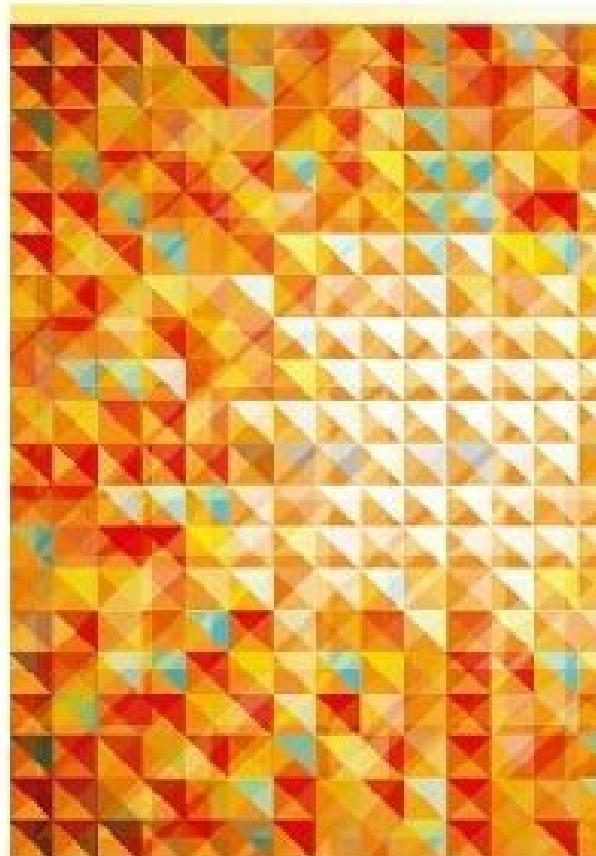




高端图像与视频新技术丛书

H.265/HEVC 视频编码 新标准及其扩展

▶ 朱秀昌 刘峰 胡栋 编著



在给出最新视频编码国际标准H.265/HEVC及其扩展部分全貌的基础上，着重介绍相关的视频编码原理、高效编码工具、高层语法语义和主要技术规范。



中国工信出版集团



电子工业出版社
<http://www.phei.com.cn>

高端图像与视频新技术丛书

H.265/HEVC——视频编码新标准及其扩展

朱秀昌 刘峰 胡栋 编著

电子工业出版社

Publishing House of Electronics Industry

北京·BEIJING

内容简介

本书从视频编码国际标准，特别是HEVC标准所涵盖的技术内容出发，主要介绍了新一代视频编码标准所采用的技术规范、基本工作原理、核心编码技术和新的编码工具。同时还介绍了HEVC标准新近扩展部分的主要内容，包括扩展的技术、工具和指标。

全书共13章，第1、2章分别简要介绍了视频编码的基础知识和历年来的国际标准；第3~10章，分8个部分按顺序介绍了HEVC的核心技术内容，包括编码结构、帧内预测、帧间预测、变换和量化、熵编码、环路编码、并行处理和高层语法；第11、12章介绍了HEVC标准的最新扩展内容，即多层编码、可分级编码、多视点编码和3D编码；第13章从应用的角度简要介绍了HEVC的实现复杂度和软硬件实现的考虑。

本书可作为从事信号处理、通信工程、计算机应用、广播电视、机器人视觉及自动控制等领域的工程技术人员、大专院校相关专业的教师、高年级学生或研究生学习、理解和掌握视频编码最新国际标准和关键编码技术的参考用书。

未经许可，不得以任何方式复制或抄袭本书之部分或全部内容。

版权所有，侵权必究。

图书在版编目（CIP）数据

H.265/HEVC：视频编码新标准及其扩展/朱秀昌，刘峰，胡栋编著.—北京：电子工业出版社，2016.7

ISBN 978-7-121-29038-1

I.①H... II.①朱...②刘...③胡... III.①视频编码—研究 IV.①TN762

中国版本图书馆CIP数据核字（2016）第128704号

责任编辑：赵娜

印刷：

装订：

出版发行：电子工业出版社 北京市海淀区万寿路173信箱

邮编 100036

开本：787×1092 1/16

印张：21.75

字数：550千字

版次：2016年7月第1版

印次：2016年7月第1次印刷

定价：59.00元

凡所购买电子工业出版社图书有缺损问题，请向购买书店调换。若书店售缺，请与本社发行部联系，联系及邮购电话：（010）88254888，88258888。

质量投诉请发邮件至zlt@phei.com.cn，盗版侵权举报请发邮件至dbqq@phei.com.cn。

本书咨询联系方式：（010）88254694。

前言

伴随着大数据的洪流和无处无时不在的移动通信发展，以及互联网、物联网的全面覆盖，一个以大数据、大流通为特征的信息时代已经开启。由于视频信息具有真实、具体、生动、全面、信息量大等特点，人们对各种视频信息的需求与日俱增，使得大数据中海量视频信息的占比已过半，因此，保证海量视频信息的高效、优质、标准化的压缩日显重要。

目前视频压缩编码技术面临着三方面的压力：一是由于视频分辨率迅速提高而引起的数据量激增，如第一代视频编码标准面对的基本上是 352×288 的CIF视频，第二代视频编码标准面对的主要也是 768×576 的SDTV视频，如今第三代视频编码标准面对最低的也是 1920×1080 的HDTV视频，需压缩视频的数据量大约是以每代4倍的速度向上翻，而视频编码效率，每代大致翻一番，这里还未考虑超高清视频、3D视频、多视点视频等数据量更大的情况。二是由于视频应用的迅速普及而引起的视频信息需求激增，现在各行各业都需要视频信息，几乎每个人都需要视频信息，移动视频还带来每时每刻和无处不在的视频需求，这些需求使得视频采集、传输、存储的数量不断增长。三是视频压缩标准化带来的压力，各方面的视频信息需要相互交流，需要有标准化的格式来保证各种类型的视频信息的畅通，这就要求制定统一的国际标准，并随着技术的发展而不断修订、扩展和换代。

尽管半导体芯片的密度不断提高，网络带宽也不断增加，可以减轻一部分对视频编码的压力，但是远远消解不了上述三方面给视频编码带来的压力。虽然目前已有一系列的国际视频编码标准，但其效率仍然远不能满足当前视频信息的应用需求。诚然，这些都是标准制定者面临的巨大压

力，同时，这也理所当然地成为从事视频编码的工程技术人员不断研究新技术、制定新标准的强大驱动力。

为达到对视频数据的高效、标准化压缩的目的，从20世纪80年代末以来，国际电联（ITU）和国际标准化组织（ISO）先是分别、后是联手矢志不渝地进行视频编码国际标准的制定工作。迄今为止，已历经三代。第一代以ITU-T的H.261、H.263建议和ISO/IEC的MPEG-1、MPEG-2标准为代表；第二代则是以两大组织联合制定的H.264/MPEG-4 AVC标准为代表；第三代也是以两大组织联合制定的并正在扩展中的HEVC标准为代表。

HEVC即“高效视频编码”是该标准的俗称或统称，在ITU-T称其为H.265建议，在ISO/IEC称其为23008-2标准。虽然名称各异，但其中的内容是一样的，现已更新为包括扩展部分的第3版。本书主要介绍新一代视频编码国际标准HEVC及其扩展中的主要压缩编码原理、高效编码工具、高层语法语义和主要技术规范。本书的要点和特点如下。

扼要介绍了和国际标准相关的视频编码原理，梳理了近30年来一系列视频编码的国际标准，简述了这些标准体系及发展历程演进。着重分析了HEVC第1版的关键编码工具，包括编码结构、帧内预测、帧间预测、变换和量化、熵编码、环路滤波、高层语法、并行处理等。在HEVC实现方面也给出了简单的说明和几个实例。针对HEVC第2版的扩展，介绍了在统一的多层编码概念、可分级视频编码、多视点和3D视频编码方面的核心编码技术的要点。HEVC在总体上继承了H.264/AVC，但在各个方面都有新发展，使得总编码效率提高了一倍，在适当的章节与H.264/AVC标准进行了简单的对比，有助于加深对HEVC的理解。

本书以从事计算机技术、通信技术以及电视技术的工程技术人员、大专院校有关专业的高年级学生、研究生或教师为主要阅读对象。全书的内容可以划分为四个部分。

第一部分（第1、2章）。第1章介绍和视频编码有关的基本概念和相关技术，如视频的数字化，数字视频信号的统计特性，混合编码的框架和主

要方法，图像质量的评价等。在此基础上，第2章介绍了ISO和ITU-T两大国际标准化组织的H.261/263和MPEG-1/2第一代标准、H.264/MPEG-4 AVC第二代标准，以及当前HEVC第三代视频编码国际标准的总体概况和标准化进程。

第二部分（第3~10章）。主要介绍了HEVC第1版（2013年3月）中重要编码工具的技术原理和技术要求。包括：第3章的视频编码结构，重点是四叉树图像的划分技术；第4章的帧内预测编码，重点是高达35种方向的帧内预测技术；第5章的帧间预测编码，重点是运动信息的高效Merge模式的生成技术；第6章的变换和量化，重点是多尺寸DCT、小尺寸DST和改进的量化、伸缩方式；第7章的熵编码，重点是高效的基于上下文的二进算术编码，尤其是它的常规模式、旁路模式和上下文模型更新技术；第8章的环路滤波，重点是简洁高效的去方块滤波和新增的样点自适应补偿技术；第9章的并行处理，重点是新增的基于片和基于波前并行处理技术；第10章的高层语法，重点是改进的NAL单元表示、新增的图像类型和参数集技术。

第三部分（第11、12章）。这两章是有关HEVC第2版（2014年10月）和第3版（2015年4月）中新编码扩展部分内容。其中第11章主要介绍了有关统一的多层编码概念和可分级视频编码的关键技术。第12章主要介绍了有关多视点视频编码和3D视频编码的关键技术。

第四部分（第13章）。简单介绍HEVC的参考软件（HM），编码复杂度比较，软硬件实现的设计考虑，给出了几个HEVC编解码实现的简例。

在本书的编写过程中，得到我们研究团队的多位教师和研究生的帮助和参与，引用了他人的研究报告、著作和论文，具体出处在书后的参考文献中一一列出。在此，对有助于本书编写的师生和参考文献的作者一并表示深切的谢意。对在本书选题、确定内容和编辑整理过程中付出辛勤劳动的董亚峰编辑表示由衷的感谢。

由于当今视频编码技术的飞速发展，同时受限于作者的学术水平，加上编写时间仓促，书中的错误和不足之处在所难免，真诚地欢迎广大读者予以批评指正。

作者

2015年10月于南京

目 录

[封面](#)

[扉页](#)

[版权信息](#)

[前言](#)

[第1章 视频编码基础](#)

[1.1 数字视频信号](#)

[1.1.1 视频信号的采集](#)

[1.1.2 视频信号的数字化](#)

[1.1.3 视频信号的显示](#)

[1.1.4 数字视频的格式](#)

[1.1.5 高清和超高清视频](#)

[1.2 视频信号的统计特性](#)

[1.2.1 图像的自相关函数](#)

[1.2.2 像素差值的自相关函数](#)

[1.3 混合编码](#)

[1.3.1 预测编码](#)

[1.3.2 变换编码](#)

[1.3.3 运动估计和运动补偿](#)

[1.3.4 混合编码框架](#)

[1.4 量化和熵编码](#)

[1.4.1 量化](#)

[1.4.2 Zig-zag 扫描](#)

[1.4.3 熵编码](#)

[1.5 率失真优化](#)

[1.5.1 图像的信源熵](#)

[1.5.2 率失真定理](#)

[1.5.3 失真率函数](#)

1.5.4 有记忆信源的处理

1.5.5 率失真优化编码

1.6 图像质量的评价

1.6.1 主观质量评价方法

1.6.2 客观质量评价方法

1.6.3 SSIM质量评价方法

本章参考文献

第2章 视频编码的国际标准

2.1 H.26x标准

2.1.1 H.261标准

2.1.2 H.263标准

2.2 MPEG-x标准

2.2.1 MPEG-1标准

2.2.2 MPEG-2标准

2.2.3 MPEG-4标准

2.3 H.264/AVC标准

2.3.1 多方向帧内预测

2.3.2 多模式运动估计

2.3.3 整数变换和熵编码

2.3.4 差错控制

2.4 AVS标准

2.5 VC-1标准

2.6 HEVC标准

2.6.1 HEVC标准的进程

2.6.2 HEVC技术概要

本章参考文献

第3章 HEVC的编码结构

3.1 H.264/AVC的编码结构

3.1.1 宏块灵活划分

3.1.2 图像的条划分

3.1.3 档次和水平

3.2 HEVC的网络适配和编码方式

3.2.1 视频编码层和网络提取层

3.2.2 三种编码方式

3.3 HEVC的四叉树划分

3.3.1 图像的取样格式

3.3.2 编码树单元和编码单元划分

3.3.3 预测单元划分

3.3.4 变换单元划分

3.3.5 CTU划分实例

3.4 HEVC的条和片划分

3.4.1 条划分

3.4.2 片划分

3.4.3 条/片划分实例

3.5 HEVC的档次、水平和等级

3.5.1 档次

3.5.2 水平

3.5.3 等级

本章参考文献

第4章 HEVC的帧内预测

4.1 帧内编码

4.1.1 空域预测编码

4.1.2 最佳线性预测

4.2 H.264/AVC的帧内预测

4.2.1 亮度 4×4 块的预测模式

4.2.2 亮度 16×16 块的预测模式

4.2.3 色度 8×8 块的预测模式

4.3 HEVC的帧内预测模式

4.3.1 帧内预测PU的划分

4.3.2 亮度PU的帧内预测模式

4.3.3 色度PU的帧内预测模式

4.4 HEVC的帧内预测过程

4.4.1 参考像素的准备

4.4.2 参考像素的平滑滤波

4.4.3 计算预测值

4.4.4 边界值的平滑

4.4.5 模式信息的编码

本章参考文献

第5章 HEVC的帧间预测

5.1 帧间预测编码

5.1.1 帧间预测方式

5.1.2 基于块的运动估计

5.1.3 运动矢量的预测

5.2 H.264/AVC的帧间预测

5.2.1 多模式宏块划分

5.2.2 高精度运动估计

5.2.3 双向预测条

5.3 HEVC的帧间预测

5.3.1 帧间预测PU的划分

5.3.2 子像素插值

5.4 HEVC的运动参数编码

5.4.1 运动参数的编码传送

5.4.2 Merge模式

5.4.3 Skip模式

5.4.4 Inter模式

5.4.5 帧间预测模式的选择

本章参考文献

第6章 HEVC的变换和量化

6.1 变换与量化

6.1.1 离散余弦变换和正弦变换

6.1.2 量化和量化失真

6.2 H.264/AVC的变换与量化

6.2.1 4×4整数DCT变换

6.2.2 变换系数的量化

6.3 HEVC残差的整数变换

6.3.1 残差四叉树 (RQT)

6.3.2 整数DCT变换

6.3.3 4×4整数DST变换

6.4 HEVC变换系数的量化

6.4.1 量化参数和量化步长

6.4.2 量化和反量化计算

6.4.3 加权量化矩阵

6.5 HEVC变换块的编码表示

6.5.1 量化后系数的扫描

6.5.2 变换系数的表示

6.5.3 变换跳过

本章参考文献

第7章 HEVC的熵编码

7.1 熵编码

7.1.1 熵编码的要求

7.1.2 定长编码

7.1.3 变长编码

7.2 算术编码

7.2.1 一般算术编码

7.2.2 自适应算术编码

7.2.3 二进制算术编码

7.2.4 自适应二进制算术编码

7.3 HEVC的算术编码

7.3.1 CABAC框架

7.3.2 二进制化

[7.3.3 上下文模型](#)

[7.3.4 常规编码模式](#)

[7.3.5 旁路编码模式](#)

[7.4 上下文建模和更新](#)

[7.4.1 上下文关系](#)

[7.4.2 上下文模型的初始化](#)

[7.4.3 上下文模型的更新](#)

[本章参考文献](#)

[第8章 HEVC的环路滤波](#)

[8.1 环路滤波](#)

[8.1.1 方块效应的产生](#)

[8.1.2 环内滤波和环外滤波](#)

[8.2 H.264/AVC的去方块滤波](#)

[8.2.1 自适应去方块滤波](#)

[8.2.2 边界强度测定](#)

[8.2.3 去方块滤波过程](#)

[8.3 HEVC的环路滤波](#)

[8.3.1 自适应去方块滤波](#)

[8.3.2 样点自适应补偿](#)

[8.4 HEVC的去方块滤波](#)

[8.4.1 去方块滤波单元](#)

[8.4.2 边界强度的判定](#)

[8.4.3 滤波强度的判定](#)

[8.4.4 去方块滤波过程](#)

[8.5 HEVC的样值自适应补偿](#)

[8.5.1 信号失真及补偿](#)

[8.5.2 SAO的两种模式](#)

[8.5.3 带补偿 \(BO \) 模式](#)

[8.5.4 边缘补偿 \(EO \) 模式](#)

[8.5.5 SAO的模式选择和参数共享](#)

本章参考文献

第9章 HEVC的并行处理

9.1 视频编码的并行处理

9.1.1 并行处理的主要方式

9.1.2 功能并行

9.1.3 数据并行

9.1.4 流水线并行

9.2 HEVC的并行处理工具

9.2.1 片并行处理

9.2.2 波前并行处理

9.3 HEVC的各级并行处理

9.3.1 GOP级并行处理

9.3.2 图像级并行处理

9.3.3 条、片级并行处理

9.3.4 块级并行处理

9.3.5 指令级并行处理

9.4 去方块滤波的并行处理

本章参考文献

第10章 HEVC的高层语法

10.1 HEVC语法特点

10.1.1 新增语法结构和元素

10.1.2 基本语法表示

10.2 H.264/AVC语法提要

10.2.1 码流的分层结构

10.2.2 NAL单元语法

10.2.3 Slice语法

10.2.4 参数集

10.3 HEVC的NAL单元

10.3.1 字节流格式

10.3.2 一般NAL单元语法

10.3.3 NAL单元头语法

10.4 HEVC的接入图像

10.4.1 帧内随机接入图像

10.4.2 前置图像

10.4.3 后置图像

10.5 HEVC的参数集

10.5.1 三类参数集

10.5.2 视频参数集 (VPS)

10.5.3 序列参数集 (SPS)

10.5.4 图像参数集 (PPS)

10.6 HEVC的参考图像集

10.6.1 参考图像集

10.6.2 参考图像列表

10.7 HEVC的SEI和VUI

10.7.1 补充增强信息 (SEI)

10.7.2 视频可用信息 (VUI)

本章参考文献

第11章 HEVC的多层和可分级编码扩展

11.1 HEVC编码扩展的进程

11.2 HEVC统一的多层编码

11.2.1 多层编码的结构

11.2.2 多层编码的工具

11.3 HEVC的多层扩展

11.3.1 层和子层

11.3.2 接入单元

11.3.3 视频参数集扩展

11.4 可分级视频编码

11.4.1 常用可分级编码方法

11.4.2 H.264/AVC的可分级编码

11.5 HEVC的可分级扩展

[11.5.1 SHVC的编码框架和性能](#)

[11.5.2 上采样滤波器](#)

[11.5.3 层间纹理预测](#)

[11.5.4 层间运动预测](#)

[11.5.5 SHVC编码一例](#)

[本章参考文献](#)

[第12章 HEVC的多视点和3D编码扩展](#)

[12.1 立体视频编码](#)

[12.1.1 立体视频和多视点视频](#)

[12.1.2 立体视频的采集和显示](#)

[12.1.3 多视点视频编码](#)

[12.1.4 H.264/AVC的多视点编码](#)

[12.2 HEVC的多视点扩展](#)

[12.2.1 MV-HEVC编码系统](#)

[12.2.2 多视点编码工具](#)

[12.2.3 虚拟视点的合成](#)

[12.3 HEVC的3D扩展](#)

[12.3.1 立体图像的深度图](#)

[12.3.2 深度图的编码](#)

[12.3.3 深度图的编码工具](#)

[本章参考文献](#)

[第13章 HEVC的实现](#)

[13.1 HEVC的参考软件HM](#)

[13.2 HEVC的复杂度](#)

[13.2.1 功能单元的复杂度](#)

[13.2.2 HM的编码复杂度](#)

[13.2.3 HM的解码复杂度](#)

[13.2.4 和H.264/AVC比较](#)

[13.3 HEVC编码器的实现考虑](#)

[13.3.1 软件实现考虑](#)

13.3.2 硬件实现考虑

13.4 HEVC的解码实验

13.4.1 HEVC的测试序列

13.4.2 基于ARM的解码

13.4.3 基于X86的解码

13.4.4 解码性能分析

13.5 HEVC的编解码器简例

13.5.1 基于DSP的HEVC解码器

13.5.2 HEVC解码器芯片

13.5.3 HEVC编码器芯片

13.5.4 HEVC编码系统

本章参考文献

缩略语 (Abbreviations)

反侵权盗版声明

第1章 视频编码基础

在信息化时代，视频技术和应用的发展，特别是高清（HD）、超高清（UHD）、3D和多视点（Multi-View）视频技术的兴起，海量的视频信息已如潮涌般深入到我们工作和生活的方方面面。据统计，视频流数据在2015年占到了整个互联网流量的90%左右，说夸张一点，整个互联网络差不多就是视频网了。尽管近年来网络带宽和传输能力增加迅速，但仍远不能满足海量视频数据的传输和存储要求。因此视频信息量的高效压缩过去是、现在是、在可预见的未来也必然是解决这一矛盾的重要技术措施之一。

本书通过对最新的视频编码国际标准High Efficiency Video Coding,HEVC，及其扩展的介绍，帮助读者了解和国际标准有关的视频编码的基本原理、主要规范和关键技术。视频编码（Video Coding）又称视频压缩（Video Compression）技术，属于数字视频信号处理的范畴，要想掌握视频编码的原理和技术，必须对数字视频处理方面的相关基本内容有一定的了解。本章将提供这方面的一些基本内容，包括视频信号的采集、显示和数字化，视频信号统计特性，图像质量评价方法，以及最基本的视频编码技术。

1.1 数字视频信号

随着视频技术的发展，传统的模拟摄像机已经逐步被淘汰，可以直接获取数字图像的数字摄像机等已占据主流地位，大大方便了人们对视频信号的采集、处理、存储和传输。但数字视频设备的普及，并不表示视频的数字化不再重要，这是因为视频信号的“源”（来源）和“宿”（归宿）的形式都是连续的、模拟的，只是在中间的处理过程中采用了数字化的方法。所以，了解视频的数字化过程有助于对数字视频的采集和显示有深入的了解，从而有助于对数字视频信号的编码处理有深入的理解。

1.1.1 视频信号的采集

摄像机是视频信号的主要来源，随着数字化的进程，数字摄像机已经成为主流的视频信号来源，它输出的是数字视频信号或压缩的数字视频信号，同时也可以输出模拟视频信号。数字摄像机和模拟摄像机的光学镜头和感光器件部分基本是一样的，两者的区别在光电转换后续的处理电路不同，下面进行简单介绍。

1. 模拟摄像机

早期的光导管模拟摄像机已遭淘汰，现在常用的是电荷耦合器件（Charge Coupled Device,CCD）和互补氧化金属半导体（Complementary Metal-Oxide Semiconductor,CMOS）摄像机。

CCD摄像机内的核心部件是一种固态半导体面阵集成电路，即CCD感光芯片，它由若干行、若干列的离散感光单元排列而成传感阵列。摄像机所对准的场景的光线通过镜头聚焦投射到阵列上，每个感光单元由于光照的作用而产生与输入光强成正比的输出电荷。这些电荷通过适当的逻辑电

路，按照逐单元、逐行的顺序，在一帧的时间内将整个阵列的所有单元的电压送出，经AD变换、信号处理和DA变换后形成标准的模拟视频信号输出。

CMOS 摄像机的传感器的工作过程和 CCD 类似，但集成度比较高，处理过程比较简单，因此采集速度可以做到每秒数百帧，功耗也比较小，耗电量大约为同类CCD的1/3。CMOS的不足之处在于传感噪声比CCD高，但这一差距正在逐步缩小，所以今天的便携式摄像设备几乎都使用CMOS传感器。

实际上，无论CCD还是CMOS本身都没有参与图像感应（光电转换），它们使用的是同一类“光电二极管”传感器（半导体PN结），能够转换光线的强弱为电压的高低，光线进入图像半导体越多，电子产生的电子也越多，从传感器输出的电压也越高。而CCD和CMOS主要作用是将图像半导体中出来的电信信号快速地储存、传输到信号处理部分。

感光平面阵列中每一行基元数（像素数）的多少和行数的多少，决定了所摄图像清晰度的高低。常用的CCD、CMOS摄像机的像素数在40~800万，如1080p视频（1920×1080）的像素数约为200万。

2.数字摄像机

数字视频信号有两种获得的途径，一种是直接的方式，另一种是间接的方式。所谓间接方式是指将模拟视频信号数字化以后产生数字视频。近年来随着电子领域数字化的进程，越来越多的直接输出数字视频信号的数字摄像机已成主流。这样的摄像机可以直接和计算机、数字图像设备相连接，而不需要经过A/D转换。

数字摄像机和模拟摄像机的构成中感光部分基本相同，不同的是数字摄像机增加了数字化信号处理部分。数字摄像机对经过CCD（或CMOS）光电转换得到的视频信号进行数字化，获得数字视频信号输出，或再经过数字信号处理、数据压缩，最终输出已压缩的数字视频信号。在这个意义上可以将数字摄像机称为“数字处理摄像机”。

现在大多数的数字摄像机都具备符合IEEE1394接口和HDMI接口规范

的输出。IEEE1394俗称“火线”(Fire Wire)接口，HDMI(High Definition Multimedia Interface)是高清多媒体接口，包括了数字视频和音频数据。它们已普遍用于和PC机或其他设备相连，高速传送视音频信号。当然，这类摄像通常还带有普通模拟复合视频输出及S-Video分量视频输出。

1.1.2 视频信号的数字化

通常意义上的图像是光强度的二维分布，是平面坐标(x,y)的函数 $f(x,y)$ 。如果是一幅彩色图像，函数值还应反映出色彩变化，可用 $f(x,y,\lambda)$ 表示，其中 λ 为波长；如果是活动彩色图像，即视频图像，还应是时间 t 的函数，可表示为 $f(x,y,\lambda,t)$ ；当然，如果是立体视频图像，还应是空间深度 z 的函数，可表示为 $f(x,y,z,\lambda,t)$ 。对常规图像来说， $f()$ 是一个连续、非负、有界函数，即 $0 \leq f(x,y,z,\lambda,t) < \infty$ 。

视频图像的连续性包含了三个方面的含义，即空间位置延续的连续性，每一个位置上光强度变化的连续性和时间方向变化的连续性。连续的视频图像无法用计算机等数字化设备进行处理、传输或存储，所以必须将连续(模拟)的视频信号转变为离散(数字)的视频信号，这个转变的过程称为视频信号的数字化，它一般包含三项操作：图像的采样、样值的量化和量化值的编码。

1.图像的采样

图像在空间上的离散化过程称为采样或取样(Sampling)，被选取的点称为采样点或样点，也称为像素(Pixel)。在采样点上的函数值称为采样值或样值。采样实际上就是用空间有限采样点的值来代替连续坐标上的函数值。一幅图像应取多少样点才能够完全由这些样点来重建原图像呢？如果样点取得过多，增加了用于表示这些样点的信息量；如果样点取得过少，则有可能会丢失原图像所包含的信息。所以最少的样点数应该满足一定的约束条件：由这些样点，采用某种方法能够完全重建原图像。实际上，这就是二维采样定理的内容。采样定理和信号的频谱有关，下面从图

像信号的频谱说起。

(1) 图像信号的频谱

定义二维图像信号 $f(x, y)$ 的傅里叶 (Fourier) 变换和反变换分别为

$$F(u, v) = \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} f(x, y) e^{-j2\pi(ux+vy)} dx dy \quad (1.1)$$

$$f(x, y) = \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} F(u, v) e^{j2\pi(ux+vy)} du dv \quad (1.2)$$

其中， u, v 是频率域坐标， $F(u, v)$ 称作 $f(x, y)$ 的傅里叶变换或频谱，它表明了空间频率成分与二维图像信号之间的相互关系，其物理意义是 $f(x, y)$ 可由频率域上的各谐波分量迭加得到。

实际的二维图像，其傅里叶变换一般是在频率域上是有界的，即信号频谱的有效成分总是落在一定的频率域范围之内。主要依据在于：图像中景物的复杂性具有一定的限度，其中大部分内容是变化不大的区域，完全像“雪花”点似的图像没有任何实际意义。另外，人眼对空间细节的分辨能力以及显示器的分辨能力都具有一定的限度。因而在频率域上观察，通常图像的频谱大多局限在一定的范围内，过高的频率分量没有太大的实际意义。

(2) 二维采样定理

图像采样要解决的问题是找出能从采样图像精确地恢复原图像所需要的水平和垂直方向采样点的最大间隔，这一问题由二维奈奎斯特（Nyquist）采样定理解决，它可看作一维采样定理的推广。

这里参照图1.1简述一下二维采样定理。图1.1（a）为原始的模拟图像 $f(x,y)$ ，其傅里叶频谱 $F(u,v)$ 如图1.1（b）所示，它在水平方向的截止频率为 u_m ，在垂直方向的截止频率为 v_m ，则只要水平方向的空间采样频率 $u_0 \geq 2u_m$ ，垂直方向的空间采样频率 $v_0 \geq 2v_m$ ，即采样点之间的水平间隔 $\Delta x \leq 1/(2u_m)$ ，垂直间隔 $\Delta y \leq 1/(2v_m)$ ，图像就有可能被精确地恢复。

对理想采样而言，在二维空间用冲激函数阵列 $s(x,y)$ 作为采样函数，这些冲激水平方向之间的距离为 Δx ，垂直方向之间的距离为 Δy ， $s(x,y)$ 可定义为

$$s(x,y) = \sum_{i=-\infty}^{\infty} \sum_{j=-\infty}^{\infty} \delta(x - i\Delta x, y - j\Delta y) \quad (1.3)$$

其中 $\delta(x,y)$ 为二维冲激函数，用理想冲激函数阵列对连续图像进行

采样后的图像为

$$f_p(x, y) = f(x, y) \cdot s(x, y) = \sum_{i=-\infty}^{\infty} \sum_{j=-\infty}^{\infty} f(i\Delta x, j\Delta y) \cdot \delta(x - i\Delta x, y - j\Delta y) \quad (1.4)$$

在频域，采样图像的傅里叶频谱 $F_p(u, v)$ 为

$$F_p(u, v) = \frac{1}{\Delta x \Delta y} \sum_{i=-\infty}^{\infty} \sum_{j=-\infty}^{\infty} F(u - i\Delta u, v - j\Delta v) \quad (1.5)$$

由此可见，采样后的频谱是原频谱 $F(u,v)$ 在 u 、 v 平面内按 $\Delta u=1/\Delta x$ 、 $\Delta v=1/\Delta y$ 周期无限重复，如图1.1(c)所示。

因此，若原图像频谱是限带的，且 Δx 、 Δy 取得足够小，使 $\Delta u \geq 2u_m$, $\Delta v \geq 2v_m$ ，则采样后频谱将不会重叠，通过低通滤波的方法就可以完全恢复原图像。

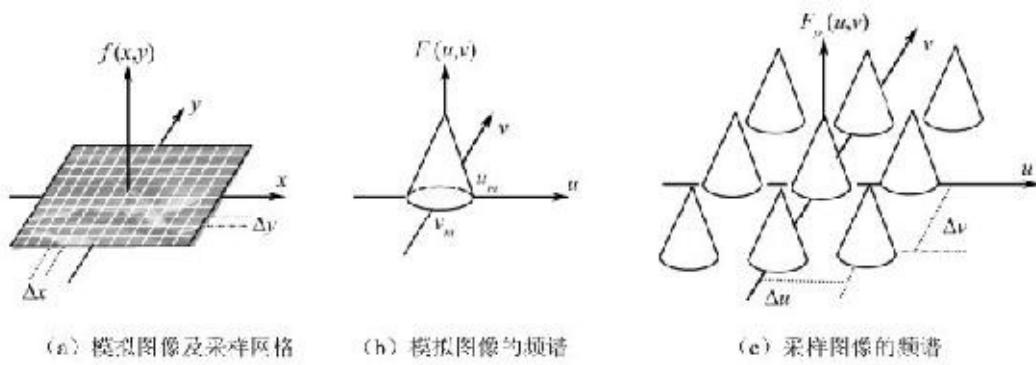


图1.1 采样图像的频谱

(3) 由采样值恢复原图像

如图1.1(c)所示，在满足采样定理条件下，各周期延拓的频谱区域互不交叠，为了从二维采样恢复原图像，只要用一个中心位于原点的理想

二维低通滤波器就可以完整地将频谱中的各个高次谐波滤除，由剩下的基频分量就可以恢复原始图像。理想低通滤波器的特性为

$$H(u, v) = \begin{cases} \Delta x \Delta y & |u| \leq u_m, |v| \leq v_m \\ 0 & \text{else} \end{cases} \quad (1.6)$$

显然，恢复图像的频谱 $F_r(u, v)$ 应该等于采样图像的频谱 $F_p(u, v)$ 和低通滤波器 $H(u, v)$ 的乘积，即

$$F_r(u, v) = F_p(u, v) \cdot H(u, v) = F(u, v) \quad (1.7)$$

可见，有了 $F_r(u,v) = F(u,v)$ ，即可通过傅里叶反变换完全恢复原图像 $f(x,y)$ 。

2. 样值的量化

经过采样的图像，只是在空间上被离散成像素（样点）的阵列，每个样点的灰度值还是一个有无穷多取值可能的连续变化量，必须将其转化为有限个离散值才能被赋予不同码字从而成为数字图像，这种转化称为量化（Quantization）。如果对每个样值进行独立处理，称为标量（Scalable）量化；如果对多个样值联合起来进行处理，则称为矢量（Vector）量化。这里主要讨论常见的标量量化，它包括两种方式：一种是将样本的连续灰度值范围进行等间隔分层的均匀量化，另一种是不等间隔分层的非均匀量化。

以均匀量化为例，将整个取样点的样值范围等分为若干部分（量化级），分界处称为判决电平，两个判决电平之间的所有灰度值用一个量化值（量化电平）来表示。量化既然是以有限个离散值来近似表示无限多个连续量，就一定会产生误差，这就是量化误差，由此所产生的失真即量化失真或量化噪声。

当量化级数少到一定程度时，量化值与连续值之间的差值（量化误差）变得很显著，引起严重的图像失真，尤其会在原先亮度值缓慢变化的区域引起生硬的“伪轮廓”失真。随着量化级数的增加，由量化引起的失真逐渐减少，但量化级数的增加意味表示图像信息的数据量增加。因此，量化的级数最终是一种在失真和数据量之间的折中，希望在量化噪声对图像质量的影响可忽略的前提下用最少的量化级数进行量化。

3. 量化值的编码

(1) PCM编码

经过采样，连续图像实现了空间的离散化，形成样点；经过量化，样点的连续灰度值实现了离散化，形成离散的样点值。对于这样的离散以后的有限的灰度量，就可以用二进制或多进制的数字来表示了，这种表示就

是“编码”：用特定的符号来表示各样点的离散灰度值。最常见的编码方法是自然二进制编码，如十进制的0,1,2,3,...编码成二进制的000,001,010,011,

...

在量化中，通常量化级数K取为2的n次幂，即 $K=2^n$,n为自然数。这样，每个量化区间的量化电平可采用 n 位（比特）自然二进制码表示，形成最通用的脉冲编码调制（Pulse Code Modulation,PCM）编码。对于均匀量化，由于是等间隔分层，量化分层越多，每一层的间隔（又称量化步长）越小，量化误差也越小，但编码时占用的比特数就越多。例如采用8比特量化，将图像灰度等级均匀分为 $2^8=256$ 层，顺序从0 ~ 255，样值落在哪个分层内，就用该分层的顺序号作为样值的量化值输出，这就是图像中最常用的 PCM 编码。例如输入某一图像样点灰度值=127.2，则量化为127，可用二进制码01111111来表示。

（2）量化信噪比

在对采样值进行n 比特的线性PCM编码时，每个量化分层的间隔（量化步长）的相对值为 $1/2^n$ ，假定采样值在它的动态范围内的概率是均匀分布，则可以证明，量化误差的均方值 N_q （相当于功率）为

$$N_q = \left(\frac{1}{12}\right)\left(\frac{1}{2^n}\right)^2 = \frac{1}{12 \cdot 2^{2n}} \quad (1.8)$$

于是，参照信噪比的定义，将峰值信号功率 S_{pp} （其相对值为1）与量化均方噪声 N_q 之比的对数定义为量化峰值信噪比（Peak Signal to Noise Ratio, PSNR），单位为dB，其表达式为

$$\text{PSNR}_q = 10 \cdot \lg \frac{S_{pp}}{N_q} = 10 \cdot \lg(12 \cdot 2^{2n}) \approx 10.8 + 6n \text{ (dB)} \quad (1.9)$$

从表征线性PCM性能的基本公式（1.9）可以看出，每个采样的编码比特数n直接关系到数字化的图像质量，每增减1比特，就使量化信噪比增减约 6 dB。每个样点的比特数又称样点深度（Depth），对于一般的应用，如电视广播、视频通信等，采用的是8 bit量化，即这些图像的8bit 的样点深度已基本能满足要求。但对某些应用，如高质量的静止图像、遥感图像处理等，需要10bit或更高精度的比特深度。

1.1.3 视频信号的显示

目前，常见的视频信号的显示设备基本上都是液晶显示器（Liquid

Crystal Display,LCD) 或发光二极管显示器 (Light Emitting Diode,LED) ,如电视机的荧光屏、计算机的显示器、大屏幕投影显示器等。曾经普遍使用的CRT显示器由于大体积、高功耗、有辐射、分辨率低等原因，现在已经难觅其踪影了。十多年前和液晶显示器竞争的等离子显示器 (Plasma Display Panel,PDP) 近年来也因为高功耗和低分辨率而渐渐消失。

1. 液晶显示器

液晶显示器 (LCD) 中的液晶，是一种在一定温度范围内呈现既不同于固态、液态，又不同于气态的特殊物质态，它既具有各向异性的晶体所特有的双折射性，又具有液体的流动性。在显示应用领域，液晶由于它的各向异性而具有电光效应，所以能够制成不同类型的显示器件。

例如，最基本的TN型 (Twist Nematic) 液晶显示，将涂有透明导电层的两片玻璃基板间夹上一层正介电异性液晶，液晶分子沿玻璃表面平行排列，排列方向在上下玻璃之间连续扭转90°。然后上下各加一偏光片，底面加上反光片，就构成了TN型液晶显示器的主体。液晶层施加电压，达到一定电压值，对行和列进行选择，出现“显示”现象，所以行列数越多，显示分辨率越高。

在TN基础上改进的实用TFT (Thin Film Transistor) 为薄膜晶体管有源矩阵液晶显示器件，在每个像素点上设计一个场效应开关管，这样就形成可真彩色、高分辨率的液晶显示器件。可实现从VGA (640×480) 到HDTV (1320×1080)，甚至更高分辨率的显示。LCD体积小、重量轻、低电压、功耗小、无软X射线，适合绝大部分应用场合。

2. 发光二极管显示器

发光二极管 (LED) 显示器是一种通过控制半导体发光二极管的显示方式，用来显示文字、图形、图像、视频等多种信息的显示屏幕。最初，LED只是作为微型指示灯，在计算机、音响等设备中使用。随着大规模集成电路和计算机技术的不断进步，平板LED显示器近年来得到迅速地发展和广泛地应用。

严格地说，LCD与LED是两种不同的显示技术，LCD是由液态晶体组

成的显示屏，而LED则是由发光二极管组成的显示屏。LCD的液晶面板本身并不发光，它只是控制透过它的透射光强度，因此，LCD的后面都必须有一块“背板”的发光源，背板的性能直接影响液晶显示的效果。目前绝大部分的电视LED显示屏并不是采用发光二极管来替代液晶，而只是用发光二极管背板来替代LCD中原来的冷阴极荧光灯背光板，做到既可节能又可降低显示器的厚度。这样的LED 显示屏虽然不是真正的由发光二极管独自显示图像，但与上述的 LCD 显示器相比，LED显示屏色彩鲜艳、亮度高、寿命长、体积小，功耗只有LCD的几分之一，刷新速率高，能提供宽达160°的视角，已成为具有绝对优势的新一代显示器，正广泛应用于不同的环境。

近来，出现了更先进的有机LED (Organic LED,OLED) 显示屏，它的单个像元的反应速度是LCD液晶屏的1000倍，在强光下也具有足够的亮度，并且适应-40°C的低温。

1.1.4 数字视频的格式

目前，在模拟视频数字化过程中，一般都是针对视频信号的1个亮度分量Y和2个色度（色差）分量R-Y、B-Y进行的，即分别对这3个分量进行采样、量化和编码，形成数字视频信号的3个相对应的分量。

1.IITU-R的视频采样格式

(1) BT.601建议

为了在世界范围内建立统一的数字视频格式，便于不同电视制式视频之间的互通，国际电联无线电通信部门 (International Telecommunications Union-Radio communications Sector,ITU-R) 制定了演播室彩色电视信号数字编码的建议，即ITU-R BT.601-6标准“Studio encoding parameters of digital television for standard 4:3 and wide-screen 16:9 aspect ratios”。从1982年制定后其间经多次修改，最近的是2007版，其主要内容如表1.1所示。

BT.601建议采用了对亮度信号和两个色差信号分别编码的分量编码方

式，对不同制式的亮度信号Y采用统一的13.5MHz采样频率。由于色差信号的带宽远比亮度信号的带宽窄，因而对色差信号B-Y（或U）和R-Y（或V）的采样频率较Y减半，为6.75MHz。每个数字有效行分别有720个亮度采样点和 360×2 个色差信号采样点。对每个分量的采样点都是均匀量化（8bit或10 bit深度）、PCM编码。这几个参数对525行、60场/秒和625行、50场/秒的制式都是相同的。

表1.1 ITU-R BT.601-6 的主要参数（亮、色采样频率比为4:2:2）

| 参 量 | | NTSC 制：(525 行, 60 场/秒) | | PAL 制：(625 行, 50 场/秒) | |
|------------|-----------|--|------|-----------------------|--|
| 亮度信号 | | $R-Y/B-Y$ | | | |
| 全行采样点数 | Y | 858 | | 864 | |
| | $R-Y/B-Y$ | 429 | | 432 | |
| 采样结构 | | 正交，按行/场则重复，每行中的 $R-Y/B-Y$ 采样与奇数（1,3,5,...）点 Y 采样同位 | | | |
| 采样频率 (MHz) | Y | | 13.5 | | |
| | $R-Y/B-Y$ | | 6.75 | | |
| 编码方式 | | 亮度信号和色差信号均为 PCM 8 bit 或 10 bit | | | |
| 每行有效采样点数 | Y | 720 | | | |
| | $R-Y/B-Y$ | 360 | | | |

(2) 采样分布格式

从表1.1中还可以看到，色度信号的采样率要比亮度信号的采样率低一倍，这样做是考虑到人的眼睛对色度信号的分辨率比亮度信号低。按照这种比例采样的数字视频格式常常又称4:2:2格式，可以简单理解为每一行里的Y、U、V的样点数之比为4:2:2。这些样点位置的几何分布如图1.2 (b) 所示。图中的水平虚线表示视频的扫描线，图1.2 (a)、1.2 (c) 分别给出了4:4:4和4:2:0格式的样点位置示意图。

需要说明的是4:2:0格式虽然不在ITU-R BT.601标准中，但这种格式在实际应用中还是相当广泛的，为了和其他格式作对比，因此也将4:2:0格式放在这里。4:2:0格式中，色度分量比4:2:2格式降低一倍，而且在不同的应用中色度样点的位置还会稍有不同。

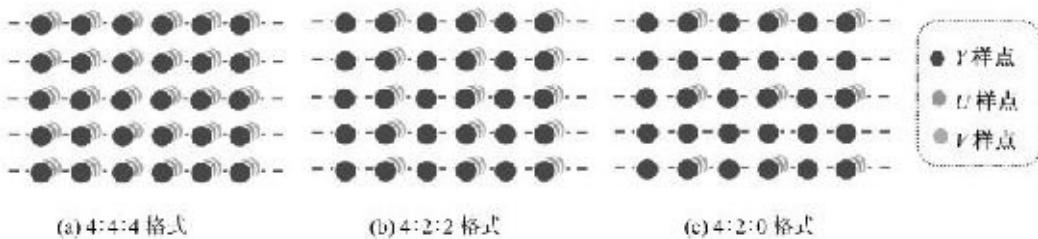


图1.2 不同格式的样点位置示意

2.IITU-T的公共中间格式 (CIF)

在一些互通要求比较高的场合，如实时视频会话，为了既可用625行的电视图像又可用525行的电视图像，ITU-T规定了一种公共中间格式（Common Intermediate Format,CIF）的视频标准，及相应的1/4CIF（QCIF,Quarter-CIF）格式和准QCIF（SQICF,Sub-QCIF）格式等。它们的分辨率分别为 352×288 、 176×144 和 128×96 ，采样格式皆为4:2:0，帧频为30帧/秒，见表1.2。在此基础上还定义了4CIF格式和16CIF格式，分辨率分别为 704×576 和 1408×1152 。

表1.2 常见视频分辨率参数

| 名 称 | 简 称 | 分 辨 率 | 高 宽 比 |
|--|----------------------------|---------------------------------------|-------|
| 4K / UHDTV Ultra High Definition Television | 视频 | 3840×2160 | 16:9 |
| | 电影 | 4096×2160 | 19:10 |
| 2K / HDTV High Definition Television | 视频 | 1280×720p 1920×1080i 1920×1080p | 16:9 |
| | 电影 | 2048×1080 | 19:10 |
| DTV Digital TeleVision | SDTV (DVD) Standard DTV | 720×576 | 4:3 |
| | LDTV (VCD) | 340×255 | |
| CIF Common Intermediate Format | CIF | 352×288 | |
| | Quarter CIF | 176×144 | |
| | Sub-QCIF | 128×96 | |

3.数字视频的文件格式

目前的数字视频都是由音频流加视频流组成，所谓视频格式就是视频流（包括音频流等）数据的封装容器，例如rmvb、avi、MP4等，目前视频文件格式纷繁复杂，下面仅介绍几种常见的格式。

(1) AVI (Audio Video Interleaved) 格式，即音频视频交错格式，1992年由微软公司推出，可封装多种视频流和音频流。所谓“音频视频交错”，就是将视频和音频交织在一起进行同步播放。

(2) MOV (MOViE digital video technology) 格式，是由苹果 (Apple) 公司推出的一种视频文件格式，现在已被Apple Mac OS、微软 Windows 在内的所有主流电脑平台支持。

(3) RM/RMVB (Real Media /RM Variable Bit rate) 格式，是由Real Networks公司开发的一种流式视频文件格式，此格式文件尺寸小，适合网络发布，因此得到迅速推广，网上直播大多采用这种格式。

(4) MPEG格式，是ISO认可的媒体封装形式，普及范围广，其储存方式多样，可以适应不同的应用环境。如mpg、dat文件等。MPEG的控制功能丰富，可以控制有多个视频、音频、字幕。MP4是SONY公司的MPEG-4视频压缩标准的文件格式。

(5) 3GP 格式，是一种3G移动流媒体的视频编码格式，是MP4格式的一种简化版本。3GP具有较低的储存空间和频宽需求，便于手机使用。

(6) DivX格式，是DivX公司在微软MPEG-4(v3)基础上衍生出的一种高效视频编码标准，其图像质量直逼DVD并且容量只有DVD的几分之一。

(7) ASF(Advanced Streaming Format)格式，由微软公司推出。用户可以直接使用Windows自带的MediaPlayer对其进行播放。由于它使用了MPEG-4的压缩算法，所以压缩率和图像的质量都很不错。

(8) WMV(Windows Media Video)格式，是微软公司的一种可以直接在网上实时观看视频节目的文件压缩格式。

1.1.5 高清和超高清视频

1.各类视频的分辨率

目前广泛使用的数字电视按分辨率可分为以下4类，如表1.2所示。

(1) 超高清晰度电视(Ultra High Definition TV,UHDTV)，主要包括两种规格，一种是UHDTV，分辨率为 3840×2160 ，采用16:9的宽高比；另一种是用于工业界的，分辨率为 4096×2160 ，采用19:10的宽高比。两者都俗称4K视频，逐行扫描，50/60fps帧频。

(2) 高清晰度电视(High Definition TV,HDTV)，常见的有3种分辨率：第一种是720p，分辨率为 1280×720 ，逐行(Progressive)扫描；第二种是1080i，分辨率为 1920×1080 ，隔行(Interlaced)扫描；第三种是1080p，分辨率为 1920×1080 ，逐行扫描。全部采用16:9的宽高比，帧频为50/60fps。其中 1920×1080 p为高清通用图像格式(High Definition CIF,HDCIF)，俗称2K视频，目前使用最为广泛。

(3) 数字电视(Digital TV,DTV)，常见的有两种，第一种是低清晰度电视(Low Definition TV,LDTV)，分辨率为 340×255 ，采用4:3的宽高比。例如以前的VCD光盘视频。第二种是标准清晰度电视(Standard Definition TV,SDTV)，分辨率为 720×576 ，采用4:3的宽高比。例如以前的DVD光盘视频，电视台标准数字视频，4CIF格式的视频等。

(4) ITU-T定义的公共中间格式 (CIF) 的视频。

2. 隔行和逐行扫描

隔行扫描 (Interleaved Scan)，是一种较为落后的扫描方式，在早期的显示器中常用。隔行扫描就是将每一帧 (Frame) 图像分割为两场 (Field)，每一场包含了一帧中所有的奇数扫描行 (Scan Line) 或偶数扫描行，通常是先扫描奇数行得到第一场，然后扫描偶数行得到第二场，得到近似两倍于原来帧频的效果，减小了传输带宽。由于视觉暂留效应，人眼将会看到运动比较平滑的场景。当然，由于不是真正的两倍帧频，有时也不可避免地会出现闪烁，使得人眼容易疲劳，但这种扫描方式等好处是可以降低传输带宽和设备成本。

逐行扫描 (Progressive Scan)，是现在最为通行的扫描方式。逐行扫描从图像第一条扫描线一直连续扫描到最后一条，而非先扫奇数条再扫描偶数条。逐行扫描可以提高运动画面的连续感，消除因隔行扫描而产生的抖动、闪烁等现象，其代价就是传输带宽较高。

1.2 视频信号的统计特性

尽管视频图像种类繁多，内容千变万化，数据量大得惊人，然而，大量的统计实验表明，图像数据本身存在一些内在联系和统计规律。例如，图像的同一行相邻像素之间，相邻行像素之间，以及相邻帧的对应像素之间往往存在很强的相关性，即通常人们所说的“冗余”信息。建立在信息论基础上的图像编码方法就是利用图像信号这种固有的统计特性，通过去除这些冗余信息来对图像数据进行压缩处理的。

1.2.1 图像的自相关函数

从信息论的角度出发，通过对图像信息的一阶熵和高阶熵的分析，来确定图像信源的统计特性。但图像熵值的计算十分困难，它要预先知道图像的概率分布等统计参数，因而在实践中用得较多的还是图像的相关函数，因为它可以直接反映任意图像像素之间的关联程度，可以在统计平均的意义上来计算它们之间的相似程度。

将数字图像 $f(i,j)$ 的每个像素值看作一个随机变量，则图像就是这些随机变量的集合，实际上形成一个二维随机场，并假设其为平稳随机场。现在考察图像内任意两点之间的相关性，设 $(i+i_0, j+j_0)$ 和 (i, j) 为 $N \times N$ 数字图像 $f(i,j)$ 中垂直相距为 i_0 、水平相距为 j_0 的任意两点，一般情况下，图像的灰度值为正实数，其归一化自相关函数可由下式表示：

$$R_f(i_0, j_0) = \frac{1}{\sigma_f^2} E\{[f(i+i_0, j+j_0) - m_f][f(i, j) - m_f]\} \quad (1.10)$$

其中， $E[\cdot]$ 表示数学期望， m_f 为图像灰度的平均值， σ_f^2 为图像灰度的方差，可近似为

$$m_f = E[f(i, j)] \approx \frac{1}{N^2} \sum_{i=0}^{N-1} \sum_{j=0}^{N-1} f(i, j) \quad (1.11)$$

$$\sigma_f^2 = E[f(i, j) - m_f]^2 \approx \frac{1}{N^2} \sum_{i=0}^{N-1} \sum_{j=0}^{N-1} [f(i, j) - m_f]^2 \quad (1.12)$$

因此式(1.10)可以用下式近似计算，即

$$R_f(i_0, j_0) \approx \frac{\sum_{i=0}^{N-1} \sum_{j=0}^{N-1} \{[f(i+i_0, j+j_0) - m_f][f(i, j) - m_f]\}}{\sum_{i=0}^{N-1} \sum_{j=0}^{N-1} [f(i, j) - m_f]^2} \quad (1.13)$$

例如，由多幅实际图像统计所得行内自相关函数 $R_f(0, j_0)$ ，在像素水平间隔 j_0 为 1 ~ 20 个像素时，自相关函数基本上呈指数规律衰减，可用式 (1.14) 的指数模型来近似，表示像素之间的相关性随着行内两像素之间的距离增加而迅速减小。

$$R_f(0, j_0) = e^{-\alpha|j_0|} = \rho^{|j_0|} \quad (1.14)$$

式(1.14)中的参数 $\rho=e^{-\alpha}$ 可通过对实际图像的统计获得。对于一般图像， ρ 的值在0.9~0.98，说明图像像素之间存在着很强的相关性。

1.2.2 像素差值的自相关函数

1. 空域像素的差值分布

可以推想，由于一幅图像内相邻像素值之间的相关性很强，相邻像素值之差的统计分布应该有相当一部分集中在零附近。图1.3是对多幅实际图像的水平方向相邻像素差值信号进行统计得到的概率密度分布示意图。大量的图像数据统计表明，对于灰度范围为0~255的常见图像，差值信号绝对值的80%~90%落在0~20的范围内。这一统计得出的结论在预测法图像压缩中是非常重要的依据。

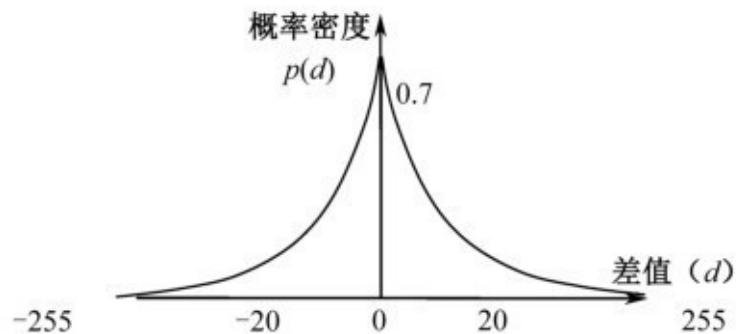


图1.3 图像差值信号的统计分布示意

图像的差值的概率密度 $p(d)$ 常用拉普拉斯 (Laplace) 分布来近似：

$$p(d) = \frac{1}{\sqrt{2}\sigma_d} \text{EXP}\left(-\frac{\sqrt{2}}{\sigma_d}|d|\right) \quad (1.15)$$

其中， d 为相邻像素值之差， σ_d 为其均方差（标准差）值。

2.时域像素的差值分布

上述对一幅（帧）图像内部相邻像素或其差值进行的统计分析，通常称为空域（帧内）统计特性。对视频图像来说，相邻帧对应位置像素之间的时间间隔很小，只有几十毫秒，图像在这段时间内发生变化的可能性不大，变化的程度往往较小。以最简单的帧间差值为例，如图1.4所示，第k帧的帧间差定义为

$$d_k(i,j) = f_k(i,j) - f_{k-1}(i,j) \quad (1.16)$$

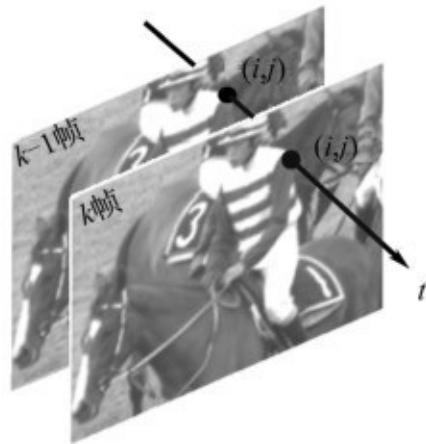


图1.4 帧间差位置示意

其中， $f_k(i,j)$ 表示第 k 帧 (i,j) 处的一个像素， $f_{k-1}(i,j)$ 表示第 $k-1$ 帧和 $f_k(i,j)$ 处于同一位置的像素。一般认为，时域的差值信号的分布特性也和空域差值信号的分布特性类似，也服从拉普拉斯分布，仅具体的均值和方差有所差别。在视频图像中，除非景物有剧烈的活动，或整幅场景更换以外，相邻帧之间往往存在着很强的相关性。

1.3 混合编码

视频编码或视频压缩就是利用图像统计特性，去除视频信号中存在的大量冗余信息的过程。目前这种基于统计的视频压缩常见有两类方法，一类方法是建立在图像的差值信号分布集中基础上便于压缩处理的预测编码，如帧内编码、帧间编码，多视点视频中的视点间预测编码等。另一类方法是建立在正交变换可以将分散分布的图像数据，在变换域集中分布的基础上便于压缩处理的变换编码，如离散余弦变换、离散正弦变换、哈达马变换等。将这两种方法结合起来使用，即形成了混合编码，可消除图像数据中不同类型的冗余信息，取得比单独使用一种方法更高的压缩率。为进一步提高压缩效率，在典型的混合编码系统中，往往还增加了针对视频图像中运动目标的运动估计和运动补偿处理。

1.3.1 预测编码

预测编码（Prediction Coding）利用相邻像素的空间或时间相关性，用已传输的像素对当前正在编码的像素进行预测，然后对预测值与真实值的差——预测误差进行编码和传输。目前用得较多的是线性预测方法，即用已传像素的线性组合对正在编码的像素进行预测。

预测编码是图像压缩技术中研究得最早，且应用最广的一种方法，它的一个重要的特点是性能较好，算法简单，易于硬件实现。图1.5是一个空间预测编码的示意图，其中的处理单元主要包括线性预测器和量化器两部分。在这种编码方案中，编码输出的不是图像像素的样值 $f(m,n)$ ，而是该样值与预测值 $\hat{f}(m,n)$ 之间的差值，即预测误差 $e(m,n)$ 。据图像信号的统计特性的分析，可以得到一组最佳的预测系数，使得预测误差的分布大

部分集中在0附近，经非均匀量化，然后传输量化后的预测误差信号 $e'(m,n)$ 。

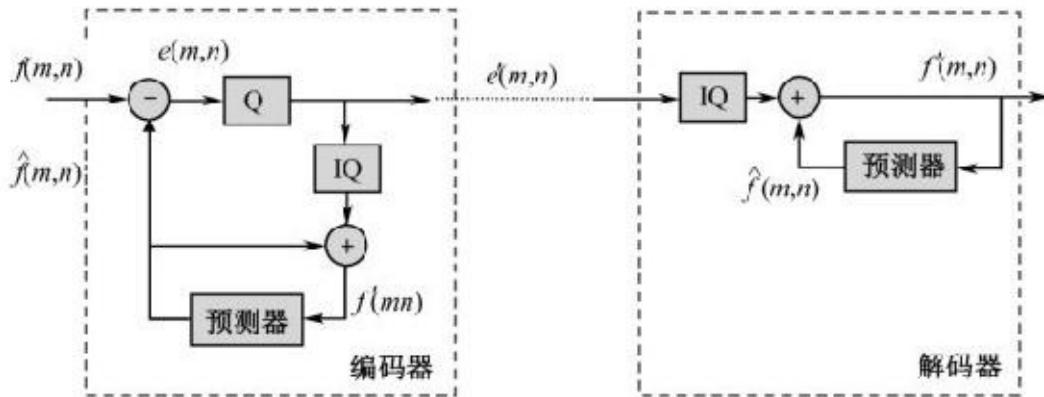


图1.5 预测编码原理框图

解码过程和预测编码相反，由解码器采用和编码器相同方法计算出的预测值 $\hat{f}(m,n)$ 加上由编码器传输来的预测误差信号 $e'(m,n)$ ，即可得到重建的图像 $f'(m,n)$ ，它和输入视频 $f(m,n)$ 的误差是由量化操作造成的。预测编码中的量化器可以根据人眼的视觉特性来设计，采用较少的量化分层，而量化噪声又不被人眼所觉察。这样，图像数据得到了压缩，而图像

的主观质量并没有明显下降。如果在编解码器取消量化和反量化单元，则可形成一个无失真预测系统，解码恢复的图像和原图象完全一致，即有 $f' (m,n) = f (m,n)$ 成立。但这种无失真系统的压缩效率要远低于有量化的有失真预测系统。

1.3.2 变换编码

变换编码（Transform Coding）将空间域描述的图像，经过某种变换（如离散余弦变换、离散正弦变换、哈达玛变换等）形成变换域中的数据（系数），达到改变数据分布，减少有效数据量的目的。

在变换编码中，正交变换是一种最常见的数据处理手段，它把统计上彼此密切相关的像素值矩阵通过线性正交变换，变成统计上彼此较为独立、甚至完全独立的变换系数矩阵。信息论的研究表明，正交变换不改变信源的熵值，变换前后图像的信息量并无损失，完全可以通过反变换得到原来的图像值。但经过正交变换后，数据的分布发生了很大的改变，变换系数在变换域坐标系分布趋于集中，如集中于少数的直流或低频分量的坐标点。数据的集中分布为数据压缩创造了条件。比如，有利于通过量化操作去除大部分零或接近零的系数，保留少量有效系数；有利于对量化后的系数采用更加有效的表示方式，如“之”（Zig-zag）字形扫描、变长编码等，从而获得对图像信息量的有效压缩。

视频编码中最常见的正交变换是离散余弦变换（Discrete Cosine Transform,DCT），以 $N \times N$ 的二维图像 $f (x,y)$ 为例，其二维DCT的正变换和反变换分别定义为：

$$F(u, v) = \frac{2}{N} c(u)c(v) \sum_{x=0}^{N-1} \sum_{y=0}^{N-1} f(x, y) \cos \frac{(2x+1)u\pi}{2N} \cos \frac{(2y+1)v\pi}{2N} \quad (1.17)$$

$$f(x, y) = \frac{2}{N} \sum_{u=0}^{N-1} \sum_{v=0}^{N-1} c(u)c(v) F(u, v) \cos \frac{(2x+1)u\pi}{2N} \cos \frac{(2y+1)v\pi}{2N} \quad (1.18)$$

其中 $c(\cdot) = \begin{cases} 1/\sqrt{2} & u, v = 0 \\ 1 & u, v \neq 0 \end{cases}$

从上面的定义可以看出，DCT本质上和傅里叶变换一样，可以反映信号的频域特性。傅里叶变换是复杂的复数运算，而DCT运算是一种相对简单的实数运算，而且正反变换基函数一样。从DCT的表达式还可以知道，二维DCT是一种可分离的变换，可以分别对两个变量进行一维DCT运算来方便地实现。

在目前常用的正交变换中，DCT 变换性能仅次于理论上最佳的可以完全除去相关的 K-L (Karhunen-Loeve) 变换，所以DCT变换被认为是一种准最佳变换。DCT变换矩阵与图像内容无关，去相关性好，有快速算法 (FDCT)，实现方便；DCT的基函数是偶对称的数据序列，可减轻在图像的分块编码中块边界处的灰度值跳变和不连续现象。DCT这些优点，使得二维DCT变换在图像编码的应用中得到普遍的使用。在历年来颁布的一系列视频压缩编码的国际标准中，都把基于块的DCT作为其中的一个基本处理模块。

采用DCT变换的图像压缩编码基本框图如图1.6所示。编码器根据DCT系数集中在低频区域，越是高频区域系数值越小的特点，利用人眼的视觉特性，通过设置不同的视觉阈值的量化电平，将许多能量较小的高频系数量化为0，可以增加变换系数中0的个数，同时保留能量较大的系数分量。

对量化后的系数还可以再进行变长编码从而获得进一步的压缩。在变换编码的解码端，其解码过程正好和编码过程相反，可以获得失真很小的解码图像。这里失真仍然是由量化器引起，因为正反变换、变长编解码都是无失真处理。

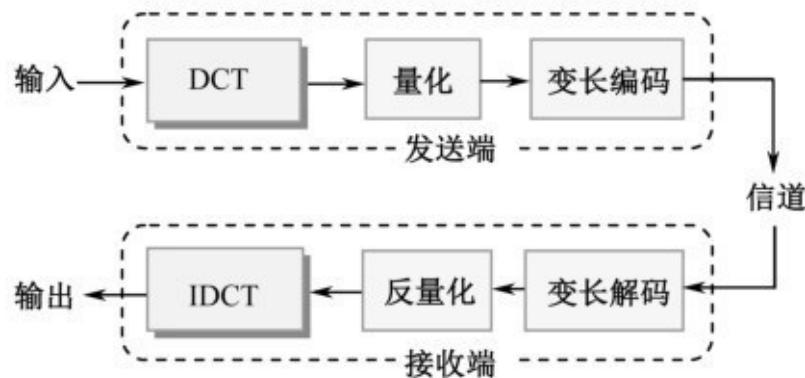


图1.6 变换编码的基本框图

1.3.3 运动估计和运动补偿

1.按运动特性分区

为了说明如何利用视频中的运动特性来进行压缩编码，首先来分析视频中如图1.7所示的一个简单的场景。在一个不十分复杂的背景前，有一个活动量不大的刚性物体，假定在第K帧中的物体相对于第K-1帧有一位移，则可以将整幅画面分为三个各具特点的区域。

