得更快。但是这会直接影响所得视频的视觉质量。另一方面，B帧的使用提供了另一种性能和质量的影响因子。解码B帧依赖可用参考帧，因此引入了延迟，但B帧的使用改善了视觉质量以及时间视频平滑度。例如在一组使用H.264编码的实验中，我们发现当在参考帧之间使用两个B帧时，相比同组的HD视频序列，FPS（画面每秒传输帧数）的平均影响约为7%，同时BD-PSNR维度的视频质量提升了约0.35dB。

类似地，诸如运动搜索范围？，搜索方法？，参考帧数量？，双遍编码？等参数的调整，都会影响视频的性能和质量。因此，在调整视频编码器的特征和参数时，始终都需要调研对性能损益和视觉质量的潜在影响。为了说明性能-质量的权衡，我们提出了两个案例研究并讨论相关的结果。

Case Study I

将一个35Mbps码率的H.264视频转码为相同格式、7Mbps码率的视频。原视频片长约5分钟，分辨率为1920×1080p，30fps。视频中包含几个不同复杂性的场景，从密集空间细节到大量平面区域，从不规则运动到静态镜头。转码中涉及将原文件全解码并用新编码参数对其进行重新编码。

转码的采用的计算机平台配置如表8-3所示。

表8-3 案例I转码采用的计算机平台配置

System Parameter	Configuration
处理器	4th-gen. Core i5
核数	4
CPU主频	2.9 GHz
CPU睿频	3.6 GHz
热设计功耗	65 W
缓存大小	6 MB
显卡	Intel (R) HD Graphics (TM) 4600
GPU最大频率	1.15 GHz
嵌入式DRAM	Yes
内存	4 GB双通道
内存频率	1333 MHz

转码采用了两种方式：仅运行在CPU上的软件转码器，和经过GPU加速的转码器（大多数计算密集型任务在专用硬件单元中完成）。这两种实现方式不同地优化了参数，但两者都提供了三种性能质量权衡的输出模式：最佳质量模式，平衡模式和最佳速度模式。

GPU加速的实现只提供了少量外部可设置的参数，而仅CPU上的转码有更多的可调参数，我们尽力使这些参数在各自的实现方式下尽可能接近。当然，两种实现方式的模式调整的确切参数存在差异，但也存在一些共性。表8-4总结了常用参数。

表8-4 两种转码方案的常用的配置

参数	最佳质量模式	均衡模式	最佳速度模式
运动估计和模型决策算			算法3（早期版本）

法			中?)
局部运动补偿	八分之一像素	四分之一像素	无
参考帧	多	少	1
自适应搜索	有	无	无
运动搜索范围	大	中	小
加权预测	有	有	无
多B帧?	有	有	无
子宏块分区	全部	部分	无
场景变化检测	有	有	无
比特率控制的前向分析	多数帧	少数帧	无

请注意，这两种实现方式中使用的参数略有不同，因此也不会产生完全相同的视频质量。另外，GPU加速实现的重点是在不损失太多视觉质量的情况下获得更高的性能，因此在此实现中只有少数参数的调整是从最佳质量到最佳速度。另一方面，仅CPU转码器中难以获得更高的性能，因此在该实现方式的最佳速度模式下更激进的关闭了几个功能（相比GPU加速转码器而言）。

以FPS来衡量两种转码器实现的三种操作模式下的性能。注意，为获得性能质量的平衡，三种模式都调整了编码参数。图8-5显示了仅CPU转码和GPU加速转码两种实现之间的性能比较，同时图中还显示了不同模式下的比率。

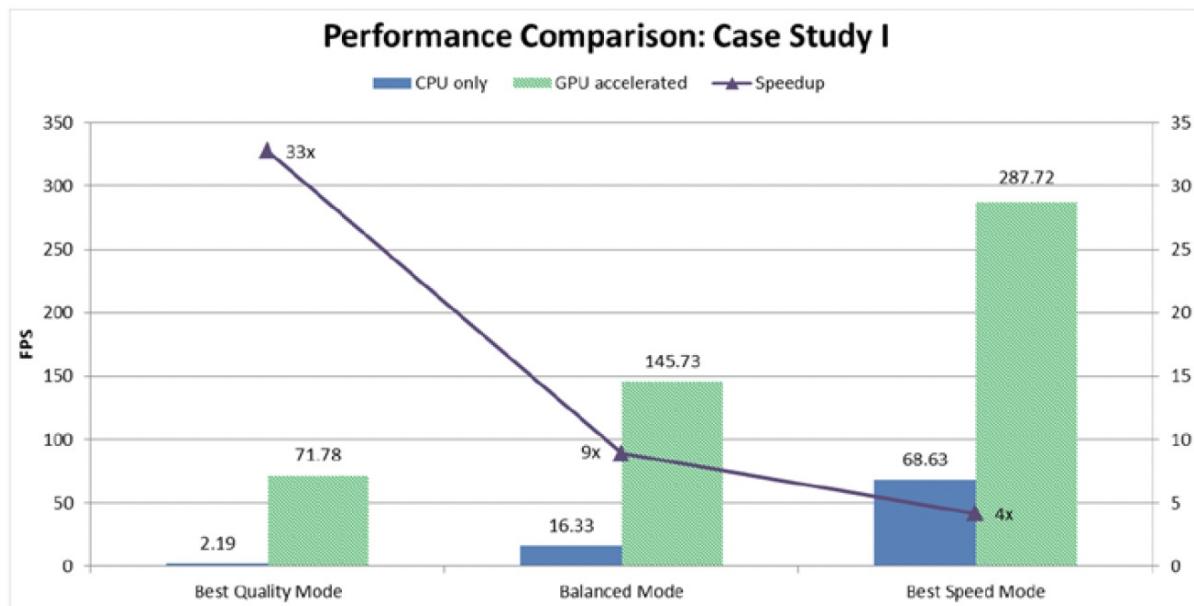


图8-5. 不同转码模式的性能对比

从图8-5可以看出，这两种实现都在速度方面进行了缩放，速度范围从最佳质量到平衡模式再到最佳速度模式。例如，GPU加速将编码从一种模式加速到另一种模式，速度提高了大约2倍。但是，由于对编码参数进行了更积极的调整，仅CPU的方式从最佳质量缩放到了平衡模式，并且利用表8-5中的优化措施实现了7.45倍的加速。类似的，从平衡模式到最佳速度模式，可获得额外的4.2倍加速。