(1) 背景区：指固定摄像机所摄取的运动物体后面的背景，一般它是静止的，若外界条件不变，则这两帧背景区的绝大部分数据相同。这意味着两帧背景区之间的帧间相关性极强。

(2) 运动物体区：若将物体运动近似看作简单的刚体平移，则第K帧与第K-1帧的运动物体区的数据也基本相同。假如能采用某种方法对运动物体的位移量进行“运动补偿”，那么两帧的运动物体区之间的相关性也是很强的。

(3) 暴露区：指物体运动后所暴露出的原来曾被遮盖的背景区域。如果有可能事先用存贮器将这些暴露区的数据暂时存贮，则经遮盖后再暴露出来的数据与原先存贮的数据应大致相同。

在理想的情况下，以上三类区域的帧间相关性都可以作为压缩编码的依据。当然，若是整个画面从一个场景切换为另一场景时，就谈不上帧间相关性的利用了。

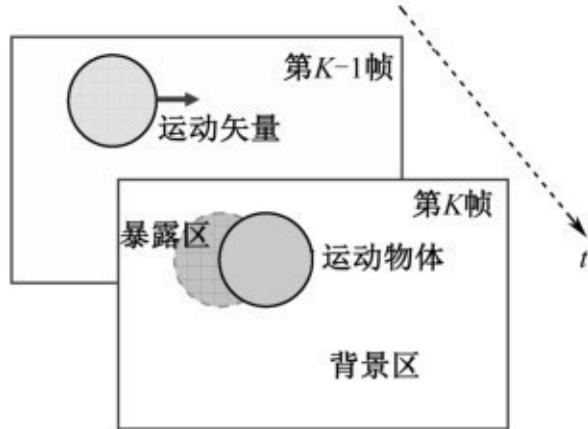


图1.7 图像按运动特性分区

2.减少帧间预测误差

在帧间预测编码中，为了达到较高的压缩比，最重要的就是要尽可能减少帧间预测误差。观察图1.7可以发现，在以上的帧间预测中，实际上仅在背景区进行预测时可以获得较小的帧间差。如果要对运动物体区域进行预测，首先就要估计出运动物体的位移，或运动矢量（Motion Vector,MV），这就是运动估计（Motion Estimation,ME）。然后再根据运动矢量找出物体在前一帧的区域位置，计算前后帧运动物体对应点之间的差值，这样求出的预测误差才比较小，这就是运动补偿（Motion Compensation,MC）。简而言之，通过运动估计和运动补偿，可以减少帧间预测误差，提高帧间预测编码的压缩效率。

从理论上讲，运动估计前必须将图像中运动的物体划分出来。由于实际的视频内容千差万别，把运动物体以整体形式划分出来是极其困难的，因此有必要采用一些简化模型，目前已得到广泛应用的就是基于图像分块

的块匹配方法 (Block Matching Arithmetic,BMA)。

3. 块匹配运动估计原理

块匹配方法将图像分为若干不重叠、紧相邻的 $N \times N$ 像素的块 (Block) 或子块 , 尺寸 N 通常取为 4,8,16,32 或更大 , 以块或子块作为运动估计和补偿的基本单元。块匹配算法就是判断某一子块是运动子块还是静止子块 , 运动子块的运动矢量又为何值。由于这种方法没有以运动物体作为运动估计的基本单位 , 因而不可避免地会产生估计误差 , 尤其是对那些既有运动物体又有静止背景的子块误差将更大 , 这就是块匹配算法的不足之处。但是 , 由于这种方法相对简单 , 因此它已成为现在应用最为广泛的运动估计算法。

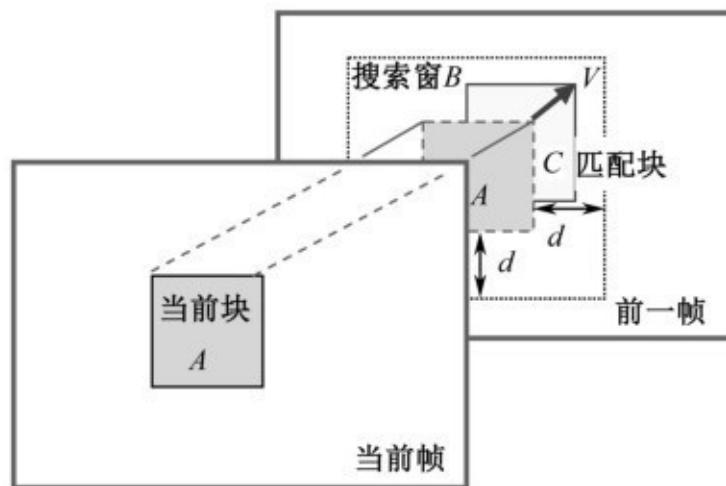


图1.8 块匹配运动估计示意

可以用图1.8来简要说明块匹配运动估计算法的原理。设A为当前帧中的一个待处理的子块，在前一帧中对应的位置为虚线块 A。在前一帧中以A为中心、上下左右各距d个像素的区域B（又称搜索窗）内寻找一个与A最相似的子块C，认为就是前一帧中的C移动到当前帧中的A的位置。那么，C与A的坐标偏移量即为所估计的运动矢量V。

上述过程实质上是一个求子块匹配的过程，通常采用均方误差（Mean Square Errors,MSE）或绝对误差和（Sum of Absolute Differences,SAD）作为匹配准则，匹配的过程就是求这些误差极小值的过程。

采用均方误差最小准则表示的运动矢量v (i,j) 为

$$v(i, j) = \arg \min_{(i, j)} \frac{1}{N^2} \sum_{m, n=0}^{N-1} [f_k(m, n) - f_{k-1}(m + i, n + j)]^2 \quad (1.19)$$

采用绝对误差和最小准则表示的运动矢量v (i,j) 为

$$v(i, j) = \arg \min_{(i, j)} \sum_{m, n=0}^{N-1} |f_k(m, n) - f_{k-1}(m+i, n+j)| \quad (1.20)$$

式中， $f_k(m, n)$ 表示当前的图像块，即 $A, f_{k-1}(m+i, n+j)$ 表示前一帧中B搜索窗内、相对A的位置偏移为 $v(i, j)$ 的一个搜索块， i, j 的变化范围为 $(-d \sim d-1)$ 。对所有的 (i, j) 计算误差匹配函数值，找到最小的误差所对应的 $v(i, j)$ 即为运动矢量，其水平和垂直分量分别为 $v_x = i$ 和 $v_y = j$ 。

采用块匹配技术进行运动估值估计时要求选择合适的方形子块尺寸 N 。子块尺寸偏小时，块内像素运动一致性好，运动估计准确度较高，但用于表示运动矢量的数据过多，会使码率增加，计算量也增大。子块尺寸偏大时，计算量减小，用于表示运动矢量的数据也变少，但含不同类型区域的子块增加，使得运动估计准确度不高，预测误差偏大，同样会使码率增加。在目前的视频编码国际标准中， N 的取值有 32, 16, 8, 4 等，另外，还出现了长方形的子块，如 $4 \times 8, 8 \times 4$ 等。

4. 块匹配运动估计算法

(1) 全搜索算法

按照式 (1.19) 或式 (1.20) 的运动估计算法，如果对搜索区域内的每一个可能的位置上都进行误差计算和比较，找到使 MSE 或 SAD 最小时的点 (i, j) 值，作为所需的运动矢量。这种方法称为全搜索 (Full Search, FS) 算法。其优点是比较准确，能够获得全局最优解，缺点是匹配运算量大，如采用 MAD 准则，需计算 $(2d+1)^2$ 次 SAD 值。

(2) 快速算法

为了克服全搜索法计算量大的缺点，在实际应用中可采用一些简化的块匹配算法，如三步法、五步法等快速算法。这些算法不在搜索区域内的每一个可能的位置上都进行误差计算和比较，只选择若干有代表性的点进行搜索，运算量自然就大为降低，但同时其性能也必然有所下降。下面简要介绍应用广泛的三步搜索（Three Step Search,TSS）算法。

在块匹配运动估计中，块的位移可以理解为中心点的位移。在三步法中，搜索范围为 ± 7 ，即在上一帧以当前子块中心为原点，将当前子块在其上下左右距离为7的范围内按一定规则移动，每移动到一个位置，取出同样大小的子块与当前子块进行匹配计算。如图1.9所示，具体分为以下三步：

① 以当前子块为中心（a），以4为步幅，将a及其周围8个位置为中心的9个子块与当前子块进行匹配，求出最佳匹配的子块中心位置，如b。

② 以第1步中求出的最佳子块为中心（b），以2为步幅，将b周围8个位置为中心的子块与当前子块进行匹配，求出最佳匹配的子块中心位置，如c。

③ 以第2步中求出的最佳子块为中心（c），以1为步幅，将c周围8个位置为中心的子块与当前子块进行匹配，求出最佳匹配的子块中心位置，如d，它与当前子块中心的位置a偏移量即为估计的位移量。

可以看到，在 $d=7$ 时，三步法仅需要 $3 \times 9 - 2 = 25$ 次匹配，它比全搜索算法的匹配次数 $(2 \times 7 + 1)^2 = 225$ 少得多。

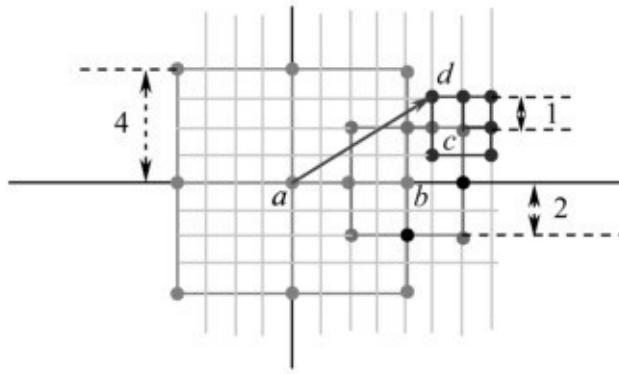


图1.9 三步搜索算法示意

5.块匹配运动补偿

运动估计算法对每个子块估计出对应的运动矢量 $V = (v_x, v_y)$ ，并将它传送到接收端。同时，在编码端利用该运动矢量进行运动补偿（Motion Compensation, MC）的帧间预测，即用前一帧图像在 $(x - v_x, y - v_y)$ 处的亮度值 $f_{k-1}(x - v_x, y - v_y)$ 对当前编码帧中 (x, y) 处的亮度值 $f_k(x, y)$ 进行帧间预测，预测值为 $\hat{f}_k(x, y) = f_{k-1}(x - v_x, y - v_y)$ ，于是，预测误差为：

$$e(x, y) = f_k(x, y) - \hat{f}_k(x, y) = f_k(x, y) - f_{k-1}(x - v_x, y - v_y) \quad (1.21)$$

对比相邻帧对应位置直接相减的方法，这种方法相差一个位移矢量。对预测误差进行变换、量化和熵编码后进行传输。在接收端经过解码后，结合上一帧的恢复图像就可以得出当前帧各子块的解码图像。

需要注意的是，在基于块的运动估计和运动补偿中有一个基本的假设，即块内所有像素具有同样的平移运动，因此子块的运动矢量就是子块内所有像素的运动矢量。对于其他类型的运动，例如缩放、旋转，以及背景区的暴露或遮盖等，这种运动补偿预测就难以适用。

1.3.4 混合编码框架

变换编码和预测编码是两类不同的压缩编码方法，如果将这两种方法组合在一起，就构成“混合编码”（ Hybrid Coding ）。一般可将“混合编码”理解为预测编码和变换编码的混合，使用DCT等变换进行空间冗余度的压缩，使用基于运动补偿的帧间预测进行时间冗余度的压缩，以提高对活动图像的压缩效率。

混合编码的核心结构如图1.10所示，其中DCT、IDCT分别为正反变换，Q、IQ分别为正反量化，属于“变换”部分。输入视频后的减法器，运动估计（ ME ）和运动补偿（ MC ）属于帧间/帧内“预测”部分。这样，“变换”和“预测”两部分组成了混合编码的框架。后面的熵编码（ Entropy Coding ），是一种变长编码（ Variable Length Coding, VLC ），可进一步提高混合编码的压缩效率。

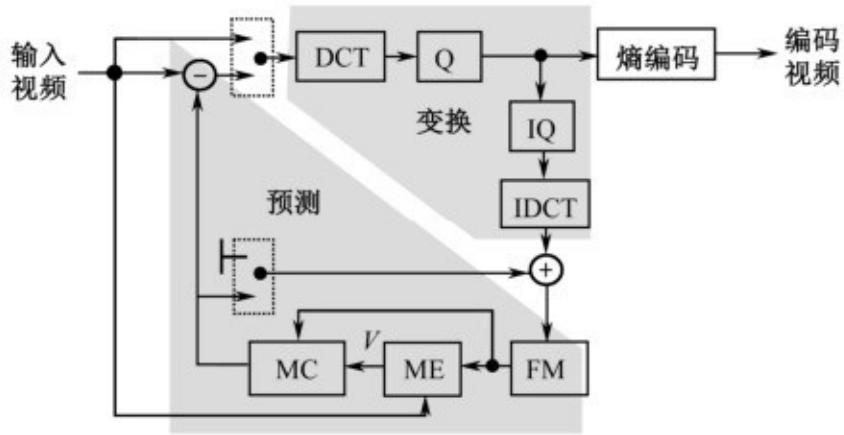


图1.10 混合编码器框架

图1.10中的编码器根据输入视频的内容不同，可工作在不同的预测模式。当两个同步的双向选择开关接到上边时，编码器工作在帧内（Intra）预测模式，输入信号直接进行DCT变换，经过量化处理后再进行熵编码，得到最后的编码输出。当双向开关同时接到下方时，编码器利用存贮在帧存（Frame Memory, FM）中的前一帧图像进行运动估计和运动补偿，提供预测信号，将输入视频信号与预测信号的相减后，对帧间预测误差（残差）进行DCT变换，变换系数经过量化、熵编码后得到最后的编码输出。

H.261建议是最早采用混合编码方法进行视频图像压缩的国际标准，它将混合编码概念具体化。除 H.261外，现有视频压缩编码国际标准，如 H.263、MPEG-1、MPEG-2、MPEG-4,H.264/AVC以及新近的HEVC中都采用了这一混合编码概念。

1.4 量化和熵编码

在混合编码框架中，量化（Quantization）和熵编码是预测和变换后的两个主要处理步骤，可以大幅度提高视频编码的压缩效率。实际上在熵编码前一般还需进行某种方式的扫描，使量化后的数据结构更有利于熵编码。下面分别对这3个步骤进行简单说明。

1.4.1 量化

从前面的变换编码和预测编码的分析可以看出，如果没有量化就不可能得到高效数据压缩。同时，预测和变换本身并未给图像数据带来失真，失真是由量化所造成的。可见量化过程是数据压缩的有效方法之一，也是图像压缩产生失真的根源之一。因此量化器的设计是一个受约束的优化问题，在允许一定失真（或保持一定图像质量）的条件下，获得尽可能高的压缩比。

量化最简单的方法就是均匀（线性）量化，但均匀量化往往效果并不好，因为它没有考虑到量化对象的概率分布。例如对DCT系数这样的数据而言，其分布大部分集中在直流和低频附近，如果采用非均匀量化，对低频区域进行细量化，对高频区域进行粗量化。可以证明，它与均匀量化相比，在相同的量化步长的条件下，其量化误差要小得多。

1.4.2 Zig-zag扫描

为提高编码效率，并不直接对量化后的DCT系数值进行熵编码，而是对重新排序后的系数进行。最常见的重排序方式就是“Z”字形（Zig-zag）

扫描，它将交流（AC）系数的2维数组以1维方式读出，顺序如图1.11所示。在扫描输出的一维数据中，将非零系数前面的“0”的游程长度（个数）与该系数值一起作为一个统计事件，然后将每一事件（“0”游程长度，非零系数值）组成的符号组进行熵编码，如Huffman编码等。

之所以在熵编码之前进行Z字形扫描，主要是因为量化后的DCT系数更加稀疏（有效系数占比小），仅少数AC系数不为零，Z字形扫描可以增加连零的长度，减少统计事件的个数，从而进一步增加对DCT系数熵编码的压缩率。

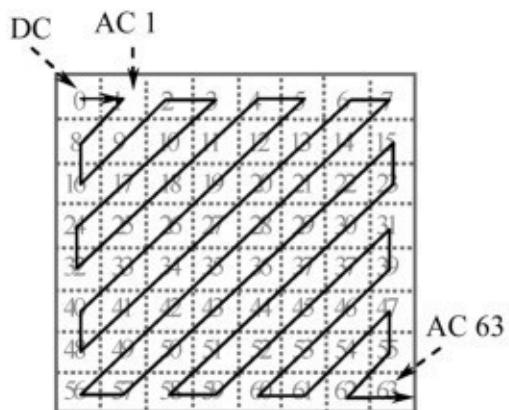


图1.11 DCT系数的Zig-zag扫描

1.4.3 熵编码

经过量化、Zig-zag扫描后的符号组，如果对它们用相同长度的二进制码表示，称为等长编码。采用等长编码的优点是编解码过程简单，但由于这种编码方法没有考虑各个符号出现的概率，实际上就是将他们当作等概率事件来处理的，因而它的编码效率较低。

和等字长编码不同的一种方法是熵编码，或变长编码（VLC）。在这种编码方法中，表示符号（或事件）的码字的长度不是固定不变的，而是随符号出现的概率而变化：给出现概率高的符号分配较短的码字，给出现概率低的码字分配较长的码字。可以证明，在非均匀符号概率分布的情况下，变长编码总的编码效率要高于等字长编码。要注意的是，变长编码是一种信息保持型编码（熵编码），即编解码的过程并不引起信息量（熵）的损失，因为它的符号和码字之间是唯一对应的。

在视频编码中最常用的变长编码方法为哈夫曼编码（Huffman）。设编码信源有K种符号，如K种灰度等级，即信源的符号集合为 $\{a_i | i=1,2,\dots,K\}$ ，与之对应的符号出现的概率集合为 $\{P(a_i) | i=1,2,\dots,K\}$ ，那么，不考虑信源符号间的相关性，对每个符号单独编码时，则平均码长为L比特：

$$L = \sum_{i=1}^K P(a_i)l_i \quad (1.22)$$

式中， l_i 是表示符号 a_i 的码字的长度，单位为bit。可以证明，若编码时对概率大的符号用短码，对概率小的符号用长码，则L会比等长编码时所需的码字短，趋近于信源熵。

除Huffman编码以外，还有多种变长编码方法，如在视频编码中常见的Golomb编码、算术编码等，在后面的章节中将会详细介绍。

1.5 率失真优化

视频编码的性能基本取决于编码码率、压缩失真和计算复杂度这三个因素，如何在这三者之间折中选择、优化设计就是视频编码的根本任务。这里暂不考虑计算复杂度，或认为在计算复杂度相同的条件下，如何决定码率和失真的关系。码率和失真的关系在理论上是由率失真定理奠定的，但是要将此定理应用到实践中，还需要考虑更多的限制条件、一定的简化措施和假设条件。

1.5.1 图像的信源熵

由于图像信息的压缩处理必须在保持信息源的信息量不变，或者损失不大的前提下才有意义，这就必然涉及信息的度量问题。为此可将信息论的有关方法运用到图像信息的度量中去，如计算图像的信源熵。

1.无记忆信源熵

设信源X可发出的消息符号集合为 $\{a_1, a_2, \dots, a_i, \dots, a_m\}$ ，各个符号出现的概率对应为 $\{P(a_1), P(a_2), \dots, P(a_i), \dots, P(a_m)\}$ ，且 $\sum_{i=1}^m P(a_i) = 1$ 。信源X发出某一符号 a_i 的信息量可以用该符号出现的不确定性来定义。不确定性越大，即出现的概率越小，越不能够预测它的出现，它一旦出现，自然带给我们的信息量也越大；不确定性越小，情况则相反。可见符号 a_i 出现的不确定性实际上和该符号出现的概率 $P(a_i)$ 大小相反，在此基础上定义符号 a_i 出现的自信息量为：

$$I(a_i) = -\log_2 P(a_i) \quad \text{bit/符号} \quad (1.23)$$

如果信源 X 各符号 a_i 的出现是相互独立的，称这类信源为独立信源，或无记忆信源（Memoryless Source）。对信息源 X 的各符号自信息量取统计平均，可得信源的平均信息量：

$$H(X) = -\sum_{i=1}^m P(a_i) \log_2 P(a_i) \quad (1.24)$$

称 $H(X)$ 为信息源 X 的熵（Entropy），单位为 bit/符号，通常也称为 X 的 0 阶熵（也有称 1 阶熵），它可以理解为信息源 X 发出任意一个符号的平均信息量。

对于实际用作观察的图像而言，要考虑的不是大量的图像（把某一幅具体图像作为一个“符号”）构成的集合，因为这样的集合其元素量巨大，例如一幅 256×256 的8bit的灰度图像，共有 $(2^8)^{256 \times 256}$ 种可能性。如果仍以图像作为基本符号单位，就难于处理而不再具有实际意义。比较直观、简便的方法是把图像分割为小尺寸图像，甚至将每个像素值作为一个信源符号，这时，公式中的 $P(a_i)$ 为各像素值出现的概率， $H(X)$ 的单位为bit/像素。

2. 有记忆信源熵

无论是经验还是实验测试都表明，具有实际意义的图像，其相邻的像素之间总有一定的联系，或者说，图像信息源是一种“有记忆”信源（Memory Source）。

对于有记忆信源，可以从联合熵的概念出发，考察图像的信源熵。为简单起见，在一个有记忆的信源中仅考虑相继的N个符号之间存在关联的情况，或者说N-1阶Markov过程：某一符号的出现只和它前面N-1个符号有关，而和它更前面第N个、第(N+1)个、……符号无关。把这些有关联的N个符号序列当作一个新符号 $B_i^{(N)}$ 。设一个原符号（如 a_i ）有m个取值（如m=256），则一个新符号有 $m^N=n$ 种不同的取值，新符号集 $\{B_i^{(N)}\}$ 共有n个新符号，即*i*=1,2,...,n。信息源发出新的符号 $B_i^{(N)}$ 的概率用 $P(B_i^{(N)})$ 表示，显然它不是原符号序列中各符号的概率乘积。对于这种信息源，每个新符号的平均信息量为：

$$H(B^{(N)}) = - \sum_{i=1}^n P(B_i^{(N)}) \cdot \log_2 P(B_i^{(N)}) \quad (1.25)$$

其中， n 是新符号的总数。 $H(B^{(N)})$ 的单位为bit/新符号。习惯上用 N 除以上面的熵值，作为每个原符号的平均熵值：

$$H_N(X) = \frac{1}{N} H(B^{(N)}) \quad (1.26)$$

在式(1.26)中，如果考虑以像素为符号，则 H_N 的单位为bit/像素。可以证明，对于同一有记忆信源， $H(X) \geq H_N(X)$ ，说明用联合熵计算，充分利用了符号之间的相关性，得到的信源熵小于将信源符号当作独立符

号时的信源熵。

1.5.2 率失真定理

视频编码的“码率”和“失真”是一对矛盾，相互制约。这两者之间的关系可以从正反两个方面来分析：一方面，在给定失真的条件下，视频能够压缩到什么程度，或码率能够降低到什么程度？另一方面，将此问题反过来，在给定码率的条件下，如何将压缩给视频带来的失真降至最低？下面介绍的率失真（Rate Distortion, RD）定理和失真率定理将给出上述两个问题的简要回答。

1. 编码模型

为了便于理解，可将图像信源的编码和解码过程类比为通信的发送和接收过程，所形成的简化模型如图1.12所示。发送端X为离散独立信源符号集，由 $\{a_i\}$ 表示，接收端Y为输出符号集，由 $\{b_j\}$ 表示。从通信系统的信息传播过程来看，信息的接收者对信源 X 发出的信息内容是从输出集Y了解的。如果是理想信道，没有噪声干扰，发送符号 a_i 和接收符号 b_j 是一一对应的，没有差错；如果是有噪声信道，发送符号 a_i 时，接收符号却不一定 是 b_j ，有可能出现差错。

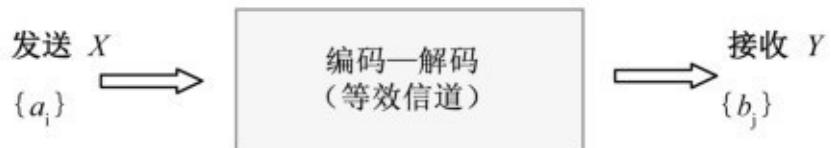


图1.12 信源编解码模型

如果把通信模型套用到编码—解码过程中，信息经编码后发送，经解码后接收，将编解码环节理解为通信信道。从图1.12可以看出，和通信系统类似，接受者收到的不是信源的符号集（码字）X本身，而是经解码输出符号集Y，由它提供了有关X的信息。现在考虑编解码过程：在无外界干扰时，如果编解码过程中采取合并、量化等措施，接收的信息和发送的信息不完全相同，相当于引入一种等效（信道）“噪声”；如果不进行编码或采用信息保持型编码，则编解码过程相当于理想信道，没有“噪声”，也没有信息损失。

如果在信息发、收过程中，没有任何信息丢失，发送集与接收集的符号是一一对应的，这时编码所采用的最佳方法就是所谓的熵编码，码率的下界由（1.24）式的信源0阶熵 $H(X) = -\sum_i P(a_i) \log_2 P(a_i)$ 确定。

在实际应用中发现，尽管信息提供的图像内容很丰富，但对接收者来说不需要或并不能完全感觉到，比如，由于人眼的识别能力、显示装置的分辨能力有限，或者某些其他原因等。在编码时可以采用量化等编码方法去掉或合并一些信息符号。这样做好处是，由于减少了信息符号集的大小，可以节省相应的编码码字。那么，最多能够去掉多少信息符号呢？或者从传输的角度看，对于有一定误差（失真）的编码，最低的码率应该是

多少呢？下面从条件信息量和互信息量的概念出发对这个问题进行简要解答。

2. 互信息量

设信源发出符号 a_i ，出现的先验概率为 $P(a_i)$ ，编码输出为 b_j ，出现概率为 $Q(b_j)$ 。用 $P(a_i, b_j)$ 表示信源发出 a_i ，同时解码输出为 b_j 的联合概率；用 $P(a_i|b_j)$ 表示已知编码输出为 b_j ，估计信源发出 a_i 的条件概率；用 $Q(b_j|a_i)$ 表示信源发出 a_i 而解码输出为 b_j 的转移概率。

由条件概率定义条件信息量为：

$$I(a_i|b_j) = -\log_2 P(a_i|b_j) \quad (1.27)$$

表示收到 b_j 后，信源发送符号为 a_i 的不确定性所形成的信息量。

$$I(b_j | a_i) = -\log_2 Q(b_j | a_i) \quad (1.28)$$

表示信源发送 a_i 后，收到符号为 b_j 的不确定性所形成的信息量。

在图 1.12 的模型中，考察在接收端收到符号 b_j 后，编码系统所传送的关于信源发送符号 a_i 的信息量。在接收端未接收到 b_j 以前，我们判断发送端发送符号 a_i 的概率为 $P(a_i)$ ，所代表的信息量为 $I(a_i)$ ；而收到符号 b_j 后，这时判断发送符号 a_i 的概率为 $P(a_i|b_j)$ ，所代表的信息量为 $I(a_i|b_j)$ 。可见，接收到符号 b_j 后，预计发送符号 a_i 的概率从 $P(a_i)$ 变为 $P(a_i|b_j)$ ，不确定性（信息量）减少，不确定性减少量所引起的信息量为：

$$I(a_i; b_j) = I(a_i) - I(a_i | b_j) = \log_2 \frac{P(a_i)}{P(a_i|b_j)} = \log_2 \frac{Q(b_j | a_i)}{Q(b_j)} \quad (1.29)$$

这种不确定性的减少（概率增大）是由于接收到 b_j 所传递的信息量实现的。 $I(a_i; b_j)$ 是传送的关于 a_i 的信息量，称为传送信息量，也称为互信息量（Mutual Information）。

从上述定义式可以看出， $I(a_i)$ 是 a_i 所含的信息量， $I(a_i|b_j)$ 表示知道 b_j 后， a_i 还保留的信息量，或者说是 b_j 尚未消除的 a_i 的不确定性。则 $I(a_i; b_j)$ 表示编码后 b_j 实际为 a_i 提供的信息量。

对于信息保持型编码，由于编码前的符号 $\{a_i\}$ 与编码后的符号 $\{b_j\}$ 之间存在一一对应的关系，因此， $P(a_i|b_j) = 1, Q(b_j|a_i) = 1$ ，因此 $I(a_i|b_j) = 0, I(b_j|a_i) = 0, I(a_i; b_j) = I(a_i)$ ，它表明 b_j 为接收者提供了与 a_i 相同的信息量。当编码中引入组合或量化后，两个符号集失去了一一对应的关系。这时 $P(a_i|b_j) \neq 1, I(a_i|b_j) \neq 0, I(a_i|b_j) < I(a_i)$ 。因此可以说，互信息量 $I(a_i; b_j)$ 是扣除了信道中量化或组合的等效噪声损失的信息量。

符号集中符号的平均信息量称为熵，从而可定义条件信息量的平均为条件熵（Conditional Entropy）：

$$H(X|Y) = -\sum_{i,j} P(a_i, b_j) \cdot \log P(a_i | b_j) \quad (1.30)$$

上式为 $I(a_i|b_j)$ 的统计平均，表示收到符号集Y的每一个符号后，符号集X还保留的平均信息量，或平均不确定性。类似的还可以定义条件熵 $H(Y|X)$ ：

$$H(Y|X) = - \sum_{i,j} P(a_i, b_j) \cdot \log Q(b_j | a_i) \quad (1.31)$$

在此基础上引入平均互信息量，它定义为：

$$I(X;Y) = \sum_{i,j} P(a_i, b_j) I(a_i, b_j) = \sum_{i,j} P(a_i, b_j) \log \frac{P(a_i | b_j)}{P(a_i)} = H(X) - H(X|Y) \quad (1.32)$$

上式表示平均每个编码符号为信源X提供的信息量，如在通信系统中则表示信道中传输的信息量。式中 $H(X)$ 为信源的0阶熵， $H(X|Y)$ 代表编码引入的对信源的不确定性，它是编码造成的信息丢失。

3. 失真度量

如前所述，在编解码系统中，如果是无失真编码，则信源符号集 $\{a_i\}$ 和输出符号集 $\{b_j\}$ 具有一一对应的关系，编解码结果没有信息损失。如果有失真编码，信源符号集 $\{a_i\}$ 和输出符号集 $\{b_j\}$ 不再有一一对应关系。例如输出符号集的符号个数小于信源符号集的符号个数，这时编解码后信息发生损失，也就是产生了失真。

这里用 $d(a_i, b_j)$ 表示信源发出符号 a_i ，而被编码成 b_j 时引入的失真量。对于数值型的符号，失真度量有多种，常用的为下面两种：

(1) 均方误差：
$$d(a_i, b_j) = (a_i - b_j)^2 \quad (1.33)$$

(2) 绝对误差：
$$d(a_i, b_j) = |a_i - b_j| \quad (1.34)$$

由于编码符号和解码符号都是随机变量，由它们表示的失真 $d(a_i, b_j)$ 也是随机变量，因此需要计算失真的统计平均，即 $d(a_i, b_j)$ 的数学期望 \bar{D}

来作为总体失真的衡量：

$$\bar{D}(Q) = E[d(i, j)] = \sum_i \sum_j P(a_i) Q(b_j | a_i) \cdot d(a_i, b_j) \quad (1.35)$$

上式中的 $\bar{D}(Q)$ 又称平均失真，是表征编解码系统性能好坏的一个重要指标。由于 $P(a_i)$ 已由信源特性所决定，因此 $\bar{D}(Q)$ 只是 Q 的函数，其大小则完全由条件概率 $Q(b_j | a_i)$ 来确定，或者说有失真编码的性能由 $Q(b_j | a_i)$ 决定。而 $Q(b_j | a_i)$ 是由某种编码方法（或编解码符号之间的对应关系）所确定的，有一种编码方法就有一套 $Q(b_j | a_i)$ ：

$$\{ Q(b_j | a_i); \quad i=1, 2, \dots, I, \quad j=1, 2, \dots, J, \quad \sum_j Q(b_j | a_i) = 1 \} \quad (1.36)$$

给定一个允许失真 D ，在平均编码失真 $\bar{D} \leq D$ 的条件下有多种编码方法，对应多套 $\{Q(b_j|a_i)\}$ ，所有满足此条件的 $Q(b_j|a_i)$ 形成一个集合，记作 Q_D ：

$$Q_D = \{ Q(b_j|a_i); \bar{D}(Q) \leq D \} \quad (1.37)$$

在给定 $P(a_i)$ 的情况下， Q_D 中任意一套 $\{Q(b_j|a_i)\}$ 所对应的平均失真 $\bar{D}(Q)$ 皆不会超过 D 。我们就是要寻找在此约束条件下的一套 $\{Q(b_j|a_i)\}$ ，它所形成的平均互信息量最小。

4. 率失真函数

平均互信息量 $I(X; Y)$ 实际上是编解码系统的编码输出的信息量，对于一个好的编码器，自然要求它在满足一定的失真条件下其平均互信息量越小越好。因为编码器的平均互信息量越小，就意味着编出的码字越少。现在编码变成了一个优化问题：在允许失真量的限制下，在 Q_D 集合中，求使平均互信息量最小的一套 $Q(b_j|a_i)$ ，即编码方案。这就是率失真函数（Rate Distortion Function）定理：

$$R(D) = \min_{Q \in Q_D} I(X; Y) = \min_{Q \in Q_D} \sum_{i,j} P(a_i) Q(b_j | a_i) \log \frac{Q(b_j | a_i)}{Q(b_j)} \quad (1.38)$$

定理表达了最小平均互信息 I_{\min} 和允许的平均失真 D 之间的函数关系 $R(D)$ 。式 (1.38) 可见，平均互信息量是由信源符号的概率，编码输出符号的概率，以及已知符号出现的条件概率所确定。在信源一定的情况下， $P(a_i)$ 和 $Q(b_j)$ 是确定的。编码方法的选择实际上是改变条件概率 $Q(b_j | a_i)$ ，它同时也决定了引入平均失真的大小。

式 (1.38) 表示从信源必须送给接收者的最小的平均信息量，接受者才能以小于或等于 D 的失真来恢复原信息。换句话说，率失真函数 $R(D)$ 是在允许失真 D 时，信源编码给出的平均互信息量 R 的下界，也就是在给定失真下信源编码能达到的极限压缩码率。离散独立信源的 R 和 D 之间的关系如图1.13所示。可以看出：平均失真 D 最小为 0，所以当 $D < 0$ 时， $R(D)$ 无定义；当 $D=0$ 时， $R(0)=H(X)$ ； D 的有效范围为 $0 < D < D_{\max}$ ，在此范围内 $R(D)$ 是正的连续下凸函数；当 $D > D_{\max}$ 时， $R(D)=0$ 。

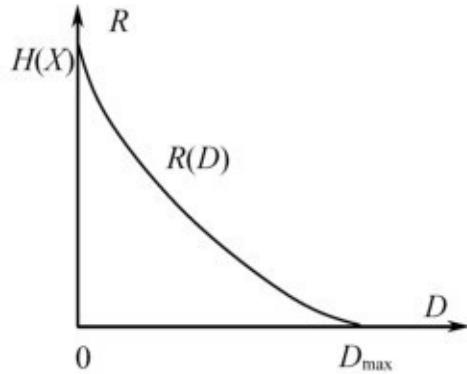


图1.13 离散信源的 $R(D)$ 的曲线

1.5.3 失真率函数

率失真函数对信源编码是具有指导意义的。然而遗憾的是，对实际信源来说，计算其 $R(D)$ 是极其困难的。一方面，信源符号的概率分布很难确知，另一方面，即便知道了概率分布，求解 $R(D)$ 也不容易，它是一个条件极小值的求解问题，其解的一般只能以参数形式给出。

实际中解决以上问题的方法通常是采用相反的思路，即给定信息率 R ，通过改变编码方法或编码参数，寻找尽可能小的平均失真 $D(R)$ ，即失真率函数（Distortion Rate Function）。 $D(R)$ 和 $R(D)$ 是同一个问题的两种不同角度的描述。在一些场合，可用 $D(R)$ 进行编码性能的比较，或者作为编码方法的选择标准。

在失真率函数的指导下，视频编码的问题可归结为：在保证比特率 R 不超过最大比特率 R_{\max} 的条件下，通过选择优化的编码方案和编码参数，使失真 D 达到最小，能够获得“最好”的重建图像质量（例如，PSNR最

高)。如图1.14所示,这一处理过程大体如下:

(1)用一套特定的编码参数(量化步长、编码模式选择等)对视频序列进行编码,检测编码比特率和解码图像质量(或失真)。这样绘出一个速率R和失真D的一个R-D工作点。

(2)用不同的若干套的编码参数重复上述编码过程,获得若干个R-D工作点。

失真率定理告诉我们,对于给定的目标速率 R_{max} ,最小的失真D点必然在 $R=R_{max}$ 这条直线上最接近于R轴的工作点上。率失真优化的目标就是寻找一套编码参数,使它所代表的工作点尽可能地接近理想R-D曲线。

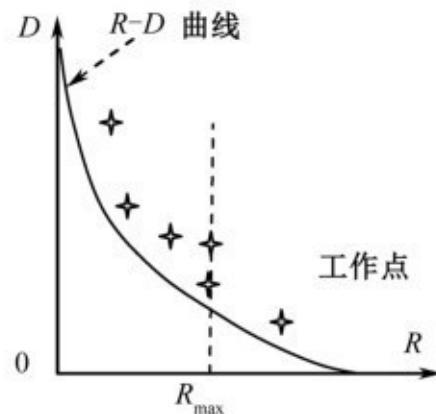


图1.14 R-D曲线和实际工作点

至此可以看出，比较视频编码器性能时常用的 Rate-PSNR 曲线本质上就是 R-D 曲线，图1.15所示的为其一例。这是一幅表示4种不同编码方案的 R-D 仿真实验结果的曲线。它的横坐标表示编码速率，单位为 kbit/s，它的纵坐标是以 PSNR 表示的失真 D，单位是 dB，而且越往上 PSNR 数值越大，失真越小。图中的节点是实际测量的结果，即“工作点”，将各个方案自己的工作点接起来就是各自的 R-D 特性曲线（或 D-R 曲线）。

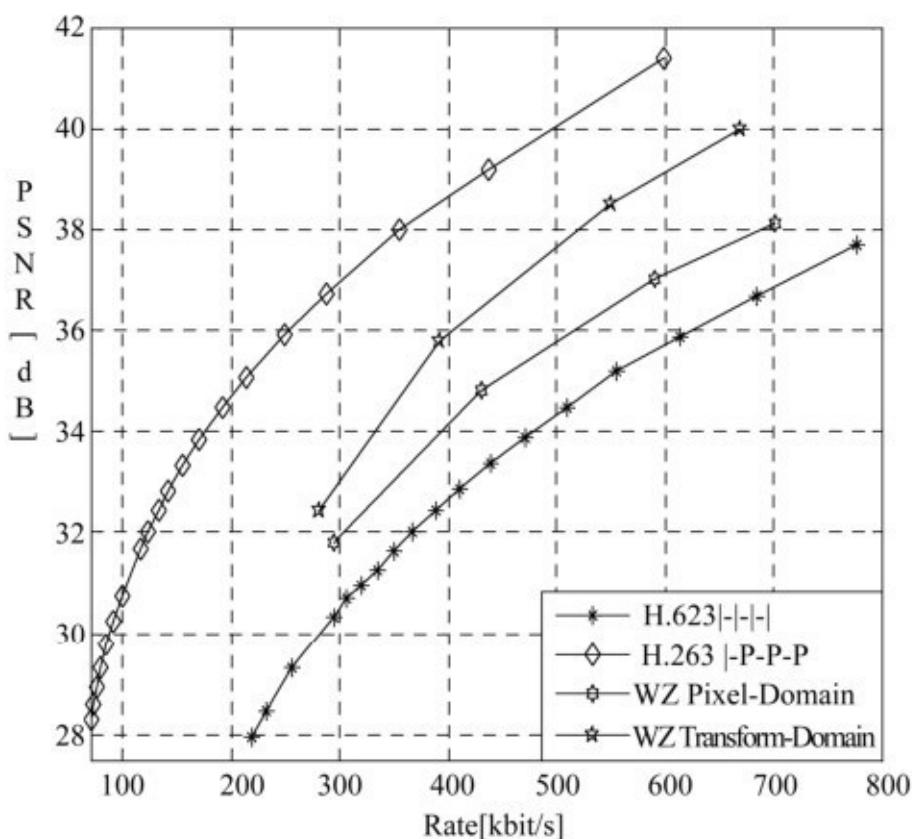


图1.15 实际编码器的R-D曲线

1.5.4 有记忆信源的处理

由于图像信源实际上是有记忆信源，如前所述，对于有记忆信源，把信源发出的N个符号序列成组计算的熵值 $H_N(X)$ 低于把信源作为无记忆时按符号计算的熵值 $H(X)$ 。因此，对有记忆信源按单个符号来编码效率是不高的。为此可以按符号序列成组进行编码，或者进行某种变换，对变换域中形成分布集中、相关性较弱的新符号进行编码，达到逼近信源熵的目标。

在率失真理论中也有类似的关系，有记忆信源的率失真函数低于把信源作为无记忆信源时计算所得的率失真函数。同样，对有记忆信源也可经去相关的处理后，再按独立信源来对待，或者根据相关性先对像素值进行预测，然后再对预测误差编码。这就是图像编码中的两类基本方法——变换编码和预测编码的理论基础。

1.5.5 率失真优化编码

由上述可知，率失真定理就是在给定比特率的情况下寻找编码失真最小的编码器，其方法称为率失真优化（Rate Distortion Optimization, RDO）。因此，将率失真优化的方法应用到视频编码的某些关键部分，提高编码效率，已经取得明显的成效。

对于符合标准的视频编码，率失真优化具有更重要的应用价值。因为在标准的视频编码规范中，除码流格式有严格的规定外，对大部份基本技术也作了种种限制，如DCT变换、帧间预测、运动估计的模式和精度等。但是对如何实现这些功能，如何选择编码参数等，标准中没有规定，可以给实施者留有选择的余地。率失真优化在这些地方已经发挥了重要的作用，如基于率失真优化的图像划分选择、量化参数选择、编码模式选择、运动估计等。

1. 率失真优化方法

率失真优化方法可以定义为如下式所示的优化问题，即在保证编码比特率R不超过最大比特率R_{max}的条件下，使得失真D最小：

$$\min\{D\} \quad s.t. \quad R \leq R_{\max} \quad (1.39)$$

假设视频编码中待优化的编码单元（编码块）之间是相互独立的，即各个编码单元的码率、失真、参数等和其他单元没有关系。第k个单元的编码参数（如量化步长、编码模式、运动矢量等）为B_k，对应的编码比特为R(B_k)、编码失真为D(B_k)，则上述的RDO问题具体为在R_{max}的限制下寻找一组编码参数 \hat{B} ，利用拉格朗日乘子（Lagrange Multiplier）法使得总失真为最小：

$$\hat{B} = \arg \min_{(B_1, B_2, \dots, B_n)} \left(\sum_{k=1}^n D_k(B_k) + \lambda \sum_{k=1}^n R_k(B_k) \right) \quad (1.40)$$

由于假设各编码单元相互独立，因此可以分别对每个编码单元求极小值：

$$\hat{B}_k = \arg \min_{B_k} (D_k(B_k) + \lambda R_k(B_k)) \quad k = 1, 2, \dots, n \quad (1.41)$$

事实上，上式的最优解可看成是代价函数 $J_k(\lambda) = D_k(B_k) + \lambda R_k(B_k)$ 取得极小值时的参数 \hat{B}_k 。代价函数表示了图1.16中R-D平面上的一条斜率为 λ 的直线，在码率限定为 R_{\max} 时，和 R_{\max} 对应的R-D率失真函数

曲线上A点的切线的斜率为 λ 。

需要说明的是，这种将各个编码单元独立处理的方法，实际上是一种局部最优的方法，因为实际视频的各个编码单元之间不可能是相互独立的。

2. 率失真优化的量化

量化操作是视频编码系统中产生失真的主要因素，提高量化器性能对提高编码压缩效率有着重要意义。RDO量化技术就是在给定码率 R_{\max} 约束条件下，寻找能够使失真最小的量化参数设置。

从上面对拉格朗日代价函数的分析可知，要想确定编码块的最佳量化参数，首先必须确定拉格朗日乘子 λ 。大量的实验结果表明， λ 与量化参数QP之间存在着密切的联系。例如，根据R-D关系的分析和大量的实验结果，在H.264视频编码中，如果失真度用SAD计算，用于宏块编码模式选择的拉格朗日乘子 λ_{mode} 可由下式计算：

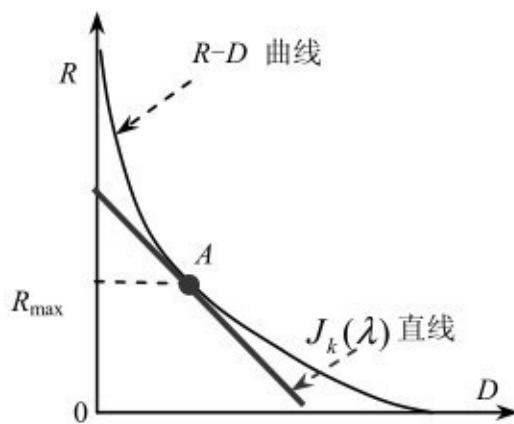


图1.16 拉各朗日最优解示意

$$\lambda_{\text{mode}} = 0.85 \times 2^{(\text{QP}-12)/3} \quad (1.42)$$

3. 率失真优化的模式判决

在视频编码中，对一个图像块可采用多种编码方法，或称多种编码模式。如在HEVC视频编码中，图像块的帧内编码，包含Intra 4×4 ~ Intra 32×32 多种模式；帧间编码，包含 Skip、Inter、Merge的各种尺寸的预测模式……所需选择的模式繁多，同一图像块采用不同的编码模式编码，得到的码率和失真各不相同，我们自然要选择一种码率最低、失真最小的模式来对编码块进行编码。这一任务可由前述的率失真优化方法来完成。

在拉格朗日乘子 λ_m 和量化参数QP选定之后，视频编码器通过最小化拉格朗日代价函数实现对每一个块的编码模式的选定。编码块 B_k 的拉格朗日代价函数如下式所示：

$$J_m(B_k, M_k | \text{QP}, \lambda_m) = D(S_k, I_k | \text{QP}) + \lambda_m \cdot R(B_k, M_k | \text{QP}) \quad (1.43)$$

其中， M_k 为相应块的编码模式。

在不同的编码模式下，编码后输出比特流的比特率R与失真度D的计算方法也不完全相同。在帧内模式下， $R(B_{k,intra}|QP)$ 为熵编码后的输出比特率，失真度D($B_{k,intra}|QP$)则由宏块的原始像素和重建像素决定，常用SAD方式计算，因为它不需要做平方运算，比较简单。

对于Skip模式，由于无须残差信号，因此编码后输出比特率R近似为1bit/block，和失真度D、量化参数QP无关。失真度D则由宏块的原始像素值和预测值决定。

4. 率失真优化的运动估计

在块运动估计方法中，从理论上说，最优运动矢量的选择不一定是SAD值最小所对应的那个运动矢量，而是应该看它对所形成的编码图像的最终比特数的影响来决定。例如，当估计一个编码块的某个可能的运动矢量(MV)时，编码器必须为它进行残差变换、量化和熵编码，由此得到相应的比特数；然后再计算这种情况下图像的失真是多少；在对所有可能的MV都进行如上的计算后，选出一个失真最小、产生的比特数最少的MV，即为最优的MV。只有这样做，才能真正选出最佳的MV值。然而，一个编码块的运动矢量有数百上千种可能，对每个可能都要计算它对图像最终产生的影响，则计算量太大，在实际的视频编码中难以实现。这时，往往借助于前面提到拉格朗日目标函数来计算。实验结果表明，用于实现块运动矢量估计的拉格朗日乘子 λ_{motion} 可由下式计算：

$$\lambda_{\text{motion}} = \sqrt{\lambda_{\text{mode}}} \quad (1.44)$$

例如，对于采用帧间编码模式的一个宏块 B_k ，在给定 λ_{motion} 和量化参数QP的情况下，可通过最小化拉格朗日目标函数来实现 B_k 的运动估计的优化，实现公式如下：

$$J_{\text{motion}}(B_k, v | \lambda_{\text{motion}}) = D_{DFD}(B_k, v) + \lambda_{\text{motion}} \cdot R_{\text{motion}}(B_k, v) \quad (1.45)$$

其中， $v = (v_x, v_y)$ 表示运动估计得到的运动矢量， $R_{\text{motion}}(B_k, v)$ 为传输运动矢量 (v_x, v_y) 所需的比特数，而 D_{DFD} 表示图像块的预测误差，常用SAD定义来计算。

1.6 图像质量的评价

在图像编码中，要求在保持解码图像一定质量的前提下，用尽量少的码字来表示，以便节省信道带宽或存储容量，这里首先涉及的就是对图像质量的合理评价问题。

图像质量的含义包括两方面：一个是图像的逼真度（Fidelity），即被评价图像与原标准图像的偏离程度；另一个是图像的可懂度（Intelligibility），是指图像能向人或机器提供信息的能力。相比较而言，图像的可懂度属于更高一层次的问题，涉及更多人的感知判断，难以统一评价，所以当前的图像质量评价的重点主要在于图像的逼真度，考察处理后图像和原图像的忠实程度。

目前对图像逼真度的评价大体上有两类方法，一类是主观评价方法，另一类是客观评价方法。客观评价方法比较简单，容易实施，但评价结果有时和主观评价不尽一致；主观评价方法比较复杂，实施麻烦，但由于显示的图像最终是供人观看的，所以最具权威性的还是主观评价方法。此外，近年来研究和使用较多的一类充分考虑了人眼视觉特性的客观图像质量评价方法，已经取得了较好的结果，如基于结构相似度图像质量评价（Structure Similarity Image Measurement, SSIM）方法，试图将主观评价和客观评价的长处结合起来，形成一种方便、权威的方法。

1.6.1 主观质量评价方法

图像的主观评价就是通过人来观察图像，对图像的优劣进行主观评分，然后对评分进行统计平均，就得出评价的结果。主观评价大体分为两类：绝对评价和相对评价。绝对评价方法是在无标准的参考图像情况下，

由观察者根据预先规定的评价尺度及个人的经验，对待测图像给出质量判断。相对评价则是在有标准参考图像的情况下，给观察者一组图像，对它们进行相互之间的比较，按照判断对所评图像进行排序。

主观评价准确与否与观察者的特性及观察条件等因素有关。为保证主观评价在统计上有意义，选择观察者时既要考虑有未受过训练的“外行”观察者，又要考虑有对图像技术有一定经验的“内行”观察者。另外，参加评分的观察者至少要有20名以上，测试条件应尽可能与使用条件相匹配。

在图像质量的主观评价方法中，ITU-R的BT.500-1建议给出两种评价角度，就是国际上通行的5级评分的质量尺度和妨碍尺度，如表1.3所示。其中，质量尺度又常称之为“MOS分”，即平均评价分（Mean Opinion Score,MOS）一般来说，非专业人员多采用质量尺度，专业人员常使用妨碍尺度。

视频图像质量主观评价试验可依据国际标准ITU-R的BT.500-1“电视图像质量主观评价方法”和ITU-R的BT.710-2“高清晰度电视图像质量的主观评价方法”进行。其中常用的方法主要有单刺激连续质量评分法（Single Stimulus Continuous Quality Evaluation,SSCQE）双刺激连续质量评分法（Double Stimulus Continuous Quality Scale,DSCQS）。在SSCQE中，只把被评价的图像序列播放给评估者看，相当于绝对评价。而在DSCQS中，则向评估者交替展示两个视频序列，一个是未受损的“原始”序列，另一个是受损的测试序列，然后要求观察者分别给出质量评分，相当于相对评价。

表1.3 BT500-1建议的两种尺度的5级评分

| 妨碍尺度 | 得分(分) | 质量尺度 |
|--------|-------|------|
| 无觉察 | 5 | 非常好 |
| 刚觉察 | 4 | 好 |
| 觉察但不讨厌 | 3 | 般 |
| 讨厌 | 2 | 差 |
| 难以观看 | 1 | 非常差 |

1.6.2 客观质量评价方法

尽管主观质量的评价是最权威的方式，但在许多场合，由于测试条件的限制，希望对图像质量有一个方便、定量的客观评价方式。图像质量的客观评价由于着眼点不同而有多种方法，这里介绍的是几种经常使用的针对灰度图像的逼真度测量方法。对于彩色图像逼真度的定量评价是一个更加复杂的问题，实用中往往是将彩色图像的各个彩色分量作为灰度图像分别进行评价。

1. 均方误差

对于数字图像，设 $f(m,n)$ 为原参考图像， $\hat{f}(m,n)$ 为其降质图像，尺寸都是 $M \times N$ ，即 M 行、 N 列，计算对应像素差值平方的均值，即它们之间的均方误差值（Mean Square Error, MSE）定义如下：

$$MSE = \frac{1}{M \cdot N} \sum_{n=0}^{N-1} \sum_{m=0}^{M-1} [f(m,n) - \hat{f}(m,n)]^2 \quad (1.46)$$

2. 峰值信噪比

设A为图像 $f(m,n)$ 的最大灰度值，如对8比特精度的图像， $A=2^8-1=255$ ，则 $M \times N \times A^2$ 可看成图像信号的峰值功率，将 $\sum_{n=0}^{N-1} \sum_{m=0}^{M-1} [f(m,n) - \hat{f}(m,n)]^2$ 看成因图像降质而引起的噪声功率，则可以用峰值信噪比（Peak Signal Noise Ratio,PSNR）来表示图像的逼真度：

$$\text{PSNR} = 10 \cdot \lg \frac{M \times N \times A^2}{\sum_{n=0}^{N-1} \sum_{m=0}^{M-1} [f(m,n) - \hat{f}(m,n)]^2} = 10 \cdot \lg \frac{A^2}{\text{MSE}} \quad (\text{dB}) \quad (1.47)$$

上述均方误差（MSE）和峰值信噪比（PSNR）是ITU-R视频质量专家组（Video Quality Experts Group,VQEG）规定的两个简单的图像质量客观评价的技术参数，因而也是两个最为常用的参数。

虽然PSNR（也包括MSE）在研究和测试中经常被采用，但它还存在一定的局限性：一是为了获得PSNR数据，需要用原始的图像作为对比，这在不少情况下是难以实现的。二是PSNR往往不一定准确地反映主观的图像质量值，相同的PSNR值并不一定表示其主观的质量一样，主观上感觉好的图像不一定PSNR值高。为了克服PSNR方法的局限性，包括VQEG在内的

很多研究人员致力于开发更加合理的客观的测试过程，提出了多种客观测试方法，下面介绍的基于结构相似性图像质量评价（SSIM）方法就是一种和主观评价比较接近的尝试。但是，目前还没有一种客观评价方法可以完全代替主观测试的方法。

1.6.3 SSIM质量评价方法

基于结构相似度图像质量评价（SSIM）方法是一种充分考虑了人眼视觉特性的客观图像质量评价方法。SSIM方法认为，相对于亮度和对比度信息，人类视觉系统对图像中的结构信息高度敏感，具有自动从视觉感知中提取结构信息的能力，因此结构信息量的变化能够反映一个图像视觉感知失真的程度。而且，从图像形成的角度上看，结构信息反映了场景中物体的结构，它基本独立于图像的亮度和对比度，亮度或对比度的改变对图像结构信息影响不大。

SSIM方法对原始图像与失真图像分别从亮度、对比度和结构3个方面的相似性进行比较，最后综合计算出的SSIM值。实验证明，SSIM评价与主观质量评价比较一致，而且由于计算量较小，已被引入到多项视频编码标准的参考软件中，以评价编码视频的质量。

SSIM的比较对象为两幅灰度图像 x 和 y ，设其中一幅为参考图像，另一幅为与其对应的失真图像。它们的亮度、对比度和结构度的相似度分别定义如下：

亮度的相似度为：

$$l(x, y) = \frac{2\mu_x\mu_y + C_1}{\mu_x^2 + \mu_y^2 + C_1} \quad (1.48)$$

对比度的相似度为：

$$c(x, y) = \frac{2\sigma_x\sigma_y + C_2}{\sigma_x^2 + \sigma_y^2 + C_2} \quad (1.49)$$

结构的相似度为：

$$s(x, y) = \frac{2\sigma_{xy} + C_3}{\sigma_x\sigma_y + C_3} \quad (1.50)$$

上述3式中， x 的平均强度为 $\mu_x = \frac{1}{N} \sum_{i=1}^N x_i$ ，标准差为

$$\sigma_x = \sqrt{\frac{1}{N-1} \sum_{i=1}^N (x_i - \mu_x)^2}$$

y 的平均强度、标准差和 x 类似， x 、 y 之间的协方差为
 $\sigma_{xy} = \frac{1}{N-1} \sum_{i=1}^N [(x_i - \mu_x)(y_i - \mu_y)]$ ， C_1, C_2, C_3 是为了避免分母接近零时测量值不稳定而定义的小常数。公式中所有的变量都需作归一化处理。最后，将这个3个相似度组合起来成为图像 x 和 y 的SSIM指数。为了简化表达，定义SSIM指数为三个相似度的乘积，且 $C_3 = C_2/2$ ，则失真图像和原图像之间的结构相似度定义为

$$\text{SSIM}(x, y) = \frac{(2\mu_x\mu_y + C_1)(2\sigma_{xy} + C_2)}{(\mu_x^2 + \mu_y^2 + C_1)(\sigma_x^2 + \sigma_y^2 + C_2)} \quad (1.51)$$

这样得到的SSIM指数是归一化的， $0 \leq \text{SSIM}(x, y) \leq 1$ ，且
 $\text{SSIM}(x, y)$ 值越接近1，说明失真图像的主观质量越好。

本章参考文献

[1]S.Channappayya,A.Bovik,C.Caramanis,et al.SSIM-optimal linear image restoration[C].IEEE International Conference on Acoustics,Speech, and Signal Processing (ICASSP 2008) ,Las Vegas,30 March-4 April,2008: 765-768.

[2]High Efficiency Video Coding[S].ITU-T Rec.H.265 and ISO/IEC 23008-2,ITU-T and ISO/IEC JCT-VC,Mach 2013 (v.1) ,Oct.2014 (v.2) ,Apr.2015 (v.3) .

[3]Gary J.Sullivan,Jens-Rainer Ohm,Woo-Jin Han,et al.Overview of the High Efficiency Video Coding (HEVC) Standard[J].IEEE Transaction on Circuits and Systems for Video Technology,Dec.2012,22 (12) : 1649-1668.

[4]Methodology for the subjective assessment of the quality of the television pictures[S].ITU-R Rec.BT.500,2000.

[5]Parameter values for ultra-high definition television systems for production and international programme exchange[S].ITU-R Rec.BT.2020-0,2012.

[6]Studio encoding parameters of digital television for standard 4:3 and wide screen 16:9 aspect ratios[S].ITU-R Rec.BT.601-7,2011.

[7]Parameter values for the HDTV standards for production and international programme exchange[S].ITU-R Rec.BT.709-5,2002.

[8]Britanak,V.,K.R.Rao.Discrete cosine and sine transforms: general properties,fast algorithms and integer approximations[M].Academic Press,New York (2006) .

[9]Sullivan,G.J.,Wiegand,T.Rate-distortion optimization for video compression[J].IEEE Signal Process.Mag.1998,15 (6) : 74-90.

[10]Orchard,M.T.,Sullivan,G.J.Overlapped block motion compensation: an estimation theoretic approach[J].IEEE Trans.Image Process.1994,3 (5) : 693-699.

[11]Abdul H.Sadka.Compressed video communications[M].Hohn Wiley & Sons,Ltd,England,2002.

[12]Sachin Dhawan.A review of image compression and comparison of its algorithms[J].International Journal of Electronics & Communication Technology (IJECT) ,March 2011,2 (1) :22-26.

[13]Chen Xilin,Zhu Xiuchang,Wang Qiao,et al.Feature topic: future video technology[J].China Communications,2013,10 (5) :8-10.

[14]Mathias Wien.High Efficiency Video Coding: Coding Tools and Specification[M].Springer-Verlag Berlin Heidelberg,2015.

[15]Vivienne Sze, Madhukar Budagavi, Gary J.Sullivan.High Efficiency Video Coding (HEVC) :Algorithms and Architectures[M].Springer International Publishing Switzerland,2014.

[16]朱秀昌，刘峰，胡栋.视频编码与传输新技术[M].北京：电子工业出版社，2014.

[17]姚庆栋，毕厚杰，王兆华，徐孟侠.图像编码基础[M].北京：清华大学出版社，2006.[18]Zhou Wang,Bovik A.C.,Sheikh H.R.,et al,Image quality assessment: From error visibility to structural similarity[J].IEEE Transactions on Image Processing,Apr.2004,13 (4) : 600-612.