表8-5. 不同模式下，仅利用CPU的优化方法

参数	最佳质量	平衡	最快速度
Motion estimation method	Uneven multihexagon search	Hexagonal search with radius 2	Diamond search with radius 1
Maximum motion vector range	24	16	16
Sub-pixel motion estimation	Yes	Yes	No
Partitions	All (p8x8, p4x4, b8x8, i8x8, i4x4)	p8x8, b8x8, i8x8, i4x4	No sub-macroblock partitions
Use trellis for mode decisions	Yes	No	No
Adaptive quantization	Yes, with auto-variance	Yes	No
R-D mode decision	All picture types	I-picture and P-picture only	None
Max number of reference pictures	16	2	1
Number of references for weighted prediction for P-pictures	2	1	None
Number of frames to look-ahead	60	30	None
Max number of adaptive B-pictures	8	2	None
CABAC	Yes	Yes	No
In-loop deblocking	Yes	Yes	No
8×8 DCT	Yes	Yes	No
Scene change detection	Yes	Yes	No

显然，如上的优化影响了视觉质量。图8-6显示了两种实现方式的视频质量的比较。从最佳质量到最佳速度，未利用GPU加速的方式的PSNR平均损失约5 dB，而文件大小的减少则不到0.1%。另一方面，在保持视觉质量的同时提高性能的焦点上，GPU加速的方式从最佳质量到最佳速度模式下仅平均损失了~0.6 dB的PSNR。但是，与最佳质量模式相比，采用最佳速度模式时，GPU加速的模式会导致文件大小增加~1.25%，从而折衷了所获得的压缩量。

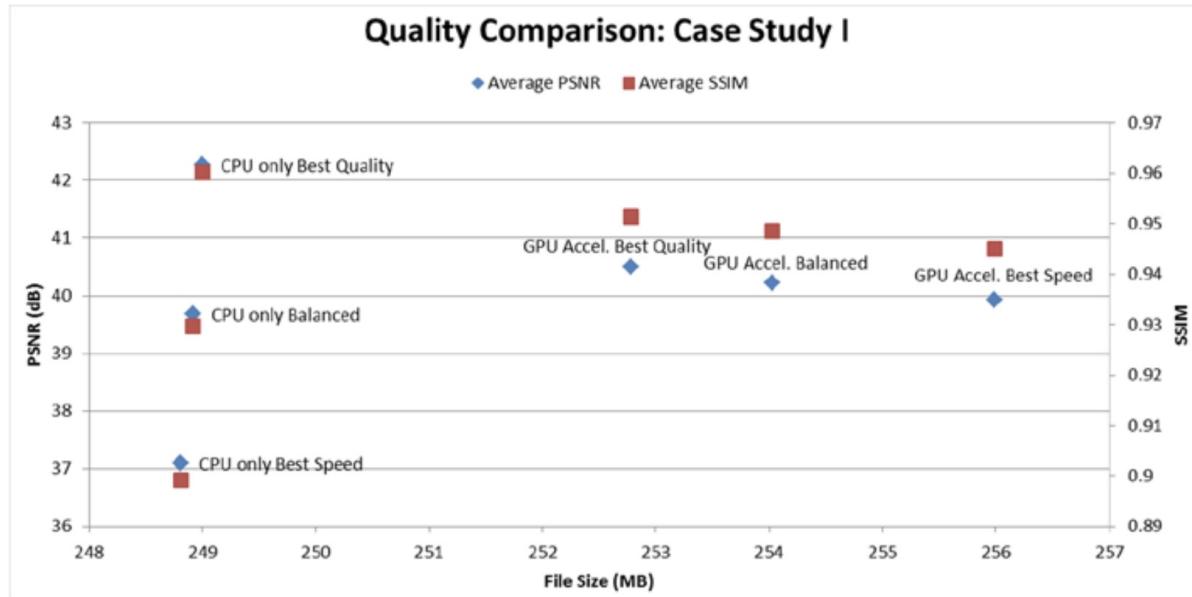


图8-6. 是否采用GPU加速模式的质量对比

从图8-5和图8-6还可以发现：就三种性能模式的速度而言，采用GPU加速比仅采用CPU时分别快约33倍、9倍、4倍。这也显示了两种实现之间在参数调整方面的对比。尽管GPU加速的实现在最佳质量模式下的性能要好得多，但与仅用CPU的最佳质量模式相比，它的PSNR平均降低了1.76 dB，文件大小增加了约1.5%。因此，GPU加速的方式已经在通过牺牲视觉质量而支持性能。此外，由于某些算法是在固定功能硬件单元实现的，因此，GPU加速的实现在更改算法的能力方面缺乏灵活性。尽管如此，在最佳速度模式下，GPU加速方式的PSNR平均高出约2.8 dB，但与纯CPU实现相比，文件大小却增大了约2.9%。这些结果证明了这两种实现中所固有的性能-质量折衷。

图8-7显示了本节的案例研究的编码视频质量与编码速度之间的关系。对于CPU-only和GPU加速的不同模式下的质量/速度存在明显的缩放，但是两种实现方式的缩放率却是不同的。

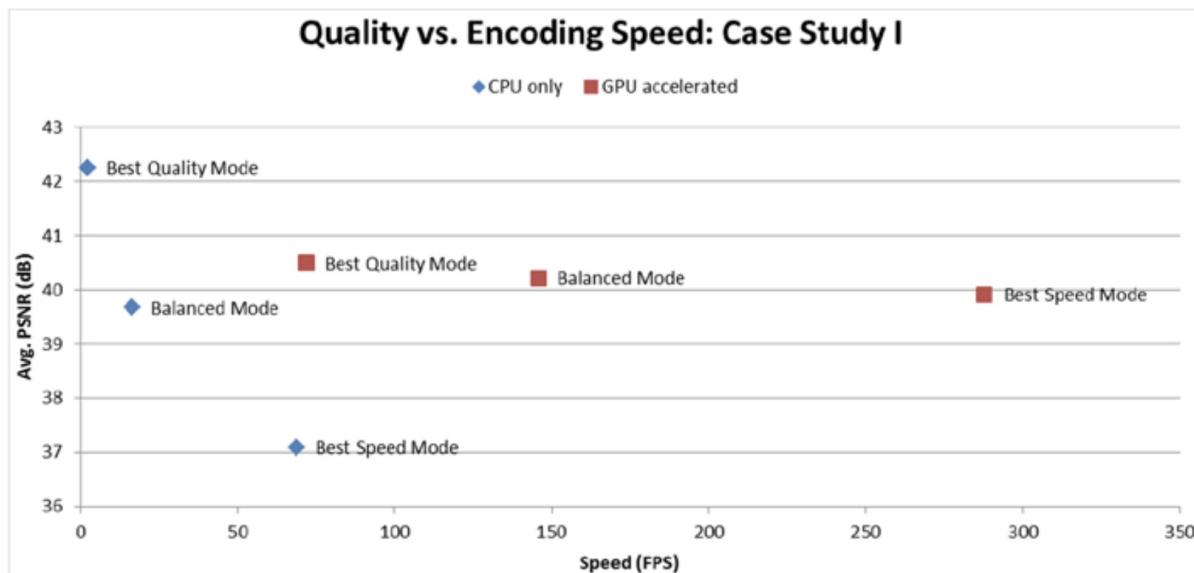


图8-7. Case Study I 的质量和速度的比较

Case Study II

第二个案例显示了两种编码解决方案在性能和质量方面的另一个比较。在该案例中，测试组的视频包含十种不同的视频内容，这些内容具有不同的运动和细节复杂性。视频分辨率均为：352×288, 720×480, 1280×720, 1920×1080。采用七组视频编码参数用以涵盖从最佳质量到最佳速度的范围。使用两种GPU加速的编码器执行编码测试。

在此案例中，尽管编码器的参数集之间存在一些差异，但两种编码器都在相似的应用程序接口上运行，因此对于两种编码器而言：参数集1提供最佳质量，参数集7提供最佳速度。例如，编码器1的参数集1包括1/8像素精度的运动补偿并使用网格执行模式决策，而编码器2的参数中未包含这些参数设置。表8-6显示了两个编码器共有的一些重要参数。

表8-6. 两种编码器共有的重要参数

	PS1	PS2	PS3	PS4	PS5	PS6	PS7
8×8 transform	Yes	Yes	Yes	Yes	Yes	Yes	Yes
1/4 pixel prediction	Yes	Yes	Yes	Yes	Yes	No	
Adaptive search	Yes	Yes	Yes	Yes	Yes	No	No
Max references	10	8	6	5	4	2	2
Multiple prediction	Yes	Yes	Yes	Yes	P-picture only	No	No

图8-8显示了两个编码器在不同参数集上的FPS方面的性能比较。两种编码器随着参数集的调整，都有明显的性能提升的趋势。但是，不同编码器的速度提升率却是不同的。在参数集3之后，编码器2的最佳性能要比实时性能快~9倍。而编码器1通过调整参数可以达到大致相同的性能提升率。

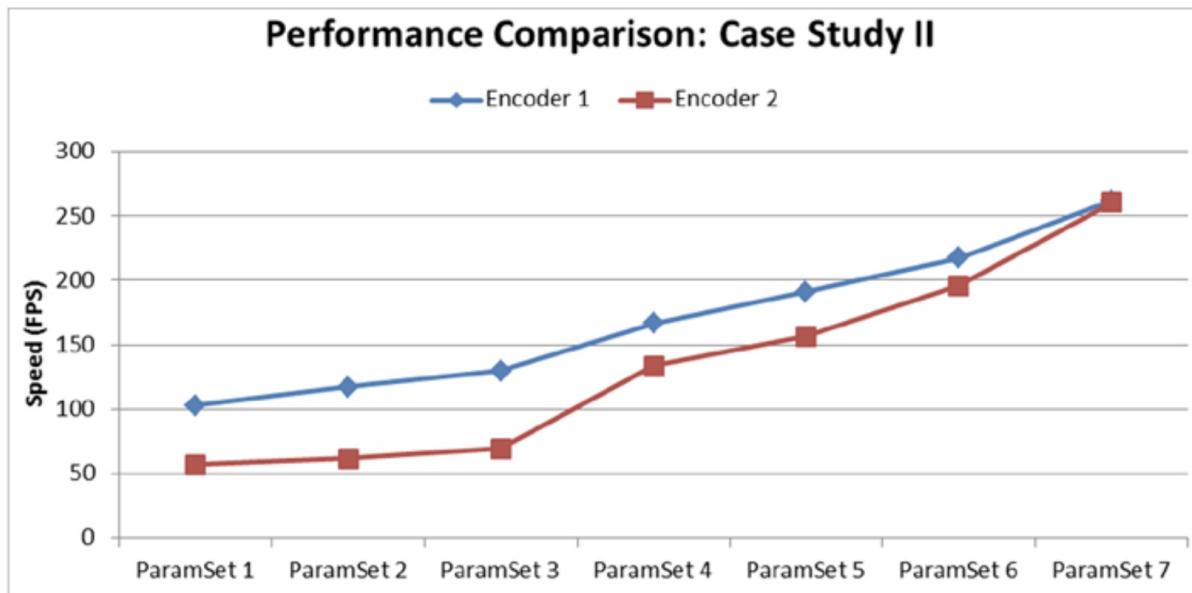


图8-8. 两种编码器的性能比较

图8-9显示了在编码参数集上，两个编码器在BD-PSNR方面的质量比较。类似的，两个编码器都存在质量逐渐降低的明显趋势。对于编码器2，调整参数最多可获得~0.45 dB的BD-PSNR增益，而编码器1则可达到~0.47 dB。但是，对于编码器1，参数调整的中间级别不会显示出明显的质量差异。编码器1在参数集7和6以及参数集2和1之间均存在明显的质量提升。