第2章 视频编码的国际标准

20世纪80年代末以来，国际电信联盟远程通信标准化组织

(International Tele-communications Union for Telecommunication Standardization Sector,ITU-T) 和国际标准化组织/国际电工委员会 (International Standardization Organization/International Electrotechnical Commission,ISO/IEC) 先后颁发了一系列有关视频编码的国际标准 (建议) ，对图像与多媒体通信的研究、应用和产业化起到了巨大的推动力。两大国际组织的视频编码标准化进程可概括为如图2.1所示的时间标注。

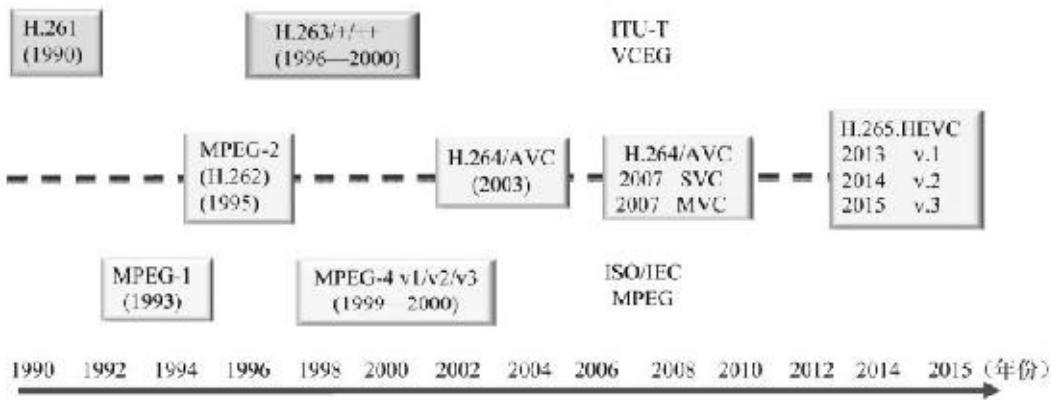


图2.1 视频编码国际标准化进程

在诸多视频编码的国际标准中，1990年ITU-T最先颁布的H.261建议是第一个实用化视频国际标准，具有里程碑意义。首先，它采用的DCT加运动补偿帧间预测的混合编码模式，被各视频编码国际标准普遍使用；其次，它的分块图像格式、编码器模块结构、输出码流的层次结构、开放的编码控制等技术和策略，对后来制定的视频编码标准产生了深远的影响；最后，H.261建议的颁布为不同生产厂商的设备互通奠定了基础，促进了视频通信的产业化发展。随后ITU-T在1995年和1996年先后发布了H.262和H.263建议，其编码框架和H.261相同，但在H.261基础上做了多项改进，编码效率有了较大的提高。

ISO/IEC的联合技术委员会自20世纪90年代以来也先后颁布了一系列视频编码的国际标准。其中，1993年颁布的MPEG-1标准用于数字存储回放系统的音视频编码，例如VCD（Video Compact Disc）光盘播放机。1995年颁布的MPEG-2标准用于通用的音视频编码，例如标清电视、高清晰电视、DVD（Digital Video Disc）光盘播放等。1999年颁布的MPEG-4标准，除对普通的音视频进行高效的编码外，还引入了音、视频对象的概念，可以处理各种不同性质的音视频对象，包括自然的、综合的、静止的、活动的，以及二维、三维等各种情况，适合各种多媒体应用。

为适应新的网络视频应用的需要，以及获得更高压缩效率，ITU和ISO的视频编码专家组成了联合视频工作组（Joint Video Team,JVT），共同制定并于2003年正式颁布了H.264/AVC，即先进的视频编码（Advanced Video Coding,AVC）。该标准仍然采用混合编码方案，对编码的多个环节进行了改进和优化，压缩率比H.263提高了一倍，编码复杂度也大为增加。

最新的视频编码国际标准——高效视频编码（High Efficiency Video Coding,HEVC）已于2013年4月由ITU-T和ISO/IEC的视频编码联合协作组（Joint Collaborative Team on Video Coding,JCT-VC）正式颁布，它主要针对高清视频的应用，其压缩性能又在H.264/AVC的基础上提高了一倍。

由于HEVC是在前面一系列视频标准基础上继承和发展而来的，因此，这里简单地回顾一下上述已经颁布的一些主要视频编码国际标准，包

括它们的主要技术参数、系统组成、技术特点等，对加深理解HEVC是很有益的。

2.1 H.26x标准

ITU-T关于视频编码的标准H.261/2/3/4，包括新的HEVC（H.265），可统称为H.26x系列标准，H.264/5在后面另作介绍，H.262标准等同于MPEG-2的视频部分，因此这里只介绍H.261和H.263。

2.1.1 H.261标准

ITU-T于1990年通过H.261建议—— $p \times 64\text{ kbps}$ 视听业务的视频编解码器，其中p代表数字话路数，范围是1~30路，64 kbps是1个话路的速率。该标准的应用目标是视频会议和可视电话。H.261标准采用的是混合编码方法，具体的视频编解码器框图如图2.2所示。

1. 图像格式和分块

H.261对非隔行视频进行编码，帧频为30000/1001（约29.97），每幅图像包括亮度分量 Y 和两个色差分量 C_b、C_r，彩色取样为4:2:0格式。支持两种图像尺寸，一种格式为CIF，亮度信号每帧288行，每行352像素，两个色差信号每帧144行，每行176像素；另一种格式为QCIF，其亮度和色差信号的水平和垂直方向的像素均为CIF尺寸的一半。

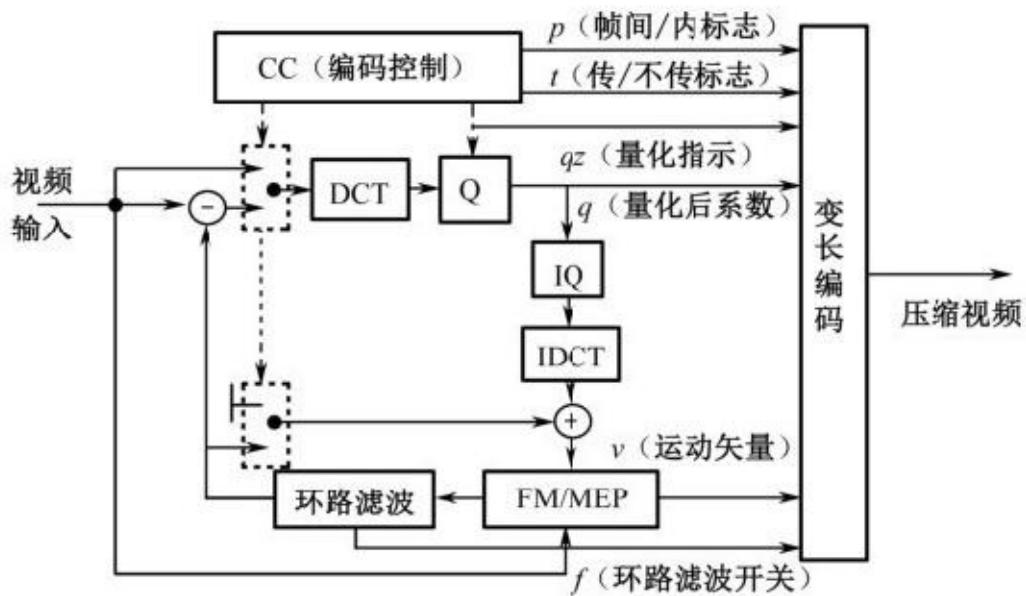


图2.2 H.261混合编码器框图

输入图像被划分为 8×8 像素的子块 (B, Block), 4个亮度子块和2个空间上对应的色差子块组成一个宏块 (Macro Block, MB), 编码以MB为单位进行。对于编码模式的选择、宏块的传与不传以及编码控制策略, H.261建议不作规定。

2. 帧内、帧间预测和DCT

H.261编码器可根据实际需要工作在不同的模式。图2.2中, 由编码控制器 (CC) 控制的两个双向模式选择开关同时接到上方时, 编码器工作在帧内 (Intra) 编码模式, 将输入视频信号直接进行DCT 变换。当双向开关同时接到下方时, 编码器工作在帧间 (Inter) 编码模式, 利用存储在帧存 (FM) 中的上一帧重建图像进行帧间预测, 将输入信号与预测信号的相减

后，对预测误差（残差）进行DCT变换。

根据应用的需要，帧间编码可以是简单的帧间预测，也可以是使用运动估计（ME）和运动补偿（MC）帧间预测，以增加预测精度。运动补偿对于编码器是可选项。运动矢量在水平和垂直方向的值均为不超过 ± 15 的像素，用于宏块中所有4个Y子块，将其除2、截尾取整后再用于相应的两个色差子块。

3.量化、Zig-zag扫描和变长编码

对于变换后的DCT系数，H.261采用的是均匀量化方式，量化步长范围是2~62，用于所有非帧内块直流分量的量化。除帧内块DCT的直流系数外，一个宏块内的所有系数使用同一个量化器，量化器的选择由编码控制部分决定。对于帧内编码子块的变换系数的DC分量使用一个单独的均匀量化器，量化步长为8，而且无中央“死区”，如图2.3（a）所示。其余所有的AC分量使用同一个均匀量化器，但中央有“死区”，即在量化器输入/输出关系曲线上，在0输入左右的 2Δ 区域输出都为0， Δ 为量化步长，如图2.3（b）所示。对于帧间编码模式的子块，则所有系数使用同一个有“死区”的量化器。

每个子块DCT系数量化后，按“Z”字形（Zig-zag）扫描，将2维数组以1维方式顺序读出，然后将（零游程长度，非零系数）组成的符号组进行变字长编码（VLC）。

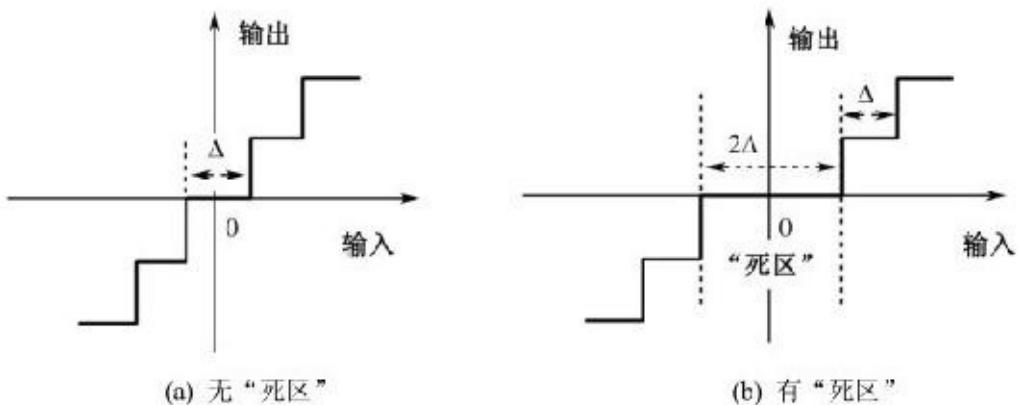


图2.3 两类均匀量化器

4. 图像结构

H.261支持 352×288 像素的CIF和 176×144 的 $1/4$ QCIF (Quarter CIF,QCIF) 格式图像。图像层次的几何结构如图2.4所示。以CIF格式图像为例：最高层为 352×288 的图像 (P)；第二层为宏块组 (Group Of Block,GOB)，每帧图像包括12个GOP；第三层为 16×16 的宏块 (MB) 层，每个GOB包括33个MB；第四层为块 (B) 层，每个MB包括4个 8×8 的亮度子块 (B) 和2个 8×8 的色度子块 (B)。

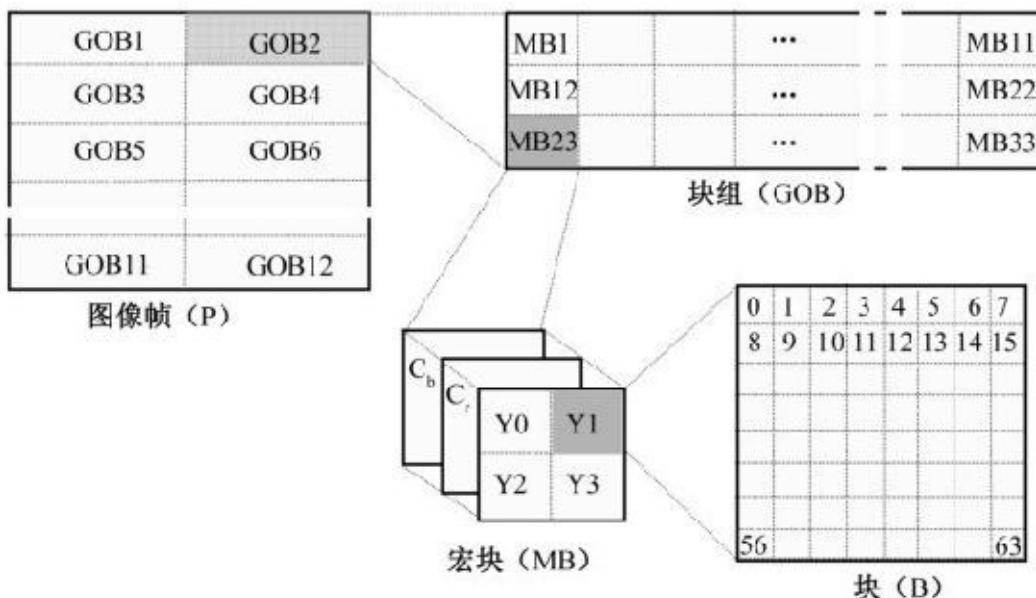


图2.4 H.261图像划分的层次结构

5.信道编码

H.261规定传输比特流中包括一个BCH (511,493) 前向纠错码，其中信息比特为493位，校验比特为18位，接收端使用它进行纠错。

6.编码控制和环路滤波

H.261编码器的工作状态由编码控制器决定，即图2.2中的CC，这部分内容对开发者是开放的，可以视具体情况决定。编码控制方法影响了编码器性能的优劣。控制的内容包括编码器工作模式的选择和各种编码参数的选择，例如：帧内/帧间编码，量化步长等。此外，在网络拥塞时还可能需要进行跳帧等控制。

H.261的环路滤波（Loop Filtering）位于编码器预测环路中的反量化、

反变换单元之后，重建的运动补偿预测参考帧之前。因而，环路滤波是预测环路的一部分，属于环内处理，而不是环外的后处理。环路滤波的目标是消除编码过程中预测、变换和量化等环节引入的以分块效应为主的失真。由于滤波是在预测环路内进行的，减少了失真，存储后可为运动补偿预测提供较高质量的参考帧。

2.1.2 H.263标准

ITU-T于1996年公布了用于低码率的视频编码标准H.263。该标准仍采用H.261建议的混合编码框架，吸收了MPEG-1/2标准的某些特性，针对低码率应用进行了优化。在H.263信源编码器中，DCT、量化、Z字形扫描、VLC等处理与H.261建议类似，但不包括信道编码部分。根据不同的应用需要，H.263增加了若干高级功能，形成 H.263+（1998年）、H.263++（2000年）等，增加的高级选项达二十余项。

1.多种图像格式

H.263不仅支持H.261中的CIF和QCIF格式图像，还增加了 128×96 的sub-QCIF格式、 704×576 的4CIF格式和 1408×1152 的16CIF等格式。由于图像尺寸的调整，在图像划分中的GOB结构与H.261也略有不同。

2.半像素精度的运动估计

H.263建议中不仅可以用 16×16 像素的宏块为单位进行运动估计，还可以根据需要对 8×8 像素的子块单独进行运动估计，即每个宏块可使用4个运动矢量；H.263的运动矢量采用半像素精度，范围为 $(-16.0, +15.5)$ 。为此，用双线性内插来得到运动估计用的半精度像素的预测值，如图2.5（a）所示。在H.263中对运动矢量信息采用二维预测的方法，如图2.5（b）所示，MV为当前宏块运动矢量预测值，它等于相邻宏块运动矢量 MV_1 、 MV_2 、 MV_3 的中值。

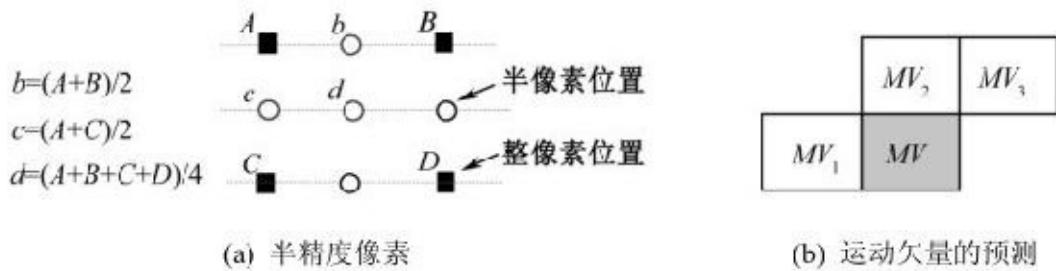


图2.5 H.263的运动矢量参数

3.三元VLC编码

H.263中对DCT系数在Zig-zag扫描后采用 (last,run,level) 组成的符号组进行三元 (变量) 变字长编码。其中，后两个变量“run”“level”与H.261的VLC相同，表示游程长度和非0系数值，而第一个变量“last”则表示当前系数是否为最后一个非零系数。这种方法可以进一步提高子块系数的熵编码效率，而且省去了“EOB” (块结束) 符号。

4.高级选项

H.263建议经历了多次修改和补充，并且通过附录的形式增加了20多个高级功能附加项，下面简要介绍其中最基本的4个高级选项。

(1) 无限制的运动矢量模式。在一般情况下，运动矢量的范围被限制在参考帧内，而无限制的运动矢量模式取消了该限制。当出现某一运动矢量所指向的参考像素超出编码图像区域时，使用边缘的图像值代替“这个并不存在的像素”。这种方法对改进边缘有运动物体的图像编码效果特别有效。

(2) 基于语法的算术编码。所有的变长编解码过程都用算术编解码过

程取代。在相同重建图像质量的前提下，使用这一模式可降低5%左右的码率。

(3) 高级预测模式。这种模式包括两种措施，可以减少方块效应，改进图像质量。一是对P帧的亮度分量采用所谓交叠块运动补偿(Overlap Block MC,OBMC)方法，即某一 8×8 子块的运动补偿由本块和周围块的运动矢量共同决定；二是对某些宏块使用4个运动矢量，每个子块都有一个运动矢量，取代原来一个宏块一个运动矢量的方式。

(4) PB帧模式。PB帧模式是提高视频图像压缩比的重要方法之一。如图2.6所示，和MPEG标准P帧、B帧不同，H.263中一个PB帧单元是由P、B两帧组成的一个编码单元。其中 P 帧是以前一幅解码后得出的 P 帧作为前向预测；而 B 帧是双向预测，以前一幅解码后的P帧作为前向预测，又以当前正在解码的P帧作为后向预测，预测值为前向预测和后向预测的平均值。

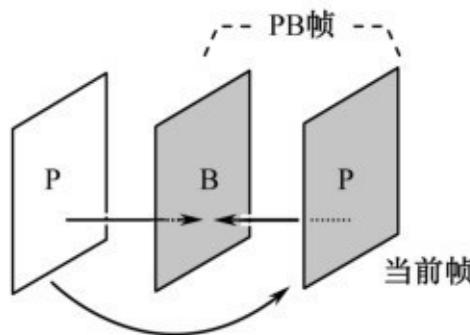


图2.6 H.263中的PB帧模式

2.2 MPEG-x标准

MPEG-1、MPEG-2和MPEG-4都是ISO/IEC的活动图像专家组(Moving Picture Expert Group,MPEG)制定的视音频编码标准。MPEG-1：“信息技术——具有1.5Mbps数据传输率的数字存储媒体活动图像及其伴音的编码”，标准号为ISO/IEC 11172,1993年颁布，主要包括系统、视频、音频等5部分。MPEG-2：“通用活动图像及其伴音的编码”，标准号为ISO/IEC 13818,1995年颁布，主要包括系统、视频、音频等10部分，其中的“视频”部分也被ITU-T接纳，成为H.262标准。MPEG-4：“视听对象的编码”，标准号为ISO/IEC 14496,1999年颁布，主要包括：系统、视觉信息、音频、一致性、参考软件、多媒体传送集成框架、优化软件、IP中的一致性、参考硬件描述、AVC等。MPEG-4一直在不断扩充，目前已有27个部分，这里主要关注其第2部分“视频编码”以及第10部分“高级视频编码”它和ITU-T H.264标准是一致的，故又称为H.264/AVC。

2.2.1 MPEG-1标准

MPEG-1标准的目标是满足各种存储媒体对压缩视频统一格式的需求，可用于对普通视频进行压缩处理，以1.5Mbps左右的码率提供连续的、活动图像的编码表示，最普及的应用如VCD、视频会议、视频监控、计算机存储等。MPEG-1的压缩原理、系统构成与主要技术和H.261相差无几，主要的不同之处在于。

1.约束参数

MPEG-1标准支持4:2:0格式的逐行亮度信号Y和两个色差信号C_b、C_r输入，对编码图像和比特流给了一定的范围和限制。如图像的宽度≤768像

素，图像的高度≤576行，帧率≤30Hz，运动矢量范围在-64～+63.5，输出比特率≤1865000 bps，等等。对于典型的应用，除CIF格式外，MPEG-1又定义了一种352×240的SIF（Source Input Format）格式。

2.4 种帧类型

MPEG-1引入了4种编码帧：I帧、P帧、B帧和D帧。其中I帧和P帧相当于H.261中的帧内编码帧和帧间预测帧。B帧是双向预测帧，能提供最大限度的压缩，它需要过去和将来的参考帧（I帧或P帧）进行用双向运动估计和补偿，但它本身不能用作预测参考帧。D帧为仅含DCT的直流分量的帧，用它可提供一种简单而具有一定质量的画面质量，如用于快进模式等。

3. 编解码顺序

MPEG-1的主要编解码过程与H.261类似，只是在有B帧时，要分别存储过去和将来的两个参考帧，以便进行双向运动补偿预测。另外在用到B帧时，编码时须对图像的帧顺序先进行调整，因为B帧在预测时要用到它的过去和将来的I帧或P帧作为参考图像。

4. 码流的层次

MPEG-1规定了编码视频比特流的语法，共分为6层，低4层（P、GOB、MB 和 B）和H.261中相同，增加了两个高层：由若干图像组成的图像组层，由若干GOP组成的序列层。

GOP是视频随机接入单元，其长度随意，可以包含一个或多个I帧。编码器必须根据需要选择GOP的长度以及I、P、B帧出现的频率和位置。在要求能随机操作、快速播放、回放等应用场合，可以使用较短的GOP。

2.2.2 MPEG-2标准

MPEG-2的视频部分也包括了MPEG-1的工作范围，从而使MPEG-1成为MPEG-2的一个子集，即MPEG-2的解码器可以对MPEG-1码流进行解码。MPEG-2的视频编码方案与MPEG-1大体类似，增加了以下多个新的编

码功能。

1. 档次和水平

MPEG-2为了区别不同应用，在编码参数上使用了“档次”和“水平”的概念，表2.1列出了MPEG-2的6种档次和4种水平的24种组合中常用的12种。例如，目前的标清数字电视（SDTV）采用的是主档次和主水平（MP@ML），而高清数字电视（HDTV）采用的是主档次和高水平（MP@HL）。

表2.1 MPEG-2的档次和水平

| 水平 档次 | High Level 1920×1080×30 或 1920×1152×25 | High 1440 Level 1440×1080×30 或 1440×1152×25 | Main Level 720×480×30 或 720×576×25 | Low Level 352×288×30 |
|--------------------------|--|---|--|-------------------------|
| Simple Profile | | | SP@ML | |
| Main Profile | MP@HL | MP@H1440L | MP@ML | MP@LL |
| 4:2:2 Profile | | | 4:2:2 P@ML | |
| SNR Scalable Profile | | | SNRP@ML | SNRP@LL |
| Spatial scalable Profile | | SSP@H1440L | | |
| High Profile | HP@HL | HP@H1440L | HP@ML | |

2. 基于场或帧的DCT

MPEG-2在把宏块数据分割为块时，可以按帧分割，也可以按场分割，相应地就可以在帧或场的模式下进行DCT编码，得到更好的压缩效果。当序列是逐行扫描时，或图像是帧方式时，采用的分割方式与MPEG-1相同；但对隔行扫描的帧图像，既可以采用上述按帧的分割方式，也可以采用按场的隔行分割方式。选择的标准是帧的行间相关系数和场的行间相关系数的大小。一般而言，对于静止或缓变图像区域宜采用按帧的DCT编码；反之，对于大的运动区域，则宜采用按场的DCT编码。

3. 多种帧间预测模式

MPEG-2规定了4种图像的运动预测和运动补偿方式，即基于帧的预测模式、基于场的预测模式、16×8的运动补偿以及双基（Dual Prime）预测模式。在具体使用时，必须考虑编码是针对帧格式图像还是场格式的图像。

4. 编码的可分级性

MPEG-2引入了3种编码的可分级性，即空间可分级性（Spatial Scalability）、时间可分级性（Temporal Scalability）以及信噪比可分级性（SNR Scalability）。可分级编码的特点是将整个码流分为基本码流和增强码流两部分，其中，基本码流仅提供一般质量的重建图像，但如果解码器“叠加”上增强层码流的质量增量部分，解码视频质量就可以得到很大的提高。可分级编码的优点是同时提供不同的编码服务水平，例如可以在一个公共的信道实现HDTV和SDTV的同传。

2.2.3 MPEG-4标准

与MPEG-1、MPEG-2不同，MPEG-4不仅着眼于定义不同码流下具体的压缩编码标准，而是更多地强调多媒体通信的灵活性和交互性，形成了MPEG-4自身的特点。

1. 场景描述和对象表示

MPEG-4规定了各种音视频对象的编码，除包括自然的音视频对象外，还包括图像、文字、2D和3D图形以及合成语音、音乐等。MPEG-4通过场景描述信息（各种对象的空间位置和时间关系等）来建立一个多媒体场景，并将它与编码对象一起传输。由于对各个对象进行独立的编码，可以达到很高的压缩比，同时也为在接收端根据需要对内容进行操作提供了可能，适应多媒体应用中人机交互的要求。

在MPEG-4中任何一个场景都被理解为若干视频对象（Video Object, VO）的组成。它们可以是一个矩形的视频帧或其中的一部分。在某个时刻的VO被称为视频对象平面（Video Object Plane, VOP）。各VOP的空间和时间上的关系用结构文本来描述。图2.7（a）是一个简单场景的例子，它包括三个对象：VOP₀（背景）、VOP₁（树）、VOP₂（人），图2.7（b）是三个对象在场景中组成的逻辑关系。根据需要，一个VO还可作进一步分解，例如图中VOP₁可以分成树冠和树干等。

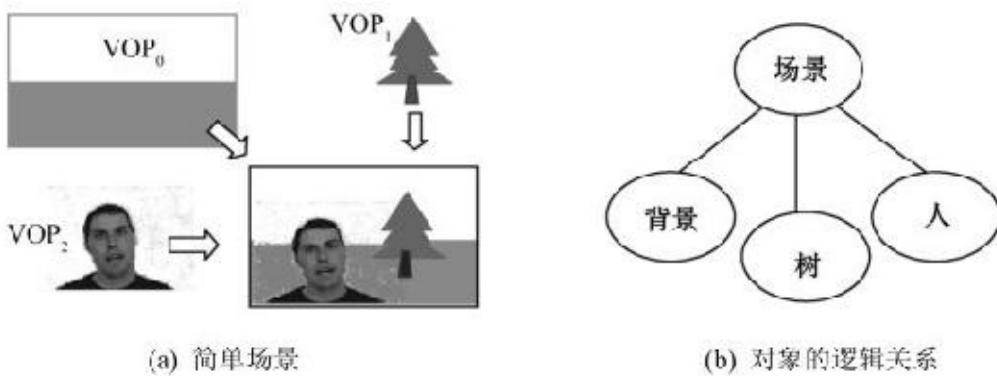


图2.7 MPEG-4的场景结构

2.基于对象的编码

MPEG-4采用的是基于对象的编码方式，不同的对象可以采用不同的编码工具。MPEG-4为每个VOP定义一个 α 平面或 α 通道，用它表示每个对象在场景中占据的位置，以及相互间的混合效果。MPEG-4的视频对象编码模块的作用就是对亮度、色度以及 α 信息进行高效编码。由于这里所需处理的是具有实际意义的单元，即具有任意形状的视频对象，因此要进行对象形状参数的处理，这是过去国际标准中没有的、新引入的参数。图2.8是MPEG-4视频对象的编码框图。在MPEG-4的验证模型（Verification Model, VM）中，对形状 α 平面信息一般采用基于上下文的算术编码（Arithmetic Coding, AC）或DCT算法，对于活动视频仍采用H.263混合编码，对静止图像采用JPEG编码或小波变换，根据应用场合的要求来决定。

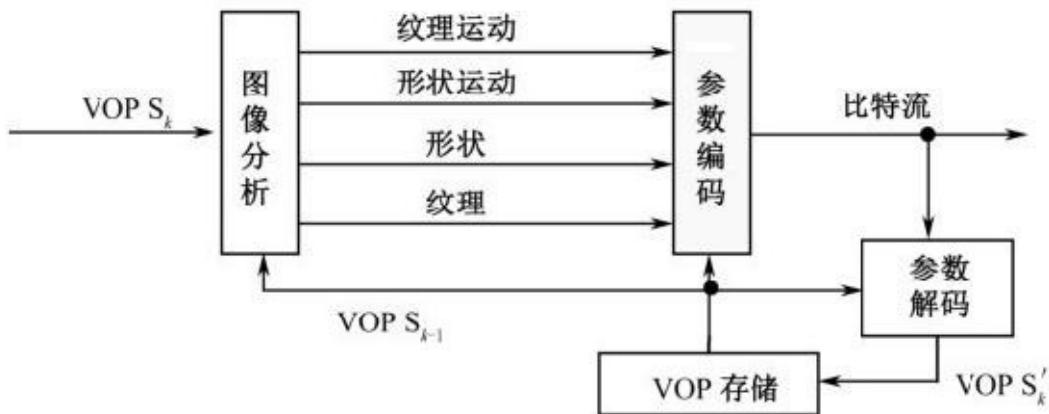


图2.8 MPEG-4中视频对象的编码

MPEG-4视觉信息的编码部分包括了许多内容，以适应各种多媒体应用的要求。例如，同时支持自然和合成可视信息（如图形、计算机动画等）的编码，支持对网格图形对象、3D人体编码等。在此基础上还提供一些高层次的编码方法，如基于内容的编码，允许对任意形状视频对象进行的编码。

3. 编码工具集

MPEG-4为视信息编码提供了一个包含多种算法的工具集，对各种应用提供不同的解决方法，供用户选择。工具集可以不断地扩充进新的编码工具，甚至于用户自己的编码方法也可放入工具集。这样，使得标准可随时跟上技术的发展而保持长时间有效。

2.3 H.264/AVC标准

H.264/AVC是ITU-T的视频编码专家组 (Video Coding Expert Group,VCEG) 和ISO/IEC的活动图像编码专家组 (MPEG) 的联合视频组 (Joint Video Team,JVT) 开发的又一代数字视频编码标准 , 它既是ITU-T的H.264 , 又是ISO/IEC的MPEG-4的第10部分 : AVC。如图2.9所示 , H.264/AVC 和以前的标准一样 , 也是帧间预测加变换的混合编码模式 , 但在多方向帧内预测、多模式运动估计、整数变换、自适应VLC编码、环路滤波、分层编码语法等方面都有它的独到之处。因此H.264/AVC算法具有很高的编码效率 , 目前已得到了广泛的应用。

H.264/AVC 的编码效率大约是 H.263 (或 MPEG-2) 的一倍。必须说明的是 , 优越性能的获得不是没有代价的 , 其代价是计算复杂度的大大增加 , 据估计 , 编码的计算复杂度大约相当于H.263的3倍 , 解码复杂度大约相当于H.263的2倍。

H.264/AVC主要有3个档次分别适用于不同的业务需求 : 基本档次主要面向会议电视、可视电话等实时视频通信应用 ; 主要档次主要面向数字电视应用 ; 扩展档次主要面向网络视频、移动视频传输及其他应用场合。

从编码传输的角度来看 , H.264/AVC可以分为两层 : 视频编码层 (Video Coding Layer,VCL) 负责高效的视频内容表示 ; 网络提取层 (Network Abstraction Layer,NAL) 负责以网络所要求的方式对数据进行打包和传送。这样 , 高效率的视频编码和顺畅的网络传输任务可分别由VCL 和NAL来完成。

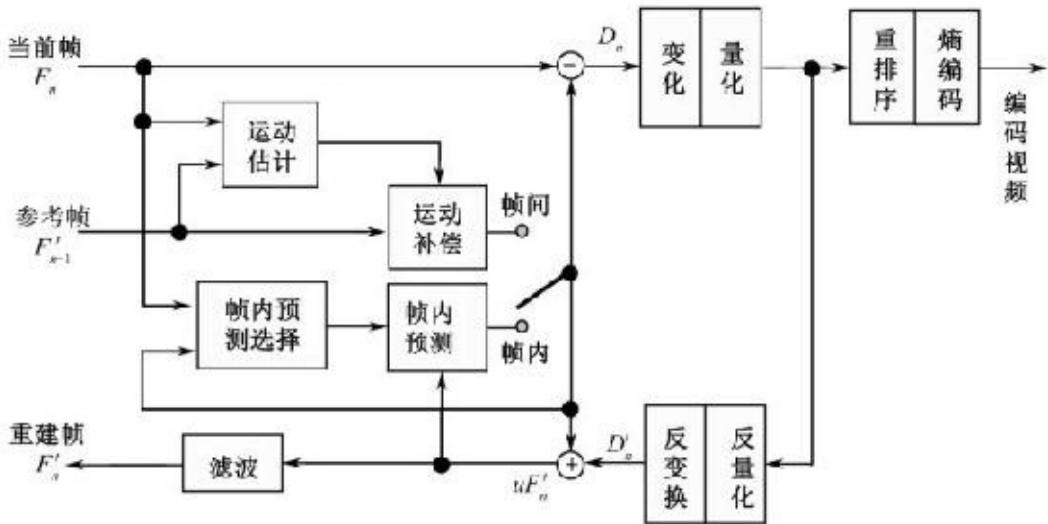


图2.9 H.264/AVC编码结构

2.3.1 多方向帧内预测

在以往的标准中，对帧内图像直接将宏块像素进行DCT变换、量化和熵编码就行了。为提高编码效率，H.264/AVC引入了空间域帧内预测技术，即充分利用相邻块之间的相关性进行压缩。预测基于 4×4 的小块进行，但对大面积缓慢变化的图像也可基于 16×16 的宏块。

H.264/AVC中，为适应图像中不同的纹理走向，对 4×4 子块的帧内预测共设计了9种预测模式（预测方向）选择，模式0~8，编码时从中选择一个最优的模式，以求预测值更接近预测对象（稍具体介绍见4.2节）。

2.3.2 多模式运动估计

在帧间预测编码时，亮度宏块可划分成形状不同的运动估计区域。划分类型包括 16×16 , 16×8 , 8×16 , 8×8 4种。当选用 8×8 模式时，可以进一步划分成 8×4 , 4×8 和 4×4 3种，共7种模式。这种多模式的灵活和细致的划分，更切合图像中实际运动物体的形状，有效提高了运动估计的精确程度。

H.264/AVC利用整像素点的亮度值进行 $1/2$ 和 $1/4$ 内插，使运动估计精度达到 $1/4$ 像素。内插过程先是通过6抽头的滤波器来获得 $1/2$ 像素精度，然后用线性滤波器来获得 $1/4$ 像素的精度。由于4:2:0采样的关系，色度的运动精度实际达到 $1/8$ 像素，也是通过线性滤波器插值得到的。

H.264/AVC 可以采用多参考帧图像来进行运动预测，最多前向和后向各16帧。和 H.263比较，由于H.264使用多参考帧预测，对周期性运动、平移封闭运动和不断在两个场景间切换的视频具有非常好的运动预测效果。如图2.10所示是一个近似的周期运动，在刚刚编码好的若干参考帧中，H.264/AVC 编码器为每个目标宏块选择能给出更准的预测帧，图中虚线为在多参考帧中选择的最相当的参考帧，而不仅仅是前帧，并为每一宏块指示是哪一帧是被用于预测的。此外，使用了多参考图像后，H.264/AVC 不仅能够提高编码效率，同时也能实现更好的码流误码恢复，但需要增加额外的时延和存储容量。

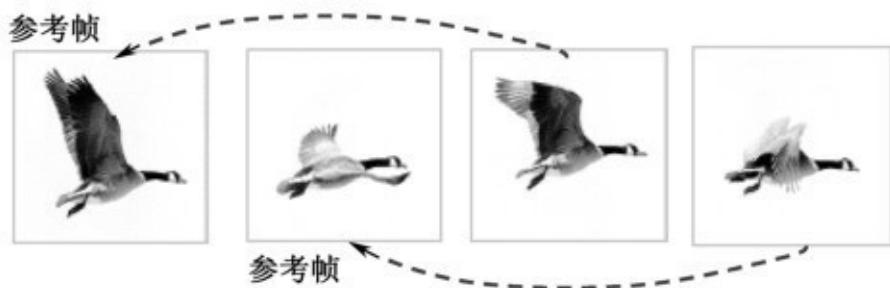


图2.10 H.264多参考帧预测示意

2.3.3 整数变换和熵编码

1. 整数变换

H.264/AVC与之前的标准相似，对预测残差采用基于块的变换编码，但变换是整数操作而不是实数运算，其过程和DCT基本相似。变换的单元是 4×4 块，而不是以往常用的 8×8 块。由于变换块尺寸的缩小，运动物体的划分更精确，这样，不但变换计算量比较小，而且在运动物体边缘处的衔接误差也大为减小。为使小尺寸块的变换方式对图像中较大面积的平滑区域不产生块之间的灰度差异，还对帧内宏块亮度数据的16个 4×4 块的DC系数进行第二次 4×4 块的哈达玛（Hadamard）变换，对色度数据的4个 4×4 块的DC系数进行 2×2 块的哈达玛变换。

2. 自适应熵编码

H.264/AVC 中熵编码有两种方法，一种是对所有待编码的符号采用内容自适应变长编码（Context-based Adaptive VLC, CAVLC），另一种是采用内容自适应的二进制算术编码（Context-based Adaptive Binary Arithmetic

Coding,CABAC)。CABAC是可选项，其编码性能比CAVLC稍好，但计算复杂度也较高。

2.3.4 差错控制

H.264/AVC中包含了用于差错消除的工具，增加了压缩视频在误码、丢包多发环境（如移动信道或IP信道）中传输的鲁棒性。例如，为抵御传输差错，H.264/AVC视频流中的时间同步通过采用帧内图像刷新完成，空间同步由条结构编码（Slice Structured Coding）支持；同时为了便于误码以后的再同步，在一幅图像的视频数据中还提供了一定的重同步点；此外，还有数据分段、灵活宏块排序（Flexible Macroblock Ordering,FMO）、任意条带排列（Arbitrary Slice Ordering,ASO）、误码掩盖（Error Concealment）等措施以保证视频的服务质量。

2.4 AVS标准

基于我国自主创新技术和国际公开技术所构建的音视频编码标准（Audio Video coding Standard,AVS）正式名称是信息技术——先进音视频编码，包括系统、视频、音频、数字版权管理4类技术和性测试等多个部分。最早的标清AVS第2部分“视频”于2006年3月成为国标颁布实施（GB/T 20090.2—2006）。后续开发的用于高清的AVS标准已完成，并成为IEEE1857广播组标准，同时也是AVS标准的P16部分。

AVS 视频与现有国际标准 H.264/AVC 相比，性能相当，在技术上处于国际先进水平。但AVS方案简洁，编码复杂度相当于H.264/AVC的30%，解码复杂度相当于H.264/AVC的70%。

AVS视频遵循当前主流的技术路线，采用混合编码框架，包括变换、量化、熵编码、帧内预测、运动估计、帧间预测、环路滤波等技术模块。不同之处在于AVS提出了一批具体的优化技术，降低了编码复杂度。这些改进的技术主要包括：8×8整数变换、量化、帧内预测、1/4精度像素插值、特殊的帧间预测和运动补偿、二维熵编码、去块效应环内滤波等。

目前，AVS 标准还在不断地扩展和进化，2012年7月公布了 AVS+标准，基本等同于H.264/AVC标准。2014年又推出了目标为超高清视频（UHDTV）应用的AVS-2标准。

下面简单介绍 AVS 帧内预测模式和多模式运动估计，仅从以下两点就可以看出它与H.264/AVC的不同之处。

（1）帧内预测。AVS视频的帧内预测基于8×8块大小，亮度分量只有5种预测模式，降低了帧内预测模式选择的计算复杂度，但性能与H.264十分接近。除预测块尺寸及模式种类的不同外，AVS视频的帧内预测还对相邻像素进行了滤波处理来去除噪声。

(2) 多模式运动补偿。可变块大小的运动补偿是提高运动预测精确度的重要手段之一，对提高编码效率起重要作用。AVS将最小宏块划分定为 8×8 ，这一限制大大降低了编解码器的复杂度。这是因为小于 8×8 块的运动估计模式对低分辨率图像编码效率影响较大，而对高分辨率图像编码则影响甚微，实验数据表明，去掉小于 8×8 块的运动预测模式，整体性能降低2%~4%，但其编码复杂度则可降低30%~40%。

2.5 VC-1标准

VC-1是2003年微软公司在WMV9 (Windows Media Video 9) 的基础上提出的一种新的视频压缩编码标准，并于2006年由美国电视电影工程协会（ SMPTE ）正式颁布。VC-1标准整合了MPEG和H.264/AVC的优点，复杂度比H.264/AVC约低，压缩率与H.264/AVC接近，现在已广泛应用于数字视频压缩领域。

VC-1视频编码如图2.11所示，它与H.264/AVC类似，也是采用基于块运动估计预测和空间变换的混合编码技术。与H.264/AVC不同的是，VC-1帧内预测是在DCT域中进行的，且熵编码采用的是自适应变长编码（ Adaptive VLC ），而不是CAVLC或CABAC方式。

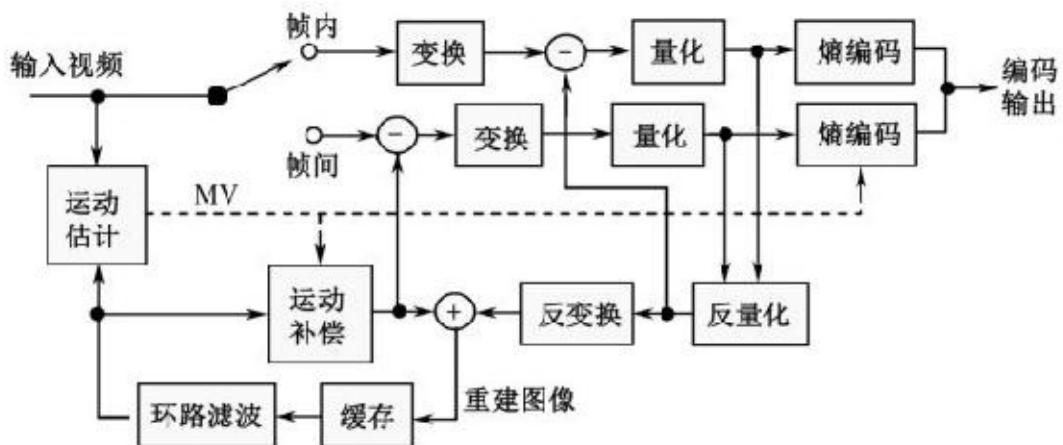


图2.11 VC-1编码器框图

与H.264/AVC比较，VC-1视频编码具有以下几个特点。

(1) 为减小缓存容量，VC-1的帧间预测参考帧只有一帧，即前一帧的重建帧。

(2) VC-1对于帧内宏块采用固定的 8×8 的变换方式，而帧间宏块则采用自适应分块变换技术，可将 8×8 块进一步划分为更小的子块，如 8×4 、 4×8 、 4×4 等。

(3) VC-1的熵编码采用多码表方式的变长编码，对不同的数据，如量化后的变换系数、运动矢量、编码块模板、宏块编码模式标识等，采用不同统计特性的码表。

(4) 为克服基于块的视频编码中常见的方块效应，VC-1在编码环路中引入了去除方块效应的环路滤波器。同时，对帧内模式压缩的区域采取重叠变换技术来弥补环路滤波的不足。

2.6 HEVC标准

为满足海量视频数据的传输和存储要求，在2003年制定H.264/AVC视频编码标准获得巨大成功后，新一代高效视频编码（High Efficiency Video Coding,HEVC）国际标准第1版在ITU-T的VCEG 和ISO/IEC的MPEG成立的视频编码联合协作小组（Joint Collaborative Team on Video Coding,JCT-VC）的通力合作下，于2013年4月正式发布。此后，包含多项扩展的HEVC第2版于2014年10月发布，进一步扩大了HEVC的应用范围。紧接着在2015年4月发布了包括3D扩展附件的第3版。

HEVC的技术特点和优势集中表现在以下三个方面。

（1）HEVC的核心目标：在H.264/AVC HP（High Profile）的基础上，在保证相同视频质量的前提下，视频流的码率减少50%。在提高压缩效率的同时，允许编码端适当提高复杂度（3倍以下计算复杂度）。

（2）HEVC的编码框架：沿用H.264/AVC的混合编码框架，用帧间和帧内预测编码来消除时间域和空间域的冗余度，对残差进行变换编码来消除变换域的冗余度，用熵编码来消除统计域的冗余度。在混合编码框架内，设计了多种新的编码工具或技术，提高了视频压缩效率。

（3）HEVC的技术创新：灵活的图像四叉树（Quad-tree）结构和条、片划分技术，多角度帧内预测技术，运动信息融合技术，高精度运动补偿技术，自适应环路滤波技术，利于并行计算的波前技术以及基于上下文的算术编码技术。

2.6.1 HEVC标准的进程

1. 视频应用的新需求

无论是在消费视频、专业视频，还是网络视频应用领域，对高分辨率的需求都在日益增加。目前，正处于从标清（Standard Definition,SD）视频到高清（High Definition,HD）视频应用的转化过程中，而且已经开始向超高清（Ultra-HD,UHD）视频演进。有代表性的数字视频分辨率，也是HEVC支持的分辨率详见表2.2。表中“1K”为水平清晰度，约等于1000线。从表2.2中可见，HEVC支持各类规格的视频，从CIF（320×288）到HD1080p（1920×1080），直至超高清视频UHD4320p（7980×4320）。

表2.2 常见数字视频的分辨率

| 视屏格式 | 分辨率 | 显示比 | 标准 |
|----------|----------------------|------|---------------|
| CIF | 352×288 隔行 | 4:3 | H.261 |
| SD | 720×576 / 720×480 隔行 | 4:3 | ITU-R BT 601 |
| HD (2K) | 1920×1080 隔行/逐行 | 16:9 | ITU-R BT 709 |
| UDH (4K) | 3840×2160 逐行 | 16:9 | ITU-R BT 2020 |
| UHD (8K) | 7680×4320 逐行 | 16:9 | |

与此同时，在有些场合对数字视频的帧率、像素的精度要求也出现更高的要求，帧率从30 fps向60 fps、120 fps甚至240fps升级，像素的比特深度从8bit向10bit、12bit扩展。

近年来在数字视频应用中，已出现了2个明显的需求或发展趋势，一个是从单一视点向立体视频、多视点视频发展，另一个是从单层视频向多层可分级视频发展。

由于上述数字视频应用的种种新的需求和发展趋势，如果继续采用以往的视频编码标准，就出现如下一些局限性。

(1) 每幅图像的宏块个数也随之增长，导致用于编码宏块的预测模式、运动矢量、参考帧索引和量化级等宏块级参数信息所占用的码字过多，而用于编码残差部分的码字所占比例有所下降。

(2) 单个宏块所表示的图像内容的信息减少，导致相邻的4×4或8×8块变换后的低频系数相似程度也大大提高，导致出现大量的冗余。

(3) 表示同样运动的运动矢量的幅值增大，用来对运动矢量进行预测的压缩率有所降低。

(4) 待编码的数据迅速增加，以往的视频编码标准比较适合串行计算，制约了用并行计算的方式来提高编码速度的实行。

为克服以上局限适应新的视频应用需求，ITU-T和ISO/IEC联合制定了适合高清晰度视频的高效视频编码标准HEVC。

2. 主要的时间节点

HEVC标准制定是一个长期的开发、补充、修改和扩展的过程，今后若干年还有大量工作要做。

(1) 前期工作

2004年，ITU-T VCEG和ISO/IEC MPEG完成了H.264/AVC的High Profile之后，就开始着手准备下一代的视频压缩标准的准备工作。此后，VCEG和MPEG对多种H.264/AVC标准的潜在改进技术进行了调研，达成了现有编码标准的效率可以显著提升的共识。

VCEG在2005年初确定了今后新标准制定的关键技术领域（Key Technical Areas, KTA），建立了一个通用的KTA软件代码库。KTA代码库由H.264/AVC的参考软件JM（Joint Model）开发而来，它引入并验证了很多新技术，用于后来几年中测试和验证各种新提案。

2005—2008年，MPEG开始探索显著提升编码效率的可能性。它组织了数个工作小组，并在2009年4月为这些技术发布了一个“征集证据”。随后专家开始观测征集而来的评估和测试结果。

2010年以前，VCEG 的主要工作是一个名为H.265和 H.NGVC（Next Generation Video Coding）尝试项目，2010年联合MPEG将其演进为HEVC。NGVC的基本要求是在相同的主观图像质量条件下，比H.264/AVC HP的码率降低50%，而计算复杂度不超过H.264/AVC HP的3倍。NGVC在和H.264/AVC HP的相同视觉质量条件下，可以提供25%的码率降低和50%的复杂度降低，在更高复杂度时还可提供更多的码率降低。

MPEG 从2007年开始了类似 NGVC 的名为高性能视频编码（High-performance Video Coding,HVC）项目。2007年7月，将获得码率降低50%的指标作为该项目的主要目标。MPEG开发的HVC在KTA参考软件编码器

上完成了指标评估工作。到2009年7月，实验结果表明，HVC和H.264/AVC HP相比较平均码率降低约为20%。这些结果促进了MPEG努力联合VCEG共同开发新标准。

（2）JCT-VC成立后的工作

2010年1月，由ITU-T的VCEG和ISO/IEC的MPEG联合成立了视频编码联合小组（Joint Collaborative Team on Video Coding,JCT-VC），共同制定下一代视频编码标准。同时发布了一份正式的关于视频压缩技术的提案征集书（Call for Proposals,CfP）。

2010年4月，在JCT-VC的第1次会议上评估了所征集的27份提案。评估结果表明，有些提案技术在和H.264/AVC同样的视觉质量下，在多次测试中只需要一半的码率，其代价是计算复杂度增加2~10倍。而另一些提案取得了较好的主观质量和适当的码率，且计算复杂度比AVC HP参考编码器更低。在这次会议上，采用HEVC作为JCT-VC联合开发标准的名称。自这次会议开始，JCT-VC集成若干测试提案的技术为单一的软件编码库，即待考虑测试模型（Test Model under Consideration, TMuC），用于完成今后评估各种提案指标的测试实验。虽然TMuC在以往标准基准上有显著的效率提升，但在每个功能块中它仍存在多余的编码工具，呈现为一个来自多个提案的简单集合体。

2010年7月，在JCT-VC第2次会议上，通过测试TMuC每个组件的方式，选择每个功能块所需的编码工具，形成TMuC的工具集。

2010年10月，JCT-VC第3次会议上，在详尽的组件测试的基础上，发布了HEVC规范的第1版工作草案（Working Draft 1,WD1）和相应的测试模型版本1（HEVC test Model 1,HM 1）。与TMuC相比，HM 1删除了不必要的高复杂度的编码工具，大大简化了代码库。在此后的一系列会议上，对HEVC的编码工具和指标进行了许多修改。

2012年2月，通过了在第6版工作草案文档基础上的HEVC委员会草案。

2012年5月，JCT-VC宣布用于可分级视频编码（Scalable Video

Coding,SVC) 的HEVC提案的评估将于2012年10月举行。这导致了对HEVC的不断扩展，使其增加对SVC的支持。

2012年6月，MPEG通过了基于第8版工作草案（WD8）的HEVC国际标准草案。

2013年1月，ITU-T宣布HEVC获得了ITU-T AAP（Alternative Approval Process）的第一步通过。JCT-VC将继续进行HEVC的扩展工作，如支持12bit和4:2:2/4:4:4彩色取样的视频。同时，MPEG宣布HEVC已经提升为国际标准最后草案状态。

（3）HEVC正式发布

2013年4月，HEVC/H.265正式通过成为ITU-T标准，完成了第一个里程碑。标准文本预先发表在2013年4月18日ITU-T的网页上。

2014年1月，HEVC标准正式由ISO/IEC颁布，这是第1版，它在ITU-T称为H.265，在ISO/IEC称为MPEG-H/Part2 23008—2。

2014年6月，HEVC/H.265标准正式在ITU-T的网络平台上发布，供大家自由下载。

2014年10月，HEVC 的第2版发布，包括3个新近的扩展的附件：范围扩展（Range Extensions,RExt），多视点扩展（Multi-View Extensions）——MV-HEVC（MultiView HEVC），可分级扩展（Scalability Extensions）——SHVC（Scalability HEVC）。

2015年4月，HEVC的第3版发布，正式包含了3D-HEVC扩展的附件Annex I-3D high efficiency video coding。

（4）产业响应几例

2012年6月，提供专利授权方案的公司MPEG LA宣布他们将开始为HEVC专利创建联合许可的工作。

2013年4月，Ateme宣布基于OpenHEVC的第一个开源HEVC软件播放器的实现。开源软件OpenHEVC解码器支持Main档次的HEVC，可在单核CPU上实现1080p/30fps的视频解码。

2014年9月，MPEG LA宣布了他们的HEVC专利池，它包含来自23个

公司的专利，专利费用为每个HEVC产品0.2美元。Apple公司宣布在iPhone6和iPhone6+手机的视频交互软件FaceTime中支持HEVC/H.265。

2015年2月，AMD公司宣布，在他们的加速处理单元（APU）设置了一个统一视频解码器（UVD）专用芯片，这是第一个基于x86 CPU的HEVC硬件解码器。

2015年7月，腾讯公司的腾讯视频播放器在9.8新版本的电影频道增加了HEVC专区，可以更流畅地观看高清视频。

3.HEVC的压缩性能

2014年5月，英国BBC和西苏格兰大学对HEVC的MP（Main Profile）和H.264/AVC的HP的主观视频质量进行了比较测试。对一组视频分别用HEVC的HM 12.1和H.264/AVC的JM 18.5进行编码。结果显示，与H.264/AVC比较，用HEVC编码的码率平均降低59%。不同分辨率视频的平均码率降低如表2.3所示。从表中还可以看出，HEVC 改进编码效率的优势更多地体现在对高清视频的压缩上。

表2.3 相比H.264/AVC平均码率降低量

| 编解码标准 | 编码视频的分辨率 | | | |
|-------|----------|------|-------|--------|
| | 480p | 720p | 1080p | 4K UHD |
| HEVC | 52% | 36% | 62% | 64% |

4.HEVC的应用

随着视频应用范围的扩展和视频质量的提升，HEVC的应用领域正以意想不到的方式和速度在发展。就目前和不久将来的估计，常见的HEVC应用大约包括以下几方面。

（1）消费电子方面。如各类电视广播，包括在光纤、铜线、卫星、陆地网络上的电缆电视，录像，视频内容生产和发布，数字影院和家庭影院等。

（2）网络视频方面。如英特网视频流、下载和播放、移动流媒体、视频通信、实时会话服务、如视频会议、可视电话、远程呈现等。

（3）视频监控方面。如远程视频监控和网络视频监控等。

(4) 卫生、教育方面。如远程医疗、医疗图像、远程教育、网络课程等。

(5) 媒体存储方面。如存储媒体，云存储、网络视频存储、海量视频库、无线显示等。

总之，作为新一代视频编码技术，HEVC有着前面几代编码技术（如MPEG-4、H.264/AVC）等在压缩率、图像质量、并行处理等方面具有无可比拟的优势，在兼容性方面也是如此。目前，已有HEVC的初步应用也证实了上述优势，HEVC的广泛普及和应用即将实现。

5.HEVC文档提要

以ITU-T于2015年4月颁布的H.265（HEVC）第3版文档为例，共有600多页，包括11项条款内容和9个附录，这里仅给出它们的标题，从中可大体了解HEVC包含的内容。

2013年4月，ITU-T发布了H.265（HEVC）第1版，主要内容如下。

Clause 0 概述（Introduction）

Clause 1 范围（Scope）

Clause 2 标准参考（Normative references）

Clause 3 定义（Definition）

Clause 4 缩略语（Abbreviation）

Clause 5 惯例（Convention）

Clause 6 比特流和图像格式、划分，扫描过程和邻域关系（Bitstream and picture formats,partitioning,scanning processes and neighbouring relationships）

Clause 7 语法和语义（Syntax and semantics）

Clause 8 解码过程（Decoding process）

Clause 9 解析过程（Parsing process）

Clause 10 子比特流提取过程（Sub-bitstream extraction process）

附录A 档次、等级和水平（Profile,tiers and levels）

附录B 字节流格式（Byte stream format）

附录C 假设参考解码器 (Hypothetical reference decoder)

附录D 补充增强信息 (Supplemental enhancement information)

附录E 视频可用信息 (Video usability information)

2014年10月，ITU-T发布了H.265 (HEVC) 第2版，在第1版的基础上增加了3个附录，相当于HEVC的扩展：

附录F 多层扩展的公共指标 (Common specifications for multi-layer extensions)

附录G 多视点HEVC (Multiview high efficiency video coding)

附录H 可分级HEVC (Scalable high efficiency video coding)

2015年4月，ITU-T发布了H.265 (HEVC) 第3版，在第2版的基础上增加了1个附录，相当于3D-HEVC扩展：

附录I 3D-HEVC (3D high efficiency video coding)

对于关心技术的读者，HEVC文档中的Clause 6到Clause 9无疑是重要的部分。本书随后的分析将有助于对这几项理解，但由于HEVC文档的内容庞杂、细节繁多，本书也不可能面面俱到，而是主要集中于HEVC的几项关键技术。

2.6.2 HEVC技术概要

HEVC的基本编码原理和H.264/AVC基本一致，即预测加变换的分块编码；在编码细节上和H.264/AVC也很接近，包含帧内预测、帧间预测、运动估计与补偿、正交变换、量化、环路滤波、熵编码和重建等编解码模块。但是，与H.264/AVC相比，HEVC几乎在每一个编码环节上都采取了重要的改进措施。

这里简要介绍HEVC的几项技术特点，详细的内容将在以后各章叙述。

1. 编码结构

(1) 三种编码方式

为应对不同应用场合，HEVC设立了GOP的三种编码方式：全帧内（All Intra,AI）编码、低延时（Low Delay,LD）编码和随机接入（Random Access,RA）编码。在全帧内编码中，每一帧图像都是按帧内方式进行空间域预测编码，不使用时间参考帧。在低时延编码中，只有第一帧图像按照帧内方式进行编码，随后的各帧都作为一般的P帧或B帧进行编码，便于交互式实时通信应用。在随机接入编码中，主要是等级B帧结构（Hierarchical B Structure），周期性地（大约每隔1s）插入一纯随机接入（Clean Random Access,CRA）帧，成为编码视频流中的随机接入点，方便信道转换、搜索及动态流媒体服务等应用。

（2）条和片的划分

HEVC允许将图像帧划分为若干条，条的划分以编码树单元（Coding Tree Units,CTU）为界，成为独立的编码区域。

为了支持并行计算和差错控制，某一个条可以划分为更小的条，称为分条或条分割（Slice Segment,SS）。每个SS包含整数个CTU，并同属于一个NAL单元。每个SS都可独立地进行熵编码（熵编码前须初始化），无须参考其他SS，但去方块滤波可以跨边界进行。例如，在多核的并行处理中，就可以安排每个核单独处理一个SS。

在HEVC中新引入了片的划分，用水平和垂直的若干条边界将图像帧划分为多个矩形区域——片，每一个片包含整数个CTU，片之间也可以互相独立，以此实现并行处理。

（3）四叉树单元划分

HEVC依然采用分块编码方式，但块的尺寸是可以自适应改变的。HEVC将编码帧分为若干相邻但不重叠的方形编码树块（Coding Tree Blocks,CTB），它是进行预测、变换、量化和熵编码等处理的基本单元，其尺寸可以是 16×16 、 32×32 或 64×64 。同一位置的一个亮度CTB和两个色度CTB，再加上相应的语法元素形成一个编码树单元。

CTU又可以按照四叉树结构递归地分解为若干部码单元（Coding Units,CU），最多可有4层分解。每个CU包含一个亮度编码块（Coding

Blocks,CB) 、两个色度CB及相应的语法元素。

CU还可以分解（或不分解）为更小的预测单元（Prediction Units,PU）和变换单元（Transform Units,TU）。自然，每个PU包含亮度、色度预测块（Prediction Blocks,PB）和相应的语法元素，是进行预测运算的基本单元。每个TU包含亮度、色度变换块（Transform Blocks,TB）和相应的语法元素，是进行变换和量化的基本单元。显然，TB和PB在几何位置上有可能是重合的。

2.帧内预测

HEVC 的帧内预测采用基于块的多方向帧内预测方式来消除图像的空间相关性，但比H.264/AVC预测方向更细、更灵活。HEVC为亮度信息定义了33种不同的帧内预测方向，连同平面和直流（DC）模式，总共35种帧内预测模式。DC模式特别适用于图像的平坦区域。预测方向不是用几何角度来表示的，而是用像素的个数或格数来表示的。帧内预测单元的尺寸从 4×4 到 32×32 。

为提高帧内预测的效率，HEVC对 8×8 或更大的PU的预测参考像素进行平滑滤波预处理。平滑滤波器实际上是一个简单的一维有限冲激响应低通滤波器，用它对预测方向上的参考像素进行滤波，可以减少噪声对预测的影响，提高预测的精度和效率。

3.帧间预测

HEVC的帧间预测编码总体上与H.264/AVC相似，但进行了如下几点改进。

（1）可变尺寸的预测块

每个CTU都可以递归地分解为更小的方形CU，这些小的帧间CU还可划分成为更小的预测单元PU。帧间预测的CU可以使用对称的和非对称运动划分（Asymmetric Motion Partitions,AMP），允许非对称地划分 64×64 到 16×16 的CU为更小的PU,PU可以是方形的，也可以是矩形的。由于采用AMP方法，HEVC使帧间编码的预测单元PU在运动估计和运动补偿中更精确地符合图像中运动目标的形状，因此可以提高编码效率。

(2) 高精度运动估计

HEVC亮度分量的运动估计精度为1/4像素。为获得非整数样点的数值，不同的非整数位置亮度的插值滤波器的系数是不同的，分别采用一维8抽头和7抽头的内插滤波器产生1/2和1/4像素的亮度值。对于一般的4:2:0数字视频，HEVC色度分量的运动估计精度为1/8像素。色度分量的预测值由一维4抽头内插滤波器用类似亮度的方法得到。

(3) 运动参数的高效编码模式

HEVC对运动参数的编码有三种模式：对运动参数直接编码的Inter模式、改进的Skip模式和Merge模式。这样，对于每个帧间编码的PU，可以在上述三种编码模式中选择。三种模式中，前两种模式和 H.264/AVC 类似，Merge 模式是 HEVC 新引入的一种“运动合并”技术。在以往的标准中，对每个帧间预测块都要传输一套运动参数。而在HEVC的Merge模式中，为进一步提高编码效率，将毗邻的运动参数相同或相近的几个 PU 合并起来形成一个所谓的“小区”，只为每个小区传输一次运动参数即可，不必为每个PU分别传输运动参数。

4. 变换和量化

(1) 离散余弦和正弦变换

HEVC的变换运算与H.264类似，也是对预测残差进行近似DCT的整数变换。HEVC在一个编码单元CU内进行变换运算时，可以将CU按照编码树层次细分，从 32×32 直至 4×4 的小块。HEVC还支持 4×4 的离散正弦变换

(Discrete Sine Transform,DST)，用于对 4×4 块的帧内预测残差编码。在帧内预测块中，那些接近预测参考像素的像素，如左上边界的像素，比那些远离参考像素的像素预测得更精确，预测误差较小，而远离边界像素的预测残差则比较大。DST对编码这一类的残差效果比较好，这是因为不同 DST 基函数在起始处很小，往后逐步增大，与块内预测残差变化的趋势比较吻合；而DCT基函数在起始处大，往后逐步衰减。

(2) 率失真优化的量化

HEVC 的量化是在整数 DCT 变换时一并完成的，并采用了率失真优化

的量化 (Rate Distortion Optimized Quantization,RDOQ) 技术，在给定码率的情况下选择最优的量化参数使重建图像的失真最小。量化操作是在变换单元 TU 中分别对亮度和色度分量进行的。在 TU 中所有的变换系数都是按照同一个量化参数进行量化和反量化的。

5. 熵编码

(1) 系数的扫描方法

在HEVC中，将TU中量化后的系数通过特定的扫描方式形成一维数据，然后对它进行熵编码。系数的扫描是以 4×4 的“系数组”为单位进行的，一个大的TU可分为若干个 4×4 的系数组。如果 TU 是帧内预测模式，则可使用水平、垂直和对角三种方式扫描。在帧间预测块中，由于可以使用矩形的变换，因此扫描方式要随TU的形状而改变。

(2) 自适应算术编码

HEVC需对量化、扫描后的变换系数进行熵编码，以获得进一步的信息压缩。HEVC采用了和H.264/AVC非常类似的基于上下文的自适应二进算术编码 (CABAC) 算法进行熵编码。

(3) 波前方式的并行处理

考虑到高清、超高清视频编码的巨大运算量，HEVC提供了基于条和基于片的并行编码和解码处理的机制。然而，这样又会引起编码性能的降低，因为这些条和片是独立预测的，并行运算打破了穿越边界的预测相关性，每个条或片用于熵编码的统计必须从头开始。为了避免这个问题，HEVC提供了一种波前并行处理 (Wavefront Parallel Processing,WPP) 的熵编码技术，在熵编码时不需要打破预测的连贯性，而是尽可能多地利用上下文信息。

6. 环路滤波

环路滤波的目标就是消除编码过程中预测、变换和量化等环节引入的失真。由于滤波是在预测环路内进行的，减少了失真，因此存储后为运动补偿预测提供了较高质量的参考帧。HEVC除采用和 H.264/AVC 类似环内去方块滤波 (De-Blocking Filter,DBF) 外，还增加了新的样值自适应补偿

(Sample Adaptive Offset,SAO) 环内滤波工具 , 形成了两个按顺序进行的处理——去方块滤波和 SAO 滤波。

为降低 DBF 的复杂度 , 利于简化硬件设计和并行处理 , HEVC 仅对排列在 8×8 取样栅格边缘的亮度和色度像素进行滤波 , 仅定义了 3 个边界强度等级 (0、1 和 2) , 只滤波边界附近的像素 , 省略了对非边界处像素的处理。

样值自适应补偿 (SAO) 对 DBF 后要重建的图像逐像素进行滤波。它先按照像素的灰度值或边缘的性质 , 将像素分为不同的类型 , 然后按照不同的类型为每个像素值加上一个简单的补偿值 , 以达到减少失真的目的。

本章参考文献

[1]Video codec for audiovisual services at p×64 kbit/s[S].ITU-T Recommendation H.261.v2.Mar.1993, Geneva.

[2]Video coding for low bit rate communication[S].ITU-T Recommendation H.263++.1998, Geneva.

[3]Coding of moving pictures and associated audio for digital storage media at up to 1.5Mb/s[S].MPEG-1 (ISO/IEC,11172) .1992.

[4]Generic coding of moving pictures and associated audio[S].MPEG-2 (ISO/IEC,13818-2) .1993.

[5]Information technology—coding of audio-visual objects-Part 2: Visual[S].MPEG-4 (ISO/IEC,14496-2) .2001.

[6]Advanced Video Coding for generic audiovisual services[S].ITU-T Rec.H.264 and ISO/IEC 14496-10 (MPEG4-AVC) .v4.July 2005.

[7]High Efficiency Video Coding[S].ITU-T Rec.H.265 and ISO/IEC 23008-2,ITU-T and ISO/IEC JCT-VC,Mach 2013 (v.1) ,Oct.2014 (v.2) ,Apr.2015 (v.3) .

[8]Information technology—Advanced coding of audio and video—Part 2: video[S].China National Standard: GB/T 200090.2-2006,2006.

[9]朱秀昌，李欣，陈杰.新一代视频编码标准—HEVC[J].南京邮电大学学报，2013,33 (3) :1-11.

[10]朱秀昌，刘峰，胡栋.视频编码与传输新技术[M].北京：电子工业出版社，2014.

[11]Iain E.G.Richardson.H.264 and MPEG-4 video compression[M].Hohn Wiley & Sons,Ltd,England,2004.

[12]Thomas Wiegand,Gary J Sullivan.Overview of the H.264/AVC video coding standard[J].IEEE Trans.on Circuits and Systems for Video Technology,2003,13 (7) : 560-567.

[13]Gary J.Sullivan,Jens-Rainer Ohm,Woo-Jin Han,Thomas Wiegand.Overview of the High Efficiency Video Coding (HEVC) standard[J].IEEE Transaction on Circuits and Systems for Video Technology,Dec.2012,22 (12) : 1649-1668.

[14]L.Hanzo,P.J.Cherriaman,J.Streit.Video compression and communications[M].v.2.0,John Wiley & Sons,Ltd,2007,West Sussex PO198SQ,England.