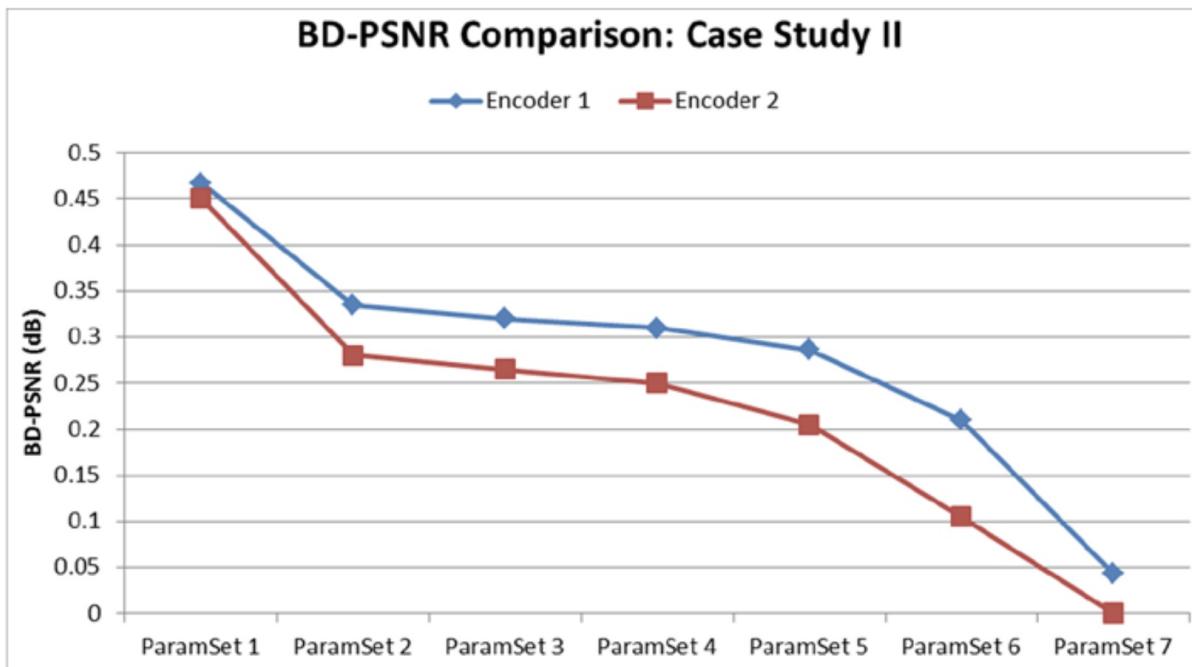


图8-9. 两个编码器的质量比较

图8-10显示了第二个案例研究的质量与编码速度的关系。通常，对于所有参数集，对于给定的编码速度而言，编码器1可以提供更好的质量。显然，参数集1确实代表了最佳质量模式，而参数集7代表了两种编码器的最佳速度。对于编码器1，参数集6似乎是最有效的，

因为它提供了最大的质量差异 (~ 0.1 dB BD-PSNR)，而且与此同时，它的编码速度也比编码器2提高了11%以上。

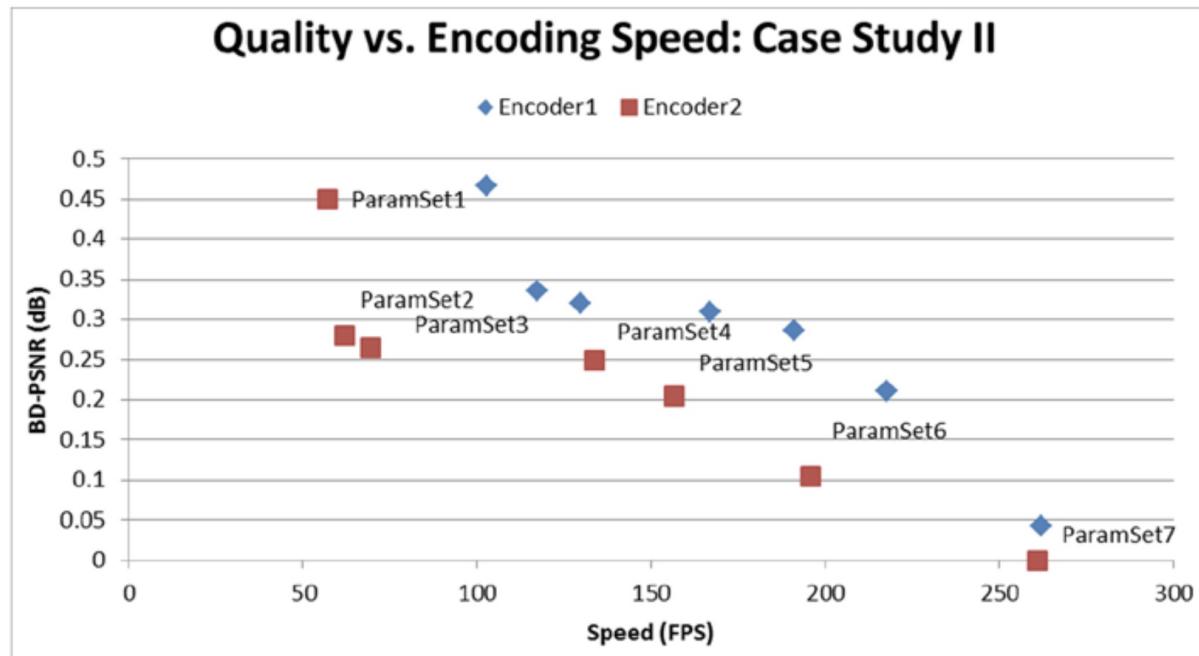


图8-10. 案例研究II中的质量和编码速度的比较

权衡功耗和质量

噪声是数字图像和视频中最关键的问题之一，尤其是在弱光条件下。亮度和色彩噪声相对量取决于曝光设置和相机型号。低光且无闪光的照片和视频会遇到严重的噪音问题。具有色度噪声的视频的感知质量可以通过色度噪声的降低来改善，这被称为色度去噪。但是彻底消除亮度噪声可能导致画面不自然，彻底消除色度噪声会引入假色，因此去噪算法应根据输入的局部特性调整滤波强度。该算法需要在降低噪声和保留细节之间进行折衷。GPU 加速的色度去噪滤波器的实现，作为一种视频处理能力，通常会是英特尔处理器显卡提供的图像增强色彩处理（IECP, Image-Enhancement Color Processing）的选项。

为了权衡功耗和质量，我们进行了色度去噪滤波的案例研究。在回放视频时，色度去噪滤波器使用时间滤波器分别检测两个色度平面（U和V）中的噪声。噪声估计值以8-bit 表述，保留在帧间并融合。基于GPU加速的色度降噪能为实时处理提供足够的性能，因此在现代处理器上性能通常不是问题。然而，尽管预期的视觉质量会提高，但色度去噪滤波器带来的额外操作意味着带来了额外功耗。本案例研究了功耗和质量之间的权衡。

案例研究

本案例使用第三代Intel Core i7系统，CPU主频2.7GHz，最大睿频3.7GHz，热设计功耗45W，显卡最大动态频率1.25GHz。屏幕分辨率为1920×1080，与视频内容的分辨率相同。使用均衡的操作系统电源策略，工作温度保持在50°C（CPU风冷系统的常规运行温度）。采用两种工作模式，包括AVC编码的视频播放和支持色度去噪滤波器的VC-1编码蓝光光盘播放。注意，与AVC编码的内容相比VC-1编码的内容具有更高的场景复杂度。

图8-11显示了色度去噪滤波器对功率的影响。从功率的角度看，使用色度去噪滤波器带来了平均约7%的额外功率，约0.48瓦。这是一个明显的功耗。

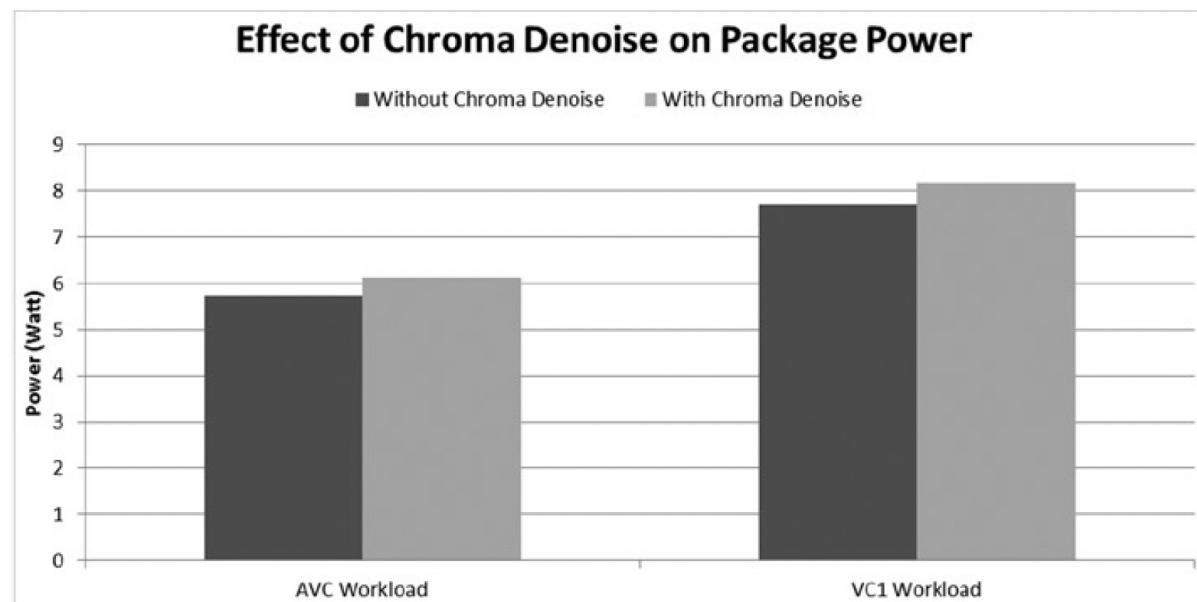


图8-11. 色度去噪滤波器对功率的影响

图8-12显示了色度去噪滤波器对CPU和GPU的整体活动影响。从活动角度看，使用色度去噪增加了约8.5%的活动量。这与功耗的增加相对应，并且说明处理器的忙碌时间明显增加。