[15]L.Yu,S.Chen,J.Wang,Overview of AVS-video coding standards[J].Signal Process.Image Commun.2009 (4) : 247-262.

[16]Mathias Wien.High Efficiency Video Coding: coding tools and specification[M].Springer Verlag Berlin Heidelberg,2015.

[17]Vivienne Sze,Madhukar Budagavi,Gary J.Sullivan.High Efficiency Video Coding (HEVC) :algorithms and architectures[M].Springer International Publishing Switzerland 2014.

[18]K.R.Rao,Do Nyeon Kim,Jae Jeong Hwang.Video Coding Standards: AVS China,H.264/MPEG-4 PART 10,HEVC,VP6,DIRAC and VC-1[M].Springer Dordrecht Heidelberg New York London,2014.

[19]J.Ostermann,J.Bormans,P.List,et al,Video coding with H.264/AVC: tools,performance ,and complexity[J].IEEE Circuits and Systems Magazine,2004,4 (1) : 7-28.

第3章 HEVC的编码结构

从某种程度上可以说HEVC是高级的H.264/AVC，不仅在总体结构上是如此，在各个模块方面也大体如此。从视频编码结构的角度衡量，HEVC和H.264/AVC也是基本相似的，或者说是在它基础上发展而成的。明白了H.264/AVC的编码结构，就不难理解HEVC的编码结构。因此，本章首先简要介绍H.264/AVC的编码结构，然后在此基础上比较详细地分析HEVC的编码结构，主要包括HEVC的四叉树图像划分，条和片的划分以及不同档次的设置等几部分。

3.1 H.264/AVC的编码结构

从编码原理看，H.264/AVC采用的是混合编码方式。从编码层次结构看，H.264/AVC采用的是分块式编码结构，将图像划分为若干宏块，主要编码操作针对宏块进行。宏块向下可以划分为多个更小的块或子块。向上可由若干宏块组成一个条，一帧图像则由数量不等的条组成。若干图像帧就形成了一个图像组（Group Of Picture, GOP），视频序列包含一系列的GOP，如图3.1所示。

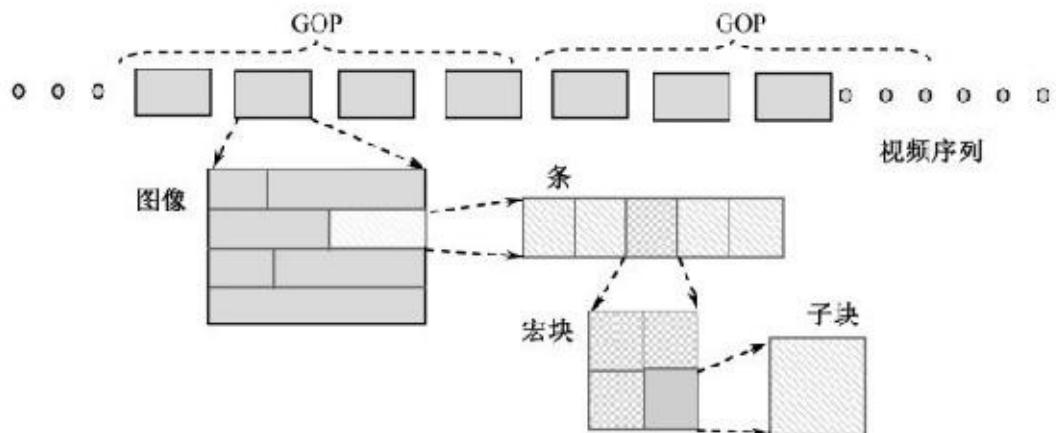


图3.1 H.264/AVC视频分层结构

3.1.1 宏块灵活划分

为适应编码图像中千变万化的运动目标形状，H.264/AVC的帧间预测块尺寸从 16×16 到 4×4 。H.264/AVC的每个宏块（ 16×16 ）可以4种方式分割：1个 16×16 块，或2个 16×8 块，或2个 8×16 块，或4个 8×8 块。而每个 8×8 子宏块还可以4种方式分割：1个 8×8 ，或2个 4×8 块，或2个 8×4 块，或4个 4×4 块。

在帧间预测中，每个分割后的子宏块或块都需有一个独立的运动矢量（Motion Vector,MV）。每个MV必须被编码、传输，分割方式的信息也需编码到压缩比特流中。分割尺寸的选择会直接影响压缩性能。对一个宏块而言，不分割或分割尺寸大，MV 和分割方式信息只需少量的比特表示，但运动补偿后的残差信息往往需要大量比特来表示；对小的尺寸分割，运动补偿后的残差信息所需比特很少，但需要较多的比特表征MV和分割方式的信息比特。

图3.2显示了一帧间预测残差帧（没有进行运动补偿），即像素的预测值和原始值相减的结果。H.264/AVC 编码器为图像的每个部分选择了最佳分割尺寸，使传输信息量最小。为了表明宏块的灵活划分，图3.2中将选择的分割线加到右边的残差帧上。可以看出，在帧间变化小的区域（残差显示灰色），选择 16×16 分割；多运动区域（残差显示黑色或白色），选择更有效的小的尺寸分割。

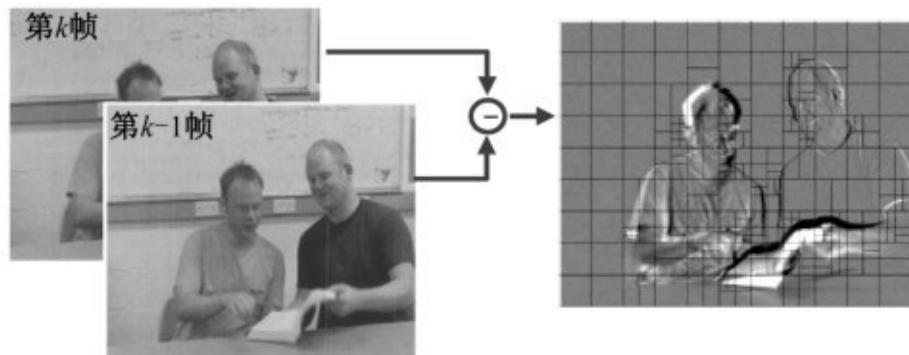


图3.2 帧间预测残差

3.1.2 图像的条划分

1. 条

一幅视频图像可划分成一个或多个条来独立编码，每条包含连续的整数个宏块（MB），即每条至少一个 MB，最多时一个条包含整个图像的宏块。总之，一幅图像中条的个数、每个条中的宏块的个数都可以灵活规定。

编码图像中设条的目的是限制误码的扩散和传播，增加编码传输的鲁棒性。在编码中，是以条为单位编码的，条和条相互间是独立的。某条的预测不能以其他条中的宏块为参考图像，这样某一条中的预测误差才不会传播到其他条中去。

编码条共有5种不同类型，常见的有3种，即I条、P条和B条，另外2种为切换预测（Switching Prediction,SP）条和切换帧内（Switching Intra,SI）条，主要应用于码流间切换（Switching）、拼接（Splicing）、随机接入

(Random Access) 等场合。

2.条组

顾名思义，条组是条的集合，可包含一个或多个条，也是一幅编码图像中若干MB的集合。在一个条组中，每个条的MB按光栅扫描次序被编码，如果整幅图像作为一个条组，则该图像中所有的MB均按光栅扫描次序编码。除非使用任意条次序（Arbitrary Slice Order,ASO）方式。

还有一种条组，叫灵活宏块顺序（Flexible Macroblock Order,FMO），它可用灵活的方法，把编码MB序列映射到解码图像中，MB的分配用MB到条组之间的映射来确定，它表示每一个MB属于哪个条组。这种灵活宏块顺序的安排，可以增加压缩视频流网络传输的抗干扰能力，尤其是抗突发干扰。

3.1.3 档次和水平

为达到设备之间的互连互通目的，H.264/AVC 标准规定了一系列的档次（Profiles）和水平（Levels）（实际上就是规定了一系列的兼容指标）。不同视频压缩产品只要共同支持某一档次的某一水平，它们就能够正确解码对方的数据流。设计了这些指标，为同一标准中具有相似功能需求的不同应用之间提供了互操作性的保证。档次定义了一套编码工具或算法 用于产生兼容的比特流，而同一水平中的不同档次则对比特流等关键参数加以限制。

对于H.264/AVC而言，档次与水平是同等重要的参数。不同档次提供不同算法的要求和限制，编码器可以只支持某档次内的部分特性，而相同档次解码器，则必须能够解码该档次支持的所有特性。

1.档次（profile）

H.264/AVC定义了7种档次、包括基本档次、扩展档次、主档次、高档次、高10档次、高4:2:2档次和高4:4:4档次，每种档次支持一组特定的应用和编码功能。

(1) 基本档次 (Baseline Profile,BP)。支持可视电话、视频会议、无线通信等实时视频通信应用。基本档次的解码器支持以下特性：I条和P条类型；1/4像素精度运动估计；三级运动分块，最小块尺寸为 4×4 ；逐行扫描、去方块滤波；4:2:0色度采样格式；基于上下文的自适应变长编码(CAVLC)的熵编码等。

(2) 主档次 (Main Profile,MP)。主要应用于数字视频存储和数字广播电视领域。主档次的解码器除支持基本档次中所有特性外，还支持以下特性：B条，隔行扫描，加权预测，基于上下文的自适应二进制算术编码(CABAC)。

(3) 扩展档次 (Extended Profile,EP)。主要应用于流媒体中。扩展档次除支持基本档次中的所有特性外，还支持以下特性：加权预测、B条、SP和SI条类型、数据分层条、场编码、帧场自适应编码等。

(4) 高档次 (High Profile,HP)。主要用于分布式数字存储和数字广播电视领域，尤其是高清电视应用(如HDTV和蓝光碟)。高档次在主档次基础上新增：8 \times 8帧内预测(Intra Prediction)、自定义量化(Custom Quant)，无损视频编码(Lossless Video Coding)、CABAC编码。

(5) 高10档次 (High 10 Profile,H10P)。主要用于超高清消费产品和某些专业领域的视频编码。高10档次是建立在高档次基础之上的，解码图像的比特深度可为10。

(6) 高4:2:2档次 (High 4:2:2 Profile,H4:2:2P)。主要针对的是使用隔行扫描视频的专业应用。该档次是建立在高10档次基础之上的，支持4:2:2色度块采样格式，解码图像的比特深度可为10。

(7) 高4:4:4预测档次 (High 4:4:4 Predictive Profile,H4:4:4PP)：该档次是建立在高4:2:2档次基础之上的，支持4:4:4色度块采样格式，解码图像的比特深度可达14。此外还支持高效无损的区域编码，且每幅彩色图像都可作为三个独立的颜色平面来编码。

H.264/AVC最常用的是三个档次，即基本档次、主档次和扩展档次。它们各自包含的基本功能和相互关系如图3.3所示。H.264/AVC这三个档次

所具有的功能之间是存在交集的，如扩展档次包括了基本档次的所有功能，而不能包括主要档次的所有功能。

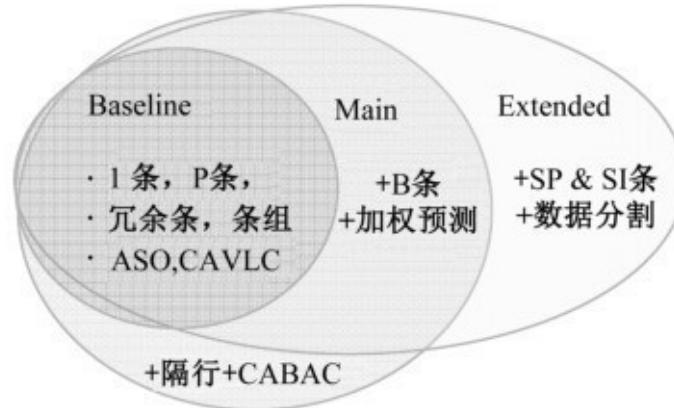


图3.3 H.264/AVC的3个常用档次

2.水平 (Level)

H.264/AVC 除定义了档次 (Profile) 外，对一个指定的档次，又分为不同的水平。水平的具体指标主要和编码图像的尺寸、目标码率等有关，应用中一般都是根据计算机或编码器的运算能力和内存容量来选择特定的编码水平。H.264/AVC定义了从水平1.0至水平5.2，共17种不同的水平。每

种水平包含8项主要指标，如每秒最大处理的宏块数，每帧最大宏块数，解码缓存的最大容量、最高码率等，见表3.1。

表3.1 H.264/AVC不同水平的指标

| 水平 | 最大 MB 处理速率 (宏块数/秒) | 最大帧尺寸 (宏块数/ 帧) | 最大解码缓存容量 (BM) | 最大视频码率 (1000pbs 或 1200pbs) | 最大 CPB 尺寸 (1000pbs 或 1200pbs) | MV 垂直分量范围 (亮度/帧样点) | 最小压缩率 | 每 2 个相邻 MB 的最大 MV 数目 |
|-----|-----------------------|----------------------|------------------|----------------------------------|-------------------------------------|-----------------------|-------|----------------------|
| 1.0 | 1485 | 99 | 396 | 64 | 175 | [-64,+63.75] | 2 | — |
| 1b | 1485 | 99 | 396 | 128 | 350 | [-64,+63.75] | 2 | — |
| 1.1 | 3000 | 396 | 900 | 192 | 500 | [-128,+127.75] | 2 | — |
| 1.2 | 6000 | 396 | 2376 | 384 | 1000 | [-128,+127.75] | 2 | — |
| 1.3 | 11880 | 396 | 2376 | 768 | 2000 | [-128,+127.75] | 2 | — |
| 2.0 | 11880 | 396 | 2376 | 2000 | 2000 | [-128,+127.75] | 2 | — |
| 2.1 | 19300 | 792 | 4752 | 4000 | 4000 | [-256,+255.75] | 2 | — |
| 2.2 | 20250 | 1620 | 8100 | 4000 | 4000 | [-256,+255.75] | 2 | — |

续表

| 水平 | 最大 MB 处理速率 (宏块数/秒) | 最大帧尺寸 (宏块数/ 帧) | 最大解码缓存容量 (BM) | 最大视频码率 (1000pbs 或 1200pbs) | 最大 CPB 尺寸 (1000pbs 或 1200pbs) | MV 垂直分量范围 (亮度/帧样点) | 最小压缩率 | 每 2 个相邻 MB 的最大 MV 数目 |
|-----|-----------------------|----------------------|------------------|----------------------------------|-------------------------------------|-----------------------|-------|----------------------|
| 3.0 | 40500 | 1620 | 8100 | 10000 | 10000 | [-256,+255.75] | 2 | 32 |
| 3.1 | 108000 | 3600 | 18000 | 14000 | 14000 | [-512,+511.75] | 4 | 16 |
| 3.2 | 216000 | 5120 | 29480 | 20000 | 20000 | [-512,+511.75] | 4 | 16 |
| 4.0 | 243760 | 8192 | 32768 | 20000 | 25000 | [-512,+511.75] | 4 | 16 |
| 4.1 | 243760 | 8192 | 32768 | 50000 | 62500 | [-512,+511.75] | 2 | 16 |
| 4.2 | 522240 | 8704 | 34816 | 50000 | 62500 | [-512,+511.75] | 2 | 16 |
| 5.0 | 589842 | 22080 | 110400 | 135000 | 125000 | [-512,+511.75] | 2 | 16 |
| 5.1 | 983040 | 36864 | 184320 | 240000 | 240000 | [-512,+511.75] | 2 | 16 |
| 5.2 | 2073600 | 36864 | 184320 | 240000 | 240000 | [-512,+511.75] | 2 | 16 |

3.2 HEVC的网络适配和编码方式

为了更好地理解HEVC的高效视频编码方式，人们常常将它分为两部分（层），一部分是有关具体视频编码的技术和数据，从通信分层的角度属于低层信息；另一部分是有关网络传输和视频应用方面，一般和具体的视频细节无关，属于高层信息。基于这样的考虑，本节主要介绍如下两方面内容。

（1）为适应网络传输，增加对网络的“友好”性，HEVC采取了分层处理的方法，分为视频编码层（VCL）和网络提取层（NAL）。

（2）为适应不同应用场景的需要，提供了全帧内、低延时和随机接入三种不同的编码方式。

3.2.1 视频编码层和网络提取层

与 H.264/AVC相同，HEVC在编码结构上也分为视频编码层（Video Coding Layer,VCL）和网络提取层（Network Abstraction Layer,NAL），如图3.4所示。前者独立于网络，主要包括核心的视频压缩引擎和图像分块的语法定义。后者主要是定义数据的封装格式，把 VCL 产生的比特字符串适配到各种各样的网络环境中。

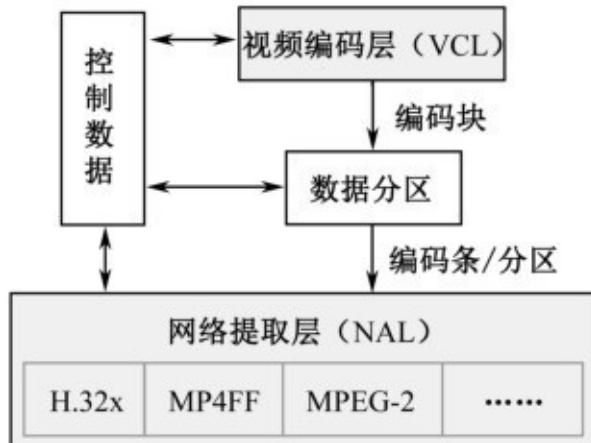


图3.4 HEVC的编码结构框图

原始视频经过VCL层，被编码成视频数据，然后经过NAL层，封装成一个个NAL包以适应不同网络的视频传输。HEVC码流在应用过程中与H.264/AVC码流的区别就在于 NAL层，具体情况见第10章的HEVC高层语法。

1.视频编码层 (VCL)

HEVC视频编码层沿用以往视频标准的混合编码方式，其编码器的框图如图3.5所示，它的输入是原始视频，输出为符合HEVC标准的比特流，简要编码过程如下。

(1) 将每一帧图像划分为不同大小的图像块单元，并将相应的块划分信息加入到码流中，传到解码器。

(2) 对每个单元进行帧内或帧间预测，原始像素值和预测值相减形成该单元的残差；在帧间预测时需要进行运动估计和运动补偿，对需要用到的重建图像事先要进行去方块滤波和自适应样值补偿 (ASO) 滤波。

(3) 对每个单元的残差进行整数变换（近似离散余弦变换或正弦变换），对形成变换系数进行量化和扫描。

(4) 对量化后的变换系数、预测信息、模式信息、运动信息和头信息等进行熵编码，形成压缩的视频码流（语法元素）输出。

解码过程虽然没有在图3.5中给出，但基本上和编码过程相反，主要包括：

(1) 对压缩码流进行解析，得到各类编码信息（语法元素）；

(2) 对每类信息进行熵解码，得到每一帧图像残差系数的量化值及其他参数；

(3) 对残差系数量化值进行反量化和反变换，得到图像的残差值；

(4) 将残差值和预测值相加得到重建图像。

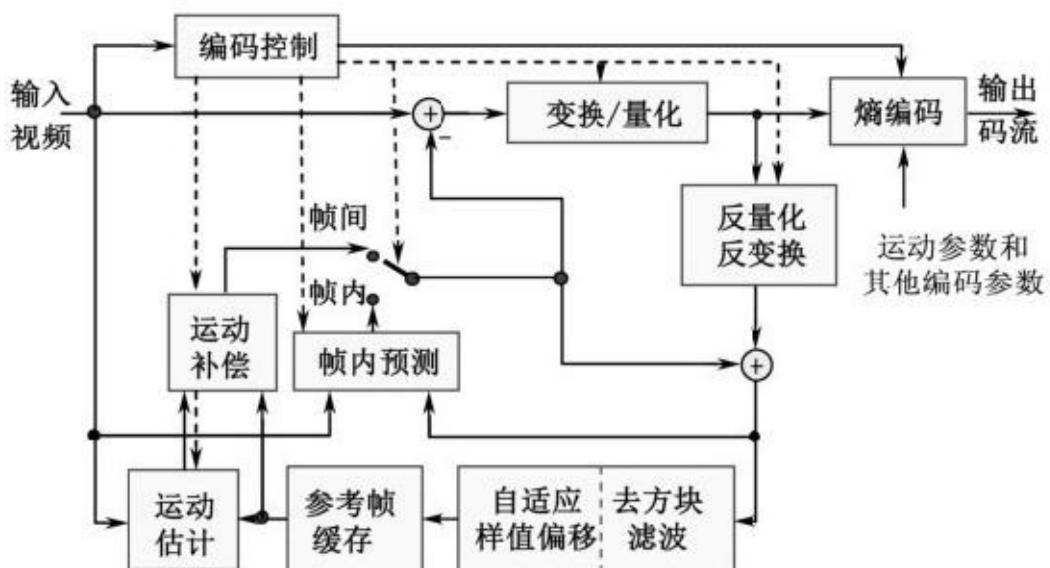


图3.5 HEVC编码框图

2. 网络提取层 (NAL)

HEVC网络提取层 (NAL) 包含了大量的从H.264/AVC的NAL继承来的语法元素。HEVC的VCL层产生的每个语法结构都会被放进网络提取层 (NAL) 单元的数据包中。NAL单元的头部容量从H.264/AVC的单字节扩充为两个字节，增加的内容主要用于标识它所装载数据的类型。NAL单元根据是否装载有编码视频或其他相关数据而分为VCL NAL单元和non-VCL NAL单元两类。HEVC 标准中，还包括了为解码器初始化和随机接入的目的而识别图像分类的若干VCL NAL单元类型。

NAL单元负责把视频编码层 (VCL) 表示视频内容的数据映射到不同传输层上，如RTP/IP、ISO MP4、H.222.0/MPEG-2系统等，并提供丢包恢复处理机制。

详细的NAL单元类型列表以及对NAL设计的一般概念，如NAL单元、参数集、存取单元、码流格式、包格式等见第10章或HEVC文档。

3.2.2 三种编码方式

为应对不同应用场合，HEVC设立了GOP的三种编码结构 (Coding Structures)，即全帧内 (AI) 编码、低延时 (LD) 编码和随机接入 (RA) 编码。

(1) 全帧内编码

在全帧内编码结构中，每一帧图像都是按帧内方式进行空间域预测编码，不使用时间参考帧。

(2) 低时延编码

在低时延编码结构中，只有第一帧图像按照帧内方式进行编码，并成为及时解码刷新 (Instantaneous Decoding Refresh, IDR) 帧，随后的各帧都作为普通P和B帧 (Generalized P and B picture, GPB) 进行编码。主要是为交互式实时通信设计的。

(3) 随机接入编码

在随机接入编码结构中，主要是等级B帧结构，周期性地（大约每隔1秒）插入一纯随机接入（CRA）帧。这些HEVC新定义的具有一定的密度CRA帧，成为编码视频流中的随机接入点（Random Access Point,RAP）。对随机接入点（帧）的解码可以独立进行，不需要参考比特流中前面已经解码的图像帧。HEVC 设置随机接入方式，有力地支持了信道转换、搜索以及交互流媒体服务等应用。

在HEVC视频编码中，视频序列图像组（GOP）是最大的编码单位，它的第一帧图像（或随机接入点的第一帧图像）只能使用帧内预测方式编码。对GOP中剩余的图像帧，或者随机接入点之间的帧，允许采用帧间预测方式编码。

3.3 HEVC的四叉树划分

在图像分块方面，HEVC的一个重要革新之处就是为预测和变换编码目标而对图像进行的基于四叉树的划分。HEVC对某一视频序列编码的层次结构如图3.6所示。

HEVC将一个视频序列分为相继的若干图像组（GOP），每一组由该序列中连续的多帧图像组成。帧是四叉树划分的基本单位，每一帧图像经过四叉树划分，形成覆盖全帧的多个同样尺寸的方形编码树块（CTB）。CTB还可以划分为更小的编码块（CB）。CB是实施视频编码算法的基本单位，它还可以划分为预测块（PB）和变换块（TB）。

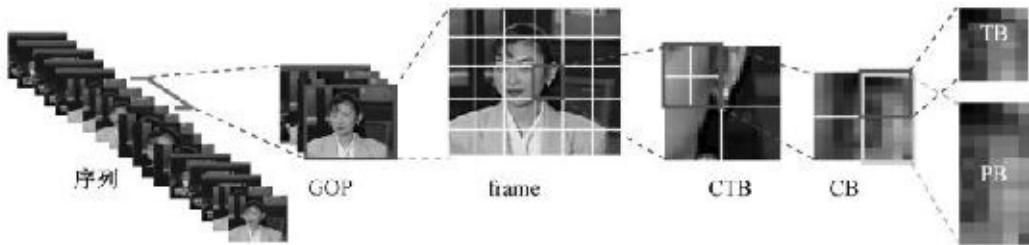


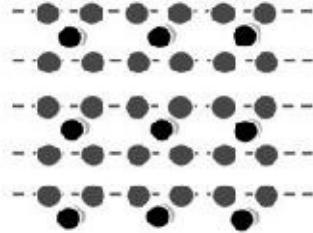
图3.6 HEVC编码顺序与结构

3.3.1 图像的取样格式

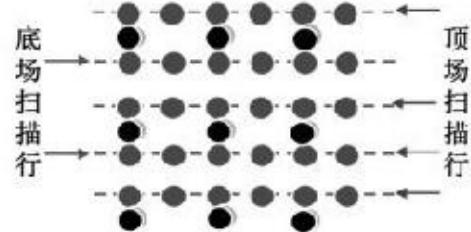
1. 彩色取样格式

HEVC原则上支持ITU-R BT.601建议规定的彩色取样的4:4:4、4:2:2格式和4:2:0彩色取样格式。如第1章所述，4:4:4取样结构中两个色差信号（ C_b 和 C_r ）的取样频率和亮度信号（Y）取样频率一致，几何位置重合，这样亮度信号与两个色差信号的取样点数之比为4:4:4。在4:2:2取样结构中，两个色差信号的取样频率在水平方向均为亮度信号取样频率的一半，这样亮度信号与两个色差信号的取样点数之比为4:2:2。在4:2:0取样结构中，两个色差信号的取样频率也为亮度信号取样频率的一半，但每隔一行才传送色差信号，亮度信号与两个色差信号的取样点数之比为4:1:1。

三种方式中，比较常用的为4:2:0的格式，也是HEVC目前支持的主要格式。在以往的视频标准中，对4:2:0格式的数字视频的安排是有所不同的。对于逐行视频，MPEG-1的取样点位置如图3.7（a）所示，彩色样点有亮度半像素间隔的偏移。对于隔行视频，MPEG-2的顶场和底场取样点位置如图3.7（b）所示。从这两幅图中可以看出，亮度和色度的取样点数的比例都一样，亮度样点的位置也都一样，不同的是色度信号的位置。在隔行视频的顶场中，色度样点的垂直位置相对于亮度采样格点下移了1/4个亮度样点的垂直间隔。在底场中，色度样点的垂直位置相对于亮度采样格点上移了1/4个亮度样点的垂直间隔。将顶场和底场合成一帧，其彩色样点的位置如图3.7（b）所示。HEVC的4:2:0格式也采用这种彩色样点安排方式，HEVC的附录E提供了对不同色度位置的安排规范。



(a) MPEG-1 逐行视频



(b) MPEG-2 隔行视频

图3.7 两种4:2:0彩色取样格式

2.比特深度

像素值的最大比特数表示图像的灰度分辨率，又称为比特深度或精度，其值通常为8bit，即灰度误差不会超过1/256。这对于人眼的观察已经足够了，但对于某些特殊应用场合和编码处理而言，有时显得精度不够，往往会带来的计算误差和累积误差。为此，HEVC 提供了一种对编码器内外有别的比特深度表示方式，即内部比特深度增加（Internal Bit Depth Increase,IBDI）方式。这种方式允许在编码器内采用高于8bit的像素深度，如用10bit进行编码运算，而编码器外部，即编码器的输出图像仍然是8bit深度。

3.隔行处理

随着数字视频技术的迅速发展，视频的隔行扫描方式日渐势微，因此不同于H.264/AVC,HEVC 没有提供专用于隔行视频的工具，而是将隔行视频的一帧看作两个独立的场，对各个场数据分别进行编码，简化了编码器的实现。

3.3.2 编码树单元和编码单元划分

1. 编码树块 (CTB) 和编码树单元 (CTU)

HEVC采用的是基于块编码方式，块的尺寸是可以通过划分自适应改变其大小的。类似于H.264/AVC中的宏块 (MB), HEVC将一帧编码图像划分为统一大小、紧邻但不重叠的若干 $2N \times 2N$ 样点的方形编码树块 (CTB)，其尺寸或所包含的像素数可以是 16×16 、 32×32 或 64×64 ，可由编码器的序列参数集 (Sequence Parameter Set, SPS) 中的语法元素根据具体情况选择，一般说来比较大的尺寸可以获得较高的压缩。

同一位置的亮度 CTB 和两块色度 CTB，再加上相应的语法元素以及所包含的编码单元 (CU) 形成一编码树单元 (CTU)。由于第1版HEVC只支持4:2:0格式的彩色亚采样，所以色度块CTB的像素数为同等亮度块的 $1/4$ ，即水平和垂直方向都是亮度块的一半。

CTB是方形的，在语法表示中其尺寸用边长表示：CtbSizeY，单位是亮度像素。相应的色度CTB的宽度和高度分别为CtbWidthC和CtbHeightC。

2. 编码块 (CB) 和编码单元 (CU)

CTU可以按照四叉树结构分解为若干部方编码单元 (CU)，同一层次的CU必须是同一尺寸的4个方块，最多可有4层分解，即 64×64 , 32×32 , 16×16 和 8×8 。如果不分解，则这个CTU仅包含一个CU，此时亮度CTB的尺寸就是亮度CB的最大尺寸。

每个CU包含一个亮度编码块 (CB)、两个色度CB以及相应的语法元素，如预测模式 (帧内或帧间)、PU划分、RQT语法、从属的PU和TU信息等。CU是决定进行帧内预测还是帧间预测的单元，也就是说，整个CU只能是一种预测模式，不是帧内就是帧间。

另外，由于图像中一定大小的CTU是按照光栅扫描的方式排列的，因此在图像的右方或下方边缘处，某些CTU覆盖的区域可能会部分地处于图像边界之外，这意味着CTU必须继续进行四叉树分裂，减小CB尺寸，直至最小允许的亮度CB尺寸，以使得所有的CB可完全安放到图像中。

3.CTB至CB的4叉树划分

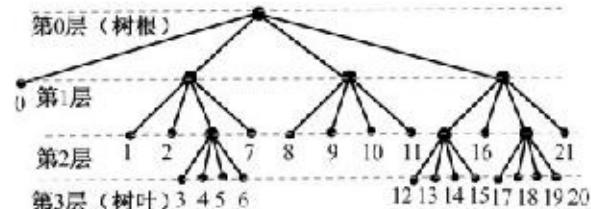
CTB和CB之间的关系可表示为一棵四叉树，CTB为树的根节点。CTB还可以划分为更小的方块，相当于由根节点向下层分叉的4个分枝，每个分枝又可以继续分叉为下一层的4个分枝，直至不再分的编码块（CB），相当于编码树的最底层——树叶。亮度CB的尺寸从 8×8 到最大 64×64 样点。此外，由编码器选择、并标注在比特流中的实际最大和最小的CB尺寸可施加更紧的限制。

图3.8（a）给出了一个CTB到CB的划分示例。CTB的尺寸为 64×64 ，一共有4层划分，形成22个CB，编码处理的顺序服从划分层次（深度）优先的光栅扫描规则，编号从0到21。其中0号是最大的CB, 32×32 ，此外还有 16×16 的CB，如1、2号等，最小的CB为 8×8 ，如3、4号等。图3.8（b）是和图3.8（a）对应的四叉树的逻辑结构图。

按照所给定的编码树将CTB递归划分为CB后，每个CB就成了编码树的一个树叶，还可以进一步划分为预测块（PB）和变换块（TB）。

| | | 1 | | 2 |
|----|----|----|----|----|
| 0 | | 3 | 4 | 7 |
| | | 5 | 6 | |
| 8 | 9 | 12 | 13 | 16 |
| | | 14 | 15 | |
| 10 | 11 | 17 | 18 | 21 |
| | | 19 | 20 | |

(a) CTB 的空间划分



(b) 相应的四叉树表示

图3.8 CTB到CB四叉树的划分实例

4.CB的划分

编码块 (CB) 还可以进一步划分为一个或多个预测单块 (PB) 和变换块 (TB) , 同一个CB可以同时进行两种划分 , 一种是预测块的划分 , 另一种是变换块的划分。这两种划分都在这个 CB 的几何位置中进行 , 但两者基本上是互不相干 , 而且划分的方法也不同 , 尽管划分的结果还是在同一个CB中。

CB到PB最多允许一层划分 , 可以是1分2 , 也可以是1分4 , 还允许不对称的1分2。

CB到TB的划分和CTB到CB的划分类似 , 也是一种四叉树划分 , CB是“树根” , 最多可进行3层的1分4,TB是“树叶”。

3.3.3 预测单元划分

1.CU中的PU划分

预测单元PU是HEVC进行预测运算的基本单元，只能定义在不再划分的最低层的CU中，包括帧内预测和帧间预测两类。CU决定了本单元包含的所有PU的预测方式和划分方式。

(1) PU的预测方式

CU中所有的PU为同一种预测方式，即在帧内预测和帧间预测中二选一。

(2) PU的划分方式

一个编码单元CU可以划分为一个或多个预测单元PU,CU到PU仅允许一层划分，最小的PU为 4×4 。划分可以是对称的，也可以是不对称的。如图3.9所示，一个 $2N\times 2N$ (N 可以是4、8、16、32) 的CU可划分为8种包含PU方式（图3.9中第2排和第3排）。帧间预测时可以在这8种方式中任意选择，帧间预测的跳过模式中只允许选择 $2N\times 2N$ 这种方式，帧内预测只允许选择 $2N\times 2N$ 或 $N\times N$ 方式。

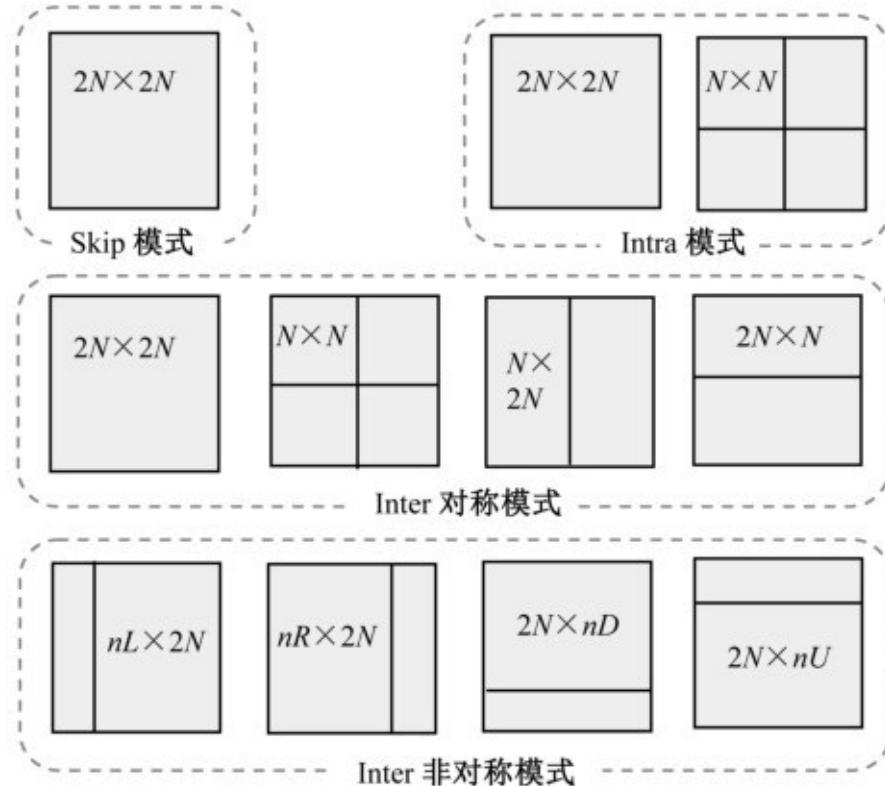


图3.9 $2N \times 2N$ 的CU的PU划分方式

2. 帧内预测块

当CU为帧内预测方式时，PU尺寸应和CU的尺寸一样。但对于最小尺寸的CU（ 8×8 ）例外，它可以继续按四叉树方式划分为4个 4×4 的PU，每个都可有自己的帧内预测模式。可见，帧内预测块的最小尺寸可选择为 4×4 。

在4:2:0色度取样模式下，当对 4×4 亮度块进行帧内预测操作时，色度帧内预测也使用 4×4 的块（不允许出现 2×2 的色度块），它覆盖了和4个 4×4

亮度块相同的图像区域。也就是说，当编码单元的尺寸为 8×8 ，且亮度预测单元的模式为 4×4 时，色度PB的尺寸为 4×4 ，不再进行分解。

帧内预测中，一个亮度 PU 有一个帧内亮度预测模式，而相应的两个色度 PU 则共享一个帧内色度预测模式。在每个PU中，同一个帧内预测模式被用于预测该PU内的每个样点。

帧内预测的参考像素来自邻近的已重建的TU样点。

3.帧间预测块

当CU为帧间预测方式时，PU的划分可以在图3.9中下面2排的8种方式中选择，下排的4种划分为非对称划分，中排的4种划分为对称划分。其中，非对称划分为可选模式，可以通过编码器配置开启或关闭。

在8种划分方式中，当亮度CU尺寸为 16×16 或更大时才允许不对称划分，一个CU划分为2个PU，每个PU分别覆盖CU的 $1/4$ 和 $3/4$ 面积。为了限制编码器的复杂度，HEVC禁止使用 4×4 尺寸的帧间预测块，至于 4×8 或 8×4 尺寸的帧间预测块只允许在单向预测时使用。

3.3.4 变换单元划分

变换单元（ TU ）是进行变换和量化操作的基本单元，和PU划分类似，它也是在CU的基础上划分的，但它基本不受所在CU的预测单元（ PU ）划分的限制。方形CU到TU的划分也是一种四叉树划分，又称“变换树”或残差四叉树（ Residual Quad Tree,RQT ），CU是这棵树的根，TU是这棵树的叶，也是方形。

1.CU中亮度的TU划分

RQT可以多层次地对CU进行4叉分裂，每向下一层则划分为4个小的正方形TU。与编码单元CU四叉树类似，残差四叉树通过深度优先、光栅扫描的顺序进行遍历。图3.10中显示一个深度为3的RQT例子。变换单元的最大尺寸以及残差四叉树的层级可以根据不同的应用进行相应的配置，对实时性或复杂度要求较低的应用可以通过增加残差四叉树的层级来提高编码

效率。

TU的最小尺寸为 4×4 ，最大尺寸为 64×64 ，其中 64×64 尺寸的TU实际上隐含着必须进一步划分为4个 32×32 ，因为最大的DCT变换运算的尺寸为 32×32 。最小和最大可用的变换块（TB）尺寸，以及最大可用变换树深度这3个参数定义在SPS中。变换块尺寸用2的整数次方表示，从2到5，即 2^2 到 2^5 分别表示从 4×4 到 32×32 。变换深度从0到3。例如，变换树深度设置为0，则此变换树降格为变换块（TB）。在这种情况下，不需要在TU中包含变换树标识。

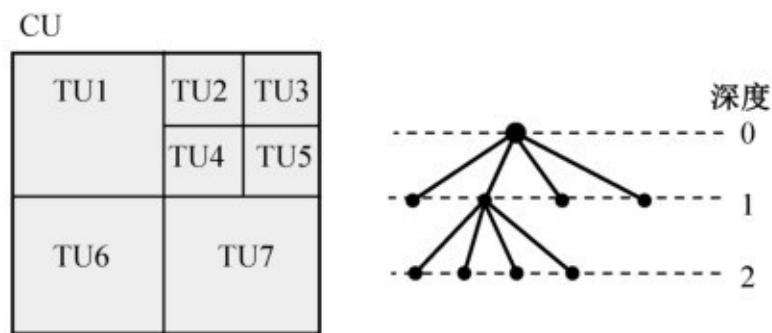


图3.10 残差四叉树结构一例

2.CU中色度的TU划分

对于常用的4:2:0彩色格式，色度TU尺寸在水平和垂直方向都是亮度TU尺寸的一半，除非亮度TU的尺寸是 4×4 。这种情况下最小的色度TU尺寸不能为 2×2 ，也就是说， 4×4 的色度块不能再分解，必须保持为 4×4 ，它覆盖的区域为4个 4×4 亮度TB。此时，亮度RQT和色度的RQT实际上是有所不同的。

3.CU的划分和PU无关

CU的变换树的构成和PU的划分基本上是没有关系的。对于帧间预测CU，所选的TU尺寸可以大于所划分的PU子块尺寸，但TU的尺寸不能大于CU尺寸，因为CU是变换树结构的根。因此，编码器可以选择使用穿越PU界线的残差变换。例如，对多个包含PU界线的PU残差联合起来进行变换、量化和熵编码，有助于节省编码比特。

在一个CU中，TU划分和PU之间关系的3个示例可见图3.11。从图中可以看出，图3.11（a）表示PU没有划分，和CU等同，但TU进行了3层划分。图3.11（b）表示PU对CU进行了1层划分，成为上下两个PU，在上下两个PU中，各自都进行了TU划分。但上述两种情况的TU划分都没有跨越PU边界。图3.11（c）表示PU对CU进行了1层划分，成为不对称的左右两个PU，其中右下角的TU跨越了PU的界线。

尽管TU的划分和PU无关，但在帧内预测CU的情况下，最邻近的TU（无论是在CU内还是外）的解码样值可用作帧内预测的参考数据。

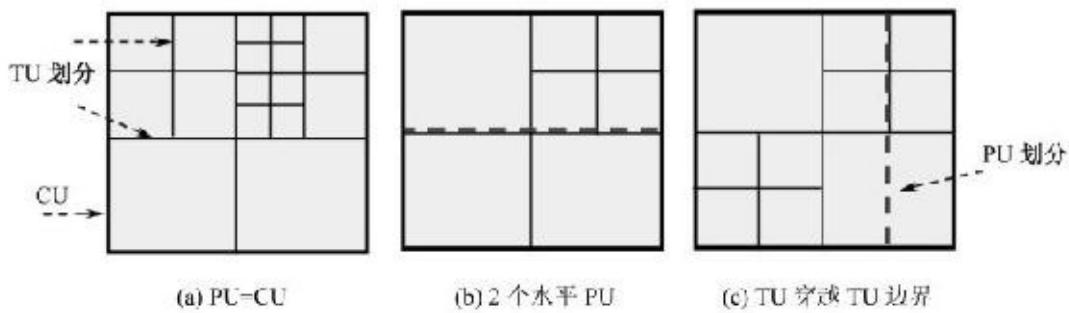


图3.11 CU划分为PU（虚线）和TU（实线）的示例

4. 帧内亮度 4×4 TU的DST

HEVC为 4×4 、 8×8 、 16×16 、 32×32 方形变换块（TB）定义了类似离散余弦变换（DCT）的整数变换。对于 4×4 亮度帧内预测的残差，也可采用另外一种源于离散正弦变换（DST）的整数变换。

3.3.5 CTU划分实例

图3.12是一个 64×64 CTB的四叉树CB、PB和TB划分实例。图3.12中的上半部是将此CTB划分为更小、不同尺寸的24个CB。每个CB还可以进一步的划分，如14号 32×32 的CB被划分，一次被划分为2个预测块PB（左下），一次被划分为不同尺寸的残差变换块TB（右下）。这种划分仍然按照四叉树结构规则进行。

在图3.12的每一个划分步骤中，解码器对某个特定的起始块都定义了一个唯一的处理顺序。通过以深度（划分的层次）在先的方式遍历编码四叉树来处理CB，形成一个递归的“Z”字形扫描顺序，如图3.12的右部所

示。这个CTB划分为25个CB，编号由0开始，逐个递增到24。

“Z”字形扫描方法还用于定义PB和TB的划分顺序。图3.12左上是14号CB（ 32×32 ）的PB划分，只允许一层划分，均匀分为左右两个PB，顺序编号为0,1。图3.12左下是14号CB的TB划分，允许多层四叉树划分，共划分13个TB，顺序编号为0,1,...,12。

图像的亮度和色度分量的块划分是相同的，从而可使用相同的语法来表示这种划分。

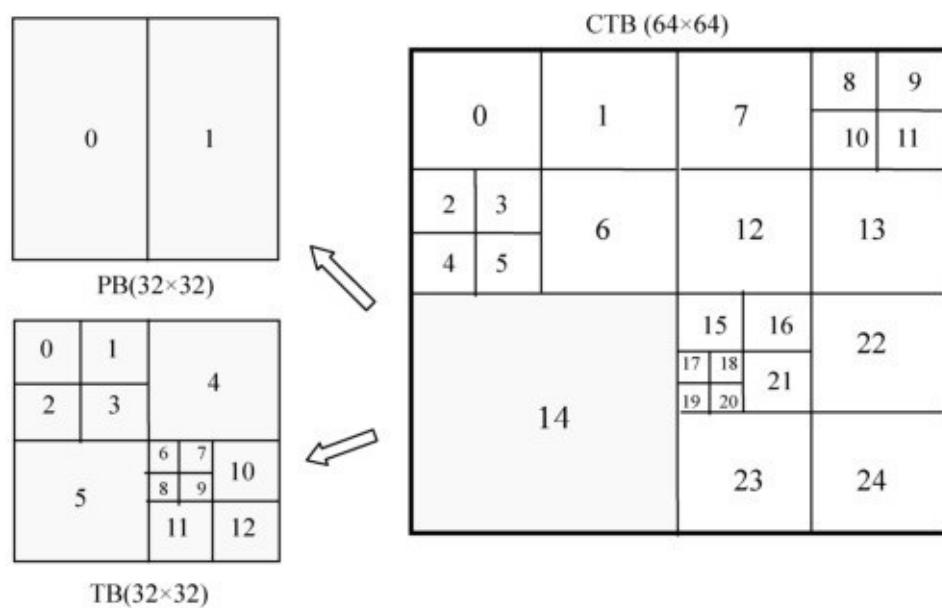


图3.12 64×64 CTB的编码四叉树的划分实例

图3.13为一帧实际图像(局部)的四叉树CB分割的结果,图3.13(a)为原始图像,图3.13(b)为分块划分以后的图像,共划分为 6×6 个CTB,每个CTB的尺寸为 64×64 ; CTB经CB划分后成为大小不等的多个CB,一个明显的趋势就是在图像内容比较简单的地方,CB的尺寸较大,最大为 64×64 ,在图像内容复杂的地方,CB的尺寸较小,最小为 8×8 。

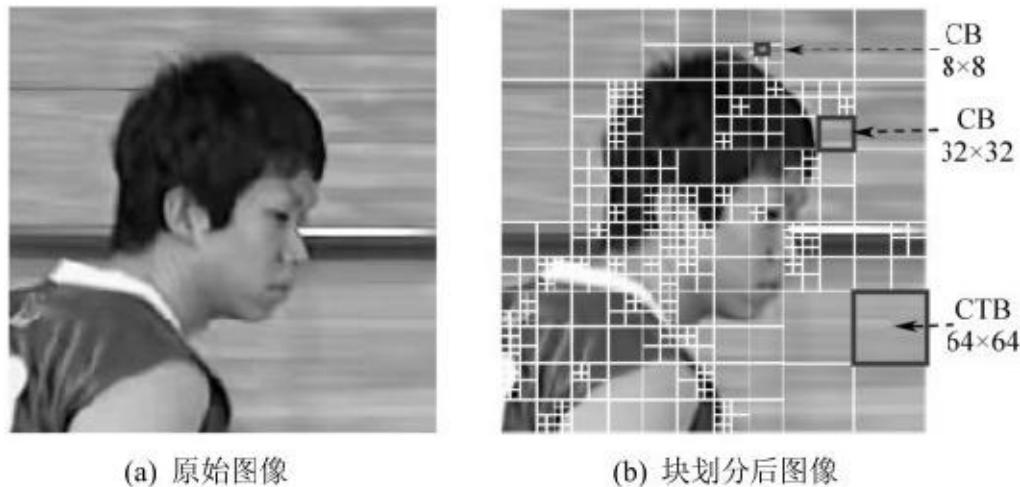


图3.13 图像的CTB、CB划分

3.4 HEVC的条和片划分

一帧图像可以被划分为条 (Slice) 或片 (Tile)，条还可以再划分为分条 (SS)。组成条的一系列分条是由一个独立分条开头，后面跟随的是非独立分条或独立分条 (如果有)。条 (或) 片在HEVC图像结构中处于CTU的上一层，一个条或分条是由整数个相继排列的CTU组成的，类似，一个片也是由整数个相继排列的CTU组成的。

3.4.1 条划分

视频解码一般包含两个基本处理步骤，首先是由熵解码器实现的对经熵编码压缩后的比特流进行解析，获得视频量化后的变换系数和其他必要的编码参数，然后进行解码处理，包括反量化、反变换、运动补偿等，获得解码的重建图像。由于视频压缩编码的本质在于充分降低视频信息的冗余度，因此压缩码流中数据前后之间的依赖性大为增加，因而只要一点差错就可能导致不能处理的误码，并累及后续的比特流。同样的差错，将会引起比非压缩视频大得多的重建图像质量下降。因此在实际应用中，必须考虑如何应对传输差错问题。除纠错编码以外，早先的视频编码标准将图像分为若干独立的编码部分 (如若干条)，使得解码器可独立解析、解码这些部分，以减轻传输差错给解码图像质量带来的损害。

HEVC中也采取了这种条的划分措施，并且有所改进。

1.条的划分方法

条的划分在H.264/AVC中就有，HEVC也允许将图像帧划分为若干条，它是HEVC中独立的编码区域，由一系列的按光栅扫描顺序处理的CTU组成。也就是说，一帧图像可以分解为一个或若干个条，每个条都包

含一组编码树单元（CTU），每个条能够被独立地编码、解码和重建。图3.14显示了一帧图像中3个条的划分情况，条的划分以CTU为界。

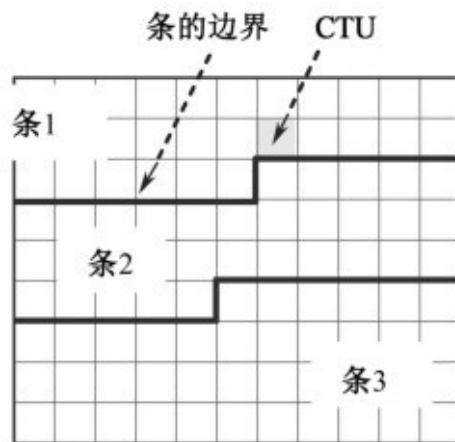


图3.14 图像中的条划分一例

每个条的编码可以使用如下不同的编码类型：

- (1) I条：这类条中的所有的CU只使用帧内预测方式编码。
- (2) P条：P条中除了允许帧内预测编码的CU外，还允许某些CU使用帧间预测方式编码，每个预测块PB最多可有一个运动补偿的预测信号，即单向预测。单向预测的P条仅使用参考图像列表0（List0）。

(3) B条：B条中除允许帧内预测和单向帧间预测编码的CU外，还允许某些CU使用双向帧间预测的方式编码，每个预测块PB最多可有2个运动补偿的预测信号。双向预测的B条可使用参考图像列表0（List0）和列表1（List1）。

2.条的容量考虑

因为条之间的编码是相互独立的，一个传输错误，最多只能影响该条内传输，而不会传到其他的条，有利于数据损失后可以再同步。从克服传输差错的目的出发，希望条的容量越小越好。但条的容量小了以后，图像空间相关性的利用大打折扣，用于条的头信息开销也会大增，导致编码效率下降。因此，必须综合考虑条的容量。

在HEVC的码流中，每个条封装为一个NAL单元传输。一个条所包含的CTU数目常常变化很大，这取决于视频场景的活动性，因此常常用一个最大的比特数来限制条的容量，例如为了在有线网络中打包传输，条的容量要求小于最大网络传输单元（Maximum Transmission Unit,MTU）容量。如果在其他网络，如无线网络、移动网络，对条的容量又必须另加考虑。

3.条的进一步划分

如有必要，条还可以划分为若干个更小的分条（SS）。在划分后的若干分条中，第一个分条为独立分条，可以独立解码，不必依赖于其他分条；其余的分条皆为依赖性分条，它们的解码需借助其他分条的数据。条的这样进一步划分有利于并行处理，实现高速、低时延解码。但这种情况下，对压缩性能可能会产生一些不利影响。SS是HEVC编码压缩数据输出的最小完整单位，每个VCL单元至少要包含一个SS,SS所需的图像层信息可从图像参数集（PPS）中获得。

这样，一幅图像可以由一个或多个条构成。一个条又可以由一个或多个分条（SS）构成，SS之间是非重叠的。这样，图像的每个CTB只包含在一个SS内，全部图像区域都被条覆盖。作为条和SS划分的一个重要特性，每个条以及属于该条的SS可被独立解码，与该帧中其他条无关。这意味着不允许从另外的条来预测本条，包括帧内预测和帧间预测。而且，即使图

像中的其他条已经丢失，图像中这个条也可以被正确解码。但在属于同一条的所有SS之间，相互预测是允许的。

每个SS都包含关于本条中第一个CTB的位置信息，这样可以确定SS所表示的图像中的区域。因此，在有数据丢失时，条和SS结构是一种非常有用的恢复和重同步的工具。还有，一幅图像中的所有的条都参考同一参考图像集，即参考图像集的只需要为图像中的第一个被解码的条构建就行了。对于帧间图像预测，一幅图像中所有的条也统一使用相同的参考图像列表。

3.4.2 片划分

1. 片划分方法



图3.15 图像中片划分一例

在HEVC中新引入了可选的片划分，用水平和垂直的若干条边界将图像帧划分为多个矩形区域，每个区域就是一个片，同时也是一个独立的编码单位。每一个片常包含大体相同的整数个CTU，但并不强求。图3.15显示了一例图像中9个片的划分情况。片划分时并不要求水平或垂直边界均匀分布，可根据并行计算和差错控制的要求灵活掌握。在编码时，图像中的片也是按光栅扫描顺序进行处理，每个片中的CTU也是按光栅扫描顺序进行。按照这一原则，图3.15中给出了一帧图像9个片的99个CTU的处理顺序编号。

HEVC引入片划分的主要目的是增强并行处理的能力，而不主要在于提供差错鲁棒性。片划分提供的是一种较简单的粗粒度（子图像）并行处理机制，它的使用不需要提供复杂的线程同步支持。

在HEVC中，允许条和片在同一图像帧中同时使用，既可以一个条中包含若干片，共享头信息，也可以一个片中包含若干条。

2. 片划分的优点

（1）更高的空间相关性

在这种灵活的矩形片划分中，一个片可以比一个包含相同数目CTB的条在空间上更加紧凑。这样带来的好处就是它比条具有更高的像素间的空间相关性。

例如图3.15中，在第一个片（左上）中，按光栅扫描的顺序标注CTB的编号为1~9。扫描完第一个片，接着就是第二个片（按片光栅扫描顺序），CTB的编号为10~21。需特别注意的是，图3.15中在第二个片中的第10号CTB紧挨着第一个片中的第9号CTB，一般情况下比相邻条边界两边的相邻CTB之间的距离要近得多。

（2）有利于并行处理

片的划分除改变CTB扫描过程外，片的边界还表示对编码依赖的切断。特别是穿越片边界的熵编码和重建的依赖是不允许的，在运动矢量预测、帧内预测和上下文选择中也是如此。环路滤波是唯一的例外，它允许穿越边界，但也可被比特流中的某个标志所禁止。这样，分离的片可以在

不同的处理器中被编码，只需少量的处理器之间的通信，非常有利于多个片同时并行处理。

为实现并行解码，片的位置必须在比特流中标注。在HEVC比特流中，紧随条头信息的第一个片的位置解码器是已知的，因此比特流中除第一个片以外其他所有片的偏移都需要在条的头信息中准确传送。

(3) 减少行缓存容量

片扫描方式具有减少用于运动估计的行缓存容量的优点。在CTB帧间预测时，需要片上存储(On-chip Memory)重建的像素数据(来自先前编码帧)，它是运动补偿的候选。这一数据加载到片上存储器并保留到不再需要为止。

如图3.16所示，在没有片划分时，一幅图像的光栅扫描形成的存储样本数据等于 $P_w \cdot (2 \cdot V_y + B_y)$ ，这里 P_w 是图像的宽度， V_y 是运动矢量最大的垂直尺寸(以像素为单位)， B_y 是CTB的高度(以像素为单位)。

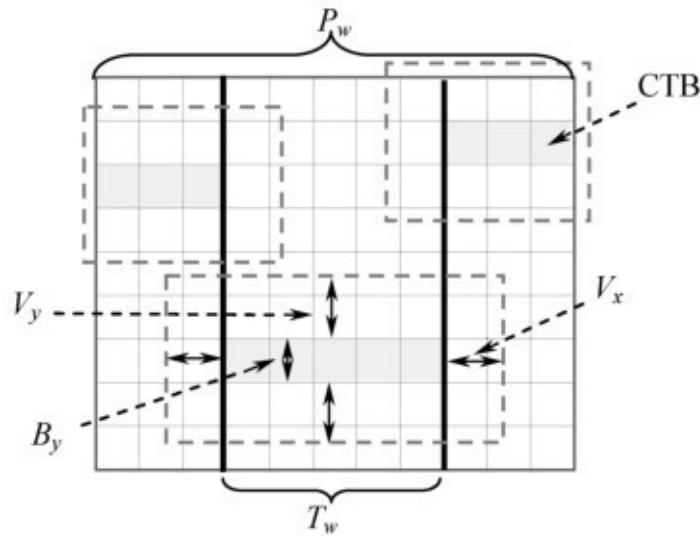


图3.16 片划分减少运动估计的行存储容量

在有片划分时，修改的扫描模式形成的取样数据存储量需求大约为 $(T_w + 2 \cdot V_x) \cdot (2 \cdot V_y + B_y)$ ，这里 T_w 是一个片的宽度， V_x 是运动矢量的最大水平分量的尺寸。在片划分中，通常 P_w 明显大于 $(T_w + 2 \cdot V_x)$ ，样本数据存储量就可大大减少。

可见，对于矩形片划分，运动补偿的行缓存的容量需求可明显降低，即使只有2个片，减少量也接近50%。因为减少片的存储需求，还可以适当扩大编码器运动矢量的垂直搜索范围，这将有利于编码效率的进一步提高。同样的道理可以用于环路滤波器，也可以降低环路滤波对存储量的需求。

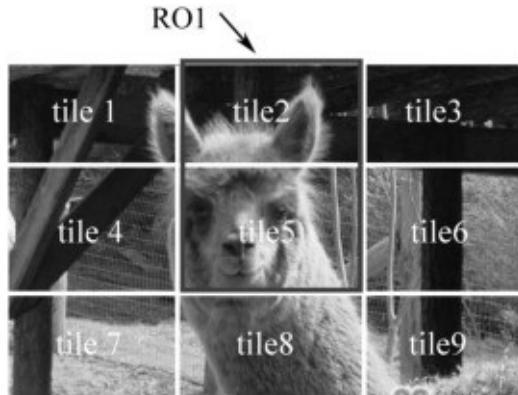


图3.17 图像中2个片组成的ROI

(4) 减少头信息开销

片在一帧图像内是一些可独立编解码的区域，由于片是比较规则的矩形划分，为了提高编码效率而不需要另设头信息。因此，和条对比，片具有比较低的头信息开销。

(5) 支持感兴趣区域编码

片的设置，除上述优点外，在某些特殊的应用场合中有它的独到之处。例如，片所提供的矩形像素数据划分有利于感兴趣区间（Region Of Interesting, ROI）编码，可以很简洁地标注出感兴趣区间。图3.17显示了一个实例，一幅图像的9个片中有2个片表示的是图像中的ROI，即按光栅扫描顺序的第2和第5个片。在识别ROI的片基础上，编码器对不同类型的片分配不同的编码资源，可为标注为ROI的片分配更强计算能力。

3.4.3 条/片划分实例

1. 分条划分的一例

在HEVC中，一帧图像可以划分为若干条，一个条又可以进一步划分为若干个分条（SS），分条分两类：独立的分条和非独立的分条。在图3.18中，每个小方块表示一个CTU，一帧图像共包含 9×11 个CTU。图中的实线将图像分为上下两个条，条1和条2。上面的条1中包含3个分条，其中最上面是一个独立的分条，包含4个CTU；中间一个是非独立的分条，包含32个CTU；下面是一个也是非独立的分条，包含24个CTU。下面的条2只有一个独立的分条，包含39个CTU。解码时，独立的分条中含有当前条中的一些全局配置信息，且不同条之间可以并行解码。

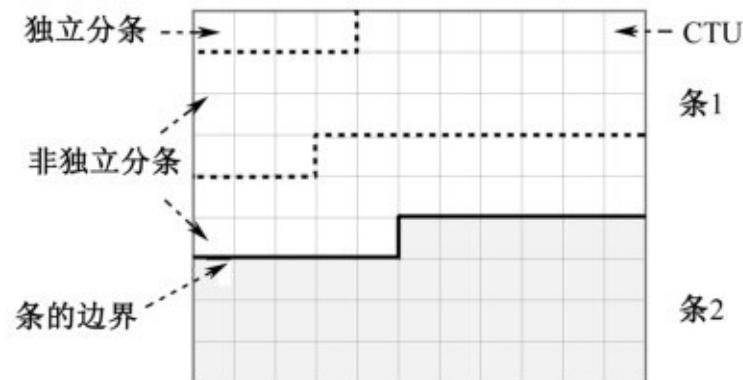


图3.18 图像中的条和分条的划分

2.条和片相互包含之例

在HEVC中，允许在同一帧图像内出现条的划分和片的划分。片必须是矩形，其内必须是整数个 CTU。在一个片中包含多个条的情况下，这些 CTU 可以分布在多个条中；条不一定是矩形，其内也必须是整数个 CTU。在一个条中包含多个片的情况下，这些 CTU 可以分布在多个片中。也就是说，无论条还是片，都是以CTU为最小组成单位。

一个条或分条内的所有的CTU必须属于同一个片，一个片内的所有的 CTU必须属于同一个条或分条。可见，条和片之间的包含必须是以整体的方式进行，一个条可以包含若干片，一个片可以包含若干条。在解码时，不同片之间也可以并行解码。

在图像中允许同时出现条划分和片划分，但必须明确包含关系，不是条包含在片中，就是片包含在条中。在图3.19中，每个小方块表示一个 CTU，一帧图像包含 9×11 个CTU。图3.19（a）为片中包含条的情况，一帧图像分为左右2片，第1片中包含3个条，第2片中包含2个条，处理顺序为片优先，片中的条再按光栅顺序扫描；图3.19（b）为条中包含片的情况，一帧图像分为上中下3条，第1条中包含3个片，即片1、片4和片7。第2条和第3条也都各包含3个片，处理顺序为条优先，条中的片再按光栅顺序扫描。

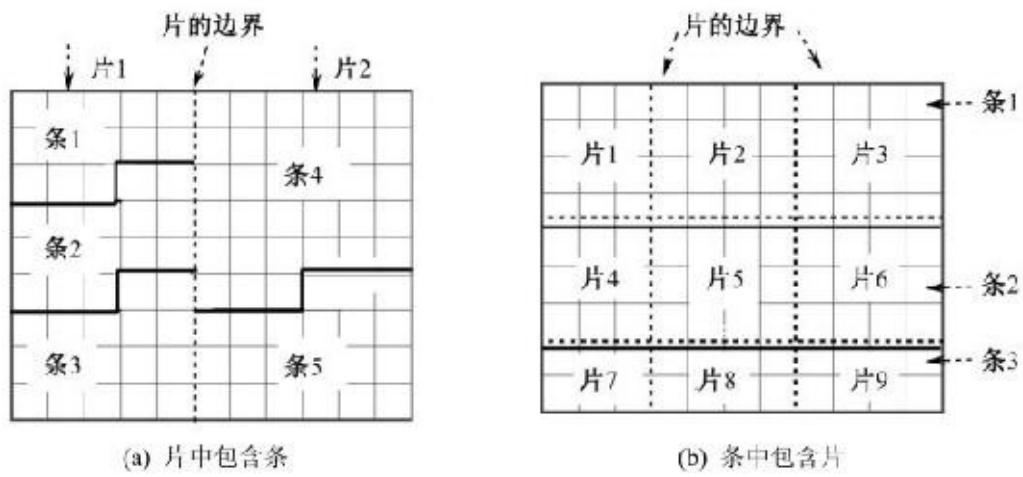


图3.19 片和条的包含关系示例

3.5 HEVC的档次、水平和等级

为提供不同应用之间的兼容互通，HEVC在附录A中定义了编码的不同档次（Profile）、水平（Level）和等级（Tier）。

（1）档次规定了码流中使用了哪些编码工具和算法。

（2）水平规定了对给定档次、等级所对应的解码器处理负担和存储容量参数，主要包括采样率、分辨率、码率的最大值、压缩率的最小值、解码图像缓冲区（Decoded Picture Buffer,DPB）的容量、编码图像缓冲区（Coded Picture Buffer,CPB）的容量等。

（3）等级规定了每个水平的码率的高低。

在编解码器的兼容性方面，要求支持某个档次的解码器必须支持该档次及低于该档次中的所有特性；要求支持某个水平和等级的解码器可以解码所有等于或低于这个水平和等级的码流。在编码器方面，支持某一档次的编码器，并不要求它支持该档次中所有的特性，但生成的码流必须是符合HEVC标准，可被支持该档次的解码器的正确解码。

3.5.1 档次

HEVC的档次规定了一套用于产生符合标准的不同用途码流的编码工具或算法。

1. 常用的3个主档次

HEVC第1版定义了常用的有3个主档次，即常规8比特像素精度的Main档次，支持10比特精度的Main10档次和支持静止图像的Main Still Picture档次。

（1）主档次（Main profile）

Main profile的主要技术指标包括：

- ① 像素的比特深度限制为8 bit。
- ② 色度亚采样限制为4:2:0格式。
- ③ CTB的尺寸为 16×16 到 64×64 。
- ④ 解码图像的缓存容量限制为6幅图像，即该档次的最大图像缓存容量。

⑤ 允许选择波前和片划分方式，但不是必需的，也不能同时选择。如果使用片，其尺寸至少高为64像素，宽为256像素。

(2) 10 bit主档次 (Main 10 profile)

Main 10 profile的主要技术指标和主档次类似，最主要的特点是它的像素比特深度限制为高精度的10 bit。

(3) 静止图像主档次 (Main Still Picture profile)

Main Still Picture profile的主要特点是不支持帧间预测编码。

归纳起来HEVC第1版的3个主档次的基本特征如图3.20所示。

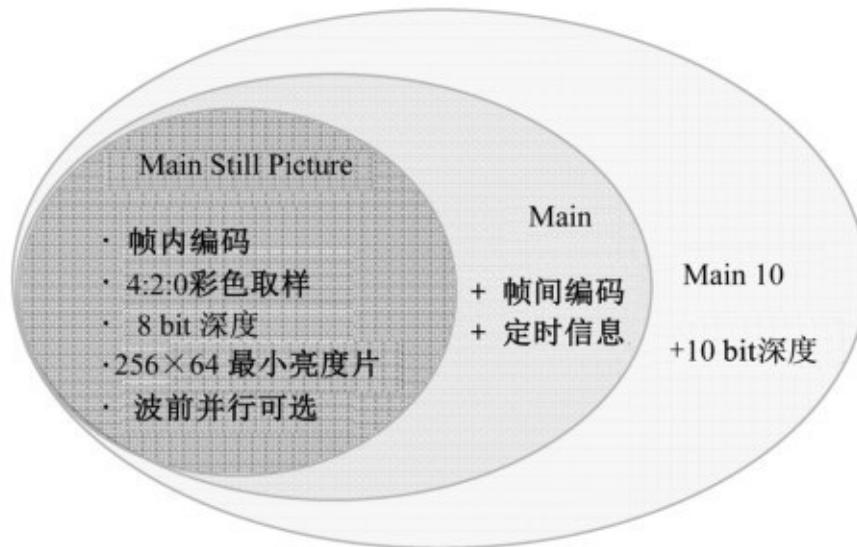


图3.20 HEVC第一版的3个主档次

2.格式范围扩展档次

HEVC第2版在3个主档次的基础上又定义了一系列的格式范围扩展档次，主要包括：

- (1) 单色档次，单色12档次，单色16档次。
- (2) 主12档次。
- (3) 主4:2:2 10档次，主4:2:2 12档次。
- (4) 主4:4:4 档次，主4:4:4 10档次，主4:4:4 12档次。
- (5) 主帧内档次，主10帧内档次，主12帧内档次；主4:2:2 10帧内档次，主4:2:2 12帧内档次；主4:4:4帧内档次，主4:4:4 10帧内档次，主4:4:4 12帧内档次，主4:4:4 16帧内档次。
- (6) 主4:4:4静止图像档次，主4:4:4 16静止图像档次。

3.格式范围扩展高吞吐量档次

HEVC 第2版还定义了一类格式范围扩展高吞吐量档次 (Format range extension high throughput profiles) , 目前主要包括高吞吐量4:4:4 16帧内档次。

3.5.2 水平

HEVC设置了1,2,2.1,3,3.1,4,4.1,5,5.1,5.2,6,6.1,6.2 13个水平 , 一个水平实际上就是一套对编码比特流的一系列编码参数 (最大采样频率、最大图像尺寸、最大比特率等) 的限制。如支持4:2:0格式视频 , 定义的图像分辨率从 176×144 (QCIF) 到 7680×4320 (8K×4K) , 限定最大输出码率等。如果说一个解码器具备解码某一水平码流的能力 , 则意味着该解码器具有解码这一水平以及低于这一水平所有码流的能力。

3.5.3 等级

对同一水平 , 按照最大码率和缓存容量要求的不同 , HEVC设置了两档等级 , 定义为高等级和主等级。主等级可用于大多数场合 , 涵盖13个水平 , 要求码率较低 ; 高等级可用于特殊要求或苛刻要求的场合 , 包括4和4以上的8个水平 , 允许码率较高 , 在同一水平大约高3 ~ 4倍。

表3.2 (HEVC第2版的Annex A的表A.4) 定义了不同等级的各个水平的限制值。表3.3 (HEVC第2版的Annex A的表A.5) 定义了Main档次和Main 10档次的不同等级的各个视频的限制值。

表3.2 一般等级 (Tier) 和水平 (Level) 限制

| 水平 (Level) | 最大亮度取样尺寸 MaxLumaPs (samples) | 最大 CPB 尺寸 (MaxCPB) (1000 or 1200 bits) | | 每帧最大 SS 数 (MaxSliceSegments PerPicture) | 最大行数 (MaxTileRows) | 最大列数 (MaxTileCels) |
|---------------|------------------------------------|--|--------------------|---|-----------------------|-----------------------|
| | | 主等级 (Main tier) | 高等级 (High tier) | | | |
| 1 | 36864 | 350 | — | 16 | 1 | 1 |
| 2 | 122880 | 1500 | — | 16 | 1 | 1 |
| 2.1 | 245760 | 3000 | — | 20 | 1 | 1 |
| 3 | 352960 | 6000 | — | 30 | 2 | 2 |
| 3.1 | 983040 | 10000 | — | 40 | 3 | 3 |
| 4 | 2228224 | 12000 | 20000 | 75 | 5 | 5 |
| 4.1 | 2228224 | 20000 | 50000 | 75 | 5 | 5 |
| 5 | 8912896 | 25000 | 100000 | 200 | 11 | 10 |
| 5.1 | 8912896 | 40000 | 160000 | 200 | 11 | 10 |
| 5.2 | 8912896 | 60000 | 240000 | 200 | 11 | 10 |
| 6 | 35651584 | 60000 | 240000 | 600 | 22 | 20 |
| 6.1 | 35651584 | 120000 | 480000 | 600 | 22 | 20 |
| 6.2 | 35651584 | 240000 | 800000 | 600 | 22 | 20 |

表3.3 Main档次 (Profile) 的等级 (Tier) 和水平 (Level) 限制

| 水平 (Level) | 最大亮度取样率 MaxLumaSr (samples/sec) | 最高比特率 (MaxBR) (1000bits/sec) | | 最小压缩率 (MinCrBase) | |
|---------------|---------------------------------------|---------------------------------|--------------------|----------------------|--------------------|
| | | 主等级 (Main tier) | 高等级 (High tier) | 主等级 (Main tier) | 高等级 (High tier) |
| 1 | 552960 | 128 | — | 2 | 2 |
| 2 | 3686400 | 1500 | — | 2 | 2 |
| 2.1 | 7372800 | 3000 | — | 2 | 2 |
| 3 | 16588800 | 6000 | — | 2 | 2 |
| 3.1 | 32177600 | 10000 | — | 2 | 2 |
| 4 | 66846720 | 12000 | 30000 | 4 | 4 |
| 4.1 | 133693440 | 20000 | 50000 | 4 | 4 |
| 5 | 262386880 | 25000 | 100000 | 6 | 4 |
| 5.1 | 534773760 | 40000 | 160000 | 8 | 4 |
| 5.2 | 1069547520 | 60000 | 240000 | 8 | 4 |
| 6 | 1069547520 | 60000 | 240000 | 8 | 4 |
| 6.1 | 2139095040 | 120000 | 480000 | 8 | 4 |
| 6.2 | 4278190080 | 240000 | 800000 | 6 | 4 |

本章参考文献

- [1]High Efficiency Video Coding[S].ITU-T Rec.H.265 and ISO/IEC 23008-2,ITU-T and ISO/IEC JCT-VC,Mach 2013 (v.1) ,Oct.2014 (v.2) ,Apr.2015 (v.3) .
- [2]Advanced video coding for generic audiovisual services[S].ITU-T Rec.H.264 and ISO/IEC 14496-10 (AVC) ,2012.
- [3]Information technology-Generic coding of moving pictures and associated audio information-Part 2: Video[S].ISO/IEC 13818-2: 2013 (MPEG-2) .
- [4]Misra,K.,et al.An overview of tiles in HEVC[J].IEEE Journal of Selected Topics in Signal Processing,2013,7 (6) : 969-977.
- [5]Thomas Wiegand,Gary J Sullivan.Overview of the H.264/AVC video coding standard[J].IEEE Trans.on Circuits and Systems for Video Technology,2003,13 (7) : 560-567.
- [6]Sjöberg,R.,Ying Che,Akira Fujibayashi,et al.Overview of HEVC high-level syntax and reference picture management[J].IEEE Trans.on Circuits and Systems for Video Technology,2012,22 (12) : 1858-1870.
- [7]Dong Hyeok Kim,Yong Hwan Kim,Woo Chool Park.Adaptive PU mode estimation algorithm for HEVC encoder[C].IEEE International Symposium on Consumer Electronics (ISCE 2015) :1-2.
- [8]F.Bossen.Common HM test conditions and software reference configurations[R].Doc.JCTVCI1100.9th meeting: Joint Collaborative Team on Video Coding (JCT-VC) of ITU-T SG 16 WP3 and ISO/IEC JTC 1/SC 29/WG,Geneva,CH,2012-4-11.

[9]Jaeho Lee,Seongwan Kim,Kyungmin Lim,et al.A fast CU size decision algorithm for HEVC[J].IEEE Trans.on Circuits and Systems for Video Technology.March 2015,25 (3) : 411-421.

[10]I.Kim,Junghye Min,Tammy Lee,et al.Block partitioning structure in the HEVC Standard[J].IEEE Trans.on Circuits and Systems for Video Technology,Dec.2012,22 (12) : 1697-1706.

[11]Yih Han Tan,Chuohao Yeo,Hui Li Tan,et al.On Residual Quad-Tree coding in HEVC[C].IEEE Multimedia Signal Processing (MMSP) Workshop,2011: 1-4.

[12]Mathias Wien.High Efficiency Video Coding: Coding Tools and Specification[M].Springer Verlag Berlin Heidelberg,2015.

[13]Vivienne Sze,Madhukar Budagavi,Gary J.Sullivan.High Efficiency Video Coding (HEVC) :Algorithms and Architectures[M].Switzerland Springer International Publishing,2014.

[14]Guilherme Correa,Pedro A.Assuncao,Luciano Volcan Agostini,et al.Fast HEVC encoding decisions using data mining[J].IEEE Trans.on Circuits and Systems for Video Technology.2015,25 (4) : 660-673.

第4章 HEVC的帧内预测

基于图像统计特性进行数据压缩的一类基本方法就是预测编码。它是利用图像信号的空间或时间相关性，用已重建的像素对当前正在编码的像素进行预测，然后对预测值与真实值的差——预测误差进行编码和传输。预测的方法有很多，目前用得较多的是线性预测方法，即用已重建像素的线性组合对正在编码的像素进行预测。

预测编码是图像压缩技术中研究得最早且应用最广的一种方法，它的主要特点是性能较好，算法简单，易于软硬件实现。实用的视频信号预测方法主要有两类：一类是以消除图像空间冗余信息为目标的帧内预测（Intra Prediction），即用于预测的参考像素和正在编码的像素在视频的同一帧内，而且一般同处于相近的邻域内；另一类是以消除图像时间冗余信息为目标的帧间预测（Inter Prediction），即用于预测的像素和正在编码的像素不在视频的同一帧内，但一般处于相邻近帧的相同或附近的位置上。

本章主要介绍HEVC的帧内预测技术，首先简要介绍帧内预测的原理和H.264/AVC的帧内预测，这是HEVC帧内预测的“发源地”，然后重点分析HEVC的帧内预测，包括多模式定义、预处理、预测的具体方法和预测过程。

4.1 帧内编码

4.1.1 空域预测编码

从第1章介绍过的预测编码的基本原理可知，帧内预测是一种较为简单和实用的图像压缩编码方法。预测压缩编码后传输的并不是像素值本身，而是编码像素的预测值和真实值之差，即预测误差或残差。

为什么用像素的预测误差作为压缩的结果或传输的对象？这是因为大量统计结果表明，同一幅图像的邻近像素之间有着很强的相关性，或者说这些像素值很相近，邻近像素值之间发生突变的概率很小。这样，编码像素的预测值往往和它的真实的像素值相差无几，预测误差很多为0或很小，我们就可以用较少的比特来表示和传输。**从信息论的角度来看，因为预测误差的分布比较集中（很不均匀），大多集中在“0”周围，而分布不均匀的信源符号其熵值比较小。而图像像素值的分布一般比较分散，或比较均匀，其熵值比较大。理想的压缩编码的平均码字长度总是趋于信源熵的，因此，将分布较均匀的像素转化为分布较集中的预测误差，为下一步的压缩编码提供了可能。**

这里的“邻近”像素的概念是指：和正在编码像素在同一扫描行内前后的几个像素，也包括上下相邻行紧挨编码像素的若干像素。如果是实时编码，按照扫描的时间顺序和因果关系，用于预测的像素的必须是当前编码像素的上面几行以及同一行左边的像素。因为在编码像素下面的行和同一行右边的像素尚未编码，不可利用。

如图4.1所示是一种简单的帧内预测方法的框图，可以用它来说明预测编码的原理。图4.1 (a) 中， X_0 为当前待编码的像素， X_1 为其同一行内已

编码的邻近像素， X_2, X_3, X_4 为上一行已编码的邻近像素。这些邻近的像素和 X_1 的距离很近，相关性很强。当然，其他像素和 X_1 也有相关性，只是距离越远相关性越弱。为简单起见，只考虑前值预测的情况，编码端以 X_0 前面的像素 X_1 的值作为 X_0 的预测值 \hat{X}_0 ，则 X_0 的预测误差 E_0 为

$$E_0 = X_0 - \hat{X}_0 = X_0 - X_1 \quad (4.1)$$

显然，由于临近像素之间相关性强，编码输出 E_0 的值一般非常小，可用较少的比特表示，从而达到压缩的目的。

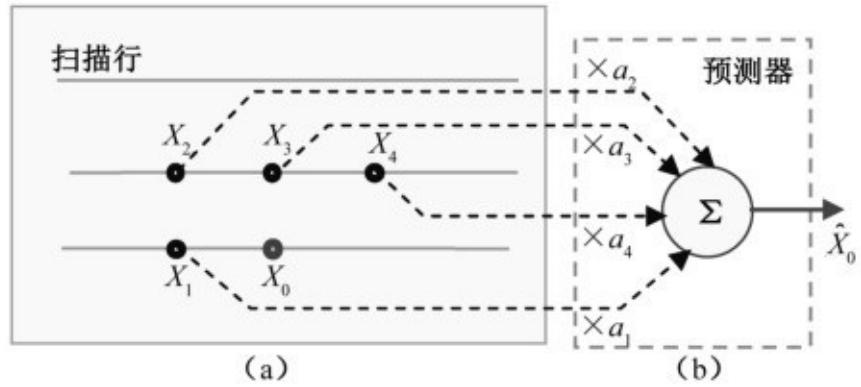


图4.1 预测像素和预测器

在解码端，接收到预测误差 E_0 后，将它和前面已经解码出来的 X_1 像素相加，就可以完全恢复出 X_0 的值，即

$$E_0 + X_1 = X_0 \quad (4.2)$$

从上面这一简单的预测编码过程，可以看出预测编码的方法是一种无损编码的方式，可以在接收端无误差地恢复原图像。还可以看出，如欲获得较高的压缩，则希望预测误差 E_0 尽可能小，也就说尽可能预测准确。为了更准确地预测 X_0 ，我们常常可以用多个像素值的线性组合来预测，如图4.1（b）中所示用 $X_1 \sim X_4$ 的线性组合（预测器）来作为 X_0 的预测值，对应的加权系数为 $a_1 \sim a_4$ 或更多像素的线性组合，会获得比简单的前值预测更加精确的值。

4.1.2 最佳线性预测

预测编码性能的优劣很大程度上取决于预测器的设计，主要是确定预测器的阶数 N ，以及各个预测系数。图4.1（b）表示一个具体预测器的运算结构，是一个4阶线性预测器，预测器的输出是输入数据的线性组合。

对于预测阶数，由图像的统计特性可知，一帧图像内像素之间的相关系数在较小的范围内可用指型衰减曲线近似。当像素距离增大时，其相关性急剧减弱，因此，预测器的阶数不宜取得过大。实验表明，对于一般图像， N 的取值范围在0~10就可以了。在预测阶数确定以后，就可设计一个性能最佳的线性预测器（实际上就是找到一组最佳的预测系数）。

为不失一般性，可以将图4.1（b）的预测器推广到 N 阶。假定当前待编码的像素为 X_0 ，其前面 N 个已编像素分别为 X_1, X_2, \dots, X_N ，若用它们对 X_0 进行预测，并用 \hat{X}_0 表示预测值， $\{a_i | i=1, 2, \dots, N\}$ 表示线性预测的系数，可写成

$$\hat{X}_0 = a_1 X_1 + a_2 X_2 + \cdots + a_N X_N = \sum_{i=1}^N a_i X_i \quad (4.3)$$

则预测误差为

$$e = X_0 - \hat{X}_0 = X_0 - \sum_{i=1}^N a_i X_i \quad (4.4)$$

因为 e 的均值为0，其均方值也就是方差，即

$$\sigma_e^2 = E[(X_0 - \hat{X}_0)^2] \quad (4.5)$$

最佳预测就是求解预测系数集合 $\{a_i\}$ ，使它所产生的预测误差的方差 σ_e^2 达到最小。这等价于求方差的极小值，可用 $\{a_i\}$ 对 σ_e^2 求偏导并令其为0，整理后得

$$R_{0j} = \sum_{i=1}^N a_i E(X_i X_j) = \sum_{i=1}^N a_i R_{ij} \quad j = 1, 2, \dots, N \quad (4.6)$$

式中， R_{ij} 表示 X_i, X_j 之间的相关函数。这是一个N阶线性方程组，可由此解出N个预测系数 $\{a_i | i=1, 2, \dots, N\}$ 。由于它们使预测误差的均方值极小，

因此称其为最佳预测系数。可以证明，在这种情况下，预测误差的方差比原始图像信号的方差要小，甚至可能小很多，这就说明预测数据大量地集中在0值左右，为进一步的压缩创造了条件。

由于图像内容的变化，导致最佳预测系数 $\{a_i\}$ 也随之而变，在实际应用中要完全做到随图像而变化的预测系数是难以实现的。因此，在视频标准的帧内压缩中，常常是根据图像的局部统计特性，近似设计出随内容变化的预测系数来为编码像素计算预测值。

4.2 H.264/AVC的帧内预测

在前面介绍预测编码时是以像素为编码单位，逐像素进行，每个编码像素的预测模式（用于预测的像素位置及对应的预测系数）都一样。在H.264/AVC中则有所不同，编码是以块或宏块为单位进行的，不同块的预测模式不一定相同，同一块中不同位置像素预测所用的参考像素和预测系数也不一定相同。

在H.264/AVC的帧内预测编码中，对当前块的预测是基于邻近已编码重建的块进行的。对亮度像素而言，独立进行预测的块可以是 16×16 宏块或 4×4 子块。 4×4 的亮度子块有9种可选的预测模式，适于对具有大量细节的图像进行预测； 16×16 的亮度块有4种预测模式，适于对具有平坦区域的图像进行预测。和 16×16 亮度宏块对应的色度块尺寸为 8×8 ，也有4种预测模式，类似于 16×16 亮度块预测模式。

由此可见，在H.264/AVC的帧内预测编码中，一个子块或宏块将面临多种预测模式的选择，不同的模式选择将产生不同的压缩率，最佳模式选择由此而成为帧内预测编码中的一个重要问题。选择最佳预测模式的方法有很多种，其中率失真优化（RDO）方法是一种性能良好的选择方法，其原则是选择能获得最准确预测、产生码字最少的那种预测模式。

4.2.1 亮度 4×4 块的预测模式

如图4.2（a）所示，帧内预测模式中待编码的 4×4 亮度共有16个像素，其值标识为a~p，帧内预测的任务就是为这16个待预测像素的每一个确定其预测值。编码块的上方和左方为已编码的块，预测用到的仅是紧靠编码块上方和左方的一行、一列像素，其值标识为A~M，称之为预测参考像

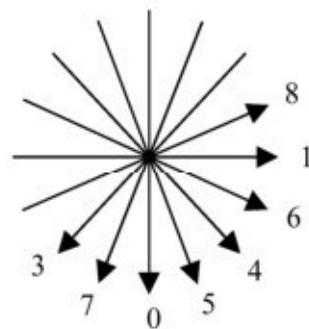
素。

预测操作实际上就是建立待预测像素值 $a \sim p$ 和预测参考像素值 $A \sim M$ 之间的关系。在H.264/AVC中，预测关系并不像上述预测原理中的那种固定关系，而是随着 4×4 亮度块的内容而自适应变化的，共有9种模式可选择。每个 4×4 亮度块只使用其中的一种预测模式，也就是说，小块中的16个像素都是统一测预测模式。H.264/AVC没有规定这9种模式是如何得到的，也不规定编码块如何确定适应自己的预测模式，仅规定那9种模式，每种模式的具体情况如何。

在9种预测模式中，模式0、1、3~8为8个不同的预测方向，如4.2 (b) 中箭头所示。 $a \sim p$ 中某个像素的预测值就通过对 $A \sim M$ 中某一方向上的参考像素进行加权平均（线性组合）求得的。

| | | | | | | | | |
|-----|-----|-----|-----|-----|-----|-----|-----|-----|
| M | A | B | C | D | E | F | G | H |
| I | a | b | c | d | | | | |
| J | e | f | g | h | | | | |
| K | i | j | k | l | | | | |
| L | m | n | o | p | | | | |

(a) 4×4 编码块与相邻像素的关系



(b) 4×4 编码块的 8 种预测方向

图4.2 4×3 亮度块的帧内预测

图4.3给出了 4×4 亮度块帧内预测9种模式中编码像素和预测像素之间的关系，用虚线箭头根部的参考像素（1个或多个）的加权值来预测箭头经过的待编码像素的预测值。在H.264/AVC中，为每种模式的每个或每一类待编码像素确定了预测公式。

图4.3中模式0为垂直模式，由 -90° 方向参考像素A、B、C、D加权得出预测值；模式1为水平模式，由 0° 方向参考像素I、J、K、L加权得出相应预测值；模式2为DC模式，由A～D及I～L平均值得出所有预测值；模式3为下左对角线模式，模式4为下右对角线模式，分别由 -45° 方向和 -135° 方向参考像素加权得出相应预测值；模式5为右垂直模式，模式6为下水平模式，模式7为左垂直模式，模式8为上水平模式，分别由 -63.4° 方向、 -26.6° 方向、 -116.6° 方向、 26.6° 方向参考像素加权得出相应预测值。

例如，根据9种模式的定义，模式2为直流方式，即取参考像素A～M的平均值作为预测值。再如在模式3中，b的预测值为 $(B+2C+D+2)/4$ 。在实际编码过程中确定一个 4×4 的亮度块的帧内预测模式可以有多种方法，常见的一种简单的模式选择方法就是计算待编码的亮度 4×4 块的9种预测模式产生的9个预测块，分别计算9个预测块的预测误差的绝对误差和（Sum of Absolute Error, SAE），其中最小SAE对应的预测模式即为与当前块最匹配的模预测模式，因为它最接近于编码 4×4 块，将产生的预测误差会最小。

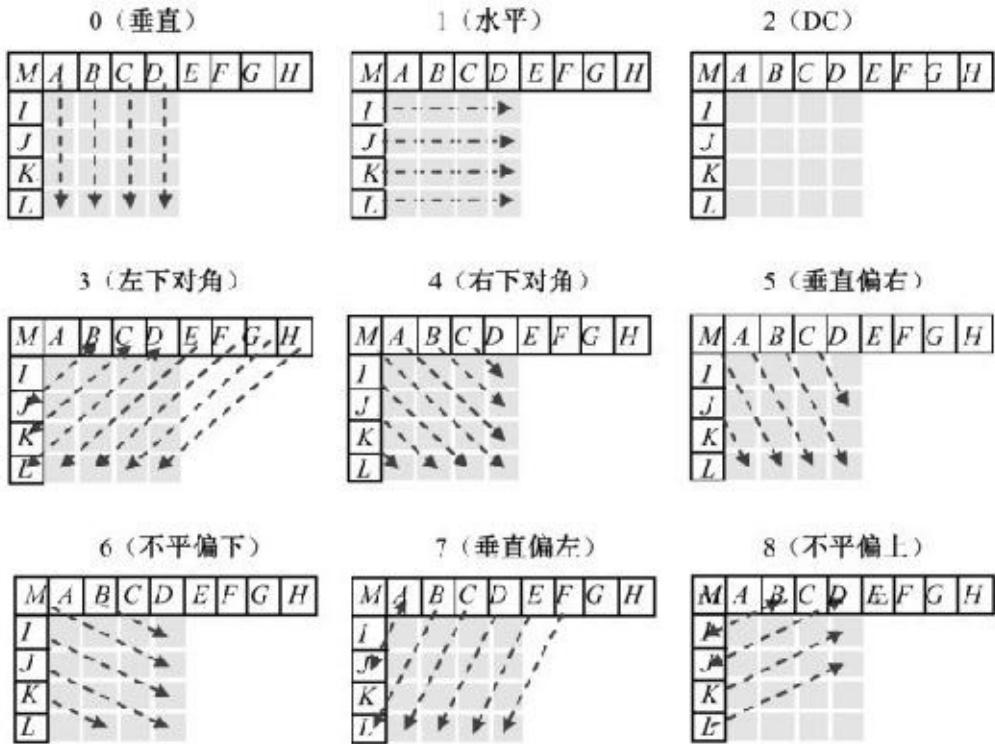


图4.3 4×4 亮度块的9种帧内预测模式

以4×4亮度块的垂直偏左模式7 (Intra_4×4_Vertical_Left) 为例，其16个像素的预测值的计算参见图4.4，中间灰色部分为4×4预测块，左边和上边的13个像素为用于预测的已编码参考像素，垂直偏左的箭头表示该预测块的预测方向，每个像素的坐标值标注在像素的方格中。按照H.264/AVC的定义，可以计算出预测块的所有的16个像素的预测值 $\text{pred}_{4\times4_L}[x,y]$ 。

$p[x,y]$ 为像素的二维坐标，当 $y=0$ 或 $y=2$ 时，预测值公式为：

$$\text{pred4} \times 4_L[x,y] = (p[x+(y>>1), -1] + p[x+(y>>1)+1, -1] + 1) >> 1$$

当 $y=1$ 或 $y=3$ 时，预测值公式为：

$$\begin{aligned}\text{pred4} \times 4_L[x,y] = & (p[x+(y>>1), -1] + 2*p[x+(y>>1)+1, -1] \\ & + p[x+(y>>1)+2, -1] + 2) >> 2\end{aligned}$$

具体到当 $x=2, y=0$ 时，像素 $(2,0)$ 的预测值为
 $\text{pred4} \times 4_L[2,0] = (p[2,-1] + p[3,-1] + 1) \gg 1$ ，表示该预测值为参考像素 $p[2,-1]$ 和 $p[3,-1]$ 的平均。当 $x=2, y=1$ 时，像素 $(2,1)$ 的预测值为
 $\text{pred4} \times 4_L[2,1] = (p[2,-1] + 2*p[3,-1] + p[4,-1] + 2) \gg 2$ ，表示该预测值为参考像素 $p[2,-1]$ 、 $p[3,-1]$ 和 $p[4,-1]$ 的加权平均，以在同一方向的参考像素 $p[3,-1]$ 为

主，占权重的一半。其他像素点的帧内预测值可照此类推得到。

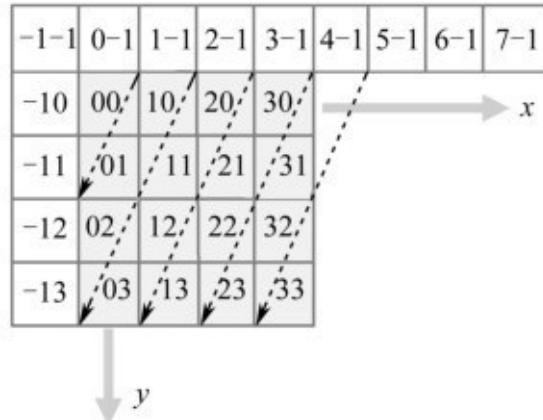


图4.4 预测值计算一例

4.2.2 亮度 16×16 块的预测模式

一般来说，图像内容变化不大的区域，可以采用较大的 16×16 亮度宏块进行整体预测。参与预测的已编码像素为待编码宏块上方的一行16个像素和左方的一列16个像素。 16×16 亮度宏块的帧内预测有4种模式，如图4.5所示。模式0为垂直模式，由待编码块上边像素推出相应的预测值；模

式1为水平模式，由左边像素推出相应的预测值；模式2为DC模式，由上边和左边像素平均值推出相应的预测值；模式3为平面模式，利用线性“plane”函数（双线性内插）及左、上像素推出相应的预测值，适用于亮度变化平缓区域。

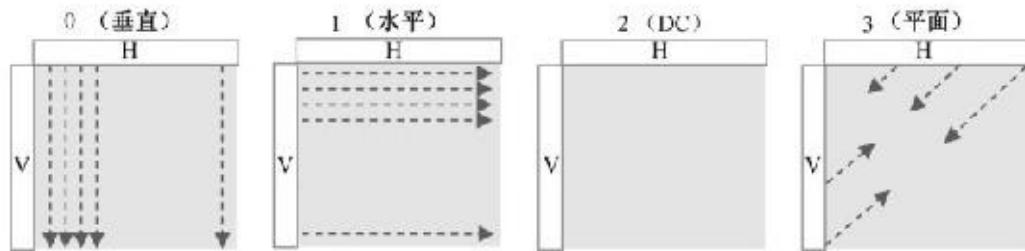


图4.5 16×16亮度宏块的帧内预测模式

4.2.3 色度8×8块的预测模式

对于4:2:0色度取样模式，8×8色度块对应的是16×16的亮度块，即和16×16的亮度块几何位置重合。考虑到人眼对色度信号的分辨率一般低于亮度信号，因此在色度信号的帧内预测中统一取8×8块的尺寸，不再细分。每个帧内编码宏块的8×8色度块的预测由已编码左方和上方的已编码

的色度像素提供，并且两种色度成分Cb和Cr常用同一种预测模式。色度帧内预测的4种模式类似于帧内 16×16 预测的4种预测模式，只是模式编号不同。其中DC预测为模式0、水平预测为模式1、垂直预测为模式2、平面预测为模式3。

4.3 HEVC的帧内预测模式

HEVC的帧内预测技术和H.264/AVC类似，采用基于块的多方向帧内预测方式来消除图像的空间相关性，但比 H.264/AVC 预测方向更细、更灵活。HEVC 预测块的预测参考像素也和H.264/AVC类似，来自预测块左方的一列和上方的一行已编码的像素。HEVC为亮度预测块定义了33种不同的帧内预测方向，连同平面和直流模式，总共35种帧内预测模式。HEVC 中色度块的帧内预测模式和对应的亮度块有一定的关联，可用5种模式来表示。为了提高帧内预测的效率，HEVC对 8×8 或更大的预测块在预测前对参考像素进行了简单的平滑滤波预处理。

4.3.1 帧内预测PU的划分

先简单回顾一下HEVC的分块编码结构：忽略相关的语法元素，一帧图像可以划分为若干方形CTU，尺寸为 64×64 、 32×32 或 16×16 ,1个CTU包含一个亮度CTB和2个对应的色度CTB;1个CTU可根据四叉树原则划分为若干CU，尺寸为 $64\times 64 \sim 8\times 8$,1个CU包含一个亮度CB和2个对应的色度CB。CU是HEVC的基本编码单元，是HEVC编码树的“树叶”，即最低编码层。预测单元PU和变换单元TU是在编码单元CU的基础上再进行划分而得。1个CU可以一次划分成1个或多个PU，同时可按照四叉树原则经1层或多层划分为多个TU,CU是“变换树”的“树根”,TU是它的“树叶”。每个PU包括1个亮度预测块（PB）和2个对应的色度预测块（PB）。每个TU也如此，包括1个亮度变换块（TB）和2个对应的色度变换块（TB）。CU中的PU和TU的划分是相对独立的，但在帧内模式中要求TB的尺寸小于PB的尺寸。

既然CU是基本的编码单位，那么，采用帧内预测或帧间预测编码模式

也是由CU决定的。当 $2N \times 2N$ 的CU定义为帧内预测模式时，CU到PU仅允许一层划分，因而有两种划分选择，可见第3章图3.9中CU划分的右上角的情况。一种是保持和CU相同的尺寸的划分（PART_2N×2N），即 $2N \times 2N$ ，实际上是不用划分；另一种是对于最小尺寸的CB（8×8）可划分为4个 $N \times N$ 的大小相同的正方形块PU（PART_N×N）。HEVC的这种划分一方面比较适应图像的局部统计特性，可获得更加准确的帧内预测结果，当然其计算复杂度也会增加；另一方面HEVC只允许最小尺寸的帧内CU可以分裂为4个小的方形PU，不能适应某些非方形图像纹理的情况，这一限制虽然对编码效率会产生一点影响，但它可显著减少编码复杂度。

按照上述的划分原则，在帧内编码中，亮度PU的尺寸范围为 $32 \times 32 \sim 4 \times 4, 4 \times 4$ 是最小的亮度PU块，同时也是最小的色度PU块（4:2:0色度模式）。因为在帧内CU中，不管最小CU是否划分为4个亮度PU，对应的色度CU都不再划分，覆盖整个CU的面积，CU的最小尺寸为 8×8 ，决定了它的最小色度CU的尺寸为 4×4 。如果CU被划分为4个PU，则允许各个亮度PU采用不同的帧内预测模式；同一亮度PU的两个色度PU则共享一个彩色帧内预测模式。

4.3.2 亮度PU的帧内预测模式

由上述的PU划分结果可知，当CU使用帧内预测模式编码，而且CU的尺寸不等于最小CU尺寸时，PU的尺寸始终等于CU的尺寸。

1. 基于TU的预测

尽管帧内预测模式是建立在PU层面上，但帧内预测的对象是TU。当一个CU分裂为多个TU时，帧内预测操作实际是顺序对每个TU进行的，而不是在PU层面上进行。这种方法的一个显著的优点就是总可以利用最邻近的已重建TU的参考像素。对比实验表明，和用PU边界的参考像素作PU帧内预测的情况相比较，这一措施可提升帧内图像编码效率约1.2%。还有一个好处，由于所有的帧内预测信息需逐PU标注，使得在同一PU内的多个

TU可共享相同的帧内预测模式信息。

2. 预测方向的定义

为了匹配图像块的内容结构，提高预测精度，HEVC比H.264/AVC更加细分了预测角度，大大增加了预测方向。HEVC共有35种预测模式，其中33种为方向预测模式，或称角度预测模式。图4.6显示了亮度块预测33个方向的定义，连同直流和平面模式，共计35种预测模式，它们都有相应的编号。

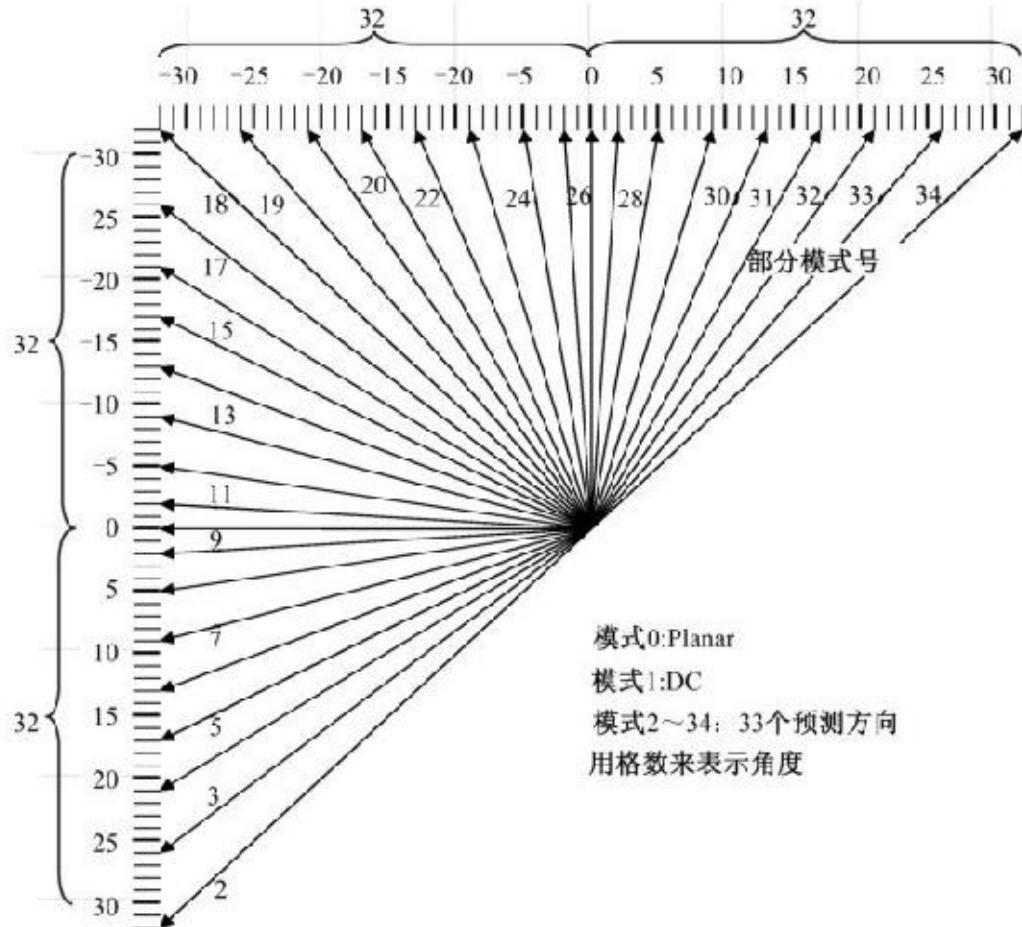


图4.6 帧内预测模式和角度

预测方向并没有用几何角度来表示，而是用像素的个数或格数来表示的。帧内亮度预测单元（PU）的尺寸从 4×4 到 32×32 ，对于所有尺寸的预测单元，一律采用所有的35种模式。在使用重构参考像素预测当前块时，对于各种尺寸的预测块均采用1/32插值精度，对应32个像素标注格，而不随预测尺寸大小不同而采用的插值精度不同。

从图4.6中可以看出，左侧的预测模式对应的预测角度和上方预测的预测模式对应的预测角度都是对称的，都是关于“0”格标度对称，一边32格。图中的箭头表示不同的预测方向，一个方向代表一种模式，从2到34共有33种不同的方向模式，其模式号也标注在图中。表4.1逐个给出了不同编号的预测模式所对应的预测角度（用格数来表示）。

表4.1 HEVC帧内预测模式编号和角度对应表

| 模式号 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 |
|-------------------|-----|-----|-----|-----|-----|----|----|----|----|----|----|----|----|----|----|----|----|
| 角度 (格数 δ) | 32 | 26 | 21 | 17 | 13 | 9 | 5 | 2 | 0 | 2 | 5 | 9 | 13 | 17 | 21 | 26 | |
| 模式号 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 | 29 | 30 | 31 | 32 | 33 | 34 |
| 角度 (格数 δ) | -32 | -26 | -21 | -17 | -13 | -9 | -5 | -2 | 0 | 2 | 5 | 9 | 13 | 17 | 21 | 26 | 32 |

从图4.6和表4.1还可以看出，分别以模式10和模式26为水平和垂直坐标轴，其他方向分别是在这两个轴的两侧 $+/-[0,2,5,9,13,17,21,26,32]/32$ 的地方，正负号表示水平坐标轴和垂直坐标轴上的相对位置。当预测方向为正时，只有主要参考像素用作当前块预测。

3.三类预测模式

HEVC亮度块帧内预测的35种预测模式可分为如下的3大类：

(1) 平面 (Intra_Planar) 预测模式 (模式0)

平面 (Planar) 预测模式适合纹理比较平滑的区域，尤其是变化趋势比较一致的区域。Planar模式对当前块中每个像素使用不同的预测值，使用水平和垂直两个方向的线性插值的平均作为当前像素的预测值。

(2) 直流 (Intra_DC) 预测模式 (模式1)

DC 模式对当前块的所有像素使用同一个预测值，即预测参考像素的平均值。这种模式适用于图像的平坦区域。

(3) 角度 (Intra_Angular) 预测模式 (模式2 ~ 34)

HEVC的帧内角度预测和H.264/AVC类似，也是在空域的逐像素预测操作，但有显著的扩展，主要是PB尺寸的增加和可选预测方向数目的增加。使用Intra_Angular模式时，每个帧内预测PU都有2个参考像素集：水平参考像素行——当前预测PU的上一行像素；垂直参考像素列——当前预测PU的左一列像素。预测PB的每个像素都可从那些在本方向预测前已经重建

的参考像素集的样值进行角度预测。

在较早的HEVC测试版本HM中，由于PU的大小和图像的内容有关，在图像细节多变区域，PU的分割往往趋细，相反图像平缓的区域PU分割趋粗。在这两种情况下，预测方向的种类都可以减少：对 4×4 的细节小块，因为块尺寸小，太多的预测方向分辨不了，从而设置了17种预测方向；对 64×64 的平缓大块，因为块大且变化小，只要少量的方向就可表示，设置了5种预测方向。这样因PU尺寸不同而变化的方向数本身也给编码带来了一定的复杂性。经过权衡，在HEVC的测试模型HM7.0以后，为了更加精确的预测和省略不同尺寸PU的方向数带来麻烦，对每种尺寸的PU一律设置了35种编码模式，其中预测方向为33种。

4.3.3 色度PU的帧内预测模式

由于在彩色视频中同一位置的色度信号和相应的亮度信号的特性大多类似，因此在视频处理中，常常对色度信号采用和相应的亮度信号相同的方法。HEVC也不例外，在帧内编码中色度信号也采用相应亮度信号所采用的预测模式。

对于色度块的帧内预测，HEVC支持所有的亮度预测模式原则上都可以用于色度预测，从而在预测准确度和模式量之间取得了良好的折中，可直接对所选择的色度预测模式进行编码，无须使用最可能预测模式的预测机制。对于色度信息，HEVC有5种帧内预测模式，即模式0~4：色度模式0，即Planar模式（相当于亮度模式0）；色度模式1，即垂直模式（相当于亮度模式26）；色度模式2，即水平模式（相当于亮度模式10）；色度模式3，即DC模式（相当于亮度模式1）；色度模式4，又称导出（derived）模式，采用和对应亮度块相同的预测模式。可见，虽然色度模式只有5个不同的编号，当由于和相应的亮度模式关联，实质上仍然可表示35种帧内预测模式。

根据色度块最佳预测角度和对应亮度块的预测模式，由表4.2可决定色

度块的帧内预测模式。表4.2的最左边一列为色度块的最佳预测角度，最上面的一行为对应亮度快的预测模式，由这两个数据就可以查出当前色度块的编码模式。其中模式34表示非0,26,10,1的其他31种角度中的任意一种。

表4.2 色度块的预测模式

| 色度块模式编号 | | 对应的亮度预测块的模式 | | | | |
|-----------------|------------|-------------|---------|---------|--------|----|
| | | 0 (Planar) | 26 (垂直) | 10 (水平) | 1 (DC) | 34 |
| 色度块 预测角 度 | 0 (Planar) | 4 | 0 | 0 | 0 | 0 |
| | 26 (垂直) | 1 | 4 | 1 | 1 | 1 |
| | 10 (水平) | 2 | 2 | 4 | 2 | 2 |
| | 1 (DC) | 3 | 3 | 3 | 4 | 3 |
| | 34 | 0 | 1 | 2 | 3 | 4 |

对于和亮度块对应的色度快，首先计算它的最佳预测“角度”，和亮度块一样也有35种不同的“角度”。

如果该色度块对应的亮度块模式不是0,10,26,1中之一，也就是说是模式34，则直接确定色度块的模式，即表4.2中的最右一列，如色度的预测角度为26，对应的亮度模式为34，则此色度块的据预测模式为“1”。再如色度的预测角度为19（属于模式34），对应的亮度模式为22（也属于模式34），则此色度块的预测模式编号为“4”，意味着此色度块预测角度为“22”，和对应的亮度块一致，而不取色度预测角度为19。

如果该色度块对应的亮度块模式是0,10,26,1中之一，且色度的预测角度和对应的亮度模式一致，则此色度块的据预测模式为“4”，对应表4.2中的右下对角线。如色度和对应亮度块的预测角度都是10，则此色度块的预测模式编号为“4”。

如果该色度块对应的亮度块模式是0,10,26,1中之一，但色度的预测角度和对应的亮度模式不一致，则此色度块的预测模式对应为表4.2中的除对角单元和最右列单元外的其他单元中的模式编号。如色度的预测角度为26，对应的亮度模式为0，则此色度预测块的预测模式编号为“1”。

为了避免冗余信令，当相应的色度块属于前面的4个色度模式之一（平面、垂直、水平和直流），而且是和亮度预测模式相同，则代之而采用

Intra_Angular[34]模式，不用再设置新的标识，在解码器中只要检测到帧内预测色度块的“Intra_Angular[34]”模式，则自动认为和相应的亮度块同模式。

4.4 HEVC的帧内预测过程

HEVC帧内预测的过程大致可分为三步：参考像素的填充，参考像素的平滑滤波，利用参考像素值求出预测块像素的预测值。下面主要介绍亮度信号的帧内预测过程，色度信号的帧内预测过程和亮度相仿，不再具体介绍。

4.4.1 参考像素的准备

填充参考像素的目的是准备好当前预测块周围所有的参考像素值，为接下来的帧内预测做准备。与H.264/AVC不同的是，HEVC中当前块的左下方和右上方的像素也可以作为参考像素。这是因为在HEVC中允许不同尺寸的编码块，所以经常出现左下方和右上方的像素属于已编码块的情况。假设预测块的大小为 $nTbS \times nTbS$ ，且预测块左上角的坐标为(0,0)，则参考像素为该预测块的左侧、左下、上方、右上，以及左上角的像素组成，如图4.7所示。参考像素值表示为 $p[x][y]$ ， x 表示横坐标， y 表示纵坐标，左边参考像素的坐标为 $x=-l, y=-l, -2, \dots, -nTbS \times 2 + l$ ，上边的参考像素的坐标为 $x=0, 1, \dots, nTbS \times 2 - l, y=-1$ 。如预测块的大小为 32×32 ，则左参考像素为 $x=-l, y=-l, -2, \dots, -63$ ，上参考像素为 $x=0, 1, \dots, 63, y=-1$ 。

值得注意的是，在某些情况下参考像素有可能不可用。例如，参考像素的位置在图像外、条分割外，或者参考像素所属的编码块的不是帧内预测模式且被限制不能作为帧内预测的参考块等情况。因此参考像素是否能够用于帧内预测，需要具体检查。HEVC 中根据参考像素可用和不可用的情况，规定了三种不同的填充处理方法。

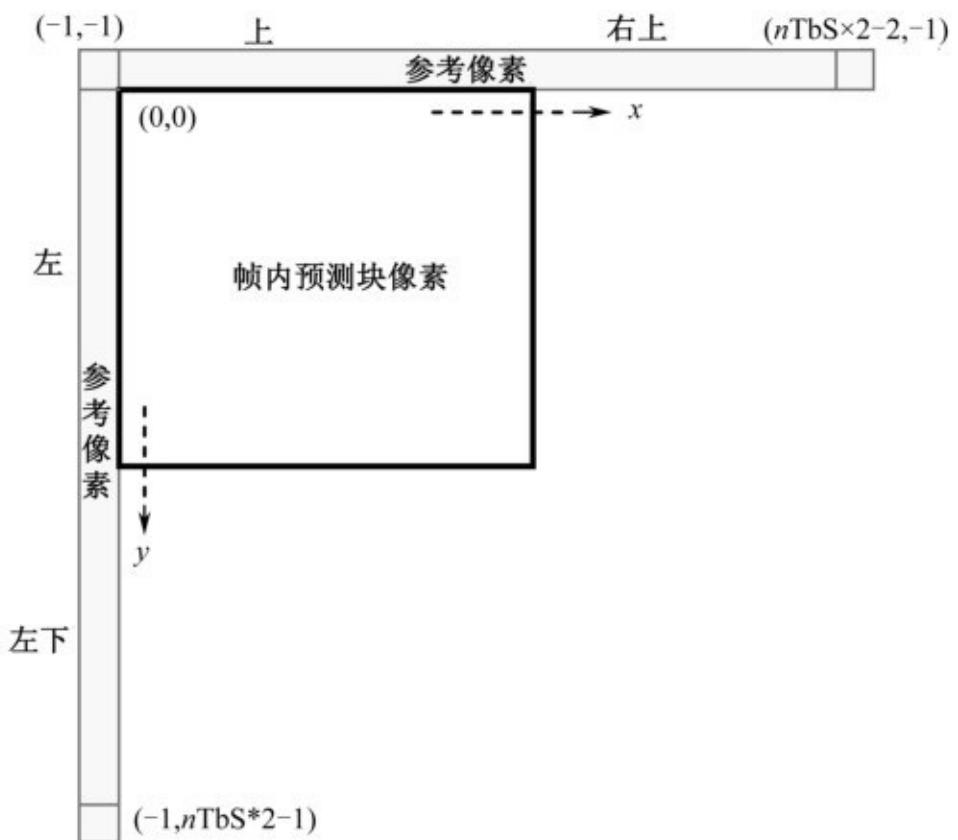


图4.7 帧内预测的参考像素位置

- (1) 如果所有的参考像素 $p[x][y]$ 都被标记为可用于帧内预测，则 $p[x][y]$ 取重建图像中对应像素的值，即已编码重建的像素值。
- (2) 如果所有的参考像素 $p[x][y]$ 都被标记为不能用于帧内预测，则所有的 $p[x][y]$ 的值填充为 $2^{\text{bitDepth}-1}$ ，其中 bitDepth 代表的是比特深度，如在常见的 $\text{bitDepth}=8$ 比特情况下， $p[x][y]$ 的值为 $2^{8-1}=128$ 。

(3) 如果至少有一个 $p[x][y]$ 但不是所有的 $p[x][y]$ 被标记为不可用于帧内预测时，参考像素值处理如下。

首先，查看最下方的参考像素即 $p[-l][nTbS \times 2-l]$ 是否被标记为可用，如果不可用则从下往上，然后从左往右，依次遍历，直到找到被标记为可用于帧内预测的像素点 $p[x][y]$ 为止，并将该 $p[x][y]$ 值赋给最下方的参考像素 $p[-l][nTbS \times 2-l]$ 。如果 $p[-l][nTbS \times 2-l]$ 标记为可用，则不需要遍历和赋值的过程。

其次， $p[-l][nTbS \times 2-l]$ 的值确定后，顺序检查左边从 $p[-l][nTbS \times 2-2]$ 到 $p[-l][-l]$ 的像素点，如果当前像素点 $p[x][y]$ 不可用，则用其下方紧邻的像素点 $p[x][y+1]$ 的值代替 $p[x][y]$ 。

最后，顺序检查上方从 $p[0][-1]$ 到 $p[nTbS \times 2-l][-1]$ 的像素点，如果当前像素点 $p[x][y]$ 不可用，则用其左方紧邻的像素点 $p[x-l][y]$ 的值代替 $p[x][y]$ 。

经过上述的检查，即完成了当前预测块的所有的帧内预测的参考像素的设置。

4.4.2 参考像素的平滑滤波

为了提高帧内预测的效率，HEVC 对 8×8 或更大的帧内 PU 的部分参考像素在预测前进行平滑滤波预处理，以减少噪声对预测的影响，提高预测的精度和效率。平滑滤波器实际上是一个简单的一维 3 抽头有限冲激响应低通滤波器，用它对处于水平或垂直方向上的参考像素进行一维滤波。

1. 需要平滑滤波的参考像素

HEVC 根据预测块的尺寸和帧内预测模式的不同，选择性地对预测块的参考像素集进行平滑处理。

对 4×4 预测块的参考像素集，不需要进行平滑滤波。其他所有尺寸的预测块，如果是直流 (DC) 预测模式，其参考像素集是不需要平滑滤波的；如果是平面 (Planar) 预测模式，则需要平滑滤波。

对于角度模式，总体上越是较大的预测块和越远离垂直或水平方向的

预测模式，越是要对参考像素进行平滑处理。具体情况如下。

(1) 对 8×8 预测块的参考像素集，只需要对3个对角方向的模式，即模式2、18和34进行常规平滑滤波。

(2) 对 16×16 预测块的参考像素集，除水平方向附近的模式9、10和11，垂直方向附近的模式25、26和27不需要滤波外，其他27个角度模式都需要进行常规平滑滤波。

(3) 对 32×32 预测块的参考像素集，除水平模式10和垂直模式26外，其余31个角度模式都要进行平滑滤波。 32×32 的块的参考像素集的平滑滤波有两者模式，一种是常规滤波模式，另一种是强平滑滤波模式。

由此可见，对大于 4×4 预测块的参考像素集，都要施行低通滤波，对角方向模式(2,18,34)总是需要滤波的，而且使用滤波的方向数目随着块尺寸的增加而增加。

2. 常规平滑滤波

参考像素的常规平滑滤波可通过3抽头滤波器来实现，如图4.8(a)所示，抽头系数为 $[1,2,1]/4$ ，参考像素 $p[x][y]$ 滤波后对应的像素值为 $pF[x][y]$ ，如式(4.7)~式(4.11)所示。

左上角参考像素的滤波处理如下，

$$pF[-1][-1] = (P[-1][0] + 2 \times p[-1][-1] + p[0][-1] + 2) \gg 2 \quad (4.7)$$

左方参考像素的滤波处理如下，

$$pF[-l][y] = (p[-1][y+1] - 2 \times p[-1][y] + p[-1][y-1] + 2) \gg 2 \quad (4.8)$$

在式(4.8)中，如果左下角参考像素 $p[-1][nTbS \times 2 - l]$ 没有相邻参考像素，则滤波后的值保持不变，如式(4.9)：

$$pF[-l][nTbS \times 2 - l] = p[-l][nTbS * 2 - l] \quad (4.9)$$

上方参考像素的滤波处理如下，

$$pF[x][-1] = (p[x-l][-1] + 2 \times p[x][-1] + p[x+l][-1] + 2) \gg 2 \quad (4.10)$$

在式(4.10)中，如果右上角参考像素 $p[nTbS \times 2 - 1][-1]$ 没有相邻参考像素，则滤波后的值保持不变，如式(4.11)：

$$pF[nTbS \times 2 - 1][-1] = p[nTbS \times 2 - 1][-1] \quad (4.11)$$

以上各式中，开头和结尾的像素不滤波， $x=0, \dots, 62, y=0, \dots, 62$; “ \gg ”表示二进制数左移n位操作。

3. 强平滑滤波

对 32×32 的块，有可能使用强滤波，强滤波标记

`strong_intra_smoothing_enabled_flag` 在SPS中。如标记有效，需估计沿着参考像素集的局部活动性。如果活动性低于某个门限值，则用一个双线性插值滤波器对参考像素进行强平滑滤波。

为判断 32×32 的预测块是否要进行强平滑滤波，需考察是否同时满足下述条件：

- (1) 预测块尺寸为 32×32 ，即 $nTbS=32$
- (2) 允许强滤波标志`strong_intra_smoothing_enabled_flag=1`
- (3) $Abs(p[-1][-1] + p[nTbS \times 2 - 1][-1] - 2 \times p[nTbS - 1][-1]) < (1 \ll (BitDepthY - 5))$ (4.12)
- (4) $Abs(p[-1][-1] + p[-1][nTbS \times 2 - 1] - 2 \times p[-1][nTbS - 1]) < (1 \ll (BitDepthY - 5))$ (4.13)

如果同时满足上述条件，则需要进行强平滑滤波。其中条件(3)和(4)是判断参考像素活动性的不等式，不等式的左边借助参考像素的水平和垂直分枝中最末样点和中心样点上的2阶导数值来表示其局部活动性的大小。判断的阈值为 $1 \ll (BitDepthY - 5)$ ，其中`BitDepthY`表示亮度信号的比特深度，如8比特深度，则阈值 $(1 \ll (BitDepthY - 5)) = 8$ 。如果小于该阈值，则表明参考像素处在低纹理区域，强度为常数或接近线性变化。对这些块，参考像素值被最末像素值和中心像素值之间的线性内插取代，如图4.8(b)所示，见下式：

$$p[-1][y] - [(63-y)p[-1][-1] + (y+1)p[-1][63] + 32] \gg 6 \quad (4.14)$$

$$p[x][-1] = [(63-x)p[-1][-1] + (x+1)p[63][-1] + 32] \gg 6 \quad (4.15)$$

其中， $y=0, \dots, 62, x=0, \dots, 62$ 。

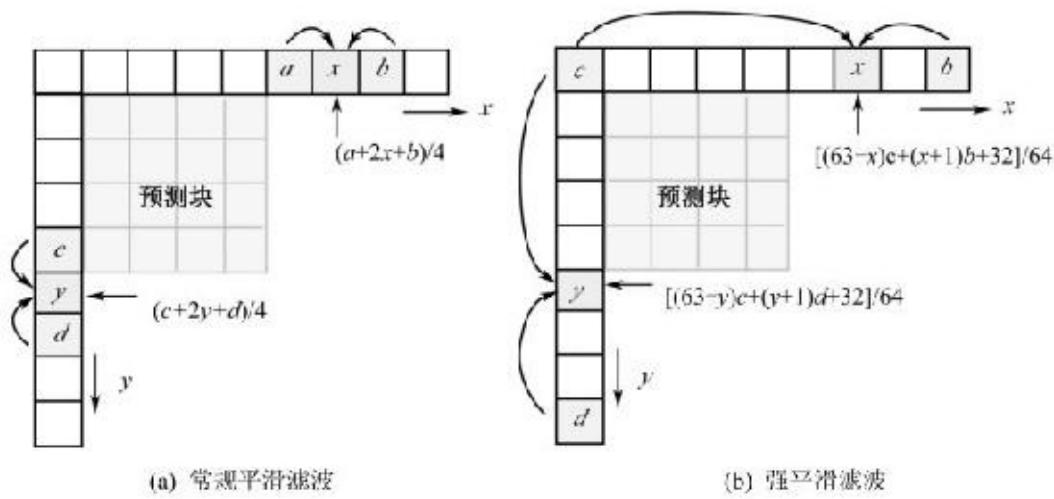


图4.8 参考像素滤波模式

这就是强滤波，通过用沿着每个方向的线性内插值取代真实样点值，取消了小的局部变化。这样的变化尽管很小，但在 32×32 块的大样点区域上进行帧内预测，还是可以产生可见的结构性失真的。

4.4.3 计算预测值

HEVC在帧内预测计算中主要包括角度类预测算法和非角度类预测算法，非角度类中又包含直流（DC）预测和平面（Planar）预测两类。

1. 平面（Planar）模式的预测值

Intra_Planar模式在H.264/AVC的 16×16 的宏块中也有类似的Plane定义，但HEVC解决了H.264/AVC中Plane模式容易在边缘造成不连续性的问题，同时还支持所有的块尺寸。

平面模式适合对图像中像素值呈渐变趋势的区域进行预测。当预测模式为平面预测时，如图4.9所示，预测块内像素p的预测值由4个参考像素来决定：该像素在水平方向上的左侧参考像素b和右上方参考像素a，该像素在垂直方向上的上方参考像素d和左下角参考像素c。对于不同位置的像素，b和d的值随之而变，但a和c的值不变，由这4个参考值加权平均得到，如式（4.16）所示。

$$\text{predSamples}[x][y] = ((nTbS-1-x) \times p[-1][y]) + (x+1) \times p[nTbS][y] + (nTbS-1-y) \times p[x][-1] \\ + (y+1) \times p[-1][nTbS+y] \gg (\text{Log2}(nTbS)+1) \quad (4.16)$$

上式中参考像素为 $p[x][y]$, $a=p[nTbS][-1]$, $b=p[-1][y]$, $c=p[-1][nTbS]$, $d=p[x][-1]$, $[x][y]$ 点的预测值为 $\text{predSamples}[x][y]$, x 为预测点距左侧参考列的距离, y 为预测点距上面参考行的距离。从式(4.16)还可以看出,经过平面预测模式预测后PU块的预测值,从左到右、从上到下呈渐变的趋势。

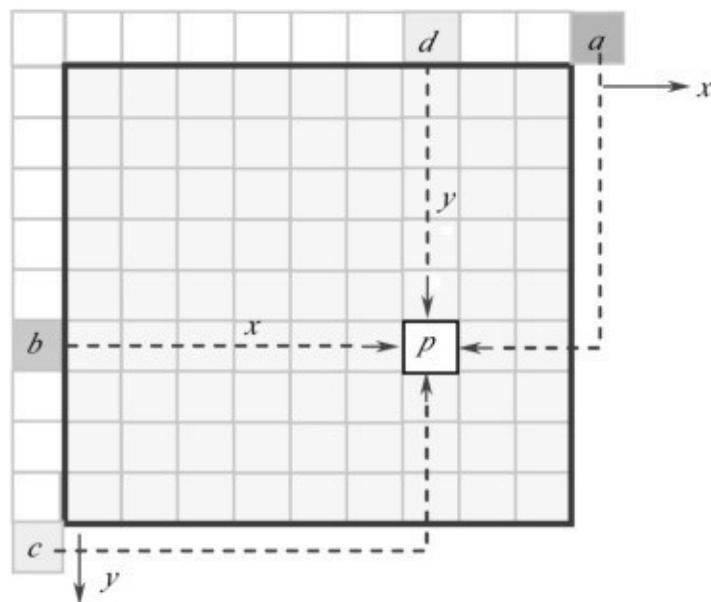


图4.9 Planar预测示意

2. 直流 (DC) 模式的预测值

HEVC中，当帧内预测模式为DC模式时，则预测值由左方参考像素和上方参考像素进行适当的平均而得。首先计算所有参考像素的平均值dcVal 作为基本的直流量，如式 (4.17) 所示：

$$dcVal = \left(\sum_{x'=0}^{nTbS-1} p[x'][-1] + \sum_{y'=0}^{nTbS-1} p[-1][y'] \right) \gg (\log 2nTbS + 1) \quad (4.17)$$

对于DC模式帧内预测块，块内像素的预测值都等于该直流量dcVal，即

$$predSamples[x][y] = dcVal, \quad x, y = 0, \dots, nTbS-1 \quad (4.18)$$

但对于亮度预测块，为改进边缘的过渡特性，块内紧邻参考像素的边缘像素点需要和参考像素进行加权处理，即

$$\text{predSamples}[0][0] = (p[-1][0] + 2 \times \text{dcVal} - p[0][-1] + 2) \gg 2 \quad (4.19)$$

$$\text{predSamples}[x][0] = (p[x][-1] + 3 \times \text{dcVal} + 2) \gg 2, \quad x = 1, \dots, n_{\text{TbS}} - 1 \quad (4.20)$$

$$\text{predSamples}[0][y] = (p[-1][y] + 3 \times \text{dcVal} + 2) \gg 2, \quad y = 1, \dots, n_{\text{TbS}} - 1 \quad (4.21)$$

对于相应的色度块预测，边缘的加权处理可省略。

3.角度 (Angular) 模式的预测值

相比H.264/AVC支持8个预测方向，HEVC由于TB尺寸的增加和可变，支持总数为33个预测方向，表示为Intra_Angular[k]，其中 k 表示模式编号，从2 ~ 34。预测角度设计的倾向是为接近水平、接近垂直的角度提供更密集的覆盖。

当预测模式为角度 (方向) 预测模式时，左侧角度模式预测值和上方角度模式预测值的计算方法类似。

从图4.6和表4.1可以看出，对于模式18 ~ 34，各个模式的格点 (角度) 分布是不均匀的，越是靠近垂直方向，分布越是细密，越是靠近对角方向分布越是稀疏。这种模式设计的根源是图像中反映的自然界中景物的垂直

运动或边缘占很高的比例。水平方向预测模式的分布特性也是如此。在帧内预测的33种模式中，我们称2~18为水平方向类预测模式，因为这些方向中水平或接近水平的方向的预测占多数。根据类似的道理，称18~34为垂直方向类预测模式。

现以垂直方向类的上方角度预测模式（predModeIntra=18,...,34）的帧内预测值计算为例进行简单介绍。对某一角度模式predModeIntra，尺寸为nTbS预测块中某一位置像素p[x][y]的帧内预测过程可分为3个阶段：先获取该预测块的参考像素数组ref[x]，接着求出该角度模式所对应的索引 iIdx 和乘法因子 iFact，最后在此基础上求出不同位置待编码像素的预测值 predSamples[x][y]。

（1）获得参考像素数组ref[x]