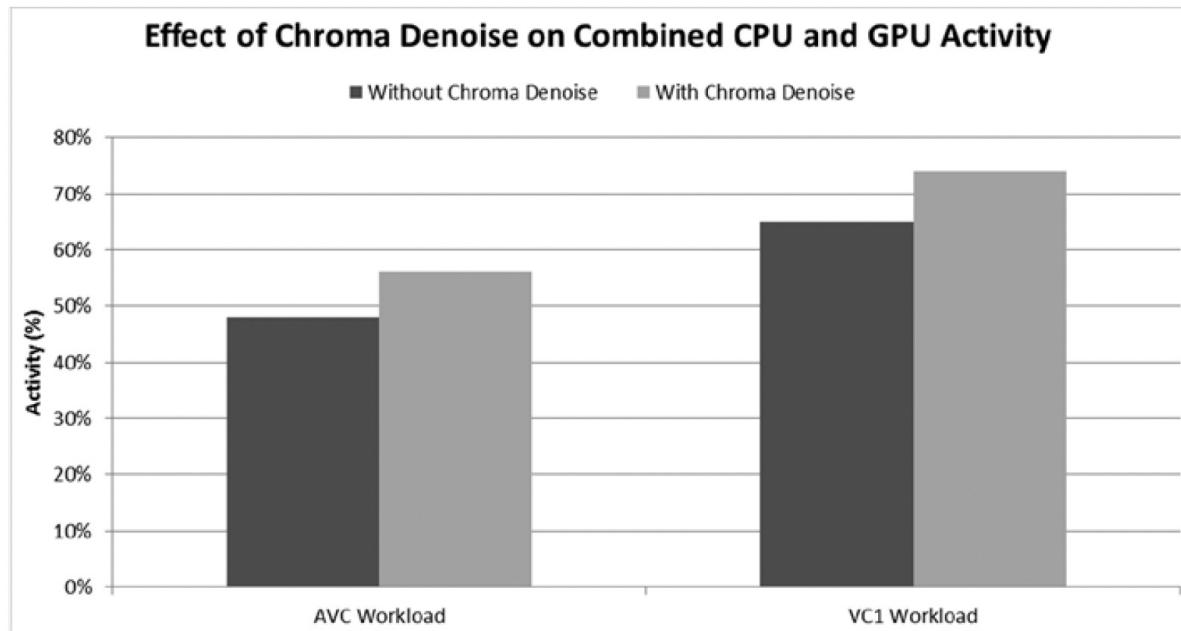


图8-12. 色度去噪滤波器对CPU和GPU的整体活动影响

图8-13显示了PSNR维度的色度去噪滤波器对视觉质量的影响（PSNR，Peak Signal to Noise Ratio，峰值信噪比）。尽管有平均约0.56dB的PSNR的改善，但这些工作负载对视觉质量的影响很小。注意，VC1工作模式下的PSNR绝对值比AVC的低约4dB，原因是VC-1视频编码与AVC编码相比有更高的复杂性。

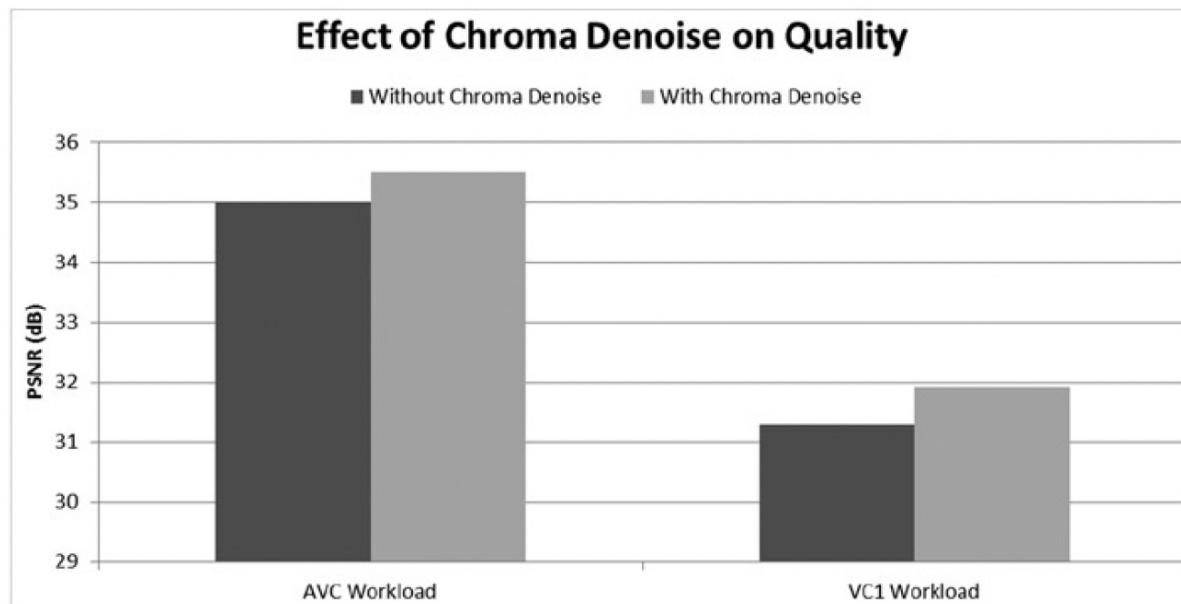


图8-13. PSNR维度的色度去噪滤波器对视觉质量的影响

图8-14显示了色度去噪滤波器功耗与质量的权衡。尽管可以观察到PSNR的改善，但是这种改善是以显著增加的功耗为代价的。因此在功耗有限的情况下应仔细权衡。例如，在检测到低电量时，可功率感知的回放程序将自动关闭色度去噪以节省功耗。

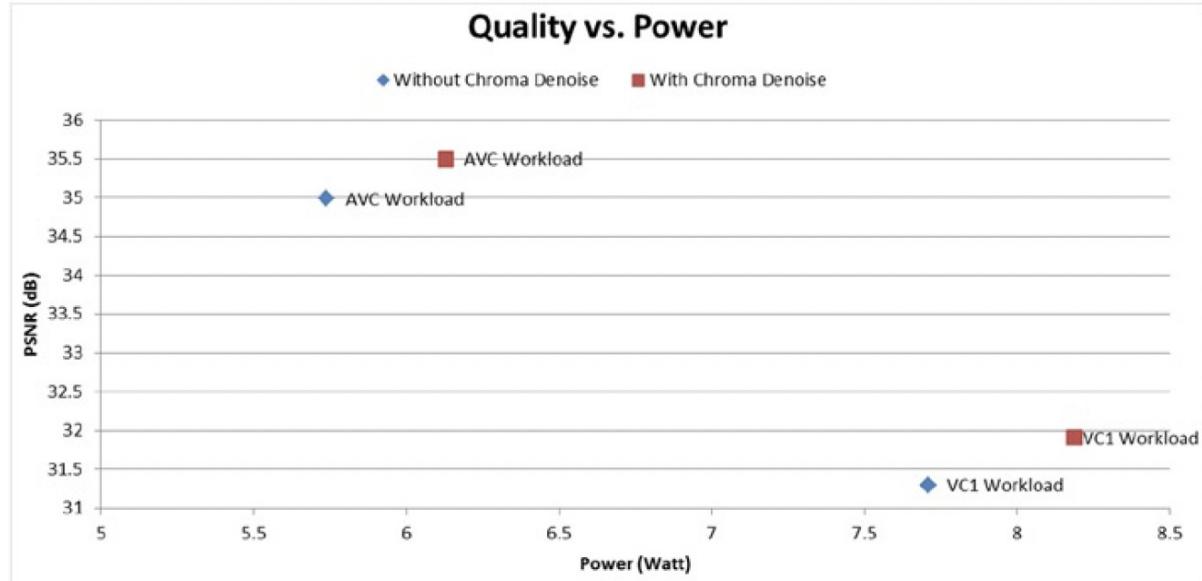


图8-14. 色度去噪滤波器功耗与质量的权衡

总结

在本章中，我们讨论了权衡分析的工作原理，并提供了四个实例。虽然权衡分析可能涉及许多不同维度，我们关注的是性能，功耗和质量，这是当今低功耗计算设备成功的首要标准。功耗决定了电池的寿命，同时使最高编码速度和最佳视觉质量的结合作为当代视频编码解决方案的最理想特征。

我们讨论了编码参数调优并研究了一些优化策略。此外，我们提供的案例研究体现了性能与功耗、性能与质量、功耗与质量的权衡。这些案例研究反映了几种不同的观点，且有助于阐明这类案例中常用的分析方法。

结语

本书是在如下的机缘而编写：数字视频及其相关的计算复杂性的不断增长的需求，视频编码标准的可用性和改进，低功耗计算设备的局限性（尤其是在移动环境中），要求越来越高的速度和对资源的高效利用，对任何给定平台上可能的最佳视觉质量的渴望，缺乏统一的方法思考并分析可用的权衡。在最后一章中，我们总结了本书的重点，并提出了对该领域未来发展的一些考虑。

重点和结论

根据本书中的要点，可以得出以下结论：

- 在各种视频度量之间可能会进行权衡，包括压缩量，视觉质量，压缩速度和功耗。
- 可以通过调整视频编码参数分析这些度量指标之间的权衡。
- 部分编码参数对某些视频度量指标的影响要大于其他编码参数。同时，根据具体应用，某些优化措施对某些度量指标的优化效果也可能会好于其它的指标。
- 分析不同的编码参数对性能，功耗和质量的影响是评估视频编码解决方案的一部分。
- 取决待优化的目标和待调整的参数，某些视频编码解决方案也会因视频类型的差异而存在差异。
- 比较两种视频编码解决方案，不仅有助于对可用解决方案进行排名，还有助于做出明智的选择。
- 采样和扫描方法，图像速率，色彩空间，色度和编码格式是ITU-R数字视频演播室标准在三个规范中定义的参数。
- 由于人类视觉系统（HVS）的自然容忍特性，并且HVS对某些类型的视觉质量损失比其他类型更敏感，因此视觉质量的降级也是一种选择。许多视频压缩技术都会利用这一事实，并以质量为代价压缩视频。
- 色度亚采样是一种充分利用HVS对颜色信息的敏感性的常用技术。许多视频都会采用使用4：2：0的色度亚采样。
- 各种技术均可用于数字视频压缩。大多数国际标准采用如下的方式进行有损编码：基于变换降低空间冗余，基于块匹配运动补偿降低时间冗余，基于可变长度代码降低频谱冗余。
- 国际标准在实际压缩，通信，存储，广播，游戏等领域定义了一系列视频应用程序。标准视频格式对于产品之间交换数字视频至关重要。这些标准定义的算法可在实际的硬件和软件系统中实现，并且在多个行业通用。
- 视频压缩受到许多因素的影响，包括输入端噪声，输入图像的动态范围，图像分辨率，伪像，码率要求，帧率，错误恢复能力，质量设置（图像之间恒定或可变），算法复杂度，平台功能等。
- 有损压缩会导致视觉质量损失，但是由于HVS限制，少量的质量损失不会导致主观视觉质量下降。常见的压缩伪像包括量化噪声，模糊，块效应，振铃，混叠，闪烁等。设备上的传感器噪声和视频特征也会影响质量。影响视频质量的视频特征主要有：空间和时间活动，压缩的数量和方法，压缩次数或生成的次数，传输过程中的错误以及后期制作过程中引入的艺术视觉效果。
- 人的意见是判断视频的视觉质量的最重要标准，但这些意见是主观的，易变的且无法可靠地重复执行。
- 诸如PSNR和SSIM之类的客观指标广泛用于视频质量评估。尽管它们与人的体验并

不完全相关，但是它们可以很好地估计视觉质量。但是，在判断编码器的输出时，仅凭这些客观手段是不够的。还必须考虑视频的带宽成本。

- 已经提出了许多视频质量评估的方法和指标，但他们的复杂程度各不相同。这是学术研究以及ITU新兴标准的活跃领域。
- 可以调整几个编码参数，以牺牲视频质量来获得性能和功耗的优化。这里的重要参数包括码率，帧率和延迟的要求，码率控制类型，可用缓冲区大小，图片结构，图片组，运动参数，参考图片数量，运动矢量精度，运动矢量搜索/插值方法，熵编码类型，编码次数，等等。
- 编码效率取决于编码所用的比特数要达到的质量。编码效率通常用于表示性能。但是，在本书中，性能是指编码速度。
- 编码速度取决于很多因素，包括平台和视频特性。平台特征包括：CPU和GPU频率，工作电压，可配置的TDP状态，操作系统电源策略，内存带宽和速度，缓存策略，磁盘访问速度，I/O吞吐量，系统时钟，图形驱动程序设置，等等。视频特征包括：视频格式，分辨率，码率，帧率，GOP和其他参数。视频场景特征包括：运动量，细节，亮度等。
- 可以利用各种并行化技术提高编码速度。但是，应认真分析任务调度和进程间通信的成本。并行化方法包括：数据分区，任务并行化，流水线，数据并行化，指令并行化，多线程，和向量化。
- 比实时编码更快的编码速度在诸如视频编辑、存档、录制、转码和格式转换等应用中很有用。
- 可视通讯和应用（投屏应用）通常需要在资源受限的客户端平台上进行低延迟的实时编码。
- 性能优化意味着最大程度地利用可用的系统资源。但是，具有功耗意识的优化方法会在尽可能短的时间内使资源利用率最大化，并使系统尽可能长时间地进入更深的睡眠状态。这与最小化资源的空闲时间的传统方法正好相反。
- 在各种性能优化方法中，需要特别关注：算法优化，代码优化，编译器优化以及冗余消除。
- 系统存在多个电源管理点，包括BIOS，CPU，图形控制器，硬盘驱动器，网络和显示器。系统还可能存在内存电源管理，但是内存电源管理很少执行。
- 通常，基于硬件的电源管理涉及各种CPU C状态和渲染C状态。操作系统或驱动程序中，基于软件的电源管理包括：CPU核离线，CPU核屏蔽，CPU负载平衡，中断负载平衡，CPU和GPU频率控制等。
- 在低功耗平台上，通常需要特殊的硬件单元管理电源。不同电压等级的多个电源点构成一个复杂的系统，这些专用单元可对电源需求进行快速而精确的管理。
- 电源管理的目标是使处理器尽可能长时间地进入各种睡眠状态，从而节省功耗。
- 总功耗包括动态功耗和静态泄漏功耗。动态功耗取决于工作电压和频率，而静态功耗取决于泄漏电流。
- 无论频率如何变化，电路运行都需要最低电压。处理器可以在最小电压下工作的最大