在预测模式号大于等于18的情况下，获得参考像素数组ref[x]的过程如图4.10所示。ref[x]的定义如下：

$$\text{ref}[x] = p[-1+x][-1] \quad (4.22)$$

其中， $x=0,\dots,n_{\text{TbS}}$ 。

由图4.10可以看出，如果 $\text{intraPredAngle} < 0$ ，并且
 $(nTbS \times \text{intraPredAngle}) \gg 5 < -1$ ，上方的参考像素序列就要进行扩
充：

$$\text{ref}[x] = p[-1][-1 + ((x \times \text{invAngle} + 128) \gg 8)] \quad (4.23)$$

其中， $x = -1, \dots, (nTbS \times \text{intraPredAngle}) \gg 5$ 。

否则， $\text{ref}[x] = p[-1+x][-1]$ ，(4.24)

其中， $x = nTbS + 1, \dots, 2 \times nTbS$ ，式 (4.23) 反角度参数 invAngle 可见表
4.3。

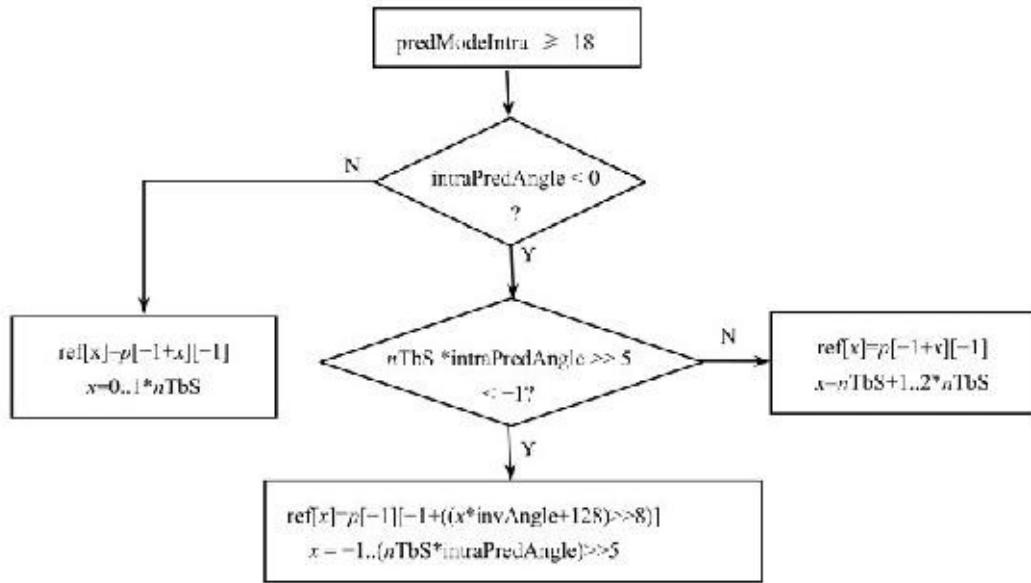


图4.10 角度预测中的ref[x]的确定

反角度参数 invAngle 的设置是为了消除帧内角度预测模式在计算预测值时麻烦的除法运算，而用简单的查表来替代。表中给出角度模式11~26的 invAngle 的值，根据对称的性质，模式2~10和模式27~34的invAngle的值是和表中的模式11~26对应相同的，只是符号相反（见表4.3）。

表4.3 反角度参数invAngle的值

| PredModeIntra | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 |
|---------------|-------|-------|------|------|------|-------|-------|------|
| invAngle | -4096 | -1638 | -910 | -630 | -482 | -390 | -315 | -256 |
| predModeIntra | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 26 |
| invAngle | -315 | -390 | -482 | -630 | -910 | -1638 | -4096 | - |

帧内角度模式预测的基本原理是通过当前像素沿某个预测方向在参考像素集上投影，“投中”的那个参考像素就是当前像素的预测值。在上面参

考像素组的获得过程中，已经可以看到，某些预测像素、某些预测角度，很多情况下并不能投中某个参考像素，原因可能是该方向上没有参考像素，也可能是投中在两个像素之间。而且对某个像素的预测有时既需要尝试投影上方参考像素集，也需要尝试投影左方参考像素集，这样计算的复杂度就会很高。因此，HEVC 设计了将上方和左方两个参考像素集转换为一个一维的参考像素集。仍然以角度模式18 ~ 34为例，组成一维参考像素集的方法就是在上方参考像素集的基础上沿着 x轴负方向左方延伸，将原来在预测块左方的参考像素值按照一定的规则投射到延伸的参考像素集上。例如，一个 4×4 块的参考像素的一维延伸如图4.11所示。

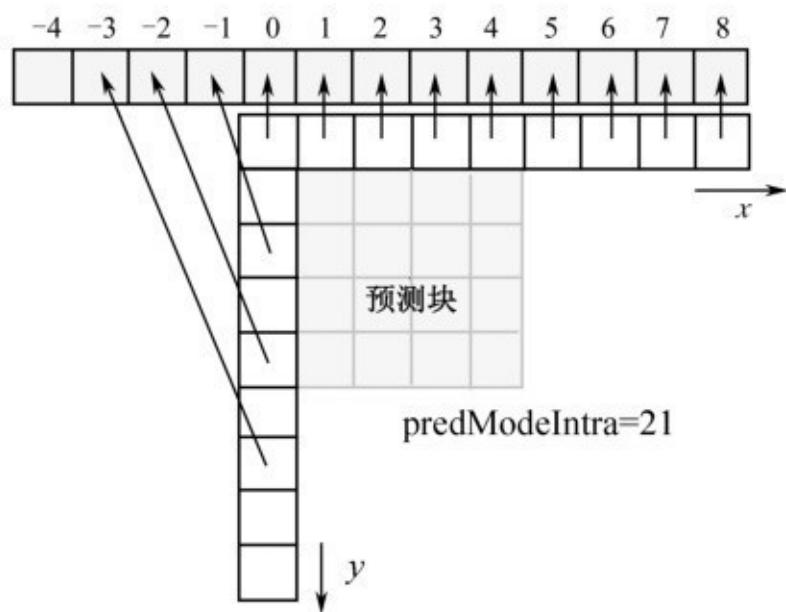


图4.11 一维参考像素集形成一例

设块尺寸 $nTbS=4$ ，角度模式为21，上方参考像素为 $p[x][-1],x=-1,\dots,2\times nTbS$ ，左方参考像素为 $p[-1][y],y=-1,\dots,2\times nTbS$ ，一维参考像素 $ref[x],x=-nTbS,\dots,2\times nTbS$ ，

对于 $x \geq 0$ ，则有： $ref[x]=p[-1+x][-1],x=0,\dots,2\times nTbS$ 。

对于 $x < 0$ ，则有： $ref[x]=p[-1][-1+Round[32\times x/intraPredAngle[k]]],x=-1,\dots,-nTbS$ 。

如角度模式 $k=21$ ，由表4.1可得 $intraPredAngle[21]=-17$ ，具体计算一维参考像素集：

对 $x \geq 0, ref[0]=p[-1][-1], ref[1]=p[0][-1], \dots, ref[8]=p[7][-1]$ ；

对 $x < 0, ref[-1]=p[-1][-1+Round(32\times (-1) / (-17))] = p[-1][1]$ ，类似可得，

$ref[-2]=p[-1][3], ref[-3]=p[-1][5], ref[-4]$ 不存在。

由此得到一维参考像素集，下面的预测就可参考这个一维数组进行。

(2) 参数iIdx和iFact的设置

设置不同模式所对应的索引iIdx和乘法因子iFact（权重参数），如式(4.25)和式(4.26)所示。其中intraPredAngle为预测模式对应的“角度”，这个“角度”不是用普通的角度单位来标识的，而是用该方向和水平（或垂直）方向偏移多少小格来表示的。例如，在图4.6中，Intra_Angular[31]模式的“角度” $intraPredAngle=17$ 。

$$iIdx = ((y + 1) \times intraPredAngle) \gg 5 \quad (4.25)$$

$$iFact = ((y + 1) \times intraPredAngle) \& 31 \quad (4.26)$$

其中“`&`”为二进制数的“与”操作。

(3) 预测值的计算

预测值计算的过程如图4.12所示。

当`iFact`不等于0时，预测值`predSamples[x][y]`由式(4.27)计算出：

$$\text{predSamples}[x][y] = ((32 - \text{iFact})$$

$$\times \text{ref}[x + \text{iIdx} + 1] + \text{iFact} \times \text{ref}[x + \text{iIdx} + 2] + 16) \gg 5 \quad (4.27)$$

当`iFact`等于0时，预测值`predSamples[x][y]`由式(4.28)计算出：

$$\text{predSamples}[x][y] = \text{ref}[x + \text{iIdx} + 1] \quad (4.28)$$

如果块尺寸小于 32×32 ，且预测模式为垂直预测(`predModeIntra=26`)时，那么预测值`predSamples[x][y]`由式(4.29)计算出：

$$\text{predSamples}[x][y] = \text{Clip}_{LY}(p[x][-1] + ((p[-1][y] - p[-1][-1]) \gg 1)) \quad (4.29)$$

其中 $x=0, y=0, \dots, n_{TbS}$ 。

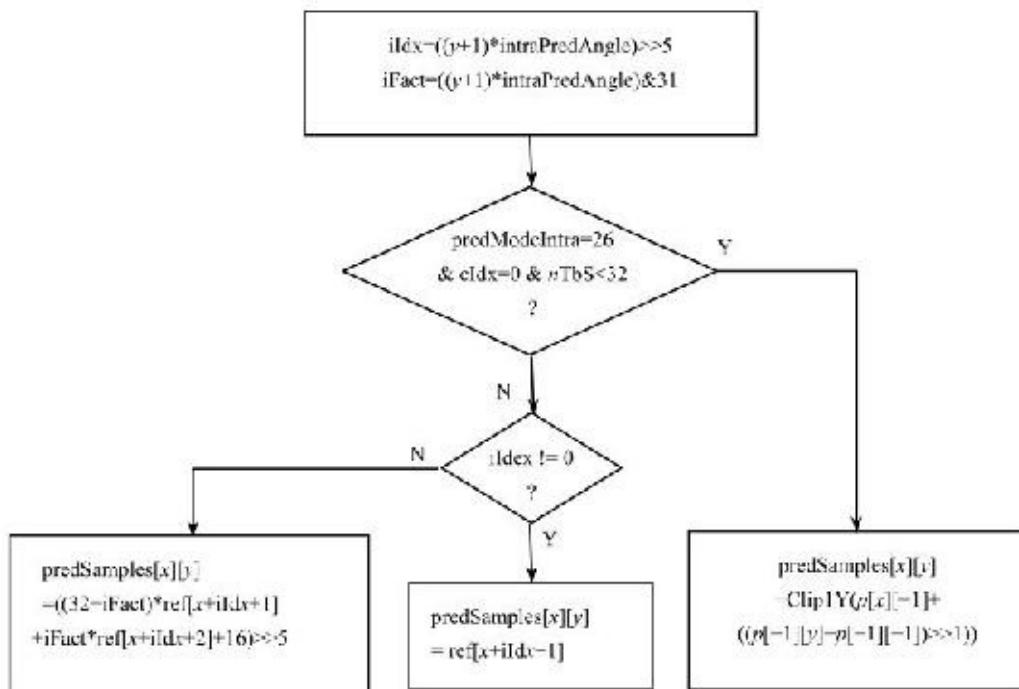


图4.12 角度预测模式预测值的确定

Intra_Angular模式的预测处理在穿越所有块尺寸和预测方向时是一致的，而H.264/AVC则是使用不同的方法来支持 4×4 、 8×8 、 16×16 尺寸的块。HEVC这种设计的一致性是特别理想的，因为较之H.264/AVC,HEVC支持更多的PB尺寸的变化和预测方向数量的显著增加。

(4) 预测值计算原理

为了更好地理解上述的参考值确定和预测值获得的原理，这里简要介绍在帧内预测模式为26~34的情况下， (x,y) 点的预测值 $P(x,y)$ 的计算方法。如图4.13所示，对于每个预测方向，根据比例关系，可以利用式(4.30)计算出预测点 $P(x,y)$ 在上方参考像素行中的投影点位置，得到投影点横坐标相对于 $P(x,y)$ 横坐标的位移 c_x ：

$$c_x / y = d / 32 \quad (4.30)$$

其中， c_x 表示待预测点 (x,y) 的横坐标和点 (x,y) 沿着预测方向投影到上参考像素行的横坐标之差，也就是HEVC定义的偏移索引iIdx; d 表示预测模式方向和垂直方向的偏移距离(格数)，范围为-32~+32，可由表4.1查得，32是中心点到上方参考像素行的距离。由式(4.30)可以定义偏移

索引iIndex和权重因子iFact如下：

$$\text{Index} = c_x = (y \cdot d) / 32 \quad (4.31)$$

$$\text{iFact} = w = (y \cdot d) \& 31 \quad (4.32)$$

然后参见与图4.13对应的实际像素点阵图4.14 (a)，有

$$i = x + c_x \quad (4.33)$$

其中， i 为过 (x, y) 点的预测方向线和参考像素行的交点， c_x 表示 x 点到 i 点的距离。式 (4.31) 中 $y \cdot d$ 除以 32 后使 c_x 成为归一化长度单位，和 x 的坐标定义相一致。 $y \cdot d$ 也表示 x 点到 i 点的距离，是以 32 格为单位，和“31”进行“与”操作后，实际上就去掉了大于 32 的部分，只剩下小于 32 的部分，即表示 $R_{i,0}$ 点到 i 点的距离，也就是权重 w 。

如图 4.14 (a) 所示，其中 i 点可以看成 (x, y) 点沿着预测方向向参考像素行的投影点， $P(x, y)$ 的预测值实际上就等于这个投影点的插值，插值由 i 点左右两个相邻像素 $R_{i,0}$ 和 $R_{i+1,0}$ 的线性内插而得，或者说加权求和而得。靠近 i 点的参考像素权重大，远离 i 点的参考像素权重小。

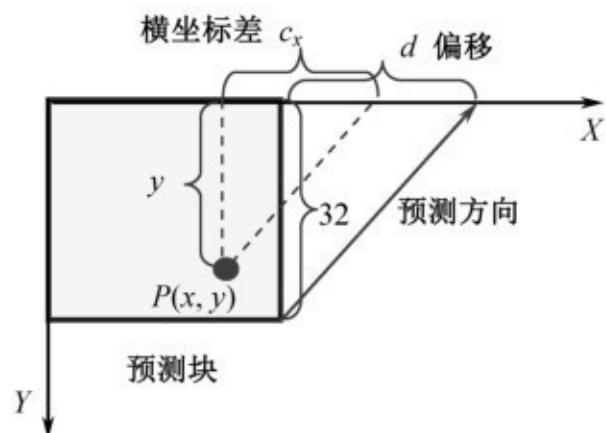


图 4.13 坐标差的计算

将图4.14 (a) 的局部放大为图4.14 (b) , 相邻两参考像素的距离为32个单位 , $R_{i,0}$ 到 i 的距离为 w , $R_{i+1,0}$ 到 i 的距离为 $32-w$ 。因此 (x,y) 点的预测值 $P(x,y)$ 就等于 i 点的插值 , 如式 (4.34) 所示 , 本质上和式 (4.27) 一样。

$$P(x,y)=((32-w_y)R_{i,0} + w_y R_{i+1,0}) \gg 5 \quad (4.34)$$

上式中的“ $\gg 5$ ”表示右移5位 , 即除以32 , 将放大的32倍的权重归一化。HEVC的帧内预测块中所有的像素都是同一种预测模式 , 因此其他像素也都采用上述的方法计算得到。这里虽然是针对帧内预测模式为26 ~ 34的情况分析的 , 其他2 ~ 25模式的预测值计算也和上述方法类似。

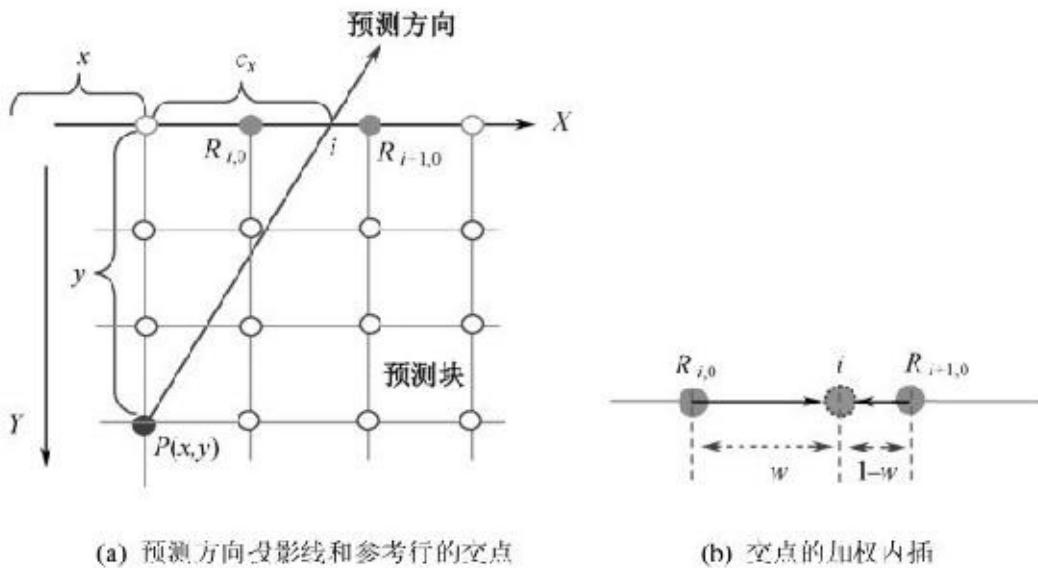


图4.14 参考值的插值计算

4.4.4 边界值的平滑

为除去沿块边界的不连续性，在3类模式中，Intra_DC（模式1）、Intra_Angular[10]（水平方向）和Intra_Angular[26]（垂直方向），当编码PB尺寸小于 32×32 时，PB内边界的像素值（紧贴参考像素的那一行和一列像素）则用滤波后的值取代。

对于 Intra_DC 模式，PB 中的第一行 $p[x][0]$ 和第一列 $p[0][y]$ 的像素被一个2抽头滤波器 $(3,1)/4$ 的输出值所替代，这个滤波器的输入就是它们的原始值和邻近参考像素值。

在水平预测Intra_Angular[10]中，PB的第一列 $p[0][y]$ 边界像素被修改，以使得它们邻近的参考像素 $p[-1][y]$ 和左上角参考像素 $p[-1][-1]$ 之差的一半

被加上去。这使得当垂直方向存在较大变化时，预测信号更平滑。

在垂直预测Intra_Angular[26]中，同样的处理用于第一行像素 $p[x][0]$ 。

4.4.5 模式信息的编码

一个亮度预测块所采用的帧内编码模式信息也需要传送的解码端，同样要求对这些模式信息进行高效编码。HEVC一共定义了35种亮度块的帧内预测模式，相比于H.264/AVC增加了许多，H.264/AVC那种只考虑1种“最可能模式”的编码方法不再适用于HEVC。

1. 亮度块的模式信息

在HEVC中对当前亮度预测块定义了3种最有可能预测模式（Most Probable Modes, MPM）：PMP[1]、PMP[2]和PMP[3]。编码时，先确定所选预测模式是否属于这3种MPM，如果是，则编码其在这3种MPM中的索引编号，只需将MPM的索引传给解码器。否则，用5bit的固定长码字来表示当前亮度预测块在剩余的32种模式中的索引，并传送给解码器。

在3种最可能的模式中，前2种模式由上边和左边已编码的亮度PB的帧内预测模式来初始决定（只要这些PB是可取的而且使用帧内预测模式编码）。任何不可取的预测模式则认为是Intra_DC。

如图4.15所示，设A表示当前预测块左边的一个已编码的预测块，编码模式号为 m_A ；B表示当前预测块上边的一个已编码的预测块，编码模式号为 m_B 。则3种MPM可根据A和B的帧内预测模式进行设置：

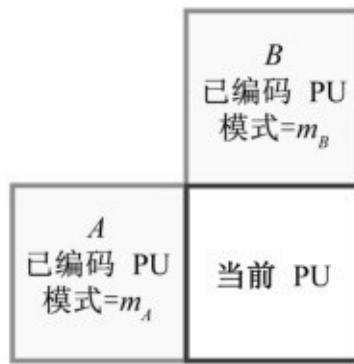


图4.15 最可能预测模式的参考块

(1) 若A或B不存在或不是帧内预测的预测块，则 m_A 或 m_B 用模式Intra_DC代替。为避免存储上一行中所有编码树块(CTB)的编码信息，当B不属于当前预测块的同一个编码树块时， m_B 也用模式Intra_DC代替。

(2) 若A和B存在，且 $m_A=m_B$ ，则执行以下操作：

(a) 如果 $m_A < 2$ ，为非方向模式，3种MPM为

$$MPM[0]=\text{Intra_Planar}$$

$$MPM[1]=\text{Intra_DC}$$

$$MPM[2]=\text{Intra_Angular}[26]$$

(b) 如果 $m_A > 2$ ，为方向性模式，3种MPM依次为 m_A 及 m_A 相邻的两个方向预测模式：

$$MPM[0]=m_A$$

$$MPM[1]=2+((m_A+29)\%32)$$

$$MPM[2]=2+((m_A-2+1)\%32)$$

式中“%32”为模32运算。

(3) 若A和B存在，且 $m_A \neq m_B$ ，则前两个模式为

$MPM[0]=m_A, MPM[1]=m_B$ ，第3个最可能模式 $MPM[2]$ 在不和前2个模式相同的情况下依次被置为Intra_Planar、Intra_DC 或Intra_Angular[26]。具体操作顺序如下：

(a) $MPM[0]=m_A, MPM[1]=m_B$ ；

(b) 如果 m_A 和 m_B 都不等于Intra_Pnalar，那么 $MPM[2]=Intra_Planar$ ；

(c) 否则，如果 m_A 和 m_B 都不等于Intra_DC，那么

$MPM[2]=Intra_DC$ ；

(d) 否则 $MPM[2]=Intra_ Angular[26]$ 。

2. 色度块的模式信息

对于色度块，HEVC定义了5 种模式：Intra_Planar（平面）、Intra_Angular[26]（垂直）、Intra_Angular[10]（水平）、Intra_DC和Intra_Derived。编码时则直接对模式编号（0 ~ 4）进行编码。Intra_Derived 表示和亮度块所选的预测模式一样。当亮度块所选的预测模式等于前4种模式中的一种时，则该模式用Intra_Angular[34]代替。

本章参考文献

[1]Jani Lainema,Frank Bossen,Woo-Jin Han,et al.Intra coding of the HEVC standard[J].IEEE Trans.on Circuits and Systems for Video Technology,Dec.2012,22 (12) : 1792-1801.

[2]High Efficiency Video Coding[S].ITU-T Rec.H.265 and ISO/IEC 23008-2,ITU-T and ISO/IEC JCT-VC,Mach 2013 (v.1) ,Oct.2014 (v.2) ,Apr.2015 (v.3) .

[3]Bossen,F.,Tk Tan,J.Takiue.Simplified angular intra prediction[R].2nd Meeting: Joint Collaborative Team on Video Coding (JCTVC) of ITU-T SG 16 WP 3 and ISO/IEC JTC 1/SC 29/WG,Doc.JCTVCB093,Geneva,2010.

[4]Sze,V.,Budagavi,M..High throughput CABAC entropy coding in HEVC[J].IEEE Trans.on Circuits and Systems for Video Technology,Dec.2012,22 (12) : 1778-1791.

[5]Bossen,F..Common test conditions and software reference configurations[R].11th Meeting:Joint Collaborative Team on Video Coding (JCT-VC) of ITU-T SG 16 WP 3 and ISO/IEC JTC 1/SC29/WG,Doc.JCTVC-K1100,Shanghai,2012.

[6]Information technology—Coding of audio-visual objects—Part 10: Advanced video coding (AVC) [S].ISO/IEC 14496—par10,2012.

[7]Mathias Wien.High Efficiency Video Coding: Coding Tools and Specification[M].Springer Verlag Berlin Heidelberg,2015.

[8]Vivienne Sze,Madhukar Budagavi,Gary J.Sullivan.High Efficiency Video Coding (HEVC) :Algorithms and Architectures[M].Switzerland Springer International Publishing,2014.

[9]Biao Min,Ray C.C.Cheung.A fast CU size decision algorithm for the HEVC intra encoder[J].IEEE Trans.On Circuits and Systems for Video Technology.May 2015,25 (5) : 892-896.

[10]Jong-Hyeok Lee,Kyung-Soon Jang,Byung-Gyu Kim,et al.Fast intra mode decision algorithm based on local binary patterns in High Efficiency Video Coding (HEVC) [C].2015 IEEE International Conference on Consumer Electronics (ICCE) ,270-272.

[11]M.P.Sharabayko,Oleg G.Ponomarev,Roman I.Chernyak.Intra Compression Efficiency in VP9 and HEVC[J].Applied Mathematical Sciences,2013,7 (137) : 6803-6824.

第5章 HEVC的帧间预测

帧间预测技术利用视频中邻近帧之间的时域相关性，使用先前已编码的重建帧作为参考帧，通过运动估计和运动补偿对当前编码帧进行预测，从而去除视频中的时间冗余信息。HEVC 的帧间预测编码总体上和 H.264/AVC相似，但在不少方面进行了改进，显著提高了编码效率。本章先简要介绍帧间预测编码的基本原理，包括H.264/AVC的帧间预测要点，然后重点叙述HEVC帧间预测编码的改进之处，特别是关于运动信息的高效处理技术。

5.1 帧间预测编码

一般而言，帧间预测编码编码效率比帧内预测更高，其原因在于视频的帧间相关性很强，信息的冗余度很高。例如，有统计结果表明，对内容变化不快的256级灰度视频序列，相邻帧之间的帧间差绝对值超过3的像素平均不到一帧像素的4%；对内容变化剧烈的256级灰度视频序列，帧间差绝对值超过6的像素平均不到一帧像素的7.5%。视频时间信息的高冗余特点告诉我们，只要充分去除帧间相关性，就有可能获得相当高的视频压缩率。

去除帧间相关性的有效方法就是帧间预测。简单的相邻帧对应像素之间的预测精度并不高，这是因为视频场景中存在的物体运动破坏了这种对应关系，因此需要引入对运动物体的特别处理，使得无论在哪一帧，对运动物体的预测都是针对运动物体本身对应点之间的预测，才可大大提高预测精度。这就要求我们掌握运动物体的运动情况，即运动估计（ME），在运动估计的基础上对预测作出修正，即运动补偿（MC）。

5.1.1 帧间预测方式

1. 单向预测

帧间预测编码的原理可用图5.1来简单说明。当前帧图像 $f_t(x,y)$ 与预测图像 $\hat{f}_t(x,y)$ 相减后的预测误差（或残差）为 $r_t(x,y)$ ，经变换、量化后输出 $e'_t(x,y)$ ，一边传送到信道，一边经反量化、反变换后形成 $r'_t(x,y)$ 。预测图像 $\hat{f}_{t-1}(x,y)$ 与 $r'_t(x,y)$ 相加，得到重建图像 $f'_t(x,y)$ ，当不计量化失真时， $f'_t(x,y)$ 和当前帧 $f_t(x,y)$ 是相同的。

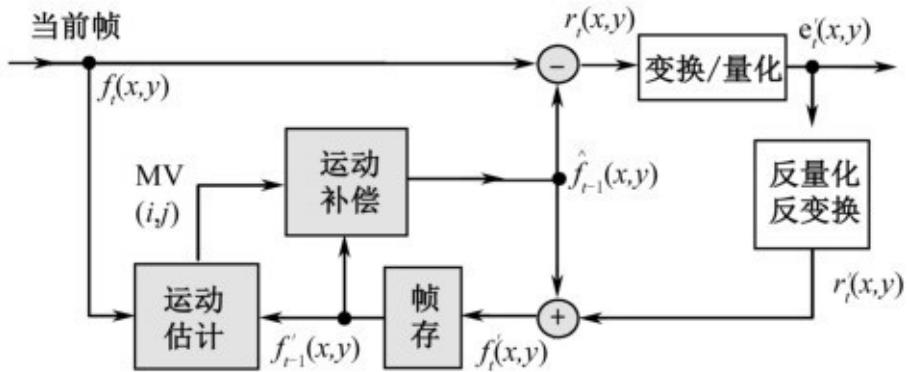


图5.1 帧间预测编码原理

把当前帧 $f_t(x,y)$ 与帧存储器输出的前一重建帧 $f'_{t-1}(x,y)$ （也称参考帧）同时输入运动估计模块，经搜索、比较后得到运动矢量（MV）。将此运动矢量输入到运动补偿模块，“补偿”输入的重建参考帧 $f'_{t-1}(x,y)$ ，得到当前编码帧 $f_t(x,y)$ 的预测帧 $\hat{f}_{t-1}(x,y)$ 。预测帧 $\hat{f}_{t-1}(x,y)$ 和当前帧 $f_t(x,y)$ 之差即为帧间预测误差 $r_t(x,y)$ 。运动矢量也要传到解码端。解码器利用反量化、反变换后的预测误差 $r'_t(x,y)$ 和经运动补偿后的重建帧 $\hat{f}_{t-1}(x,y)$ 相加就可近似得到当前重建帧 $f'_t(x,y)$ 。预测帧并不需要传给解码器，因为解码器可以用和编码器相同的方法得到预测帧 $\hat{f}_{t-1}(x,y)$ 。

与帧内预测的机理类似，帧间预测的关键也是要千方百计地提高预测的准确度，降低预测误差，预测误差越小，编码效率越高。在这里，运动估计和运动补偿的基本作用是使得预测帧 $\hat{f}_t(x,y)$ 和当前帧 $f_t(x,y)$ 更加接近，使预测误差更加接近于0，提高编码效率。

利用运动矢量将前一重建帧“补偿”后成为预测帧的方法可用下式表

示：

$$\hat{f}_t(x, y) = f'_{t-1}(x + i, y + j) \quad (5.1)$$

其中， (i, j) 为运动矢量， i 为水平方向的分量， j 为垂直方向的分量。由于这里的帧间预测只涉及前一帧图像，因而又称为“前向预测”或“单向预测”。

2. 双向预测和多帧预测

前向帧间预测是最常见的一种帧间预测，此外还有后向预测、双向预测、多帧预测等方法可应用于不同的场合。如图5.2所示，横坐标“帧号”表示时间的进展，利用“历史帧”（发生在当前帧之前）对当前帧进行预测就是上述的前向预测，利用“将来帧”（发生在当前帧之后）对当前帧进行预测就是后向预测，这两种情况统称为单向预测。如果不仅利用前一帧图像预测当前块，还同时利用后一帧图像进行预测，这就是双向预测，其预测公式为：

$$\hat{f}_t(x, y) = \alpha f_{t-1}(x+i, y+j) + \beta f_{t+1}(x+k, y+l) \quad (5.2)$$

其中， $\hat{f}_t(x, y)$ 是预测帧， (i, j) 为从前面第 $t-1$ 帧到当前第 t 帧的前向运动矢量， (k, l) 为从后面第 $t+1$ 帧到当前第 t 帧的后向运动矢量， α 和 β 分别为前向和后向预测系数，可按最佳预测公式确定。

由于双向预测可参考的信息比单向预测要多，其预测准确度有可能提高，从而编码效率一般也高于单向预测。但双向预测需要涉及将来帧，因此在编码时需要存储编码帧前后的图像，必然会带来编码时延和存储容量的加大，因此，这种方式一般不适用于视频会话类的实时通信应用，而广泛应用于广播电视、视频存储、网络视频等领域。

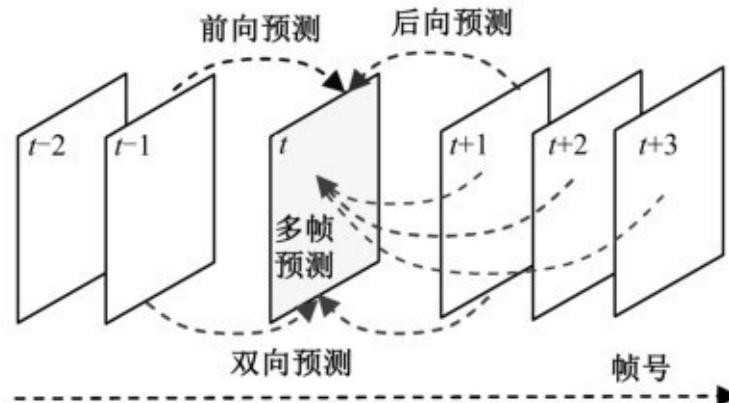


图5.2 4种帧间预测模式

为了进一步提高编码效率，从H.264/AVC起，引入了多参考帧预测技术，如图5.2所示，即每个方向允许使用若干参考帧，最多可达15帧，包括单向多帧和双向多帧。多帧参考是指在运动估计过程中，在某些情况下可采用多个参考帧来提高预测精度。为了适应多参考帧预测的要求，需要在编解码端建立一个可存储多个参考帧的缓存，当前的待编码块可以在缓存内的所有重建帧中寻找最优的匹配块进行运动补偿，以便更好地去除时域的冗余度。

5.1.2 基于块的运动估计

在视频场景中一般会有一个或多个不同的运动物体，在图像中将不同的运动物体分割出来，并给每个物体指定一个运动矢量，即基于物体的运动估计，由于图像分割技术的不成熟，目前还难以做到。另外一种方法就

是为每个像素指定一个运动矢量，即基于像素的运动估计，这种方法虽然对任何类型图像都是适用的，精度也高，但它需要的计算量大，也不能保证所运动矢量的稳定性和唯一性。而且以像素为单位进行预测，除传送帧差外，还要为每个像素传送对应的运动矢量，编码效率显著下降。既然按照运动物体或按照每个像素进行运动估计都难以实现，这就导致了在这两种方法之间的一种折中处理，即基于图像块的运动估计的方法。

基于块的运动估计将每一视频图像帧分成若干小块，设法搜索出每个图像块在邻近帧图像中的位置，并得出两者之间空间位置的相对偏移量，也就是运动矢量，搜寻运动矢量的过程也叫做运动估计。

利用运动矢量来进行运动补偿得到的帧间预测误差连同运动矢量共同发送到解码端，在解码端按照运动矢量指明的位置，从已经解码的邻近参考帧图像中找到相应的图像块，和预测误差相加后就得到了在当前帧的重建块数据。通过运动估计可以去除帧间冗余度，使得视频传输的比特数大为减少，因此，准确的运动估计是视频压缩处理中的一个重要的编码工具。

1. 块匹配运动估计

对于包含运动物体的场景，一般可认为两帧之间的物体运动一般是刚体的平移，位移量不太大。在此前提下，实际中普遍采用的方法是把一个图像帧分成多个块，使原本针对运动物体寻找运动矢量的复杂问题简化成为每个块寻找在前后帧之间的运动位移。例如将图像分成若干个 $m \times n$ 块（如 16×16 、 8×16 等），为每一个块寻找一个运动矢量，用于帧间预测编码的运动补偿。

这里以前向运动估计为例介绍块匹配运动估计的机理：设正在编码的第 t 帧图像为当前帧 $f_t(x, y)$ ，第 $t-1$ 帧图像为参考帧 $f_{t-1}(x, y)$ ，如图5.3所示。当一个物体从第 $t-1$ 帧运动到第 t 帧时，从理论上说，在当前帧中的某一正在编码的当前块，一定可以在参考帧中搜索到和当前块一样的块，称其为匹配块。运动估计就是搜索匹配块，运动矢量则表示匹配块和当前块

之间的位移。从视觉的角度可以看到运动物体在二维屏幕上从 $t-1$ 帧到 t 帧有了一个运动矢量 V ，它是一个二维矢量，包括两个分量，一个是水平方向的分量 v_x ，一个是垂直方向的分量 v_y ，即 $V = (v_x, v_y)$ 。

至于运动矢量的搜索方法、块匹配的准则已经在第1章中有所介绍，这里就不再赘述。

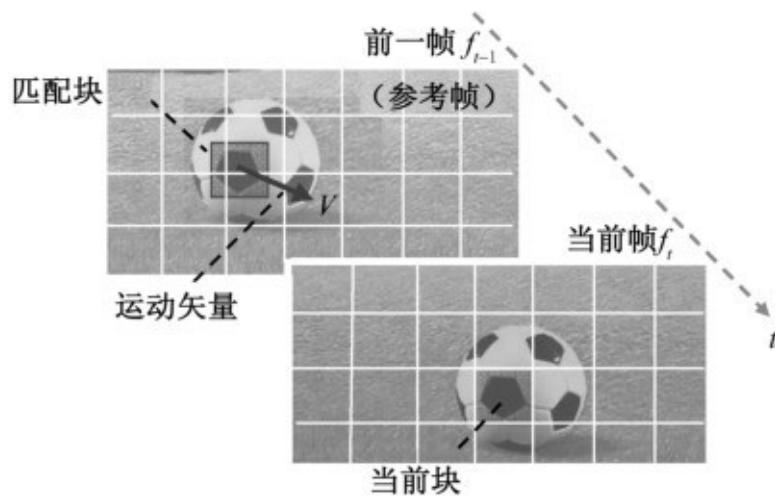


图5.3 基于块的运动估计

2.高精度运动矢量

由于自然物体运动的连续性，物体在相邻两帧之间的运动矢量不一定刚好是整数个像素单位。因此，为了提高运动矢量的精确程度，在近年的视频标准中对亮度的运动估计常采用 $1/2$ 、 $1/4$ 甚至 $1/8$ 像素精度的运动矢量。但在数字视频中并不存在分数像素处的样值，一般来说，为实现 $1/K$ 像素精度估计，必须将这些分数像素点的值近似内插出来，即对参考帧的行方向和列方向进行 K 倍内插。如 $K=2$ ，须在 $1/2$ 像素处进行插值，在插值后的图像中进行的搜索，称为半像素精度运动估计。实验证明，与整像素精度相比，半像素精度的搜索使运动估计的精度有很大提高，特别是对于低清晰度视频。在低噪声情况下，一般可采取 $1/4$ 或 $1/8$ 像素精度的运动矢量，而在高噪声情况下， $1/2$ 像素精度就够了，再提高精度受到噪声的干扰而导致收益并不明显。

5.1.3 运动矢量的预测

虽然运动估计的帧间预测大大压缩了原始视频的数据量，但仍然需要将运动矢量本身的数据传送到解码器，如果对每个块的运动矢量都进行编码，那么将花费相当数目的比特，特别是在选择小尺寸块时编码运动矢量的比特花费会显著增加。由于一个运动物体往往覆盖多个图像块，所以空间域相邻块的运动矢量具有很强的相关性；同时由于物体运动的连续性，运动矢量在时间域相邻帧之间也存在较强相关性。因此，与图像像素的预测编码道理一样，当前块的运动矢量可以根据先前已编码的空间邻近块或时间邻近块的运动矢量块进行预测。这样，只要将当前运动矢量和预测运动矢量之间的差值传输到解码端，就可以有效地压缩运动矢量的编码比特数。运动矢量预测主要可分为空域预测方式和时域预测方式，具体的预测方法较多，下面分别给出这两类方式中较为常见的一种。

1.空域预测方式

运动矢量空域预测方式中较为简单、常见的是中值预测方式，预测时

所参考的相邻块的关系参如图5.4所示。

如果当前块和邻近块的尺寸相同，如图5.4 (a) 所示，在同一帧中利用与当前块E相邻的左边块A（运动矢量为 MV_A ）、上边块B（运动矢量为 MV_B ）和右上方块C（运动矢量为 MV_C ）的运动矢量，取其中值来作为当前块的预测运动矢量（ MV_{pE} ）。

如果当前块和邻近块的尺寸不同，如图5.4 (b) 所示，E块的左侧多于一个块，那么选择最上方的块作为A,E块的上方多个块中选择最左侧的块作为B。

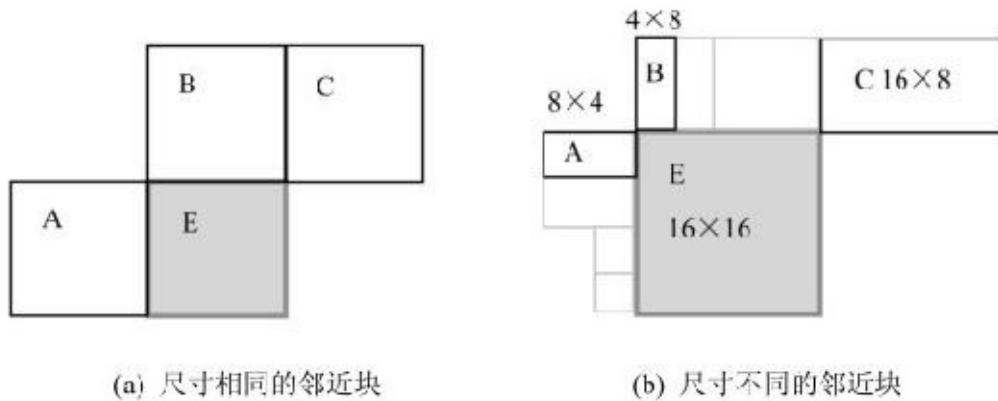


图5.4 运动矢量的中值预测

综合上述两种情况，可以得到统一中值预测公式，块E的中值运动矢量预测值为：

$$MV_{pE} = \text{Median}(MV_A, MV_B, MV_C) \quad (5.3)$$

其中，Median() 表示在括弧中取中间值的函数。如果E块的真实运动矢量为 MV_E ，则相应E块运动矢量的预测差值为：

$$MVD_E = MV_{pE} - MV_E \quad (5.4)$$

中值预测的性能随着预测块尺寸的减小而增加，这是因为当前块尺寸越小，相关性越强。

2.时域预测方式

时域运动矢量预测最简单的一种就是前帧预测，设当前块为E，取前帧中和E对应位置块的运动矢量 MV_{bE} 作为当前块运动矢量的预测值 MV_{pE} ，即

$$MV_{pE} = MV_{bE} \quad (5.5)$$

虽然这种方法比较简单，但效果并不理想。实际上，时域MV预测也可以采用多个邻近的MV线性组合的预测方式来提高预测精度。

5.2 H.264/AVC的帧间预测

H.264/AVC 的帧间预测采用的是对已编码视频帧进行基于块的运动补偿预测模式。但H.264/AVC的帧间预测块的尺寸灵活多样，运动矢量的精度提高到1/4像素，允许多参考帧参与预测等。HEVC几乎继承了全部这些特点，还在多个方面进行了改进和优化。

5.2.1 多模式宏块划分

如图5.5所示，每个 16×16 的宏块可以进一步分割为更小的块，分割方式由mb_type来表示，有3种方式分割：两个 16×8 的块，两个 8×16 子块，四个 8×8 块。其中， 8×8 的宏块分割又称子宏块（sub-MB），它还可以进一步分割，分割方式由sub_mb_type来表示，有3种方式分割：两个 4×8 块，两个 8×4 块，4个 4×4 块。

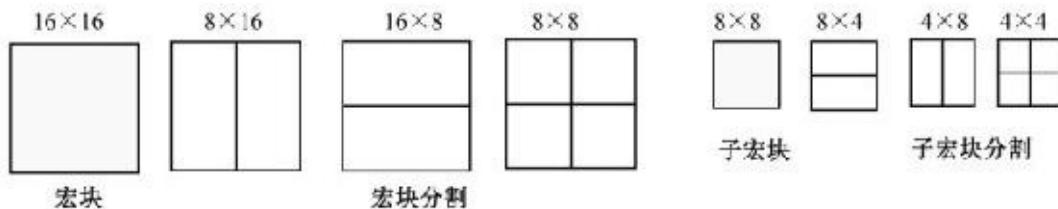


图5.5 宏块及子宏块分割

在帧间预测编码时，每个宏块或子宏块都需有一个运动矢量（MV），每个 MV 和分割方式信息都必须连同帧间预测残差数据被编码、传输。这样，对大的分割尺寸而言，MV 数据和分割方式数据只需少量的比特，但由于块尺寸较大，运动补偿后的预测残差数据的能量比较高，数据量较大；对小尺寸分割而言，由于能够获得更准确的 MV，运动补偿后的预测残差的能量比较低，但需要较多的比特编码MV数据和分割方式数据。由此可见，分割尺寸的选择影响了压缩性能，通常大的分割尺寸适合平坦区域，小的分割尺寸适合多细节区域。

对于常用的4:2:0数字视频，色度宏块（C_r和C_b）的分割方式和亮度宏块一样，只是尺寸在水平和垂直方向都减半。色度块的MV是通过相应亮度MV水平和垂直分量减半而得。

5.2.2 高精度运动估计

在H.264/AVC中，亮度MV的精度为1/4整亮度像素间隔，由于亮度和色度像素样点的几何比例，色度MV的精度为1/8整色度像素精度。1/4、1/8这些亚像素位置的亮度和色度像素并不存在于参考图像中，需利用邻近已编码像素点进行内插而得。

1. 亮度1/2像素内插

如图5.6所示，参考帧中已有的整像素点由小写字母标注，如a,b,c,d,...。为生成参考图像亮度的半像素点，可分两步插值生成：

(1) 首先考虑和两相邻整像素在一条水平线上并处于它们之间中点位置的半像素点，如x。这类像素的值可由一个对称的6抽头滤波器对水平线上的6个整数点内插得到，滤波系数为

(1/32,-5/32,20/32,20/32,-5/32,1/32), x类半像素点可由下式计算：

$$x = \text{round} ((a - 5b + 20c + 20d - 5e + f) / 32) \quad (5.6)$$

类似地，和两相邻整像素在一条垂直线上并处于它们之间中点位置的半像素点，如 y 。这类像素的值可由同样的6抽头滤波器对垂直线上的6个整数点内插得到。

(2) 还有一类半像素点，如 z ，和它在同一水平或垂直方向的直线上没有已知的整数点，但在 x 、 y 类半像素点内插完成后，这些 z 类半像素点可用在水平或垂直方向新插值出来的6个半像素点由同样的6抽头滤波器内插获得。

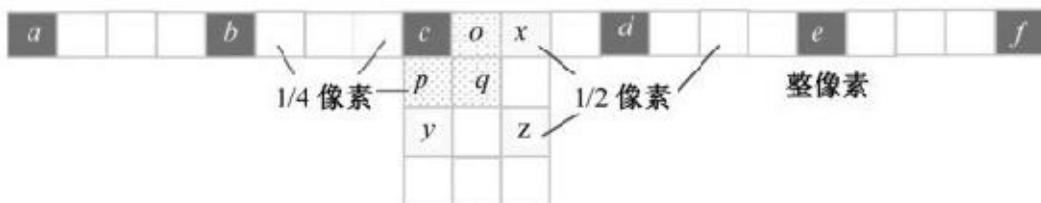


图5.6 亮度1/2和1/4像素内插

2.亮度1/4像素内插

仍然参考图5.6，半像素点计算出来以后，1/4像素点就可通过简单的线性内插得出。和已知（包括原有的整像素点和半像素点）在同一直线上的1/4像素点，如o、p等，可由邻近的2个像素内插而得，如o为

$$o = \text{round}((c+x)/2) \quad (5.7)$$

和已知像素点不在同一直线上的1/4像素点，如q等，可由一对对角半像素点线性内插得出，如q为

$$q = \text{round}((c+z)/2) \quad (5.8)$$

3. 色度1/8像素内插

由于色度整像素之间的距离是亮度的一倍，要达到和亮度同样精度的运动矢量，色度内插像素相的精度需要达到色度整像素的1/8，即常称色度像素需要1/8精度插值。如图5.7所示，色度1/8像素可通过相邻的4个整像素双线性内插得到，其中a为：

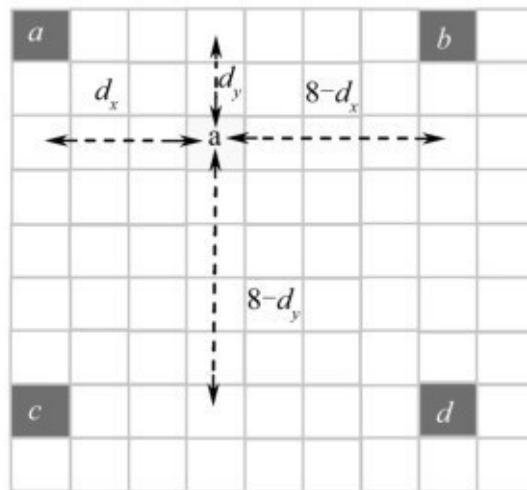


图5.7 色度1/8像素内插

$$a = \text{round}(((8-d_x)(8-d_y)a + d_x(8-d_y)b + (8-d_x)d_yc + d_xd_yd) / 64) \quad (5.9)$$

4.运动矢量的中值预测

MV预测有多种方法，H.264/AVC采用的是一种中值预测的方式，可参考前面的图5.4（a），当前编码宏块的运动矢量可以由该块的左边、上边和右上角相邻宏块的运动矢量的中值来预测。

只需要将运动矢量的预测误差（MVD）编码、传送到解码端。在解码端，采用和编码端相同方式形成预测值 MV_p ，并将它加到 MVD 上，就可还原得到 MV。可以跳过宏块，连 MVD也不需要，运动补偿宏块由 MV_p 直接生成。

5.2.3 双向预测条

双向预测条（B slice）中的帧间编码宏块的每个子块都是由一帧或两帧参考图像预测而得。该参考图像位于当前图像的前面或后面。参考图像存储于编解码器的存储区中。

1.参考帧列表

在H.264/AVC中设置了两个已编码帧列表：List 0和List 1，这两个列表都可包含前向和后向的已编码的参考帧（按显示顺序排列）。如当前帧有3幅前向参考帧和3幅后向参考帧，按照自然图像顺序编号（Picture Order Count, POC）为97、98、99、100（当前帧）、101、102、103。这些帧在List0和List1中的索引（index）号排序不同，在List0中，最靠近当前帧的前面的一帧标志为index 0，再前一帧为index1，……，前向帧排完后接着是按顺序排列的后向帧。List1中和当前帧最靠近后向帧为index 0，再后一帧为index1，……，后向帧排完后接着是按顺序排列的前向帧。具体排列如下：

| | | | | | | | |
|------------|----|----|----|-----|-----|-----|-----|
| 参考帧自然编号 | 97 | 98 | 99 | 100 | 101 | 102 | 103 |
| List0中的索引号 | 2 | 1 | 0 | 3 | 4 | 5 | |
| List1中的索引号 | 5 | 4 | 3 | 0 | 1 | 2 | |

2.预测模式选择

B 条的预测信息包括：宏块划分方式、预测模式、参考列帧表等选择。宏块划分方式有3种选择： 16×16 、 8×16 、 16×8 和 8×8 ；每种划分方式可以在4种预测模式中选择一种：直接（direct）模式、利用List0参考帧模式、利用List1参考帧模式、利用List0和List1的参考帧模式。其中，非对称划分（ 8×16 或 16×8 ）不可选择直接预测模式， 8×8 划分所选择的预测模式适用于划分中的所有子块。

3. 双向预测

双向预测中，当前块的前后参考块是由List0和List1中的参考帧提供的。从List0和List1分别得出两个运动补偿参考块（需要两个运动矢量），而预测块的像素一般取List0和List1参考块相应像素的平均值：

$$\text{pred}(x,y) = (\text{pred0}(x,y) + \text{pred1}(x,y) + 1) \gg 1 \quad (5.10)$$

其中， $\text{pred0}(x,y)$ 和 $\text{pred1}(x,y)$ 为由list0和list1参考块提供的预测像素， $\text{pred}(x,y)$ 为双向预测像素。计算出每个预测像素后，运动补偿残差通过当前宏块像素减 $\text{pred}(x,y)$ 而得。

4. 直接预测模式

直接预测模式编码的 B 条宏块或宏块划分不需传送运动矢量，由解码器计算 list0 和 list1 中已编码运动矢量，得到直接模式的双向预测运动补偿。直接预测模式包括空域和时域两种模式，由条头信息标明。

图5.8所示的是时域直接运动矢量举例，当前帧t宏块的list1中的参考帧t+2在当前帧两幅图像后出现。list1中参考帧t+2中的同位（co-located）MB的MV为 $(+2.5,+5)$ ，指向list0参考图像t-3（出现于当前图像3幅图像前）。解码器分别计算出指向list1的MV1为 $(-1,-2)$ ，指向list0的MV0为 $(+1.5,+3)$ 。

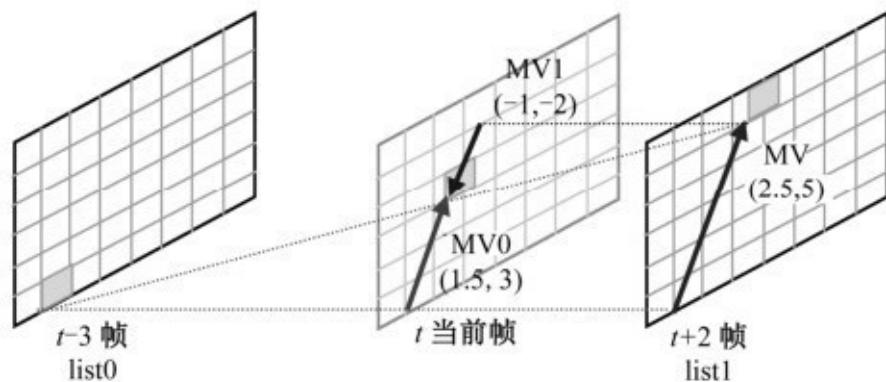


图5.8 时域直接运动矢量举例

5.3 HEVC的帧间预测

HEVC的帧间预测编码总体上和H.264/AVC相似，但主要在以下三方面进行了改进，一是采用了灵活的预测块（PU）划分，有利于使划分结果更加贴近运动物体的形状；二是采用了更合理的子像素插值算法，进一步提高了运动估计和运动补偿的精度；三是采用了新的合并模式，可以更加高效地编码传输运动参数。

5.3.1 帧间预测PU的划分

在HEVC编码中，每个CTU都可以递归地分解为更小的方形CU。CU是CTU四叉树分解最末的叶节点，视频编码的帧间预测算法就在CU上实行。如图5.9所示为一帧图像的CU划分结果，其中 11×6 个CTU的尺寸为 64×64 ，划分出的最大CU的尺寸也是 64×64 ，和CTU相等，实际上是没有划分，最小的CU尺寸为 8×8 ，其余的CU尺寸还有 32×32 、 16×16 。

CU是决定预测模式的基本单元，它决定该CU是采用帧内预测模式还是帧间预测模式。如果是帧间预测模式在有必要的情况下，帧间预测的CU还允许划分一次，成为更小的预测单元（PU）。

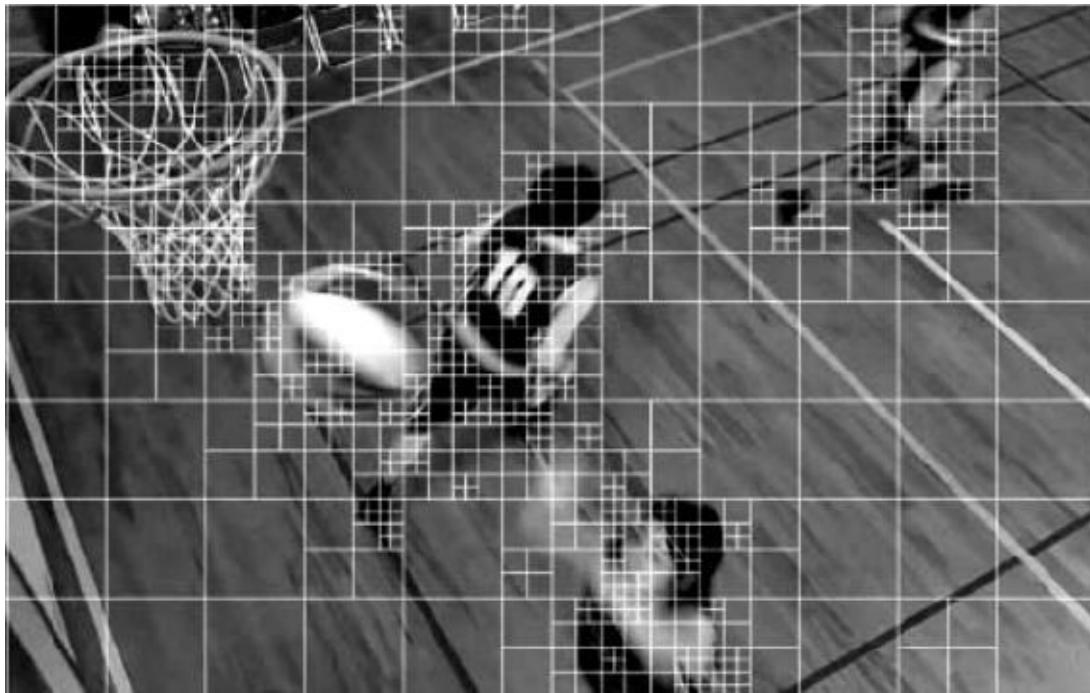


图5.9 HEVC的CTU到CU划分一例

1.PU划分

在HEVC的分块结构中，帧间预测的PU（本章的PU如没有说明都是指帧间PU）划分是很灵活的，允许将 64×64 、 32×32 、 16×16 或 8×8 的CU划分为更小的PU,PU可以是方形的，也可以是矩形的，还可以是不对称的。可参见第3章的图3.9，该图中下面2排就是帧间 $2N\times 2N$ 的CU可以划分的8种情况。其中4种属于对称划分， $2N\times 2N, 2N\times N, N\times 2N, N\times N$;4种属于非对称划分， $2N\times nU, 2N\times nD, nL\times 2N, nR\times 2N$ 。

（1）对称矩形PU划分

划分模式PART_2N×2N、PART_2N×N和PART_N×2N的定义分别对应

CU没有划分、水平划分为2个相同尺寸的矩形PU和垂直划分为2个相同尺寸的矩形PU。

(2) 对称方形PU划分

PART_N×N定义CU分裂为4个相等尺寸的PU，但这种模式仅当CU尺寸等于最小允许CU尺寸时方能采用。其原因是，既然从CTU分解到CU,CU尚未达到最小的CU，如果允许将它1分为4的话，实质上就是CTU到CU的分解深度增加一层而已。

(3) 非对称PU划分

有4种划分类型支持划分CU为2个尺寸不同矩形的PU:PART_2N×nU、PART_2N×nD、PART_nL×2N和PART_nR×2N。这4种模式称为非对称运动划分(AMP)。

2.运动参数

每个采用帧间预测方式编码的 PU 都有一套运动参数，包括运动矢量、参考帧索引和参考表标志等。因为AMP允许PU在运动估计和运动补偿中更精确地符合图像中运动目标的形状，而不需要用进一步的细分来解决，因此可以提高编码效率。可这样解释，设想一个长条形的运动物体，如果有非对称划分，可能只要一个 PU 就可以完全覆盖，只需一套运动参数即可；如果没有非对称划分，则可能需要多个划分更小的方形块拼接起来才能覆盖这个长条形物体，需要多套运动参数。

5.3.2 子像素插值

运动估计的具体方法在以往所有的视频编码标准中都是开放的，只要求开发者获得的运动矢量符合标准所要求的格式，保证解码器可识别和使用。至于如何得到视频中的运动矢量，允许开发者采用自己的办法。

HEVC 标准也是如此，没有规定具体的运动估计算法，在标准的文档中自然也就没有这方面的叙述，但是具体规定了高精度运动估计中形成子像素的插值滤波方法。如果不统一插值像素的生成方法，由于内插出的像素值

有可能不一样，就会影响不同的编解码设备之间的兼容和互通。

用于帧间预测的 PU 样点可从的参考图像（由参考图像索引给出）的相应块中取得，其位置由运动矢量的水平和垂直分量的位移来确定。除了运动矢量为整数值的情况，分数样点内插用于产生非整数取样位置的预测值。同在H.264/AVC中一样，HEVC支持运动矢量的精度为亮度样点之间的1/4单位距离。对于色度取样，运动矢量的精度取决于色度取样的格式，对4:2:0取样，色度样点之间的为1/8单位距离。

1. 亮度样点内插

H.264/AVC对于亮度分数样点的内插采用2步内插处理：第一步使用一个6抽头滤波器，滤波后进行舍入操作得到半像素点插值；第二步对1/4样点两端的整数点值和半像素点值进行平均处理，得到1/4像素位置的插值。

与H.264/AVC不同，HEVC中亮度分数样点的内插分两种情况处理，对1/2像素位置插值使用一个8抽头滤波器，对1/4像素位置插值使用一个7抽头滤波器。可见，HEVC是使用这种可分离的内插滤波器来产生所有分数位置样点，没有中间的舍入操作，可提高精度，简化分数样点内插过程。同时，HEVC使用了更长的滤波器提高了内插精度。对半像素位置采用8抽头滤波器，对1/4像素插位置采用7抽头滤波器已经足够了，这是由于1/4像素位置比半像素更接近整像素位置。内插滤波的实际抽头值（滤波系数）是从DCT基函数方程推导出来的。

从图5.10可以看出，对于1/4精度的插值，平均每一个整数像素将伴随着15个插值生成的像素。其中大写字母 $A_{i,j}$ 表示在原图像整像素位置的亮度像素值，而用小写字母标注在非整数像素位置的像素，即需要由内插产生的像素。下面以 $A_{0,0}$ 为例，给出相伴的15个插值点的计算方法，其他所有整数点的相伴插值点只要如此计算即可。

(1) 和 $A_{0,0}$ 在一直线上的6个插值点

图5.10中标注为 $a_{0,0}, b_{0,0}, c_{0,0}, d_{0,0}, h_{0,0}$ 和 $n_{0,0}$ 的内插像素是由整像素 $A_{i,j}$ 通过1/2像素位置的8抽头滤波、1/4像素位置的7抽头滤波而得。

1/2内插点 $b_{0,0}$ ，利用水平方向的8个整数点，采用8抽头滤波器hfilter[]内插得到；1/2内插点 $h_{0,0}$ ，利用垂直方向的8个整数点，采用8抽头滤波器hfilter[]内插得到。

1/4内插点 $a_{0,0}$ 和3/4内插点 $c_{0,0}$ ，利用水平方向的7个整数点，采用7抽头滤波器qfilter[]内插得到；1/4内插点 $d_{0,0}$ 和3/4内插点 $n_{0,0}$ ，利用垂直方向的7个整数点，采用7抽头滤波器qfilter[]内插得到。这些点的内插公式如下：

$$\begin{aligned}
 a_{0,0} &= \left(\sum_{i=-3}^3 A_{i,0} \cdot \text{q filter}[i] \right) \gg (B-8) & d_{0,0} &= \left(\sum_{j=-3}^3 A_{0,j} \cdot \text{q filter}[j] \right) \gg (B-8) \\
 b_{0,0} &= \left(\sum_{i=-3}^4 A_{i,0} \cdot \text{h filter}[i] \right) \gg (B-8) & h_{0,0} &= \left(\sum_{j=-3}^4 A_{0,j} \cdot \text{h filter}[j] \right) \gg (B-8) \quad (5.11) \\
 c_{0,0} &= \left(\sum_{i=-2}^4 A_{i,0} \cdot \text{q filter}[1-i] \right) \gg (B-8) & n_{0,0} &= \left(\sum_{j=-2}^4 A_{0,j} \cdot \text{q filter}[1-j] \right) \gg (B-8)
 \end{aligned}$$

在公式中，常数B是参考样点的比特深度，一般 $B \geq 8$ ，大多数应用 $B=8$ ，“ $\gg (B-8)$ ”表示二进制数算术右移(B-8)位操作。1/2插值滤波器hfilter[]和1/4插值滤波器qfilter[]的系数见表5.1。

表5.1 亮度分数样点插值滤波器系数

| Index <i>i</i> | -3 | -2 | -1 | 0 | 1 | 2 | 3 | 4 |
|----------------|----|----|-----|----|----|-----|---|----|
| hfiltcr[i] | -1 | 4 | -11 | 40 | 40 | -11 | 4 | -1 |
| qfiltcr[i] | 1 | 4 | 10 | 58 | 17 | 5 | 1 | |

(2) 中间的9个插值点

图5.10中虚线框内的中间9个点 $e_{0,0}, f_{0,0}, g_{0,0}, i_{0,0}, j_{0,0}, k_{0,0}, p_{0,0}, q_{0,0}$ 和 $r_{0,0}$ 的像素可以由相应的滤波器对刚才内插出来的垂直邻近的像素 $a_{0,j}, b_{0,j}$ 和 $c_{0,j}$ 滤波而得到，它们的插值如下：



图5.10 分数位置像素内插标记

$$\begin{aligned}
e_{0,0} &= \left(\sum_{v=-3}^3 a_{0,v} \cdot q \text{ filter}[v] \right) \gg 6 \quad i_{0,0} = \left(\sum_{v=-3}^4 a_{0,v} \cdot h \text{ filter}[v] \right) \gg 6 \quad p_{0,0} = \left(\sum_{v=-2}^4 a_{0,v} \cdot q \text{ filter}[1-v] \right) \gg 6 \\
f_{0,0} &= \left(\sum_{v=-3}^3 b_{0,v} \cdot q \text{ filter}[v] \right) \gg 6 \quad j_{0,0} = \left(\sum_{v=-3}^4 b_{0,v} \cdot h \text{ filter}[v] \right) \gg 6 \quad q_{0,0} = \left(\sum_{v=-2}^4 b_{0,v} \cdot q \text{ filter}[1-v] \right) \gg 6 \\
g_{0,0} &= \left(\sum_{v=-3}^1 c_{0,v} \cdot q \text{ filter}[v] \right) \gg 6 \quad k_{0,0} = \left(\sum_{v=-3}^4 c_{0,v} \cdot h \text{ filter}[v] \right) \gg 6 \quad r_{0,0} = \left(\sum_{v=-2}^4 c_{0,v} \cdot q \text{ filter}[1-v] \right) \gg 6
\end{aligned}$$