频率 ($F_{max}@V_{min}$) 是最省电的工作点。从这一点开始增加频率会以立方速率增加动态功耗，而以线性速率增加静态功耗。在相对较高的功耗下，可以通过简单的权衡电压-频率来降低功耗。将频率降低到最有效点（即 V_{min} 区域）以下会线性降低动态功耗，而静态功耗则保持恒定，从而产生恒定的泄漏电流。

- 可以在体系结构级别，算法级别，系统集成级别和应用程序级别优化功耗。
- 在低功耗平台，可能需要在处理器面积，功耗，性能，视觉质量，压缩量和设计复杂度之间进行一些实际的权衡。为了节省这些平台的功耗，可能有必要牺牲视觉质量。
- 对于视频应用，显示器消耗了很大比例（大多为1/3）的系统功耗。显示功耗管理技术包括：面板自刷新，使用英特尔显示节能技术的背光控制，环境光传感器和内容自适应性。
- 低功耗软件设计的注意事项包括：智能功耗意识，质量需求，硬件加速的可用性，高能效UI，代码密度和内存占用量，数据传输，缓存利用率的优化，并行和批处理等。
- 低功耗架构方面的考虑因素包括：在同一芯片上组合系统组件，优化的硬件-软件交互，工作负载从通用硬件迁移到固定功能硬件，CPU-GPU功耗共享，uncore和图形单元，功率岛，功耗感知仿真和验证等。
- 功耗优化方法包括：快速运行和关闭处理器，调度任务和活动，减少唤醒，突发模式处理，减少CPU-GPU依赖性并增加并行性，GPU内存带宽优化以及显示器和存储单元的功耗优化。
- 需要测量功耗和性能以进行权衡分析，可以校准系统并为如下参数提供合适的设置：工作温度，电压和频率，cTDP，交流或直流电源模式，操作系统电源策略，显示设置，驱动程序设置，应用程序设置，编码参数。
- 第8章讨论的权衡分析试图填补目前综合分析方法中存在的空白。特别是对于检查调整各种参数的影响以更好地了解不同视频措施的成本和收益而言非常重要。
- 了解性能，功耗和质量之间的折衷对架构师、开发人员、验证者和技术市场营销人员、编码解决方案的技术评审人员、采购人员和最终用户都很重要。

对未来的思考

我希望本书所涵盖的主题能够激发人们的讨论，从而将我们带入视频编码及其相关分析的未来。以下是将来可能会扩展到的一些领域。

增强的分析工具和指标

尽管可以在给定的编码测试的执行过程中查看性能、功耗和质量的详细信息，并了解它们之间的关系，但要确定给定指标为何会增加或减少却并非易事。当比较两个测试的结果时，确定不同编码方案的指标之间的差异尤其困难。看起来，这些差异可能因为编码器方案的不同而导致的。类似地，在同一实验中比较两个指标时，对于为什么指标彼此间会有增加或减少并不总是非常清楚。复杂性源于存在的许多变量，这些变量会对系统或视频编码参数的变化做出不确定的反应，并且相互影响。同样，这些影响对于不同的视频内容，应用程序和使用情况也不尽相同。需要进行研究并认真、花费很长时间来调试，以便能够理解这些复杂的关系。

研究人员正在尝试提出更好的视频指标和分数，尤其是在视觉质量、压缩、性能和功耗方面。分析技术有望适应未来更全面的指标。最终，所有收益将只有一个衡量指标，而视频编码的所有成本也只有一个衡量指标，并且该衡量指标将被普遍接受并用对编解码器进行评估和排名。随着新指标的提出，考虑到视频编码各个方面的增强的基准测试工具也有望实现。

改善的性能和质量

相同压缩量下提高视觉质量的技术将会持续改进。在过去的几十年中，这种趋势从MPEG-2到AVC，从AVC到HEVC非常明显。同样，性能和功耗的优化技术的改进速度甚至比质量改进的速度还要快。与上一代GPU加速视频编码相比，每一代Intel处理器在相同的功耗曲线下都能产生大约30%~200%的性能提升。即使是今天的低功耗处理器，也能够支持十年前梦想中的视频应用。可以认为，经过适当的权衡和优化，尽管平台受到限制，但日常视频应用仍将具有更好的视觉质量。

新型的应用

可穿戴设备在功耗和性能方面提出了独特的挑战，但这些新兴计算平台的新用途每天都在出现。视频是一个开放的研究领域。它包含以下概念：如何确定针对这些用途的视频编码的良好程度，如何量化它们以及使用哪些指标来量化。

无人驾驶汽车和无线电遥控无人机领域的视频编码的能力、用途和要求也在评估和开发之中。随着视频编码在资源受限系统上的处理能力的不断提高，权衡分析和优化将在设计和应用中扮演重要角色。但是，对于这些用途的方法和度量标准仍未定义。

远程医疗也处于起步阶段。高分辨率视频的压缩和通信技术日趋成熟，最终可以完美运用于远程外科手术的手持设备上。在这些情况下，性能，功耗和质量也是需要权衡的因素。

从视觉到其他感官

视觉被认为是人类五种感官中最重要的感官，但是仅依靠视觉并不能完成人类的体验。因此，视频并不是数字多媒体应用的唯一数据类型。通常，音频和视频是一起发展的，触摸和手势也在迅速发展。因此，对其他感官的测量，理解和调整将包括音频，触摸和手势。这些基于感觉的数据类型之间的关系非常复杂，需要进行深入的分析，详细的研究以及最终的权衡。

这仍然是另一个活跃的研究领域。未来，随着不断解决这些挑战，我们将体验到人类感官的真正的、全部的潜能。