(5.12)

从式 (5.12) 可以看出，所有的插值像素都放大了64倍，其目的是在中间的计算过程中保证较高的精度，待进入后续的预测环节中，这些像素都会被缩小64倍，落入正常的像素值范围内。在HEVC中，最多需要2步舍入操作来获位于1/4位置处的像素，这样当使用双向预测时，在最坏的情况下5次舍入操作就足够了。然而，在普通的应用中，其比特深度为8比特，在最坏的情况下舍入操作的总数将进一步减少至3。由于较少的舍入操作次数，累加舍入误差有所减少。

2. 色度样点内插

对色度分量的分数位置像素的内插处理类似于亮度分量的方法，不同的是其滤波器的抽头数为4。对于一般的4:2:0数字视频，色度整数样点的距离比亮度大一倍，要达到和亮度同样的插值密度，其插值精度需为1/8色度像素。HEVC为1/8像素位置定义了一套用于4:2:0色度格式的4抽头滤波器，如表5.2所示（而在H.264/AVC中仅使用2抽头双线性滤波器）。

表5.2 色度分量样点插值滤波器系数

| Index | -1 | 0 | 1 | 2 | |
|-------------------|----|----|----|----|---------------------|
| 1/8 插值 filter1[i] | -2 | 58 | 10 | -2 | 7/8 插值 filter1[1-i] |
| 2/8 插值 filter2[i] | -4 | 54 | 16 | -2 | 6/8 插值 filter2[1-i] |
| 3/8 插值 filter3[i] | -6 | 46 | 28 | -2 | 5/8 插值 filter3[1-i] |
| 4/8 插值 filter4[i] | -4 | 36 | 36 | -2 | |

滤波器系数的值标注为filter1[i],filter2[i],filter3[i]和filter4[i], $i=-1,0,1,2$ ，分别用于内插色度像素的第1/8,2/8,3/8和4/8分数位置处的值。对5/8,6/8和7/8分数位置处的值，按照镜像对称性，分别使用filter3[1-i]、filter2[1-i]和filter1[1-i]的值， $i=-1,0,1,2$ 。

一个整数色度像素须插入 $8 \times 8 - 1 = 63$ 个样值。其中，和整像素点在同一水平方向的7个内插点用水平方向的4抽头滤波器系数；和整像素点在同一垂直方向的7个内插点用垂直方向的4抽头滤波器，系数和水平方向一样。处于中间的 $7 \times 7 = 49$ 个点只用垂直方向4抽头滤波器，同时利用刚才内插出来的非整数色度像素值，滤波器系数值仍然和前面一样。

5.4 HEVC的运动参数编码

在HEVC的帧间预测编码中，每一个帧间预测单元PU都含有一组运动参数：运动矢量残差、参考帧索引和帧间预测标记等。运动矢量差（Motion Vector Difference,MVD）用于表示当前编码PU的运动矢量（MV）和参考帧中最为匹配PU的MV之间的差值；参考帧索引用于表示运动估计所使用的参考帧标号；帧间预测标记则用于标明前向预测、后向预测或双向预测方式。

这些运动参数都需要传递到解码端，才可以使解码器正确解码、重建原图像。按照传输运动参数的方式不同，HEVC定义了三种帧间预测模式：帧间模式（Inter mode）、跳过模式（Skip mode）和合并模式（Merge mode）。为提高运动矢量预测和编码的压缩效率，对这三种模式都采用了运动矢量竞争（MV Competition,MVC）机制，从所给的包含诸多空域和时域运动候选的候选列表中选择一个最佳的运动候选。

5.4.1 运动参数的编码传送

在视频编码中，运动矢量需要作为边信息（Side Information,SI）与帧间预测残差一同传输到解码端。解码端由解码得到的运动矢量进行运动补偿，即通过解码的预测残差加上参考帧中运动矢量指向的预测块像素得到重构的原始信号。在视频编码中，一般采用不同尺寸的块匹配、双向预测、多参考帧等技术，从而提高运动补偿的性能，但同时加大了运动矢量等运动参数信息在整体码流中的比重。因此，运动矢量的编码对提高整体编码效率起着至关重要的作用，特别是在低码率的情况下，运动矢量将占整体码流很大一部分。

1. 相邻运动矢量的相关性

为了提高编码效率，对运动矢量等参数进行压缩编码势在必行，其中预测编码是最常用的一种方法，它的原理和图像像素的预测编码类似，其根据也是相邻运动矢量之间存在的较大相关性。

图5.11描述了由统计平均得到的当前块和相邻块（时域、空域）的运动矢量之间的相关系数。图5.11右边为当前帧，深色的为当前块；图左边为先前帧，中心块为同位块（和当前帧中当前块处于相同的空间位置）。块中标明的数字是当前块和其邻近块的运动矢量在水平和垂直方向的平均相关系数。可以看到，当前块和邻近块的运动信息之间存在着较强的相关性，这是HEVC中采用运动信息预测的根据，即相邻块的运动信息很可能相同或相近。

从图5.11中可以看出，随着编码块在水平方向、垂直方向和时间方向远离当前块，它们的相关系数值在逐步减小。当前块和它空域上的直接左邻近块、上邻近块、右上角邻近块和左上角邻近块的相关性较高，时域方向上和同位块，以及同位块的直接上、下、左、右邻近块也具有一定的相关性，但一般比空域相关性低。

一般对运动矢量的预测编码较像素的预测编码简单，先利用相邻（包括空间和时间相邻）运动矢量对当前运动矢量进行预测，再对运动矢量的预测残差进行熵编码。所以，运动矢量的编码性能在很大程度上取决于运动矢量预测方法。在HEVC中采用多个候选运动矢量的预测方法来提高运动矢量预测的准确性，只需要对所选预测候选的索引值进行编码。

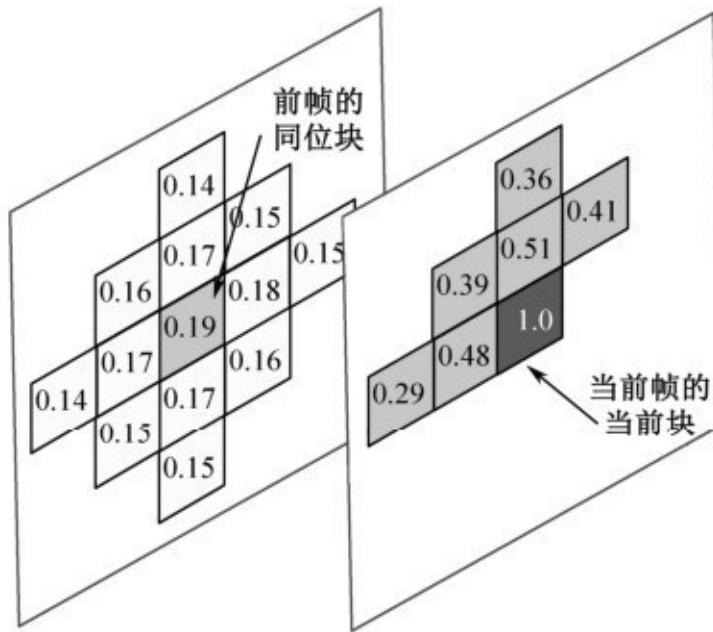


图5.11 当前块和邻域块运动矢量的相关系数

2.运动参数编码模式

HEVC对运动参数（主要是运动矢量）的编码继承并改进了H.264/AVC的方法，设有三种模式，简称为Inter模式、Skip模式和Merge模式。前两种模式和H.264/AVC类似，其中Inter模式也常称为高级运动矢量预测（Advanced Motion Vector Prediction,AMVP）模式。Merge模式是HEVC新引入的一种“运动合并”技术。这样，对于每个帧间编码的PU，可以在下列三种编码模式中选择。

(1) Inter模式，对运动参数直接编码（如对运动矢量进行预测编码）。

(2) Merge模式，采用运动合并技术处理运动参数，这种处理方式不

仅可以在Merge模式下使用，在Skip、Inter帧间模式下也可使用。

(3) Skip模式是Merge模式中的一种特殊情况。

在3种模式中，Inter 模式需要传送当前编码PU的运动矢量（MV）来完成基于运动补偿的帧间预测，运动矢量可采取空间或时间预测编码的方式（H.264/AVC仅采用空间预测）。而Skip模式和Merge模式无需传送MV信息，只传送候选PU块的索引信息，由解码端采用运动推理方法（如复制）来获得运动信息，推理依据来自空域邻近块或时域邻近块（候选块）的MV。在Skip模式中，系数残差信息也省略了，不用传送。

5.4.2 Merge模式

在以往标准的帧间预测中，要为每个帧间预测宏块都传输一套运动参数。为进一步提高编码效率，可对相邻已编码的帧间预测块（PU）进行合并处理，形成Merge帧间预测编码模式，将邻近的运动参数相同或相近的若干 PU 合并起来形成一个形状不规则的“小区”，可克服四叉树分割中固定的方块划分的缺点，且只要为每个小区传输一次运动参数，不必为每个 PU 分别传输运动参数。这就是HEVC的运动合并模式，是一种利用时空相邻的已编码预测单元（PU）的运动参数得到当前预测单元运动参数的技术。主要包括两个步骤。

(1) 构造候选列表，编码器为当前正在编码的 PU 构造一个先前已编码的邻近 PU 的候选列表，列表中的候选PU既可以是当前PU的空间邻近块，也可以是时间邻近块。

(2) 以竞争方式选择参考PU，编码器可以通过最小率失真（RD）准则，在候选表中选择并标注编码代价最小PU，作为当前PU的参考PU。使得当前编码PU在解码时只要复制选中的PU的运动参数即可。

Merge模式以预测单元（PU）为单位，每个CU中的全部或部分PU可采用Merge模式。在 Merge 模式中，运动矢量差（MVD）、参考帧索引以及帧间预测标识等运动参数都不需要在码流中传输。码流中只需编码传送

运动参数在运动参数候选表中的合并索引即可。解码端首先使用与编码端相同的运动合并技术构成运动参数候选表。然后根据码流中传送来的运动合并索引在运动参数候选表中找到对应的候选运动参数。

Merge模式的一个示例如图5.12所示，图5.12 (a) 是原图像，图5.12 (b) 中的白色边界是经四叉树CU划分后的结果，图5.12 (c) 是经过Merge模式的竞争机制得到的运动信息共享区域的界线。可见，在不少平坦区域具有相同的运动参数，“合并”成更大的运动信息共享区域，而在图像的细节之处，共享的区域虽然有，但面积要小得多。

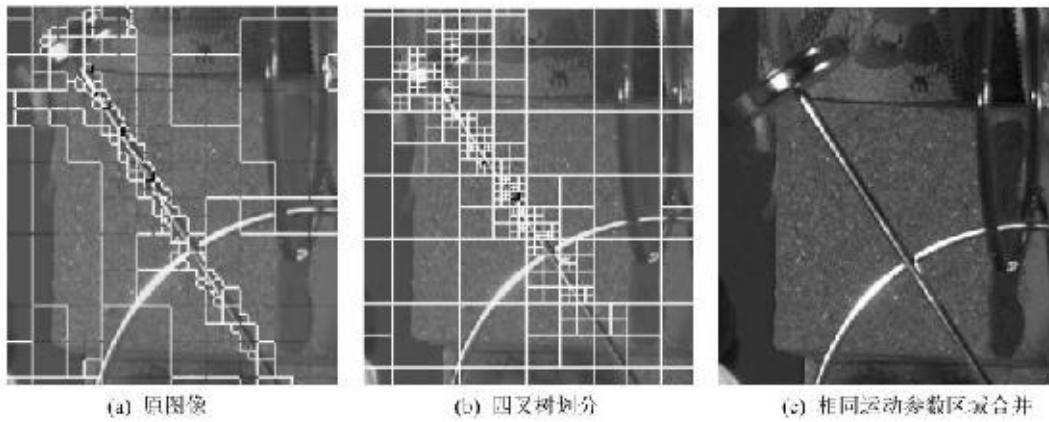


图5.12 某帧的运动参数合并模式

1. 空域候选的确定

在Merge模式中，编码的PU在运动矢量预测（ Motion Vector Prediction, MVP ）候选集中选择运动矢量预测块。MVP候选集包括5个空域MVP和2个时域MVP，如图5.13所示。5个空域MVP分别为当前PU左下角的预测块 A_0 和 A_1 ，左上角的预测块 B_2 ，右上角的块 B_0 和 B_1 ；2个时域MVP分别是先前已编码帧中和PU相同位置的预测块 T_{CT} 和右下角的 T_{BR} 。

1) 对称划分PU的运动候选

Merge 模式中空域候选运动参数的选择是通过选择相邻的已编码块进行的。按照HEVC规定，如图5.13所示，在所有7个候选块中，Merge模式（ Skip模式也是如此 ）的合并候选列表需选择5个候选：包括4个空域运动候选和1个时域运动候选。也就是说，在空间相邻块中最多可以从5个不同位置上选择其中的4个，在时间相邻块中最多可以从2个中选择一个。它们的选择顺序如下：

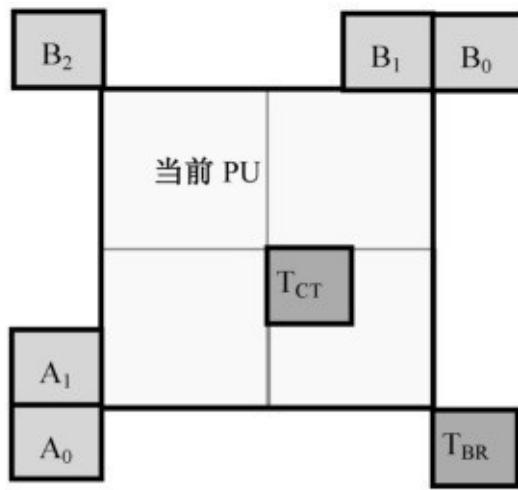


图5.13 Merge模式的时空候选的位置

- (1) 左候选 (A_1) ;
- (2) 上候选 (B_1) ;
- (3) 右上候选 (B_0) ;
- (4) 左下候选 (A_0) ;
- (5) 左上候选 (B_2 , 仅用于上边空域候选没有一个块可用时) ;
- (6) 时域候选PU (从 T_{BR} 到 T_{CT} 第一个可用的)。

所谓“不可用”是指，如果在某个位置的块是帧内预测块，或这个位置在当前条、片或图像以外，就认为它是不可用块。

HEVC的编码器可采用率失真优化（RDO）的决策机制，为Merge模式

在给定的候选集中选择最终的候选，并将所选择的候选信息通过合并索引传送到解码器。

2) 非对称划分PU的运动候选

当预测单元划分类型为 $N \times 2N$ 时，为避免“虚化”成 $2N \times 2N$ 的预测单元，即两个 $N \times 2N$ 预测单元使用同一组候选而实际形同 $2N \times 2N$ 的预测单元。如图5.14所示，对于左图中两个 $2N \times N$ 预测块中，第一个预测块（上方）可按上述的正常顺序进行选择，而对第二个预测块（下方）需要进行特殊处理，由于第2个预测块的空间候选 B_1 位置不可用作候选，所以选取空间候选集的顺序变为 $A_1 \rightarrow B_0 \rightarrow A_0 \rightarrow B_2$ 。基于同样的原因，对于右图中两个 $N \times 2N$ 预测块的第1个预测块（左方）可按照正常的候选选择顺序进行，第2个预测块（右方）的 A_1 位置不能再使用，选取的顺序变成 $B_1 \rightarrow B_0 \rightarrow A_0 \rightarrow B_2$ 。

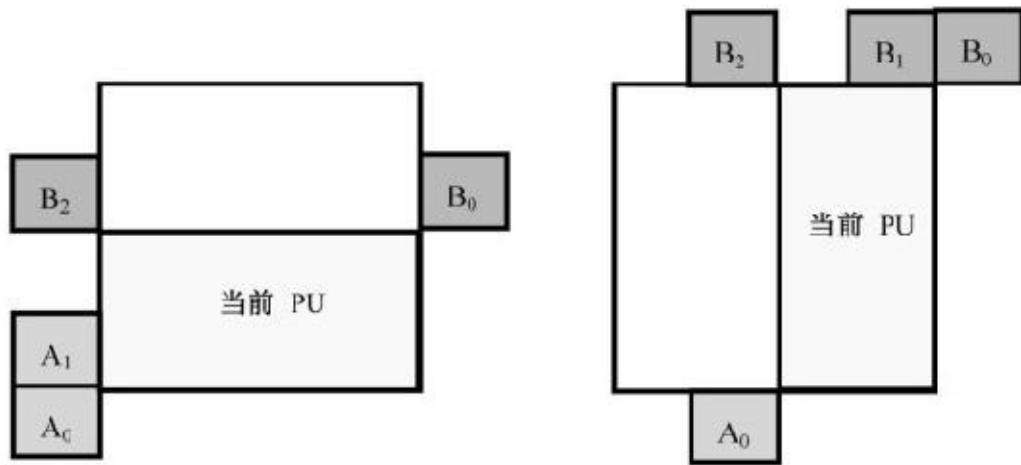


图5.14 Merge模式非对称划分时第二个PU的候选位置

2.时域候选的确定

如图5.13所示，时域运动参数的候选就是在已编码帧的 T_{BR} 或 T_{CT} 中选择一个合适的帧间预测单元，即从当前帧的参考图像队列中的参考帧中获得。但时域候选块的运动矢量确定和空域的情况不一样，时域候选一般不能直接引用候选块的运动矢量，而是需要根据参考图像的位置关系做相应比例调整。

如图5.15所示，从参考帧队列中找出与当前帧Curr-Pic的POC序号最接近的同位帧Col-Pic，当前帧的参考帧Curr-Ref，当前帧的预测单元Curr_PU，同位帧的参考帧Col_Ref。然后，从Col-Pic帧中找两个相当于 T_{BR} 或 T_{CT} 位置的同位预测单元作为候选位置，择其一（ T_{BR} 优先）为时域参考预测单元Col-PU。设当前帧Curr_Pic与Curr_Ref的时间距离为 t_b ，Col_Pic与Col_Ref的距离为 t_d ，当前预测单元Curr_PU的运动矢量 MV_{Curr_PU} 可由Col_PU的运动矢量 MV_{Col_PU} 根据 t_b 和 t_d 的比值进行缩放得到，即

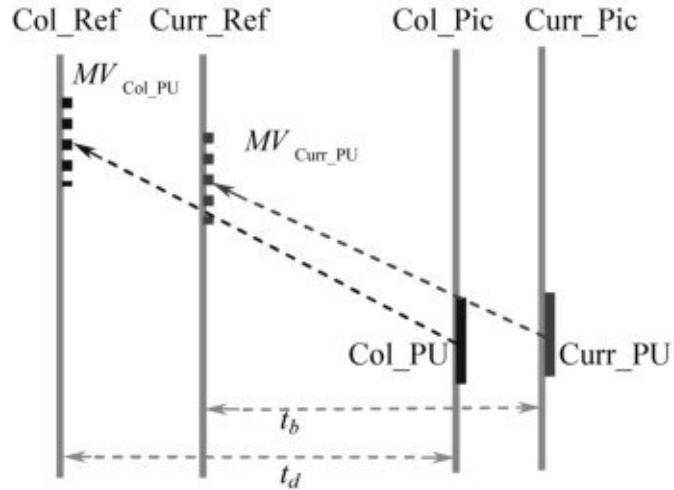


图5.15 Merge时域MV的伸缩

$$MV_{curr_PU} = \frac{t_h}{t_d} MV_{col_PU} \quad (5.13)$$

3.候选集的形成

实际上，Merge模式的运动信息采用4种类型的相邻运动信息作为候选，生成候选列表，通过传输候选列表的索引进行运动数据的表示，4类相邻运动信息候选分别为：①空域候选、②时域候选、③重组双向预测候

选、④零运动矢量候选。前面介绍的是前2种，后2种主要用于前面候选个数不满时的补充信息。并且按照这一顺序，依次添加到候选列表中，直到满足候选列表的最大个数K。

运动合并候选集包含空域候选子集和时域候选子集，它的总个数等于 MaxNumMerge Cand（默认值为5）。参见图5.16，运动参数候选表的形成可分为以下三个步骤：

（1）选取空域候选子集。从预测单元的5个空间相邻位置中选取至多4个预测代价小的预测单元作为空间候选子集。

（2）选取时域候选子集。从预测单元的2个时间相邻位置中选取一个最优运动参数集作为时域候选。

（3）子集合并。其中前两个步骤比较简单，通过某种优化选择的方法就可确定。关键是第3步子集的合并过程，此过程又包括2个步骤。

第一步，去除运动参数重复的冗余候选，在候选 PU 的选择过程中要去除其中运动参数重复的候选PU，同时还要去除其中使得与当前PU融合后形成一个等同于 $2N \times 2N$ 的PU的候选块。

第二步，当候选集总数K少于MaxNumMergeCand时，新增 MaxNumMergeCand-K个候选。

用于新增的候选集包括组合双向预测候选及零合并候选。在按顺序添加过程中，当候选个数达到MaxNumMergeCand时，其余的候选集将不再添加到运动参数候选表中。其中，组合双向预测候选只在B帧中使用。

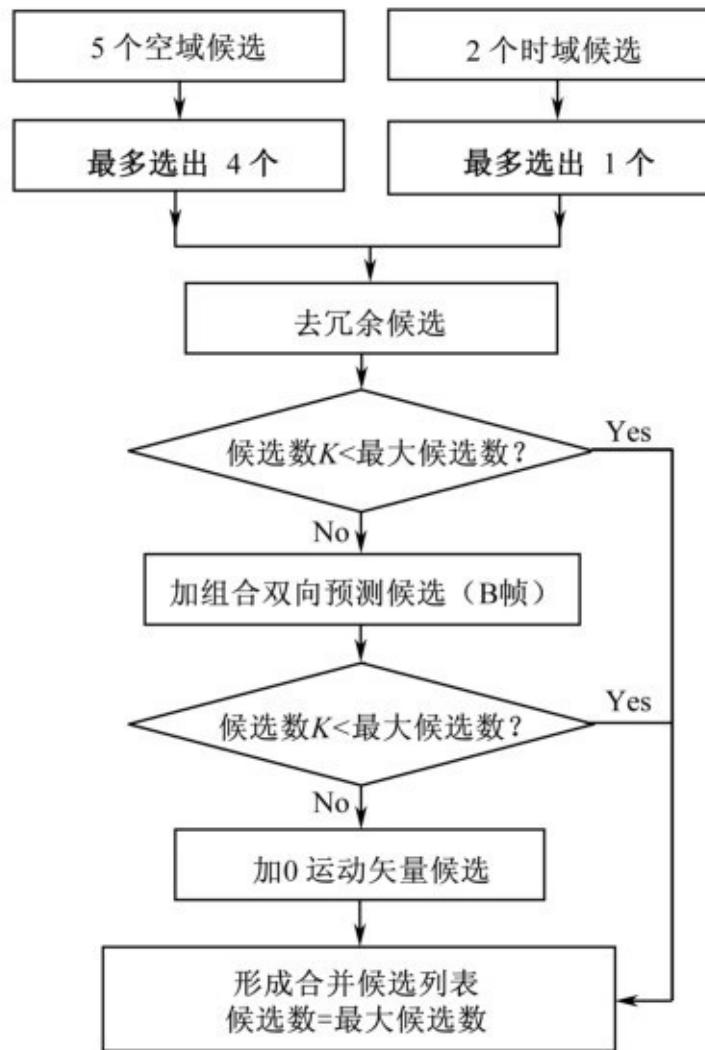


图5.16 运动合并候选列表的形成

对双向预测的B帧，使用组合双向预测候选来产生新的Merge候选。HEVC具体地定义了组合双向候选的方法。由于需要两个方向的运动矢

量，考虑在List0和List1已经导出的4个候选中选取，一个组合双向候选由List0中一个候选和List1中的一个候选组合而成。最多有16种可能的组合，去除其中两个候选相同的4种组合，最多还剩下12种可选的组合，可从中择优加入到Merge列表中（如果有）。

对P帧或B帧，由上述生成的运动参数仍然未能达到K个候选值，则生成0运动矢量，参考帧的索引值从0至最大依次选择的候选填入，直至选参数集个数达到MaxNumMerge Cand个数为止，运动参数候选表构建完毕。这种固定候选数有利于解码时的并行处理，并提高容错能力。

参数候选表完成后，接着计算表中每个运动参数对应的率失真代价。选出最佳运动参数索引，使用二进制一元截断编码（Truncated Unary Binarization, TUB）对最优运动参数索引进行熵编码，并传输到解码端进行解码。

4. 固定尺寸的候选列表

这里先回溯一下在HEVC中为何要选择固定尺寸的候选列表。一般情况下，去除冗余或不可用的候选以后，候选列表的尺寸在编码端和解码端都处于动态变化中，这样有利于索引信息的截断一元码二进制化的熵编码。尽管候选列表的变化尺寸可以带来编码增益，但它也引入了解码端的解析问题。因为时域运动候选是包含在候选列表中的，如果先前一幅图像的一个MV不能够被正确解码，在编码端候选列表和解码端的候选列表之间的就有可能出现一个错误的匹配，形成候选索引的解析错误。这个解析错误能够继续传播，使得当前图像的剩余部分也不能够被正确解析或解码。更严重的还在于，这个解析错误可影响后续帧的图像，因为这些图像认可这一时域运动候选，一个MV的解码错误就可能导致大量后续图像的解析失败。

在HEVC中，为解决上述解析问题，使用了一个固定尺寸的后候选列表来分离候选列表构建和索引解析之间的依赖。更进一步，为了补偿由固定列表尺寸引起的熵编码性能的损失，将附加的候选配置到候选列表的空位置上。在这一过程中，将候选列表索引编码成最大长度的截断一元码，

对Skip模式和Merge模式，候选最大长度的值在条头信息中传输，而在Inter模式中此值固定为2。候选列表尺寸固定后简化了解码器的解析操作，增加了它的鲁棒性，因为拆解编码数据的能力不依赖于合并候选的有效性。

5.4.3 Skip模式

Skip模式在H.264/AVC中已有，判定为Skip的块，表示它和某个参考块基本一样，因此编码器只需要传送相应的指向这个参考块的运动矢量即可，解码器凭此数据可获得该Skip块的像素数据来重构图像，大大减少了数据传输量。

在HEVC的Skip模式与 $2N \times 2N$ 的Merge模式类似，以 $2N \times 2N$ 的编码单元（CU）为基本预测单位，编码器只需要传输预测单元的Skip模式标识和运动信息索引，以保证解码时Skip PU可以复制所选中的候选者的运动参数作为自己的运动参数，并不需要传输运动补偿后的预测残差。这样，对那些帧间变化很少，或者运动缓慢的区域编码只需很少的比特。

实际上，Skip模式是特殊的Merge模式，即 $2N \times 2N$ 的编码块标志（Coded Block Flag,CBF）等于0时的Merge模式，实质上是不需要传输残差信号的Merge模式。因而也有学者将HEVC帧间编码模式分为两类，就是将Skip模式归纳到Merge模式中。

5.4.4 Inter模式

对于Inter模式的帧间预测，运动参数的候选集和Merge模式一样也是从几个候选值中选取，但HEVC仅允许2个候选用于Inter模式的运动矢量预测处理，因为编码器除了要对变化的运动矢量差值编码，此前还需完成运动估计，这是编码器中一项最耗费计算量的操作，所以允许较少的候选意味着较少的运动估计计算，可以降低复杂性。同时，因为在Inter模式中还需要编码运动矢量的预测残差，所以减少候选的数量不会给编码效率带来太

大影响。

1. 候选的确定

Inter 模式运动矢量候选的位置与 Merge 模式中空域候选的位置一样（见图5.13），包括2个空域运动候选集 $\{A_0, A_1\}$ 和 $\{B_0, B_1, B_2\}$,1个时域运动候选集 $\{T_{BL}, T_{CT}\}$ 。

空域候选的选择方法与Merge模式相同，包括下列两个步骤。

(1) 空域的左方运动候选，从当前预测单元(PU)的左下方向上搜索，即从 A_0 到 A_1 ，遇到的第一个可用的块就认作左候选。

(2) 空域的上方运动候选，从当前PU的右上方到左上方搜索，即从 B_0 、 B_1 到 B_2 ，遇到的第一个可用的块就认作上候选。

当两个空域候选具有相同的运动矢量时，则删除一个冗余的空域候选。由于Inter模式的运动候选个数规定为2个，当以上选择的空域候选少于2，且允许使用时域运动候选，即加入时域上相邻块作为第二个运动候选。这意味着，当2个空域候选都有效时，时域候选则全然无用。

时域候选的选择方法与Merge模式相同，即顺序搜索位于参考图像中，相当于当前帧编码PU右下角的块 T_{BR} 和同样位置的块 T_{CT} ，遇到的第一个可用的块就认作时域候选。

最后，若候选预测器的个数仍然小于2，则可用值为0的运动矢量填补，直到运动矢量预测候选的数目等于2为止，它保证了运动矢量预测器的数目等于2。这样，只要用一个编码标志来标明在Inter模式下使用的是哪一个运动矢量预测。对于时间相邻块得到的运动矢量预测值，需根据其参考帧与当前参考帧的不同做相应的缩放。

2. 运动矢量预测

类似于Merge模式，在Inter模式中，需用运动矢量预测值对运动矢量进行差值编码，编码的运动矢量之差和候选的索引都要传送到解码端。对于Inter模式的帧间预测块，编码器需传送一个帧间预测标志，标明当前PU采用的是list0预测（前向）、list1预测（后向），或者是双向预测。然后，如

果有多幅参考图像，则再传送1个或2个参考索引。为了给每个预测方向（前向预测或后向预测）从候选列表中选择一个运动候选，还要传送一个索引。

时域候选的参考图像又称时域同位图像。通过在条头中传送一个标识和一个参考索引来标明时域同位图像。标识用于指明使用哪个参考图像列表，参考索引用于指明参考图像列表中的那帧图像用作同位参考图像。在索引传送以后，还需再传送1个或2个运动矢量差值（MVDs）。

5.4.5 帧间预测模式的选择

至此，我们已经简单介绍了HEVC帧间编码运动信息编码的3种模式，其中Skip模式需要传送的信息最少，编码效率最高。因此在进行 PU 模式判决时，首先要看 PU 是否符合 Skip模式，然后再考虑Merge模式和Inter模式。实际上，这不仅是PU的模式判定原则，也是CU的模式判定原则。

在连同帧内模式选择在内的完整CU模式判定中，首先看它是否为帧间的Skip模式，然后再看帧间模式，包括帧间CU到PU的各种划分，最后看帧内模式，也包括不同的划分和不编码的PCM方式。这样，编码器遍历所有可能的帧间和帧内划分与模式，通过计算得到各种情况的率失真代价（RD cost），为CU选取一个最佳（率失真代价最小）的预测模式，其顺序如图5.17所示。

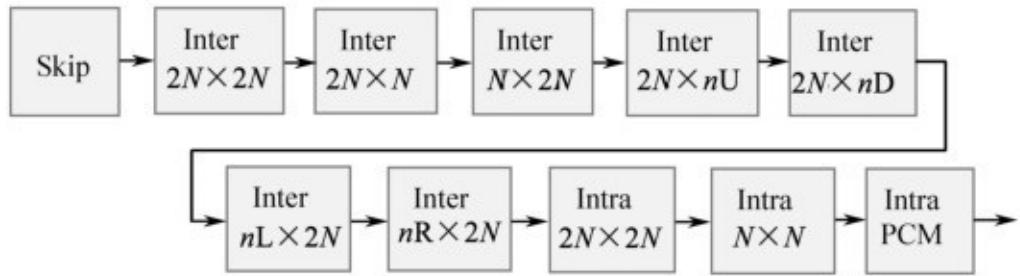


图5.17 CU内预测模式选择流程图

例如，对一个给定的CU，首先计算Skip模式的代价。Skip模式代价计算完成后，计算从inter $2N \times 2N$ 到inter $nR \times 2N$ 的帧间预测模式的代价。在选取其中最佳模式后，会检测编码块标志（CBF），如果CBF为0，那么此后所有的帧内预测模式检查就不再执行。否则将继续遍历所有帧内预测模式。编码块标志（CBF）用于标明编码块内量化后系数的情况。当图像块通过预测得到残差后，编码器会对残差进行变换和量化，如果量化后的系数全部为0，则对此块的CBF标记为0，否则标记为1，表示量化后存在非0系数。在HM参考模型软件中，CBF用来记录每个 4×4 大小的块。

本章参考文献

[1]High Efficiency Video Coding[S].ITU-T Rec.H.265 and ISO/IEC 23008-2,ITU-T and ISO/IEC JCT-VC,Mach 2013 (v.1),Oct.2014 (v.2).

[2]Jian Liang Lin,Yi Wen Chen,Yu Wen Huang,et al.Motion vector coding in the HEVC standard[J].IEEE Journal of Selected Topics in Signal Processing,Dec.2013,7 (6) : 957-968.

[3]Ugur,K.,A Alshin,E.Alshina,et al.Motion compensated prediction and interpolation filter design in H.265/HEVC[J].IEEE Journal of Selected Topics in Signal Processing,Dec.2013,7 (6) : 946-956.

[4]S.G.Biasi,I.Zupancic,E.Izquierdo,et al.Adaptive precision motion estimation for HEVC coding[C].IEEE Picture Coding Symposium (PCS 2015),144-148.

[5]Su,Y.,Segall,A.Reduced resolution storage of motion vector data[R].Doc.CTVC-D072.4th Meeting: Joint Collaborative Team on Video Coding (JCT-VC) of ITU-T SG 16 WP 3 and ISO/IEC JTC 1/SC 29/WG 11,Daegu,2011.

[6]P.Helle and K.Ugar.Block merging for quadtree-based partitioning in HEVC[J].IEEE Trans.on Circuits and Systems for Video Technology.Dec.2012,22 (12) :1720-1731.

[7]Mathias Wien.High Efficiency Video Coding: Coding Tools and Specification[M].Springer-Verlag Berlin Heidelberg,2015.

[8]Alshina,E.,Alshin,A.DCT derived interpolation filter test by Samsung[R].Doc.JCTVCF247,6th Meeting: Joint Collaborative Team on Video Coding (JCT-VC) of ITU-T SG 16 WP3 and ISO/IEC JTC 1/SC 29/WG

11,Torino,2011.

[9]E.Alshina,J.Chen,N.Shlyakhov.Experimental results of DCTIF by Samsung[R].Doc.JCTVC-D344.4th Meeting: Joint Collaborative Team on Video Coding (JCT-VC) of ITU-T SG 16 WP 3 and ISO/IEC JTC 1/SC 29/WG 11,Daegu,2011.

[10]F.Kossentini,Y.W.Lee,M.J.T.Smith,etal.Predictive RD optimized motion estimation for very low bit-rate video coding[J].IEEE Journal on Selected Areas in Communications,Dec.1997 (15) : 1752–1763.

[11]Agbinya,J.I.Interpolation using the Discrete Cosine Transform[J].Electronics Letter,1992,28 (20) : 1927-1928.

[12]Vivienne Sze,Madhukar Budagavi,Gary J.Sullivan.High Efficiency Video Coding (HEVC) :Algorithms and Architectures[M].Switzerland Springer International Publishing,2014.

[13]Bossen,F.Common test conditions and software reference configurations[S].Doc.JCTVCK 1100.11th Meeting: Joint Collaborative Team on Video Coding (JCT-VC) of ITU-T SG 16 WP 3 and ISO/IEC JTC 1/SC 29/WG 11,Shanghai,CN (2012).

[14]Advanced video coding for generic audiovisual services[S].ITU-T Rec.H.264 (AVC) .Oct.2014.

[15]K.R.Rao,Do Nyeon Kim,Jae Jeong Hwang.Video Coding Standards: AVS China,H.264/MPEG-4 PART 10,HEVC,VP6,DIRAC and VC-1[M].Springer Dordrecht Heidelberg New York London,2014.

[16]Xufeng Li,Ronggang Wang,Xiaole Cui,et al.Context-adaptive fast motion estimation of HEVC[C].IEEE International Symposium on Circuits and Systems (ISCAS,2015),2784-2787.

[17]Xiem Hoang Van,Joao Ascenso,Fernando Pereira.Improving enhancement layer merge mode for HEVC scalable extension[C].IEEE Picture Coding Symposium (PCS 2015) 15-19.

第6章 HEVC的变换和量化

采用混合编码方式的HEVC编码技术，先对视频数据进行空间预测或时间预测，随后对预测残差进行整数变换，再对变换后的系数进行量化，将量化后的系数通过特定的扫描方式形成一维数据，从中提取出重要信息后送去进行熵编码。本章主要介绍其中的变换和量化环节，以及为后面熵编码做准备的系数扫描方式。

HEVC的变换和量化大体上和H.264/AVC相似，但增加了多处改进，提高了本部分的编码效率。在变换方面的改进主要体现在三方面，一是变换块（TB）的尺寸可变范围扩大，二是整数DCT的精度提高，三是引进了 4×4 的离散正弦变换。在量化方面基本和H.264/AVC相同，但由于HEVC的整数DCT的缩放矩阵特别简单，从而使得量化处理的负担有所减轻。

6.1 变换与量化

6.1.1 离散余弦变换和正弦变换

离散变换（DCT）的基本思想是将一个实函数对称延拓成一个实偶函数，对实偶函数进行离散傅里叶变换（DFT），其结果也是一实偶函数，各取偶函数的一侧定义为离散余弦变换对。可见DCT的本质就是DFT，它的物理意义和傅里叶变换基本一样，但由于没有DFT的复数运算，计算方便。而且，无论是一维还是二维 DCT，其正反变换的变换核都相同，再加上二维DCT核是可分离函数，可采用一维DCT方法来实现信号的二维DCT，无论是软件还是硬件实现起来都比较方便。因而在信号处理领域，尤其是在图像领域得到了广泛的应用。

二维DCT变换的公式见式（6.1）、式（6.2）、式（6.3）和式（6.4）。但在离散的数字邻域，二维DCT可以用矩阵来表示，则更加直观和方便。

1. 二维DCT矩阵

$N \times N$ 的信号矩阵 f 的二维DCT的 $N \times N$ 系数矩阵 F 可以用矩阵乘法的方式来表示，其正反DCT分别如下：

$$F_c = C \cdot f \cdot C^T \quad (6.1)$$

$$f = C^T \cdot F_c \cdot C \quad (6.2)$$

为简单起见，其中所有的矩阵皆为 $N \times N$ 的方阵， $f(x,y)$ 是二维信号矩阵 f 中第 x 行第 y 列的元素（信号值）， $F(u,v)$ 是DCT系数矩阵 F_c 中第 u 行第 v 列的元素（系数值）。二维DCT变换矩阵 C ，的第 u 行第 x 列元素可表示为：

$$c_{u,x} = \sqrt{\frac{2}{N}} k(u) \cos \frac{(2x+1)u\pi}{2N} \quad k(u) = \begin{cases} 1/\sqrt{2} & u=0 \\ 1 & \text{else} \end{cases} \quad (6.3)$$

其中， $x,u=0,1,\dots,N-1$ 。对 4×4 图像块信号的DCT变换矩阵 C 为：

$$C = \begin{vmatrix} \frac{1}{2}\cos(0) & \frac{1}{2}\cos(0) & \frac{1}{2}\cos(0) & \frac{1}{2}\cos(0) \\ \sqrt{\frac{1}{2}}\cos(\frac{\pi}{8}) & \sqrt{\frac{1}{2}}\cos(\frac{3\pi}{8}) & \sqrt{\frac{1}{2}}\cos(\frac{5\pi}{8}) & \sqrt{\frac{1}{2}}\cos(\frac{7\pi}{8}) \\ \sqrt{\frac{1}{2}}\cos(\frac{2\pi}{8}) & \sqrt{\frac{1}{2}}\cos(\frac{6\pi}{8}) & \sqrt{\frac{1}{2}}\cos(\frac{10\pi}{8}) & \sqrt{\frac{1}{2}}\cos(\frac{14\pi}{8}) \\ \sqrt{\frac{1}{2}}\cos(\frac{3\pi}{8}) & \sqrt{\frac{1}{2}}\cos(\frac{9\pi}{8}) & \sqrt{\frac{1}{2}}\cos(\frac{15\pi}{8}) & \sqrt{\frac{1}{2}}\cos(\frac{21\pi}{8}) \end{vmatrix} = \begin{bmatrix} a & a & a & a \\ b & c & -c & -b \\ a & -a & -a & a \\ c & -b & b & -c \end{bmatrix} \quad (6.4)$$

其中， $a = \frac{1}{2}$ ， $b = \sqrt{\frac{1}{2}}\cos(\frac{\pi}{8}) \approx 0.6325$ ， $c = \sqrt{\frac{1}{2}}\cos(\frac{3\pi}{8}) \approx 0.2706$ 。可以看出，具有16个元素的C矩阵中其实只有3个不同的元素值（不计正负号）。

2.二维离散正弦变换

离散正弦变换（Discrete Sine Transform, DST）是一种与傅里叶变换相关的变换，类似离散余弦变换，也是一种实数变换。DST将一个一维实数信号进行奇对称延拓，长度约为原来的两倍，对这个两倍长的奇对称信号进行离散傅里叶变换，由于一个实奇信号的傅里叶变换为纯虚数奇对称函数，所以输出也为两倍长的奇对称虚数序列，各取奇函数的一侧实数部分作为一维离散正弦变换对。

按照和DCT同样的方法可以得到二维离散正弦变换。设一个 $(N-1) \times (N-1)$ 的二维实数信号为 $f(x,y)$ ，其二维离散正弦变换为 $F_s(u,v)$ ，如下所示：

$$F_s(u, v) = k(u)k(v) \sum_{x=1}^{N-1} \sum_{y=1}^{N-1} f(x, y) \sin\left(\frac{(2u-1)x\pi}{2N-1}\right) \sin\left(\frac{(2v-1)y\pi}{2N-1}\right) \quad (6.5)$$

$F_s(u, v)$ 的离散正弦反变换为：

$$f(x, y) = \sum_{u=1}^{N-1} \sum_{v=1}^{N-1} k(u)k(v) F_s(u, v) \sin\left(\frac{(2u-1)x\pi}{2N-1}\right) \sin\left(\frac{(2v-1)y\pi}{2N-1}\right) \quad (6.6)$$

其中 $x, y, u, v = 1, 2, \dots, N-1$, $k(u) = k(v) = 2 / \sqrt{2N-1}$ 。
如 4×4 图像块， $N=5$, $k(u) = k(v) = 2/3$ ，它的DST为：

$$F_s(u, v) = \frac{4}{9} \sum_{x=1}^4 \sum_{y=1}^4 f(x, y) \sin\left(\frac{(2u-1)x\pi}{9}\right) \sin\left(\frac{(2v-1)y\pi}{9}\right) \quad (6.7)$$

其中， $x, y, u, v = 1, 2, \dots, N-1$ 。

3. 二维DST矩阵

$(N-1) \times (N-1)$ 的信号矩阵 f 的二维DST的 $(N-1) \times (N-1)$ 系数矩阵 F_s 可以用矩阵乘法的方式来表示，其正反DST分别如下：

$$F_s = S \cdot f \cdot S^T \quad (6.8)$$

$$f = S^T \cdot F_s \cdot S \quad (6.9)$$

其中， $f(x, y)$ 是二维信号矩阵 f 中第 x 行第 y 列的元素（信号

值) , $F_s(u,v)$ 是DST系数矩阵 F_s 中第u行第v列的元素(系数值)。二维 DST变换矩阵 S , 的第u行第x列元素可表示为 :

$$s_{u,x} = k(u) \sin \frac{(2u-1)x\pi}{2N-1} \quad (6.10)$$

其中 , $x,u=1,2,\dots,N-1$ 。 对 4×4 图像块的DST变换矩阵 S 为 :

$$S = \frac{2}{3} \begin{vmatrix} \sin\left(\frac{\pi}{9}\right) & \sin\left(\frac{2\pi}{9}\right) & \sin\left(\frac{3\pi}{9}\right) & \sin\left(\frac{4\pi}{9}\right) \\ \sin\left(\frac{3\pi}{9}\right) & \sin\left(\frac{6\pi}{9}\right) & \sin\left(\frac{9\pi}{9}\right) & \sin\left(\frac{12\pi}{9}\right) \\ \sin\left(\frac{5\pi}{9}\right) & \sin\left(\frac{10\pi}{9}\right) & \sin\left(\frac{15\pi}{9}\right) & \sin\left(\frac{20\pi}{9}\right) \\ \sin\left(\frac{7\pi}{9}\right) & \sin\left(\frac{14\pi}{9}\right) & \sin\left(\frac{21\pi}{9}\right) & \sin\left(\frac{28\pi}{9}\right) \end{vmatrix} = \begin{bmatrix} a & b & c & d \\ c & c & 0 & -c \\ d & -a & -c & b \\ b & -d & c & -a \end{bmatrix} \quad (6.11)$$

其中， $a = \frac{2}{3} \sin\left(\frac{\pi}{9}\right)$, $b = \frac{2}{3} \sin\left(\frac{2\pi}{9}\right)$, $c = \frac{2}{3} \sin\left(\frac{3\pi}{9}\right)$, $d = \frac{2}{3} \sin\left(\frac{4\pi}{9}\right)$ 。可以看出，具有16个元素的S矩阵中有4个不同的元素值（不计正负号和0）。

6.1.2 量化和量化失真

在视频图像压缩中，在变换后面还有两个压缩数据量的关键步骤，一个是量化，另一个是熵编码。它们都是对被处理数据的另一种表示方式：量化把原数据分为不同的区间，每个区间的多个数值只用一个标号来代表，显然标号的个数要远小于数据的个数，数据量得到压缩。由于标号和数据之间失去了一一对应的关系，因而量化是一种有损信息压缩方式。熵编码按照数据的不同结构，用新的更紧凑的方式来标记，由于标记和原数据之间是一一对应的，因而熵编码是一种无损信息压缩方式。这里主要讨论量化操作。

在图像数据的量化操作中，由于丢弃（忽略）了相当一部分对人眼视觉贡献不大的数据而达到压缩的要求，它是不可逆操作，不可能通过反量化来获得量化前的数据，因而会损失一部分信息。常见量化的方法可以分

为两类，一类是标量量化（Scalable Quantization,SQ），另一类是矢量量化（Vector Quantization,VQ）。标量量化是将图像中样点的取值范围划分成若干区间，每个区间仅用一个数值（标量）“代表”其中所有可能的取值。每个样点的量化取值是一个标量，并且独立于其他样点的取值。矢量量化是将图像的每个像素看成一个n维矢量，将每个n维取值空间划分为若干个子空间，每个子空间用一个n维“代表”矢量来表示该子空间所有的矢量取值。由于矢量量化利用了多个像素之间的关联，一般说来，其压缩率要高于标量量化，代价是计算复杂度要高于标量量化。

目前广泛使用的视频编码标准中，量化环节都是采用标量量化，主要是因为这种方法相对简单，和每个像素或系数直接对应，压缩效果常常也不见得不如矢量量化，所以这里仅介绍有关标量量化以及由量化带来的量化失真的衡量问题。

1. 标量量化

量化过程是数据压缩的有效方法之一，也是图像压缩编码产生失真的主要根源之一。因此量化器的设计总是朝着最好的方向努力，既要获得尽可能高的压缩比，又要尽量减少量化失真，保持尽可能好的图像质量，以此来寻找最佳标量量化器的设计方法。

（1）均方误差最小量化

设计一个性能优良的量化器性能最重要的就是希望量化后的数据和原数据之间的误差越小越好，如果误差用均方误差来计算，就形成了均方误差最小的量化器。例如，对于如预测误差这样非均匀分布的信号而言，其分布大部分集中在“0”附近，这时如果采用非均匀量化，对概率密度大的区域细量化，对概率密度小的区域粗量化，就有可能达到均方误差最小。

下面以预测误差的量化为例来介绍最小均方误差的非均匀量化器的设计方法。如图6.1所示，设预测误差e的值域为 $[e_L, e_H]$ ，量化器的判决电平为 $\{d_i|i=0,\dots,K\}$ ，输出的量化电平为 $\{e_i|i=0,1,\dots,K-1\}$ ，当量化器输入为连续值e时，量化过程 $Q[\cdot]$ 可以用下列关系式表示：

$Q[e] = e_i$, 如果 $d_i \leq e \leq d_{i+1}, i=0,1,2,\dots,K-1$ (6.12)

此时 , 量化误差为 $q_i = e - e_i$, 则量化器输出的总的量化误差的均方值 ε 为各个量化区间量化误差的和 :

$$\varepsilon = E[e - Q(e)]^2 = \sum_{i=0}^{K-1} \int_{d_i}^{d_{i+1}} (e - e_i)^2 \cdot p(e) \cdot de \quad (6.13)$$

式中 , $p(e)$ 是预测误差信号 e 的概率密度函数。

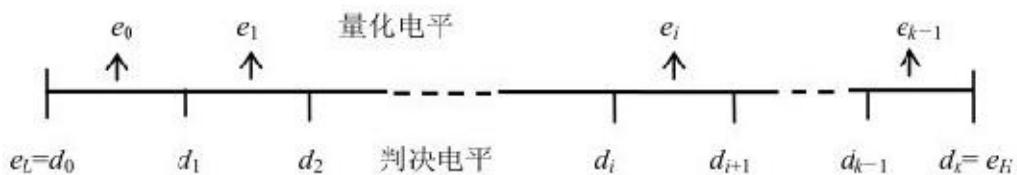


图6.1 量化器的主要参数

在一般的应用场合，量化分层数K较大，每一层的间隔很小，因此可以把p(e)在各量化分层中视为常数，通过直接对d_i和e_i求偏导，并使它们等于零来得到ε的极小值：

$$\frac{\partial \epsilon}{\partial d_i} = 0, \quad \frac{\partial \epsilon}{\partial e_i} = 0, \quad i = 1, 2, \dots, K-1 \quad (6.14)$$

经推导可得判决电平 d_i 和量化电平 e_i ：

$$d_i = \frac{e_{i-1} + e_i}{2}, \quad e_i = \frac{\int_{d_i}^{d_{i+1}} e \cdot p(e) \cdot de}{\int_{d_i}^{d_{i+1}} p(e) \cdot de}, \quad i = 1, 2, \dots, K-1 \quad (6.15)$$

可以看出，均方误差最小量化器的判决电平 d_i 应在相邻两个量化电平的中点，而量化电平 e_i 则在判决区间的形心（重心）上。

(2) 均匀量化

均匀量化可以看作非均匀量化的特殊情况，即当预测误差信号 e 的概率密度函数 $p(e)$ 在整个取值范围内为均匀分布时的结果。设 $p(e) = 1/(e_H - e_L)$ ，代入式(6.15)中间部分后得：

$$e_i = \frac{d_{i+1} + d_i}{2} \quad i = 0, 1, \dots, K-1 \quad (6.16)$$

可见量化电平 e_i 处于判决区间的中间，为两端判决电平值的平均。而各个判决区间因为是均匀量化都是相等间隔的，实际上判决电平也是在两边量化电平的中间。显然，均匀量化要比非均匀量化简单得多，所以在视频编码中得到了最广泛的应用。

当然，在有些情况下，均匀量化并不能取得最好的效果，即不能获得最小量化误差。或者说，在相同的量化分层条件下，它的量化均方误差值比非均匀量化大；在同样的均方误差条件下，它需要比非均匀量化器更多的量化分层。

2.量化信噪比

对量化器性能的衡量，除量化误差的均方值 ε 以外，还可以从量化误差相当于引入噪声出发，来分析降质图像的信噪比，即量化信噪比。当采用最佳预测时，预测误差信号的概率分布 $p(e)$ 可以用第1章所述的Laplace分布近似：

$$p(e) = \frac{1}{\sqrt{2}\sigma_e} \exp\left(-\frac{\sqrt{2}|e|}{\sigma_e}\right) \quad (6.17)$$

当误差信号的动态范围 $[e_L, e_H]$ 比预测误差信号的均方根 σ_e 大得多时，可以得到量化误差的均方根 σ_q 的近似表示：

$$\sigma_q^2 \approx \left(\frac{9}{2K^2}\right)\sigma_e^2 \quad (6.18)$$

其中， K 是量化分层总数，常取 $K=2^n$ 。于是，最佳预测的量化信噪比为：

$$\left(\frac{S}{N}\right)_q = 10\lg\left(\frac{\sigma^2}{\sigma_q^2}\right) = 10\lg\left(\frac{2K^2}{9}\right) + 10\lg\left(\frac{\sigma^2}{\sigma_e^2}\right) \approx -6.5 + 6n + 10\lg\left(\frac{\sigma^2}{\sigma_e^2}\right) \quad (\text{dB}) \quad (6.19)$$

上式中， σ 为图像信号的均方根。从该式中可以看出，数字图像信号的量化精度，体现在 n 上，直接决定了量化信噪比的大小，每增加1比特精度，信噪比就大约增加6dB。

6.2 H.264/AVC的变换与量化

在早期的视频编码中，DCT 变换和量化基本上是两个独立的处理过程。但在 H.264/AVC 中，传统的 4×4 二维DCT由近似DCT的整数变换取代，并将变换的一部分乘法移到量化环节中一并完成，这样变浮点运算为整数运算，节省了计算量，同时还能基本保证编码图像的质量。如果输入块是色度块或帧内 16×16 预测模式的亮度块，为了进一步减少传输各系数块直流分量的比特花费，还将宏块中各 4×4 块的整数余弦变换的直流分量组合起来再进行Hadamard 变换，进一步压缩码率。H.264/AVC的变换和量化的总体过程如图6.2所示。

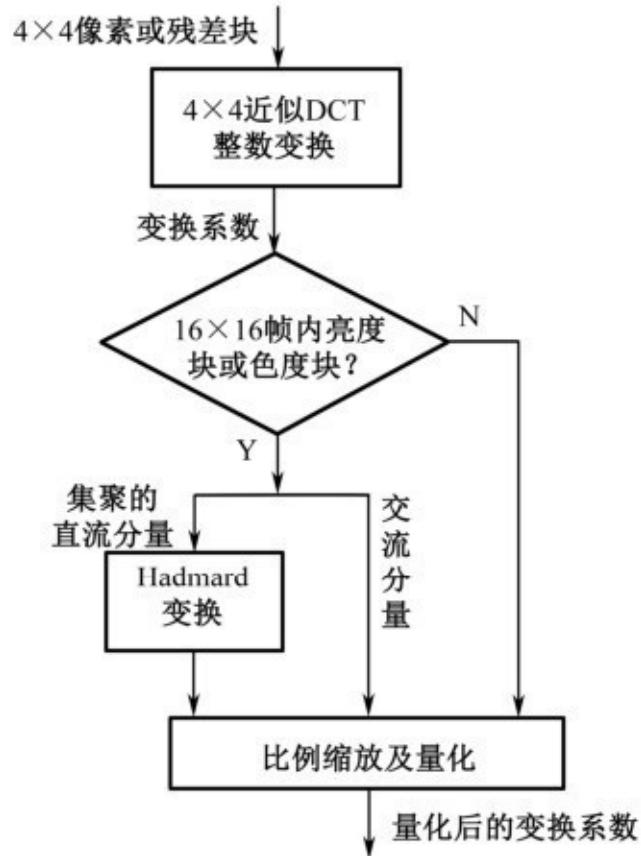


图6.2 整数变换及量化过程

6.2.1 4×4 整数DCT变换

在式(6.4)的DCT变换中，会产生两个问题：第一，由于 a 、 b 、 c 为实数，需要进行浮点数操作，从而造成运算的复杂性；第二，由于变换核都是无理数，而有限精度的浮点数不可能精确地表示无理数，再加上浮点数的运算可能会引入舍入误差，这就使得反变换的输出结果和正变换的输

入不一致。

为了克服这些问题，H.264/AVC 采用了一种用整数进行近似 DCT 变换的方法——整数变换。以二维 4×4 的变换为例，采用基于 4×4 块的整数操作而不是实数运算，使得变换操作仅用16位整数加减和移位操作就可以完成，这样既降低了计算的复杂度，又避免了正反变换的不匹配，能够得到与 4×4 DCT变换类似的变换效果，而由此带来的变换性能减少的微乎其微。

可以从标准的DCT公式推导出整数DCT正变换的公式，仍然以 4×4 的图像f为例，由式(6.1)可知其二维DCT的

$F_c = CfC^T$, $a=1/2$, $b=0.6533$, $c=0.3827$ 。为计算简单，取近似 $b \approx 2c$ ，则矩阵C可近似为：

$$C = \begin{bmatrix} a & a & a & a \\ b & c & -c & -b \\ a & -a & -a & a \\ c & -b & b & -c \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 2 & 1 & -1 & -2 \\ 1 & -1 & -1 & 1 \\ 1 & -2 & 2 & -1 \end{bmatrix} \otimes \begin{bmatrix} a & a & a & a \\ b/2 & b/2 & b/2 & b/2 \\ a & a & a & a \\ b/2 & b/2 & b/2 & b/2 \end{bmatrix} = C_f \otimes E_f \quad (6.20)$$

其中符号“ \otimes ”表示两个相同尺寸矩阵的对应元素相乘，结果仍然是同

尺寸的矩阵。

f 的二维DCT变换结果 F_c 为 4×4 的DCT系数矩阵：

$$\begin{aligned}
 F_c &= C \cdot f \cdot C^T = (C_f \otimes E_f) \cdot f \cdot (C_f \otimes E_f)^T = (C_f \cdot f \cdot C_f^T) \otimes E_f \otimes E_f^T = (C_f \cdot f \cdot C_f^T) \otimes E \\
 &= \left(\begin{bmatrix} 1 & 1 & 1 & 1 \\ 2 & 1 & -1 & -2 \\ 1 & -1 & -1 & 1 \\ 1 & -2 & 2 & -1 \end{bmatrix} f \begin{bmatrix} 1 & 2 & 1 & 1 \\ 1 & 1 & -1 & -2 \\ 1 & -1 & -1 & 2 \\ 1 & -2 & 1 & -1 \end{bmatrix} \right) \otimes \begin{bmatrix} a^2 & ab/2 & a^2 & ab/2 \\ ab/2 & b^2/4 & ab/2 & b^2/4 \\ a^2 & ab/2 & a^2 & ab/2 \\ ab/2 & b^2/4 & ab/2 & b^2/4 \end{bmatrix} \\
 &\quad (6.21)
 \end{aligned}$$

需要注意的是，此处的变换已经不是真正的DCT，仍然称其为DCT变换只是因为它是由DCT推导而来，且性能和DCT相仿。这种近似DCT变换中只有整数的加法、减法和移位（乘以2）运算（除了E矩阵），故H.264/AVC称上式中 $(C_f \cdot f \cdot C_f^T)$ 部分为整数变换，也是实际编码器的DCT单元的输出：

$$W = (\mathbf{C}_f \cdot f \cdot \mathbf{C}_f^T) \quad (6.22)$$

这样做的结果是把用整数变换替代DCT变换所产生的误差纳入到E矩阵中了。由于E矩阵的每个元素都要和整数变换的相应的系数作乘法运算，而在实际应用中，DCT运算的后续处理往往是量化处理，量化处理必须要对每个DCT系数作除法（乘法），这样就可以将E矩阵的乘法运算合并到量化中完成，而只增加很少的运算量。

6.2.2 变换系数的量化

1. 量化和反量化

在H.264/AVC编码端，量化器将每个DCT变换系数 $F_{u,v}$ 按照下式关系映射成量化值 $F_{u,v}^Q$ ：

$$F_{u,v}^Q = \text{round}\left(\frac{F_{u,v}}{Q_{\text{step}}}\right) \quad (6.23)$$

其中， $F_{u,v}$ 为残差DCT系数输入， Q_{step} 为量化步长， $F_{u,v}^Q$ 为 $F_{u,v}$ 的量化值， $\text{round}(\cdot)$ 为取整函数，其输出为与输入实数最近的整数。

在解码端，解码器需要将量化值还原为系数值，这就是反量化过程。

反量化获得的系数值为 $F_{u,v}^{IQ}$ ：

$$F_{u,v}^{IQ} = F_{u,v}^Q \cdot Q_{\text{step}} \quad (6.24)$$

在量化和反量化过程中，由于量化中 $\text{round}(\cdot)$ 函数的取整作用， $F_{u,v}^{IQ}$ 只能是 $F_{u,v}$ 的近似值，量化步长 Q_{step} 决定了 $F_{u,v}^{IQ}$ 和 $F_{u,v}$ 的近似程度，从而也决定了量化器的编码压缩率及重建图像的精度。如果 Q_{step} 比较大，量化较粗，量化值 $F_{u,v}^Q$ 的取值种类较少，相应的编码长度较短，压缩率高，但反量化时失真大，损失较多的图像细节信息；如果 Q_{step} 比较小，量化较细， $F_{u,v}^Q$ 的取值种类较多，相应的编码长度也较长，但反量化时失真小，图像细节信息损失较少。比较理想的情况是编码器能够根据变换系数的实际动态范围自动改变 Q_{step} 值，在编码长度和图像精度之间折中，达到整体最佳效果。

2.量化步长 Q_{step} 和量化参数QP

在H.264/AVC中，量化步长 Q_{step} 共有52个值，编码传送时，不用传送实际的量化步长的值，只要传它们对应的编号——量化参数QP就可以了。QP的值从0到51，和 Q_{step} 一一对应，如表6.1所示。当QP取最小值0时代表最精细的量化，当QP取最大值51时代表最粗糙的量化。QP每增加6, Q_{step} 增加一倍。对于色度分量的量化，一般使用与相应亮度分量同样的量化步长。为了避免在较高量化步长时的出现颜色量化人工效应，H.264/AVC把色度的QP最大值大约限制在亮度QP最大值的80%范围内，即色度QP的最大值是39。

表6.1 H.264/AVC量化参数和量化步长的关系

| QP | Q_{step} | QP | Q_{step} | QP | Q_{step} | QP | Q_{step} |
|----|------------|----|------------|----|------------|----|------------|
| 0 | 0.6250 | 13 | 2.75 | 26 | 13 | 39 | 56 |
| 1 | 0.6875 | 14 | 3.25 | 27 | 14 | 40 | 64 |
| 2 | 0.8125 | 15 | 3.50 | 28 | 16 | 41 | 72 |
| 3 | 0.8750 | 16 | 4 | 29 | 18 | 42 | 80 |
| 4 | 1 | 17 | 4.5 | 30 | 20 | 43 | 88 |
| 5 | 1.125 | 18 | 5.0 | 31 | 22 | 44 | 104 |
| 6 | 1.250 | 19 | 5.5 | 32 | 26 | 45 | 112 |
| 7 | 1.375 | 20 | 6.5 | 33 | 28 | 46 | 128 |
| 8 | 1.625 | 21 | 7.0 | 34 | 32 | 47 | 144 |
| 9 | 1.750 | 22 | 8.0 | 35 | 36 | 48 | 160 |
| 10 | 2 | 23 | 9.0 | 36 | 40 | 49 | 176 |
| 11 | 2.250 | 24 | 10 | 37 | 44 | 50 | 208 |
| 12 | 2.500 | 25 | 11 | 38 | 52 | 51 | 224 |

3.包含缩放的量化

据前所述，H.264/AVC量化过程中还要同时完成DCT变换中“ $\otimes E$ ”乘法运算，它可以表示为：

$$F_{u,v}^Q = \text{round}\left(\frac{W_{u,v} \cdot E_{u,v}}{Q_{\text{step}}}\right) \quad (6.25)$$

其中， $W_{u,v}$ 是式(6.22)表示的DCT单元输出矩阵W中的元素， $E_{u,v}$ 是矩阵E中的对应元素，根据系数点在变换块中的位置(u,v)取值，对于 4×4 矩阵其值可参见式(6.21)，在矩阵E的(0,0)、(0,2)、(2,0)、(2,2)处为 $a = 0.25$ ，在(1,1)、(1,3)、(3,1)、(3,3)处为 $b = 0.1581$ ，在其他位置处为 $ab/2 \approx 0.1581$ 。

为了将量化中的除法运算变为移位运算，将式(6.25)中 $E_{u,v}/Q_{\text{step}}$ 扩大 $2^{q\text{bit}}$ 倍，或右移qbit位，成为一个无须除法的新变量MF：

$$MF = \left(\frac{E_{u,v}}{Q_{\text{step}}}\right) \cdot 2^{q\text{bit}} \quad (6.26)$$

将MF代入(6.25)的量化式后得：

$$F_{u,v}^Q = \text{round}\left(W_{u,v} \frac{\text{MF}}{2^{q\text{bit}}}\right) \quad (6.27)$$

为了进一步简化计算，利用量化步长随量化参数每增加6而增加一倍的性质，可设：

$$q\text{bit} = 15 + \text{floor}\left(\frac{\text{QP}}{6}\right) \quad (6.28)$$

其中， $\text{floor}(\cdot)$ 为取整函数，其输出为不大于输入实数的最大整数。这样，MF可以取整数，如表6.2所示。表6.2只列出对应QP值为0到5的MF值，对于QP值大于5的情况，只是qbit值随QP值每增加6而增加1,2^{qbit}扩大一倍，而 Q_{step} 也扩大一倍，因此对应的MF值不变。这样，量化过程则为整数运算，MF的值只有6种，并且可以避免使用除法。

表6.2 H.264/AVC中乘法因子MF 值

| QP | (0, 0) | | (1, 1) | | 其他位置 |
|----|--------|--------|--------|--------|------|
| | (0, 2) | (2, 0) | (1, 3) | (3, 1) | |
| 0 | 13107 | | 3243 | | 8065 |
| 1 | 11916 | | 4660 | | 7490 |
| 2 | 10082 | | 4194 | | 6254 |
| 3 | 9362 | | 3647 | | 5825 |
| 4 | 8192 | | 3355 | | 5243 |
| 5 | 7282 | | 2893 | | 4559 |

采用整数运算，具体量化过程为：

$$\begin{aligned} |F_{u,v}^Q| &= \text{round}(|W_{u,v}| \cdot MF + f) \gg qbit \\ \text{sign}(F_{u,v}^Q) &= \text{sign}(W_{u,v}) \end{aligned} \quad (6.29)$$

其中，“ \gg ”为右移运算，右移一位完成整数除以2, $\text{sign}(\cdot)$ 为符号函数， f 为偏移量，它的作用是改善恢复图像的视觉效果，例如，对帧内预测图像块 f 取 $2^{\text{qbit}}/3$ ，对帧间预测图像块 f 取 $2^{\text{qbit}}/6$ 。

4.16×16亮度块的直流系数量化

对于 16×16 预测的亮度块，需要将其中16个 4×4 整数变换后系数矩阵中的直流分量抽出，按该 4×4 块在对应宏块中的排序，组成新的直流系数 4×4 矩阵，对该系数矩阵进行Hadamard变换，再对变换结果进行量化。

亮度变换系数 16×16 的宏块位置对应的色度宏块尺寸为 8×8 ，它包含色度系数Cr及Cb的4个 4×4 块。和亮度信号的直流矩阵相类似，色度Cr或Cb块的直流分量为 2×2 矩阵，对它进行Hadamard 变换后再进行量化。

6.3 HEVC残差的整数变换

和以往固定尺寸的变换不同，在HEVC中，CB到变换块TB也是基于四叉树的结构划分，CB是“树根”，TB是“树叶”，最大的TB就是CB。HEVC支持近似离散余弦变换（DCT）的整数变换，尺寸范是从 4×4 到 32×32 样点。对于 4×4 亮度帧内预测残差的变换，还可使用另一种近似离散正弦变换（DST）的整数变换。

6.3.1 残差四叉树（RQT）

从CB到TB四叉树划分主要是服务于残差的变换运算，所以又称为残差四叉树（Residual Quad Tree, RQT）。图6.3所示的一个 32×32 残差CB包括从0~12共13个不同尺寸的TB，这些不重叠的TB单元覆盖了整个CB。较大的变换块尺寸，适于较大的图像内容较为平坦的区域，可提供较好的频率分辨率。而较小的变换块尺寸，适于图像内容较为复杂的区域，提供比较好的空间分辨率。

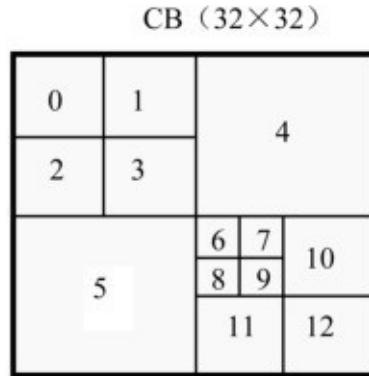


图6.3 CB到TB的RQT划分实例

1.RQT的参数和划分

TB的残差四叉树（RQT）的具体划分可由3个参数定义：最大树的深度 d_{max} , TB的最小允许变换尺寸 T_{min} 和最大允许变换尺寸 T_{max} 。 d_{max} 的变化范围从0到3。变换块宽度 T_{max} 和 T_{min} 用2进制对数表示，变化范围从2到5，对应变换块所允许的尺寸从 4×4 到 32×32 样点。

RQT最大允许的深度 d_{max} 限制CB的划分的次数。 $d_{max}=0$ 的值意味着CB不作任何进一步的划分，这时CB只包含一个TB。另外还有几种特殊的需求划分的情况：

(1) 根CB的尺寸是 64×64 ，最大的深度 $d_{max}=0$ ，但HEVC规定的最大的TB尺寸等于 32×32 ，即 $T_{max}=5$ 。在这种情况下，需定义CB至少划分1次来适合对变换尺寸的限制。

(2) 当 $d_{max}=0$ 时，出现了另一种需要划分的情况，当CB使用运动补偿的时域预测，而相应的预测划分由不止一个PB组成。

(3) 当 $d_{max} > 0$, CB为帧间预测块, 由于RQT的划分独立于PB的划分, 就可能导致变换块TB覆盖若干个来自同一个根CB的不同的PB。

然而, 对帧内预测的CB, 若干PB被一个TB覆盖是不可能的, 导致第2和第3种情况不会出现。因为对于帧内预测, 在CB内的顺序相邻的PB需要被完全重建后成为当前的PB预测信号, 一个TB就不能跨越若干帧内PB。因此, 在帧内的情况下就强迫划分CB为若干TB, 使得这些TB中没有覆盖多个PB的情况。当然, 这些TB每一个也都可能被进一步划分。

对于同一TB, 亮度分量的RQT结构也适用于色度分量, 形成亮度和色度分量相同的划分。然而, 对这一规则的例外是在4:2:0彩色格式的 4×4 亮度TB视频信号中, 这将导致相应 2×2 色度TB的划分, HEVC中不支持 2×2 的色度TB。因此, RQT允许划分一个 8×8 亮度TB, 但不允许划分相应的 4×4 色度TB, 这就导致在此特殊情况下, 亮度和色度的相应RQT结构的不同。

RQT的3个参数, 即最大RQT深度 d_{max} , 最小和最大的变换尺寸 T_{min} 和 T_{max} , 都需在比特流的序列参数集(SPS)中传送。对于RQT深度, 可为帧内和帧间预测CU指定和标记不同的值。必须记住, 在强迫划分时, 传送的深度值不必对应于划分的次数, 如上所述, 所形成的TB尺寸大于最大的变换尺寸。

2. 快速RQT结构确定

不同的RQT结构的数量随着最大RQT深度的增加而呈指数性增长。为减少在编码端估计不同RQT结构的计算复杂度, 目前已有多种快速方法出现。例如, 启发式提前跳过技术就是快速确定RQT结构的一种有效方法。对每种RQT深度, 将尚未量化的变换系数的绝对值和一个阈值比较, 当所有未量化的变换系数低于给定的阈值, 则在给定RQT节点上跳过进一步的划分。阈值选择和量化的QP有关: 对于QP值低于24, 相应的阈值可选为量化器步长尺寸的125%; 对QP值高于48, 阈值可选为量化步长尺寸的50%; 对QP值在24~48, 所有的阈值可采用在量化步长尺寸的125%和50%之间的线性变换。

6.3.2 整数DCT变换

和 H.264/AVC 类似，HEVC 采用了对预测残差进行近似的整数离散余弦变换（DCT）。但为适应较大的编码单元而进行了改进，在一个编码单元（CU）内进行变换运算时，共有4种大小的对称DCT变换尺寸： 32×32 、 16×16 、 8×8 和 4×4 。每一种大小的DCT变换都有一个相对应的同样大小的整数变换系数矩阵。大块的变换能够提供更好的能量集中效果，并能在量化后保存更多的图像细节，但是却带来更多的振铃效应。因此，根据当前块像素数据的特性，自适应的选择变换块大小可以得到更好的效果。

HEVC的变换运算的顺序和H.264/AVC不同，变换时首先进行列运算，然后再进行行运算。HEVC的整数变换的基矢量具有相同的能量，不需要对它们进行调整或补偿，而且对DCT的近似性要比H.264/AVC好。

1.4×4整数DCT

为了说明HEVC的 4×4 整数DCT，重写前面的式（6.4）， 4×4 的DCT变换矩阵C为：

$$C = \begin{bmatrix} a & a & a & a \\ b & c & -c & b \\ a & -a & -a & a \\ c & -b & b & -c \end{bmatrix} \quad (6.30)$$

其中， $a=0.5, b=0.6325, c=0.2706$ 。对 a, b, c 各自乘128，同时为了保证系数的正交性，进行了一点微调，即 $a=64, b=83.6 \approx 84, c=34.6 \approx 35$ ，而不是像H.264/AVC对系数作了比较大的近似。这样得到 4×4 近似DCT整数变换矩阵：

$$\mathbf{C} = \begin{bmatrix} 64 & 64 & 64 & 64 \\ 83 & 36 & -36 & -83 \\ 64 & -64 & -64 & 64 \\ 36 & -83 & 83 & -36 \end{bmatrix} \cdot \frac{1}{128} = \mathbf{C}_f \cdot E_f \quad (6.31)$$

其中， $E_f = (1/128)$ ，可见，HEVC的DCT比H.264/AVC的更加简单。
 4×4 图像块f的二维DCT为：

$$\begin{aligned}
F_c &= (\mathbf{C}_f \cdot f \cdot \mathbf{C}_f^\top) \otimes \mathbf{E}_j \otimes \mathbf{E}_j^\top = \mathbf{W} \otimes \mathbf{E} \\
&= \left(\begin{bmatrix} 64 & 64 & 64 & 64 \\ 83 & 36 & -36 & -83 \\ 64 & -64 & -64 & 64 \\ 36 & -83 & 83 & -36 \end{bmatrix} f \begin{bmatrix} 64 & 83 & 64 & 36 \\ 64 & 36 & -64 & -83 \\ 64 & -36 & -64 & 83 \\ 64 & -83 & 64 & -36 \end{bmatrix} \right) \cdot \frac{1}{128} \cdot \frac{1}{128} \quad (6.32)
\end{aligned}$$

2.32×32整数DCT

HEVC支持从4×4到32×32共4种不同尺寸的整数DCT，但为简单起见，根据DCT变换的性质，HEVC只定义了一个32×32点的2维整数DCT矩阵（限于篇幅，这里没有列出，具体的数据可见HEVC第2版文档的8.6.4.2节）。这个矩阵是由32个1维的32点（相当于1行）数列叠加组成一个2维数组。其他16×16、8×8、4×4矩阵的系数都可从这个32×32矩阵系数的亚抽样得到，使得HEVC仅使用一个32×32样点变换核支持所有尺寸的变换。

这个32×32矩阵有一个重要的特性：左半部16列和右半部16列是对称的，但对称的方式有所不同，偶数行（第0,2,...）是标准的对称，而奇数行是反号对称的，即右半部数值和左半部对应的数值符号相反。实际上不仅仅是32×32矩阵，其他16×16、8×8、4×4的DCT矩阵都具有这样的性质。这一性质，可以在存储矩阵系数时节省大约一半的存储量。

由32×32矩阵的亚抽样可得到其他尺寸的整数变换，例如，长度为16×16的变换H₁₆矩阵是由32×32矩阵的偶数行（第0,2,...,30行）的前16个（0,1,...,15）元素组成：

$$\begin{aligned}
 H_{16} = & \begin{bmatrix}
 64 & 64 & 64 & 64 & 64 & 64 & 64 & 64 & 64 & 64 & 64 & 64 & 64 & 64 & 64 & 64 \\
 90 & 87 & 80 & 70 & 57 & 43 & 25 & 9 & -9 & -25 & -43 & -57 & -70 & -80 & -87 & 90 \\
 89 & 75 & 50 & 18 & -18 & -50 & -75 & -89 & -89 & -75 & -50 & -18 & 18 & 50 & 75 & 89 \\
 87 & 57 & 9 & -43 & -80 & -90 & -70 & -25 & 25 & 70 & 90 & 80 & 43 & -9 & -57 & -87 \\
 83 & 36 & -36 & -83 & -83 & -36 & 36 & 83 & 83 & 36 & -36 & -83 & -83 & -36 & 36 & 83 \\
 80 & 9 & -70 & -87 & -25 & 57 & 90 & 43 & -43 & -90 & -57 & 25 & 87 & 70 & -9 & -80 \\
 75 & -18 & -89 & -50 & 50 & 89 & 18 & -75 & -75 & 18 & 89 & 50 & -50 & -89 & -18 & 75 \\
 70 & -43 & -87 & 9 & 90 & 25 & -80 & -57 & 57 & 80 & -25 & -90 & -9 & 87 & 43 & -70 \\
 64 & -64 & -64 & 64 & 64 & -64 & -64 & 64 & 64 & -64 & -64 & 64 & 64 & -64 & -64 & 64 \\
 57 & -80 & -25 & 90 & -9 & -87 & 43 & 70 & -70 & -43 & 87 & 9 & -90 & 25 & 80 & -57 \\
 50 & -89 & 18 & 75 & -75 & -18 & 89 & -50 & -50 & 89 & -18 & -75 & 75 & 18 & -89 & 50 \\
 43 & -90 & 57 & 25 & -87 & 70 & 9 & -80 & 80 & -9 & -70 & 87 & -25 & -57 & 90 & -43 \\
 36 & 83 & 83 & 36 & 36 & 83 & 83 & 36 & 36 & 83 & 83 & 36 & 36 & 83 & 83 & 36 \\
 25 & -70 & 90 & -80 & 43 & 9 & -57 & 87 & -87 & 57 & -9 & -43 & 80 & -90 & 70 & -25 \\
 18 & -50 & 75 & -89 & 89 & -75 & 50 & -18 & -18 & 50 & -75 & 89 & -89 & 75 & -50 & 18 \\
 9 & 25 & 43 & 57 & 70 & 80 & 87 & 90 & 90 & 87 & 80 & 70 & 57 & 43 & 25 & 9
 \end{bmatrix} \quad (6.33)
 \end{aligned}$$

在这个 16×16 的矩阵中也可以看出它的偶数行的对称性和奇数行的反号对称性。对 8×8 和 4×4 的的变换矩阵 H_8 和 H_4 可以分别使用 H_{16} 的第0,2,4,...,14行的前8项和第0,4,8,12行的前4项导出，如下式(6.34)所示。其中 H_4 和式(6.31)的系数是一致的。

$$\begin{aligned}
 H_8 &= \begin{bmatrix} 64 & 64 & 64 & 64 & 64 & 64 & 64 & 64 \\ 89 & 75 & 50 & 18 & -18 & -50 & -75 & -89 \\ 83 & 36 & -36 & -83 & -83 & -36 & 36 & 83 \\ 75 & -18 & -89 & -50 & 50 & 89 & 18 & -75 \\ 64 & -64 & -64 & 64 & 64 & -64 & -64 & 64 \\ 50 & -89 & 18 & 75 & -75 & -18 & 89 & -50 \\ 36 & -83 & 83 & -36 & -36 & 83 & -83 & 36 \\ 18 & -50 & 75 & -89 & 89 & -75 & 50 & -18 \end{bmatrix} \\
 H_4 &= \begin{bmatrix} 64 & 64 & 64 & 64 \\ 83 & 36 & -36 & -83 \\ 64 & -64 & -64 & 64 \\ 36 & -83 & 83 & -36 \end{bmatrix} \quad (6.34)
 \end{aligned}$$

3. 变换跳过模式

HEVC的 4×4 变换编码中的另一个例外情况就是变换跳过模式。这种模式能使帧内或帧间编码的变换环节“旁路”，这种方式对编码屏幕和图形内容效果良好。相应的功能可由图像参数集（Picture Parameter Set,PPS）中的一个特别的标志来确定，而且一旦这个标志有效，就为每个 4×4 TU传送一个变换跳过标志（transform_skip_flag），标明变换是否被旁路。

6.3.3 4×4 整数DST变换

在帧内预测时，当前块的预测参考为紧邻该块左边和上边的已编码块的数据。像素间的相关性是随距离增大而减小的，因此预测块越是左边和上边的像素的预测越是准确，预测误差越小。也就是沿着编码块水平方向向右，预测误差逐渐增大；沿着垂直方向向下，预测误差也逐渐增大。这样的预测误差数据的分布和DCT的余弦基函数并不符合，因为余弦函数的特点是起始值最大，而后逐渐减小。而这一数据分布特点正好和离散正弦变换（DST）的正弦基函数比较一致，在起始处最小，然后逐渐增大。根

据变换编码的特点，变换基函数和信号的相似程度越高，其压缩率越大，因此HEVC对帧内预测 4×4 亮度残差TB的所有的模式都采用可分离整数近似离散正弦变换（DST），以利于图像压缩率的提高。

和近似的离散余弦整数变换导出方法类似，也可从离散正弦变换（DSC）可推导出近似的离散正弦整数变换。对于 4×4 的DST核，根据前面二维DST定义的式（6.7），采用和DCT类似的整数化过程，得到DST的变换矩阵如下：

$$S = \begin{bmatrix} 29 & 55 & 74 & 84 \\ 74 & 74 & 0 & -74 \\ 84 & -29 & -74 & 55 \\ 55 & -84 & 74 & -29 \end{bmatrix} \cdot \frac{1}{128} = S_f \cdot E_f \quad (6.35)$$

HEVC将式（6.35）这样的 4×4 变换矩阵用于亮度 4×4 残差块的帧内预测，则 4×4 图像块的DST变换为：

$$\begin{aligned}
F_g &= (S_f \cdot f \cdot S_f^T) \otimes E_f \otimes E_f^T \\
&= \begin{bmatrix} 29 & 55 & 74 & 84 \\ 74 & 74 & 0 & -74 \\ 84 & -29 & -74 & 55 \\ 55 & -84 & 74 & -29 \end{bmatrix} f \begin{bmatrix} 29 & 74 & 84 & 55 \\ 55 & 74 & -29 & -84 \\ 74 & 0 & -74 & 74 \\ 84 & -74 & 55 & -29 \end{bmatrix} \cdot \frac{1}{128} \cdot \frac{1}{128} \quad (6.36)
\end{aligned}$$

以 4×4 整数DCT为例，如果不计正负号，变换核共有3个不同的量83、64和36，而DST的变换核则有5个不同的量，0、29、55、74和84，因此DST较DCT计算量稍大。但是 4×4 整数DST在帧内预测编码中大约可提供1%的比特率降低。综合考虑结果，HEVC中DST类变换的使用只限于帧内预测的 4×4 亮度残差块，因为在其他尺寸的情况下，发现其编码效率的改进是微不足道的，反而需要增加变换类型标识等耗费。

6.4 HEVC变换系数的量化

量化处理本质上就是用量化步长除以变换系数，得到变换系数更简单的表示，从而获得数据量的压缩。但量化是压缩编码产生失真的主要根源，量化步长的值越大表示量化越粗，所产生的码率越低，当然带来的失真也会越大。反之，量化步长的值越小，表示量化越细，则失真也越小，但所产生的码率也越高。因此选择恰当的量化步长，使失真和码率之间达到最好的平衡就成了量化环节的关键问题。如前所述，HEVC 的量化实际上还包含了整数变换遗留的变换系数的伸缩操作。

6.4.1 量化参数和量化步长

HEVC 中的量化也采用 H.264/AVC 中定义的均匀重建量化（Uniform Reconstruction Quantizers,URQ）方式，属于非线性标量量化。对8比特的视频信号，利用量化参数（QP）控制每个编码块的量化步长 Q_{step} 。具体的量化参数用变量QP表示，它和 Q_{step} 之间呈近似指数关系，可用下式表示：

$$Q_{step}(QP) \approx (2^{\frac{1}{6}})^{QP-4} \quad (6.37)$$

由于QP为整数， $2^{1/6} \approx 1.125$ ，QP每增加1， Q_{step} 增加1.125倍，QP每增加6， Q_{step} 则在原来的基础上翻倍。根据8比特信号的取值范围，亮度信号值的范围定义在0~51，对应的量化步长 Q_{step} 值在0.625~224，跨度相当大。相应的色度信号的QP范围定义在0~45，主要是防止量化步长过大时引起重建图像的彩色漂移现象。色度信号的量化参数QP在0~29和亮度信号所对应的量化步长是一致的，从30开始，两者之间就有了差异，具体亮度和色度的QP和 Q_{step} 之间的关系如表6.3所示。

为达到向解码器传送量化参数的目的，一个起始的QP值首先在PPS中传送，然后增量QP值可在条和CU层传送。除了这种基本的量化步长控制，为支持感知协调、依赖于频率的量化等功能，HEVC还支持选用加权量化矩阵。

表6.3 HEVC的量化参数和量化步长

| 亮度 QP | 色度 Q_{step} | 亮度 QP | | | 亮度 QP | | | 亮度 QP | | |
|----------|------------------|----------|------------------|------------|----------|------------------|------------|----------|------------------|------------|
| | | 亮度 QP | 色度 Q_{step} | Q_{step} | 亮度 QP | 色度 Q_{step} | Q_{step} | 亮度 QP | 色度 Q_{step} | Q_{step} |
| 0 | 0.625 | 13 | 2.813 | 2.813 | 26 | 12.75 | 12.75 | 39 | 35 | 57 |
| 1 | 0.703 | 14 | 3.138 | 3.138 | 27 | 14.25 | 14.25 | 40 | 36 | 64 |
| 2 | 0.797 | 15 | 3.562 | 3.562 | 28 | 16 | 16 | 41 | 36 | 72 |
| 3 | 0.890 | 16 | 4 | 4 | 29 | 18 | 18 | 42 | 37 | 80 |
| 4 | 1 | 17 | 4.50 | 4.50 | 30 | 20 | 20 | 43 | 37 | 90 |
| 5 | 1.125 | 18 | 5 | 5 | 31 | 20 | 22.50 | 44 | 38 | 102 |
| 6 | 1.250 | 19 | 5.625 | 5.625 | 32 | 21 | 25.50 | 45 | 39 | 114 |
| 7 | 1.406 | 20 | 6.375 | 6.375 | 33 | 22 | 28.50 | 46 | 40 | 128 |
| 8 | 1.594 | 21 | 7.125 | 7.125 | 34 | 23 | 32 | 47 | 41 | 144 |
| 9 | 1.781 | 22 | 8.0 | 8.0 | 35 | 23 | 36 | 48 | 42 | 160 |
| 10 | 2 | 23 | 9.0 | 9.0 | 36 | 24 | 40 | 49 | 43 | 180 |
| 11 | 2.250 | 24 | 10 | 10 | 37 | 24 | 45 | 50 | 44 | 204 |
| 12 | 2.500 | 25 | 11.25 | 11.25 | 38 | 25 | 51 | 51 | 45 | 224 |

6.4.2 量化和反量化计算

HEVC的量化操作以TU为基本处理单位，分别对TU中的亮度分量和色度分量进行，对一个TU中的所有的变换系数用统一的QP值进行均匀地量

化和反量化。但由于HEVC中整数变换的系数矩阵与H.264/AVC的不同，因而采用的量化系数也不同。通常可采用率失真优化的量化（Rate Distortion Optimized Quantization,RDOQ）技术，依据编码效率为TU选择最佳的量化参数，可提高亮度部分的编码效率4%~5%，但增加了编码端的计算复杂度。

1.量化计算

类似前述的H.264/AVC量化过程，HEVC的基本量化过程中还要同时完成整数DCT变换中伸缩因子“ $\otimes E$ ”的乘法运算，它可以表述为：

$$F_{u,v}^Q = \text{round}\left(\frac{W_{u,v} \cdot E_{u,v}}{Q_{\text{step}}}\right) \quad (6.38)$$

其中， $W_{u,v}$ 是式（6.32）矩阵W中的元素（以 4×4 的DCT为例）， $E_{u,v}$ 是矩阵E中的对应元素，实际上HEVC的所有的 $E_{u,v}$ 都为一个相同的常数E，比H.264/AVC还简单。

为了将量化中的除法运算变为移位运算，将 $1/Q_{\text{step}}$ 扩大 $2^{q_{\text{bit}}}$ 倍，或右移

qbit位，成为一个无需除法的新变量MF_q。设：

$$qbit = 14 + \text{floor}(QP / 6) \quad (6.39)$$

考慮到式(6.37)中QP和QP_{step}的关系，则变量MF_q为：

$$\begin{aligned}
 \text{MF}_q &= \frac{2^{q\text{bit}}}{Q_{\text{step}}} = \frac{2^{14+\text{floor}(QP/6)}}{Q_{\text{step}}} = \frac{2^{14+\text{floor}(QP/6)}}{2^{(QP-4)/6}} \\
 &= 2^{\frac{14+4}{6}} \cdot 2^{\text{floor}(QP/6)} \cdot 2^{-(QP-4)/6} = 26214 \cdot 2^{(QP\%6)/6}
 \end{aligned} \tag{6.40}$$

$$= \begin{cases} 26214 & QP\%6 = 0 \\ 23302 & QP\%6 = 1 \\ 20560 & QP\%6 = 2 \\ 18396 & QP\%6 = 3 \\ 16384 & QP\%6 = 4 \\ 14546 & QP\%6 = 5 \end{cases}$$

从上式可见，对所有的QP, MF_q只有6个不同的值，实际上MF_q是QP%6的函数。

将MF_q代入式 (6.38) 的量化式后得：

$$F_{u,v}^Q = \text{round}\left(W_{u,v} \frac{\text{MF}_q \cdot E}{2^{q\text{bit}}}\right) \tag{6.41}$$

伸缩因子E可以表示成2的整数幂， $E=2^{-T}$ ，代入式（6.41），考虑补偿舍入偏移后：

$$F_{u,v}^Q = \text{round}\left(W_{u,v} \frac{\text{MF}_q}{2^{q\text{bit}} + 2^T} + f\right) = \text{round}\left(W_{u,v} \text{MF}_q + f'\right) \gg (q\text{bit} + T) \quad (6.42)$$

上式中 $f=f\ll (q\text{bit}+T)$ 为等效偏移，“ \ll ”为二进制左移运算。考虑量化结果的符号后，完整的量化表达式为：

$$F_{u,v}^Q = \text{sign}(W_{u,v}) \cdot \text{round}\left(|W_{u,v}| \text{MF}_q + f'\right) \gg (q\text{bit} + T) \quad (6.43)$$

其中，“ \gg ”为右移运算，右移一次完成整数除以2, $\text{sign}(\)$ 为符号函数， f' 为偏移量，它的作用是改善恢复图像的视觉效果。例如，对帧内预测图像块， f' 取1/3，对帧间预测图像块， f' 取1/6。

2. 反量化计算

反量化的基本处理就是用量化步长 Q_{step} 乘以量化后的系数 $F_{u,v}^Q$ ，并同时考虑反变换的伸缩系数 $E_{u,v}$ ，得到反量化后的系数 $F_{u,v}^{\text{IQ}}$ ，可用如下公式表示：

$$W_{u,v}^{\text{IQ}} = \text{round}\left(F_{u,v}^Q \cdot Q_{\text{step}} \cdot E_{u,v}\right) \quad (6.44)$$

和量化处理类似，为了避免小数计算，设：

$$iqbit = 6 - \text{floor}(QP/6) \quad (6.45)$$

考虑到QP和QP_{step}的关系，引入变量MF_{iq}为：

$$\begin{aligned} MF_{iq} &= 2^{iqbit} \cdot Q_{step} = 2^{6-\text{floor}(QP/6)} \cdot 2^{(QP-4)/6} \\ &= 2^6 \cdot 2^{-4/6} \cdot 2^{-\text{floor}(QP/6)} \cdot 2^{QP/6} = 40 \cdot 2^{(QP \% 6)/6} = \begin{cases} 40 & QP \% 6 = 0 \\ 45 & QP \% 6 = 1 \\ 51 & QP \% 6 = 2 \\ 57 & QP \% 6 = 3 \\ 64 & QP \% 6 = 4 \\ 72 & QP \% 6 = 5 \end{cases} \quad (6.46) \end{aligned}$$

其中，MF_{iq}只有6个不同的值，40,...,72。将上式代入式(6.44)中

$$\begin{aligned}
W_{u,v}^{IQ} &= \text{round} \left(F_{u,v}^Q \cdot \frac{\text{MF}_{iq}}{2^{iqbit} \cdot 2^{-iT}} + f \right) \\
&- \text{round} \left(F_{u,v}^Q \cdot C(\text{QP}\%6) + f' \right) \gg (iqbit - iT)
\end{aligned} \tag{6.47}$$

其中， $E_{u,v}=2^{iT}$ 是反变换的伸缩倍数。这样，反量化过程也为整数运算， MF_{iq} 的值只有6种，并且可以用位移替代除法。由于量化处理是一种非线性处理，反量化后不能保证恢复原来的值，因此一般情况下 $W_{u,v}^{IQ}$ 的值并不等于 $W_{u,v}^Q$ 的值，两者之差就是通常所说的量化误差或量化失真。

6.4.3 加权量化矩阵

如前所述，在量化过程中HEVC还支持加权量化矩阵，现在对这个可选的矩阵给予简单说明。量化加权矩阵是一种和待量化的TB同样尺寸的矩阵，在量化时，矩阵的每个元素除以TB中每个对应频率系数，从而达到控制每个频率系数的量化粗细的目的。在HEVC中，编码器可以标注是否使用量化加权矩阵。通常，这种依赖于频率系数的伸缩对实现基于人眼视觉系统（Human Visual System,HVS）的量化是非常有用的。在这种情况下，加权矩阵对变换块中低频系数给予较细的量化，其量化步长的尺寸较小，保证重建图像的低频（平坦或缓变）区域失真较小，以适应人眼往往对图像的平坦部分灵敏度较高的特性。而对高频系数则给予较粗的量化，适应人眼对剧变的高频细节的觉察能力不强的特性。对于大多数视频序列，基于HVS的量化可以提供比独立于频率系数的量化较好的可视质量。量化加

权矩阵还可以用于对图像感兴趣区间（ROI）的编码，对重点关注的区域设置较细的量化，以保证重建图像中这一区域的图像质量。

HEVC根据变换块不同的尺寸和类型，设计了下述20个加权量化矩阵：

(1) 亮度矩阵，包括Intra 4×4 、 8×8 、 16×16 和 32×32 矩阵，Inter 4×4 、 8×8 、 16×16 和 32×32 矩阵；

(2) 色度Cb矩阵：Intra 4×4 、 8×8 和 16×16 矩阵，Inter 4×4 、 8×8 和 16×16 矩阵；

(3) 色度Cr矩阵：Intra 4×4 、 8×8 和 16×16 矩阵，Inter 4×4 、 8×8 和 16×16 矩阵。

是否采用加权量化矩阵可用语法元素scaling_list_enabled_flag来标志， 4×4 和 8×8 量化矩阵值是HEVC默认的，如图6.4所示。从量化矩阵的值可以看出：在左上角的低频处，元素值（相当于量化步长）比较小，对应于细量化；在右下角，元素值比较大，对应于粗量化。这样矩阵发挥的功能和静止图像标准JPEG中的符合视觉阈值特性的二维DCT量化矩阵类似。

| | 用于帧内亮度和色度 | 用于帧间亮度和色度 |
|------------------|---|--|
| 用于帧内和帧间 亮度和色度 | $\begin{bmatrix} 16 & 16 & 16 & 16 & 17 & 18 & 21 & 24 \\ 16 & 16 & 16 & 16 & 17 & 19 & 22 & 25 \\ 16 & 16 & 17 & 18 & 20 & 22 & 25 & 29 \\ 16 & 16 & 18 & 21 & 24 & 27 & 31 & 36 \\ 17 & 17 & 20 & 24 & 30 & 35 & 41 & 47 \\ 18 & 19 & 22 & 27 & 35 & 44 & 54 & 65 \\ 21 & 22 & 25 & 31 & 41 & 54 & 70 & 88 \\ 24 & 25 & 29 & 36 & 47 & 65 & 88 & 115 \end{bmatrix}$ | $\begin{bmatrix} 16 & 16 & 16 & 16 & 17 & 18 & 20 & 24 \\ 16 & 16 & 16 & 17 & 18 & 20 & 24 & 25 \\ 16 & 16 & 17 & 18 & 20 & 24 & 25 & 28 \\ 16 & 17 & 18 & 20 & 24 & 25 & 28 & 33 \\ 17 & 18 & 20 & 24 & 25 & 28 & 33 & 41 \\ 18 & 20 & 24 & 25 & 28 & 33 & 41 & 54 \\ 20 & 24 & 25 & 28 & 33 & 41 & 54 & 71 \\ 24 & 25 & 28 & 33 & 41 & 54 & 71 & 91 \end{bmatrix}$ |

图6.4 4×4和8×8 默认加权量化矩阵

用于16×16和32×32变换块的量化矩阵的值可由默认的8×8量化矩阵获得，其过程如图6.5 所示。图中右下角是一个HEVC默认的8×8加权量化矩阵，该矩阵的每个元素“投射”到左上角的16×16量化矩阵对应的2×2区域的4个值，深色块表示这一项的值，也就是说这4个值都等于8×8加权矩阵的那个元素的值。如果投射到32×32量化矩阵，则对应的4×4区域的16个值。这种简化的方式这样可以减少大的量化矩阵的存储量。

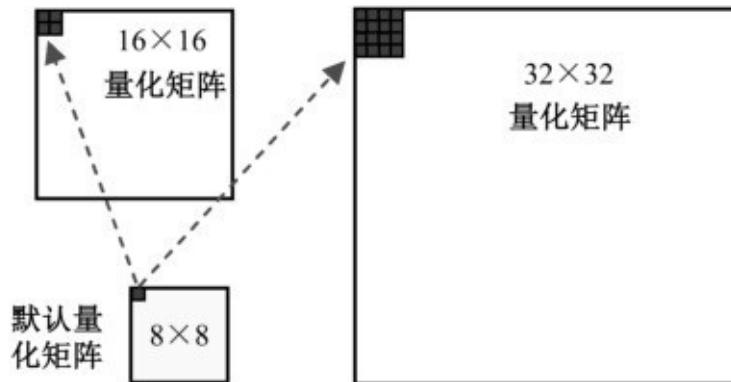


图6.5 8×8默认量化矩阵元素的投射

编码中也可选择非默认的量化矩阵，但要在比特流中将此信息传输到解码端，所需的矩阵数据可安置在序列参数集（SPS）或图像参数集（PPS）中。量化矩阵的元素采用右上对角扫描顺序、DPCM编码的方式传送。对 16×16 和 32×32 量化矩阵，实际上只需要传输 8×8 量化矩阵元素，在解码器中经上采样可获得所有的元素值。HEVC 还允许从另一个相同尺寸的量化矩阵来预测当前的量化矩阵，在 HEVC 中称为伸缩矩阵，这种情况由 SPS 中的 `scaling_list_enabled_flag` 来标识。当此标识有效时，在 SPS 和 PPS 中附加的标识来控制使用的是默认的量化矩阵或非默认的量化矩阵。

6.5 HEVC变换块的编码表示

从编码器的角度看，变换编码包括三个步骤：变换、量化以及量化后系数的熵编码。在HEVC视频编码过程中，TU中变换完的二维变换系数需要进行量化，量化后的系数再以特定的方式扫描成一维数据进行熵编码。

视频数据经变换和量化处理后，形成量化后的变换系数，又称为“水平”（level），以区别于未经量化的变换系数。由于“level”使用的地方较多，反而难以区别，本书仍然称量化后的系数为变换系数，由上下文可以界定是否是量化后的系数，特别容易混淆的地方则注明“量化后”。

编码块的编码表示在视频编码系统中处于量化模块之后、熵编码模块之前的位置，一般的编码框图上都没有特别的表示，可以理解为隐含在量化模块内了。量化后变换系数的编码表示包含两个部分：一是对量化后变换系数的扫描，将二维的变换系数按照一定规则转换为一维的数据排列方式；二是对非零的变换系数的位置和量值进行编码，形成便于熵编码的数据格式。

6.5.1 量化后系数的扫描

HEVC变换单元变换系数的扫描方法和H.264/AVC是不同的，HEVC的扫描都是在 4×4 子块（Sub-Block,SB）或 4×4 的TU基础上进行。一个大于 4×4 的TU可划分为若干 4×4 子块，每个 4×4 子块包含16个量化后的系数。HEVC对SB之间和每个SB内部使用选定的相同的扫描方式进行扫描，当前SB必须在前一个SB的所有系数被扫描后才被顺序扫描。

对一个TB内SB的处理顺序以及在每个SB内频率系数的处理顺序，HEVC都使用反向扫描方式。无论是TB内的SB，还是SB内变换系数，扫描

的顺序总是从右下角到左上角。这一顺序刚好和H.264/AVC的系数扫描顺序相反，因而谓之反向扫描顺序。反向扫描顺序又包括3类扫描方式，即水平扫描方式、垂直扫描方式和对角扫描方式。

图6.6所示的为对角扫描方式一例，从图6.6 (a) 中可看出，一个 16×16 的TB划分为16个SB,SB 的扫描顺序从右下到左上。在每个SB内，对频率系数的扫描仍然采用对角扫描方式，变换系数的扫描顺序也是从右下到左上，如图6.6 (b) 所示。

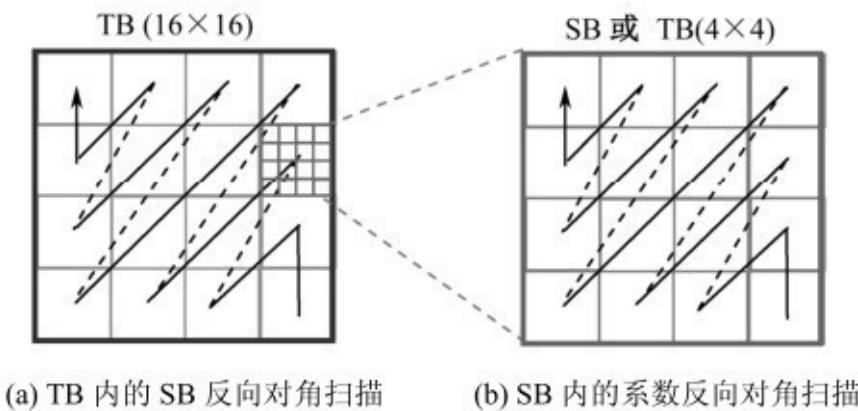


图6.6 TB (16×16) 及其内SB (4×4) 的反向对角扫描方式

扫描后的系数被编码和传输到解码器，使用一个标识符 significant_coeff_group_flag 来标记这个SB的系数是否有非0的量化后的系数。如果significant_coeff_group_flag为0，则SB中没有非0的量化系数，它的16(4×4)个量化后的0系数都不需要传送到解码器。

1. 帧内4×4 或8×8 TB的扫描顺序

如果TU是帧内模式(Intra Mode)预测的亮度信息，且为4×4 TB或8×8 TB，则可使用水平、垂直和对角三种方式扫描。扫描方式的选择和帧内预测模式的方向有关，即依赖模式的系数扫描(Mode-Dependent Coefficient Scan, MDSCS)。一旦帧内预测模式确定，则相应的扫描方式也就确定了，不需要再给解码器传送扫描模式的标识信号。如果帧内编码的亮度 TU 采用的是接近水平方向的预测，模式号为6~14，则采用垂直扫描方式；如果是接近垂直方向的预测，模式号为22~30，则采用水平扫描方式；其他方向都采用对角方式。如图6.7所示，图6.7(a)表示一个8×8 TB的4个4×4 SB及其频率系数的反向水平扫描方式，图6.7(b)则是这个TB的反向垂直扫描方式。如果是对角扫描方式，则可以参照图6.6。

帧内4×4 TB 的扫描方式相对比较简单，图6.6、图6.7中的4×4 SB分别表示对角、水平和垂直三种系数扫描方式。

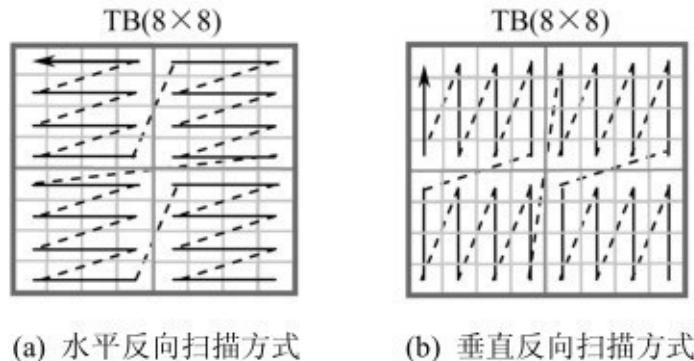


图6.7 帧内8×8 TB中的扫描方式

2. 其他TB的扫描顺序

如果TU是帧内亮度 16×16 TB或 32×32 TB，或是帧间模式（Inter Mode）预测的亮度TB，无论是什么尺寸，是否为对称划分，一律使用对角扫描方式。

对色度分量和相应的亮度分量划分一致，但是最小的色度TB不能小于 4×4 的尺寸。由第3章的编码结构可知，编码单元CB的变换块TB按照变换残差树（RQT）划分，当 8×8 的亮度分量块进一步划分为 4×4 的变换块时，对于4:2:0的彩色格式，对应的 4×4 的彩色分量不再进一步划分为更小的块，RQT保持色度块为最小的 4×4 尺寸。

3.QP的传输和更新

量化参数QP的定义范围是由PPS中的量化组（Quantization Group, QG）来控制的，QG的大小在一帧图像内是统一的，一个QG内的所有非零系数CU的量化参数QP都相同，不同的QG可以定义不同的QP。在一个CTU内可以包含一个或多个QG，其尺寸定义在亮度编码树块（CTB）尺

寸和最小亮度编码块（CB）尺寸之间，其值存放在SPS中。有可能出现一个CB包含多个QG或一个QG包含多个CB的情况。

QP值是以CU为单位进行传输和更新的。在slice或tile中，第一个CU的量化参数QP可以从PPS里的起始QP值导出。QP的大小可由slice分割（SS）的头信息中QP增量的值来更新，还可由CU层面的QP增量来进一步调整。

为了提高QP信息的传输效率，对当前QG的有效QP可从先前QG（解码顺序）预测得到，例如利用当前QG左边或上面QG进行预测，只需传输两者之间的预测误差。有效QP增量的值是放在相应CU的第一个TU中传输的。PPS和slice分割（SS）中的QP增量值用有符号的指数哥伦布（Exp-Golomb）码来进行熵编码。而在CU层面的QP增量值则用CABAC来进行熵编码。

色度分量的有效QP值可从亮度分量的QP值导出。由前可知，HEVC对QP大于30的色度信号的量化采取了较细的量化步长，主要是为了防止当大量化步长时色度变换系数的消失。

6.5.2 变换系数的表示

子块SB扫描后产生了一维的数据，HEVC并不是直接将这一串数据送给熵编码部分去进行基于上下文的自适应二进制算术编码（CABAC）去处理，因为这样的处理效率不高，而是要进行一番分析和处理，形成最合适的变换系数的表示，交由熵编码处理。

1.最后一个非零系数的位置表示

如前所述，TU中的变换系数被分为若干 4×4 大小的子块（SB），子块里的16个系数称为系数组（Coefficient Groups,CG）。对于每个CG内，按对角（或其他）方式扫描系数，同时在TU中，所有的CG也按对角（或其他）方式扫描。为了改进熵编码的效率，非0量化系数需要放置在扫描顺序中较早的位置。

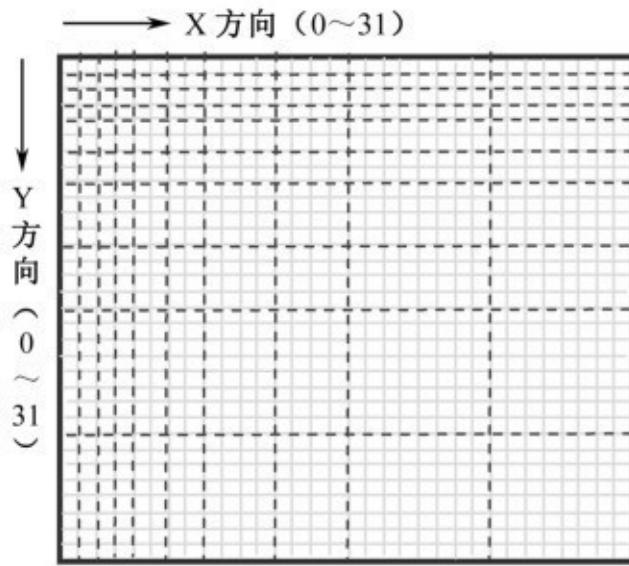


图6.8 32×32 TB中的系数位置分区

最后一个非零系数是指变换块（TB）按反向扫描方式首先碰到的那个非零（量化后）变换系数。编码器需要为这最后一个非零系数的位置信息，即它的（x,y）坐标进行编码传送。

将 TB 的水平和垂直方向分别划分为若干个区域，TB 中任意一点的位置都可以先对获得该位置所在区域的最小坐标x、y值，再加上x、y方向的偏移量就可确定该点的位置。图6.8是一个32×32 TB的分区示意图，x方向分为10个区，其坐标范围分别为0,1,2,3,4 ~ 5,6 ~ 7,8 ~ 11,12 ~ 15,16 ~ 23 , 24 ~ 31，对应的10个一元码为0,10,110,...,111111110,1111111110。y方向也是如此分区和编码。某一矩形区域位置用它中间坐标值最小的一组（x,y）表示。如果最后一个非零系数落在某一区域，但不在最小坐标位置上，还需要加上x、y方向的偏移。对0,1,2,3区，不需要加偏移，即偏移量=0；

对4~5和5~6区，偏移量为0或1，需用1比特来表示；对8~11和12~15区，偏移量为0~3，需用2比特来表示；对16~23和24~31区，偏移量为0~7，需用3比特来表示。偏移量都采用自然二进制码表示。这样，最后一个非零系数在TB中任意一个位置都可以用分区的一组(x,y)坐标和一组x、y方向的偏移量来准确表示。

2. 其他非零系数的位置表示

编码器在确定最后一个非零系数位置后，就需要确定其他所有的非零系数的位置。为此，编码器从最后一个非零系数位置开始，按照扫描规则反向扫描直到(0,0)处，即DC系数处。

首先对每个CG进行判断，检查该CG中是否含有非零系数，其结果用语法元素CSBF(coded_sub_block_flag)表示，若该CG至少包含一个非零系数，则CSBF=1，否则CSBF=0，表示该CG为全零系数。

对于CSBF=0的全零CG，编码就此结束，继续对下一个CG进行编码。

对于CSBF=1的CG，还要确定其中非零系数的位置。按照扫描顺序，检查每个系数位置，每个系数位置用1比特的sig_coeff_flag标志，如果该位置的系数为0，则sig_coeff_flag=0，如果该位置的系数不为0，则sig_coeff_flag=1。

表示其他非零系数位置的2个语法元素coded_sub_block_flag 和 sig_coeff_flag都采用常规二进制编码。

3. 非零系数的幅值的表示

在确定了非零系数位置信息后，只需要按顺序对各个非零系数的幅值进行编码。变换系数的值有正有负，因此需要有表示它符号的语法元素coeff_sign_flag。变换系数的值有大有小，因此需要有表示它大小的绝对值的语法元素。由于量化后的系数都是整数，而且其绝对值等于1或2的概率较大，因此专门设立了3个语法元素，第一个是表明系数是否大于1的coeff_abs_level_greater1_flag，第二个是表明系数是否大于2的coeff_abs_level_greater2_flag，第三个是表明系数剩余值的coeff_abs_level_remaining。

具体执行时，先检查非零系数是否等于1，如果等于1，则用coeff_abs_level_greater1_flag=0来表示这个系数，再继续检查下一个系数；如果该系数不等于1，则还需检查是否等于2，如果等于2，则用coeff_abs_level_greater2_flag=0来表示这个系数，再继续检查下一个系数；如果该系数不等于2，则其值为

```
coeff_abs_level_remaining = absCoeffLevel - baseLevel
```

其中，absCoeffLevel为该非零系数的绝对值，baseCoeffLevel为：

baseLevel=coeff_sig_flag+coeff_abs_level_greater1_flag+coeff_abs_level_g

要注意的是，由于熵编码的需要，在非零系数是否大于1的判断中，一个CG中只允许最多设置前8个系数，即coeff_abs_level_greater1_flag=1或0；在非零系数是否大于2的判断中，一个CG中只允许第一个“大于2”为真，即coeff_abs_level_greater2_flag=1。此后更多的未予设置的系数作为剩余值处理。

图6.9给出一个CG的系数表示的示例。图6.9（a）是一个CG值16个系数的位置图最后一个非零系数的位置为x=2,y=3。图6.9（b）是经过上述的

扫描和标识处理后的一维数据，这些数据即将送往熵编码单元进行编码，有的是常规模式的内容自适应二进制算术编码（CABAC），有的是旁路模式的算术编码。

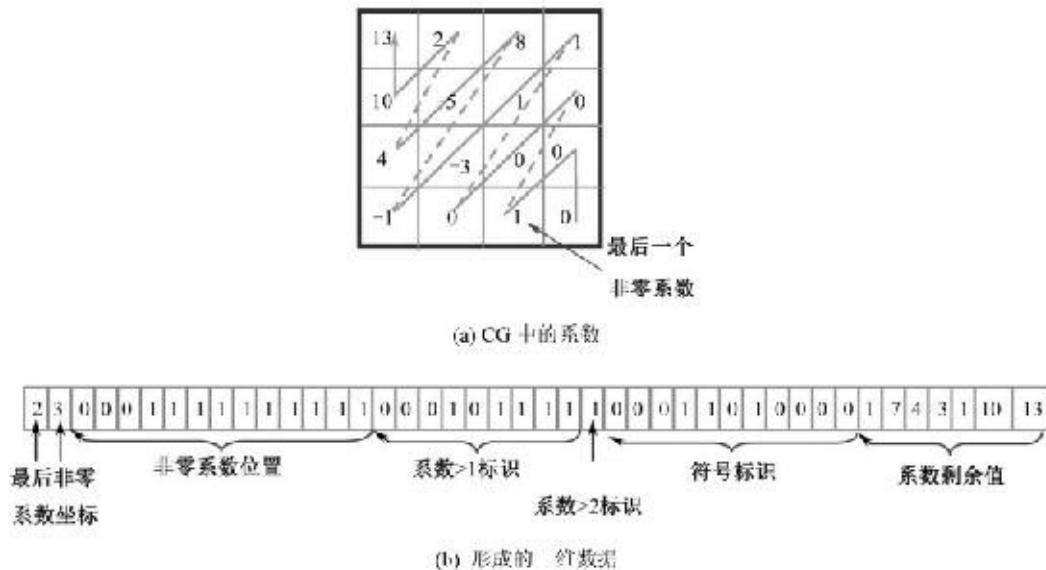


图6.9 CG的系数扫描和表示

表6.4是图6.9中CG系数的具体编码结果。表中第一行是从最后一个非零系数开始按扫描顺序排列的14个系数值。第二行是最后一个非零系数后

所有非零系数的位置表示，“1”表示该位置有非零系数，“0”则表示没有。第三行表示非零系数绝对值大于1的情况，系数值等于1则表示为0，系数值大于1则表示为“1”，按照规定，这里只表示了前8项，后面就不再处理了。第四行表示非零系数绝对值大于2的情况，系数值等于2则表示为“0”，值大于2则表示为“1”，这里只表示了第一项，后面也不再处理。最后一行表示系数大于1或大于2所剩余部分的值。

表6.4 图6.9中CG的系数表示

| 系数 | 1 | 0 | 0 | 0 | 1 | 1 | -3 | -1 | 8 | -5 | 4 | 2 | 10 | 13 |
|----------------|---|---|---|---|---|---|----|----|---|----|---|---|----|----|
| Sig coeff flag | — | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| Coeff>1 | 0 | — | — | — | 0 | 0 | 1 | 0 | 1 | 1 | 1 | 1 | — | — |
| Coeff>2 | — | — | — | — | — | — | 1 | — | — | — | — | — | — | — |
| Sign flag | 0 | | | | 0 | 0 | 1 | 1 | 0 | 1 | 0 | 0 | 0 | 0 |
| Remaining | — | — | — | — | — | — | 1 | — | 7 | 4 | 3 | 1 | 10 | 13 |

4. 符号比特隐藏

HEVC还有一个可选的符号比特隐藏工具，具体选用或是不选用由PPS中的标记来决定。如果选择，且CG中存在足够多的非零系数（3个以上），则第一个非零系数（即上述的最后一个非零系数）的符号比特可以不用编码，而是由解码器从该CG的其他符号推导得出。这样，每个CG可以节省1比特码字。

符号比特隐藏技术推断丢失的符号等于所有系数绝对值之和的最低有效位（LSB）。这意味着当编码器在编码时碰到绝对值之和的LSB和省略的符号不一致的情况时，它就需要调整非零系数中的某一个，使它的值增加1或减小1以满足上述的最低有效比特和符号比特相一致的要求。

当然，这一工具的使用有可能带来变换系数的失真，但是符号比特在熵编码中是以旁路模式编码的，没有压缩，编码花销是较大的。通过不编码一些符号比特，节约的意义要大于调整一个变换系数带来的失真。

6.5.3 变换跳过

在编码图像中，有时会遇到局部图像内容包含尖锐边缘或跳变，相关性极差。这种情况下，与其进行正常的变换操作，压缩的效果并不好，还不如跳过变换环节，直接对残差进行编码，即对残差信号不再进行变换操作，直接将它当作系数处理，这就是“变换跳过”。跳过变换的标记定义在PPS中，它是分别对每个 4×4 变换块的每个彩色分量进行标注。

量化器对变换系数的拉伸操作是独立于变换跳过的应用而完成的。如果变换跳过是针对TB的，则反变换操作就可省略，但是需要将此时的“变换系数”进一步拉伸到适应重建残差的动态范围。

由于采用变换跳过，就可消除在变换残差信号的重建过程中引入的振铃和模糊效应，只剩下量化误差对残差样点值的影响。

本章参考文献

[1]High Efficiency Video Coding[S],ITU-T Rec.H.265 and ISO/IEC

23008-2,ITU-T and ISO/IEC JCT-VC,Mach

2013 (v.1) ,Oct.2014 (v.2) ,Apr.2015 (v.3) .

[2]Advanced Video Coding for Generic Audiovisual Services[S].ITU-T Rec.H.264 and ISO/IEC 14496-10,ITU-T and ISO/IEC,Oct.2012.

[3]T.Wiegand,G.J.Sullivan,G.Bjøntegaard,et al.Overview of the H.264/AVC video coding standard[J].IEEE Trans.on Circuits and Systems for Video Technology,Jul.2003,13 (7) :560-576.

[4]J.R.Ohm,G.J.Sullivan,H.Schwarz,et al.Comparison of the coding efficiency of video coding standards including High Efficiency Video Coding (HEVC) [J].IEEE Trans.on Circuits and Systems for Video Technology,Dec.2012,22 (12) : 1669-1684.

[5]G.J.Sullivan,J.-R.Ohm,W.-J.Han,T.Wiegand.Overview of the High Efficiency Video Coding (HEVC) standard[J].IEEE Trans.on Circuits and Systems for Video Technology,Dec.2012,22 (12) : 1649-1668.

[6]T.Nguyen,H.Schwarz,H.Kirchhoffer,et al.Improved context modeling for coding quantized transform coefficients in video compression[C].IEEE Picture Coding Symposium,Nagoya,Japan,2010,378-381.

[7]J.Sole,R.Joshi,N.Nguyen,et al.Transform coefficient coding in HEVC[J].IEEE Trans.on Circuits and Systems for Video Technology,Dec.2012,22 (12) : 1765-1777.

[8]Tung Nguyen,Philipp Helle,Martin Winken,et al.Transform Coding Techniques in HEVC[J].IEEE Journal of Selected Topics in Signal

Processing,Dec.2013,7 (6) : 978-989.

[9]Yuan-Ho Chen,Chieh-Yang Liu.Area-efficient video transform for HEVC applications[J].Electronics Letter,9th July 2015,51 (14) : 1065-1067.

[10]Seishi Takamura,Atsushi Shimizu.Image coding using nonlinear evolutionary transforms[C].IEEE Data Compression Conference (DCC 2013),521-521.

[11]Mathias Wien.High Efficiency Video Coding: Coding Tools and Specification[M].Springer-Verlag Berlin Heidelberg,2015.

[12]Vivienne Sze,Madhukar Budagavi,Gary J.Sullivan.High Efficiency Video Coding (HEVC) :Algorithms and Architectures[M].Switzerland Springer International Publishing,2014.

[13]Lee Prangnell,Victor Sanchez,Rahul Vanam.Adaptive quantization by soft threshold in HEVC[C].IEEE Picture Coding Symposium (PCS 2015) ,35-39.

[14]Yih Han Tan,Chuohao Yeo,Hui Li Tan,et al.On residual quad-tree coding in HEVC[C].13th IEEE International Workshop on Multimedia Signal Processing (MMSP 2011) ,1-4.

[15]Chieh Yang Liu,Wen Quan He,Yung Ming Chang,et al.Low-cost video transform for HEVC[C].4th IEEE International Conference on Information Science and Technology (ICIST 2014) ,221-224.

[16]RyeongHee Gweon,Yung-Lyul Lee.N-level quantization in HEVC[C].IEEE International Symposium on Broadband Multimedia Systems and Broadcasting (BMSB 2012) ,1-5.

[17]Biatek,T.Raulet,M.Travers J.F.,et al.Efficient quantization parameter estimation in HEVC based on ρ -domain[C].Proceedings of the 22nd European Signal Processing Conference (EUSIPCO,2014) ,296-300.

[18]Vivienne Sze,Madhukar Budagavi.High throughput CABAC entropy coding in HEVC[J].IEEE Trans.on Circuits and Systems for Video

Technology, Dec., 2012, 22 (12) : 1778-1791.

第7章 HEVC的熵编码

在视频数据压缩中，按照压缩前后图像信息量是否有损失，可以把压缩方法分为两类：一类是信息保持型编码，常称为无失真编码或熵编码（Entropy Coding）；另一类则是非信息保持型编码，即允许一定量失真的编码，常称为有失真编码或有限失真编码。

历来的基于混合编码的视频信息压缩标准中（包括HEVC），都采用这两类压缩方法。变换、预测后的量化处理属于有限失真编码，消除的是信源的空间和时间冗余度，形成的是表示量化后的预测残差变换系数的语法元素。当然，还有其他控制信息、标识信息的语法元素。对这些语法元素，还可以用熵编码的方法进一步压缩，消除码字之间的冗余度。采用熵编码的方法，虽然用来表示信源符号的比特数比原来有所减少，但它们所代表的信息量却和未编码前相同，因而可以保证这一压缩步骤不会给解码重建图像增加新的失真。

本章在简要介绍熵编码、算术编码的基本概念基础上，说明上下文自适应二进制算术编码，着重分析HEVC中CABAC框架下的二进制化、基于上下文的建模和二进制算术编码三项关键技术。

7.1 熵编码

目前应用广泛的熵编码是一类建立在图像统计特性基础之上的信源压缩编码方法。根据香农（Shannon）信息论的观点，信源冗余度来自信源本身的相关性和信源内事件（符号）概率分布的不均匀性。只要找到去除相关性和改变概率分布不均匀的方法，就找到了信源熵编码的方法。

7.1.1 熵编码的要求

编码就是将信源集中的不同符号用不同的数码来表示，最常见的是二进制编码，用“0”和“1”的不同组合来表示不同的符号。

如果为信源集中的每个符号固定对应一个码字，称这种分配码字的方法为非奇异编码（Non-Singular Codes），否则为奇异编码（Singular Codes）。如果每个符号都是用同样长度的二进制数表示，则称为等长编码，否则为变长编码（VLC）。等长编码的编码和解码实现都比较简单，但编码效率不高。相反，变长编码可以为概率大的符号分配短码表示，概率小的符号分配长码表示，平均而言，总的编码效率要高于等长编码，但实现的复杂程度要高于等长编码。

1. 唯一可译编码

在不等长编码的情况下，为减少符号表示的平均码字长度，往往在相邻码字之间不加识别码，又要求所编码字序列能被唯一地译码出来。满足这个条件的编码称为唯一可译编码，也称单义可译码。换句话说，对于任意有限长度单义码的码字序列，只能唯一地被分割成一个个码字，而不发生歧义。单义码存在的充要条件是这个码字集合中的码字长度满足克劳夫特（Kraft）不等式，即

$$\sum_{i=1}^n D^{-t_i} \leq 1 \quad (7.1)$$

其中，D为代码中码元种类的进制数，对二进制情况，D=2。n为符号的个数或码字的种类数， t_i 为代码中第*i*个码字长度（即码元个数）。也就是说，如果代码中各个符号的码长结构符合克劳夫特不等式，一定能够找到（即“存在”）一组相同码长结构的单义代码。对具体的一组码字，目前尚没有一个准则判断它是不是单义可译码，即使它满足Kraft不等式，也不能保证它是单义可译码。如代码C={00,10,001,101}，因为是二进制码，则D=2，共有4个码字C₁=00、C₂=10、C₃=001和C₄=101,n=4。其相应的长度为t₁=2、t₂=2、t₃=3、t₄=3，代入式（7.1）可得

$$\sum_{i=1}^4 2^{-t_i} = \frac{1}{2^2} + \frac{1}{2^2} + \frac{1}{2^3} + \frac{1}{2^3} = \frac{6}{8} < 1$$

因此 $\{C_i\}$ 有可能是单义代码，事实上它确实是单义码。再看代码 $\{0,10,010,111\}$ 满足克劳夫特条件，但不是单义码，如收到“010”后有两种译码方法，一种是“0”和“10”两个码字，另一种是“010”一个码字。

2. 非续长代码

若代码中任何一个码字都不是另一个码字的续长，也就是不能在某一个码字后面添加一些码元而构成另一个码字，则称其为非续长代码。反之，称其为续长代码。如二进制代码 $\{0,10,11\}$ 即为非续长代码，而 $\{0,01,11\}$ 则为续长代码。因为后者码字集合中的“01”可由同一集合中的码字“0”后加上一个码元“1”构成。

在非续长码的码字集合中，任何一个码字均不是其他码字的字头（前缀），因此，只要传输没有错误，在接收过程中，就可以从接收到的第一个比特开始顺序考察，一旦发现一个符号序列符合某一码字，就立即作出译码，并从下一个比特开始继续考察，直至全部译码完成。因此，非续长码又称为即时码。显然，非续长码既保证了译出码的唯一性，又保证了译码的即时性。在实际应用中，往往尽可能采用单义可译的非续长码。

7.1.2 定长编码

定长编码也称等长编码，即为每个编码符号分配一个等长比特的码字。由于这种方法码字的长度是一定的，编码和解码都很简单，很容易处理和实现。例如，众所周知的原始数字视频的亮度、色度信号都是采取这种编码方法。

在 HEVC 中，描述为 $f(n)$ 的语法元素表示有一个固定 n 比特的预定值。例如，`forbidden_zero_bit` 语法元素，它是一个 $f(1)$ 码字，1 比特长，其值为 0。再如描述为 $u(n)$ 的语法元素，表示码流中无符号 n 比特整数值的语法元素。这种定长编码主要用于 NAL 单元头、slice 分割头以及多个参数集，如 PPS、SPS、VPS 等。

7.1.3 变长编码

不同于定长编码，变长编码为各个编码符号分配的比特数不一定相等，也就是说这种编码产生的码字长度是变化的。变长编码的优势是编码的平均码长不定长编码短，但编码和解码都比定长方法复杂，较难处理和实现。常见的变长编码有哈夫曼（Huffman）、香农（Shannon）编码、指数哥伦布（Golomb）编码等。例如，众所周知的原始数字视频的亮度、色度信号都是采取的这种编码方法，又称PCM编码。

1.Huffman编码

哈夫曼编码是目前最常用的一种变长编码方法，在视频编码标准中得到广泛应用。衡量编码效率的最重要的指标是它的平均码字长度，平均码长越短越好。设被编码的信源有m种符号，如m种灰度等级，即信源的符号集合为 $\{a_i | i=0,1,\dots,m-1\}$ ，且它们出现的概率对应为 $\{P(a_i) | i=0,1,\dots,m-1\}$ ，那么，不考虑信源符号间的相关性，对每个符号单独编码时，则 Huffman 编码的平均码长L为：

$$L = \sum_{i=0}^{m-1} P(a_i) \cdot l_i \quad (7.2)$$

式中， l_i 表示符号 a_i 的码字长度。可以证明，若编码时对概率大的符号用短码，对概率小的符号用长码，则 L 会比等长编码时所需的码字少。或者说在 Huffman 编码中，如果码字的长度严格按照所对应符号出现概率大小逆序排列，则平均码字长度一定小于其他任何顺序的排列方法。

例如，输入量化后的系数集为 { $y_0, y_1, y_2, y_3, y_4, y_5$ }，对应的概率集为 {0.32, 0.22, 0.18, 0.16, 0.08, 0.04}，进行 Huffman 编码（过程省略）的结果为： $y_0=00, y_1=10, y_2=11, y_3=010, y_4=0110, y_5=0111$ 。可以求出它的平均码字长度：

$$L = \sum_{i=0}^{m-1} P(a_i)l_i = 0.32 \times 2 + 0.22 \times 2 + 0.18 \times 2 + 0.16 \times 3 + 0.08 \times 4 + 0.04 \times 4 = 2.40 \text{ bit}$$

根据第1章的信源熵概念的式 (1.24) 可求得系数集的信源熵为

$$H = - \sum_{i=0}^{m-1} P(a_i) \log_2 P(a_i) = 2.352 \text{ bit}$$

6个符号集的信源熵为2.352比特，Huffman编码的平均码字长度为2.40比特，编码效率为 $\eta = \frac{H}{L} = \frac{2.352}{2.400} = 98\%$ ，可见，Huffman编码的结果已经很接近信源熵了。

2.0阶指数Golomb编码

针对Huffman编码的编译码比较复杂的缺点，可采用一种结构明确的不等长码，使编解码实现容易一些，只要将不同统计特性的信源符号根据出现的概率大小顺序映射到这种不等长码上。例如0阶指数Golomb编码就是这样一种结构明确的不等长码，如表7.1所示，将欲编码的一系列事件按概率从高到低排序，对应表中左边一列顺序编号 code_num,Colomb 编码后的码字如表7.1所示。

表7.1 0阶指数Colomb编码的码字

| code_num | 码字 | code_num | 码字 |
|----------|-------|----------|---------|
| 0 | 1 | 6 | 00111 |
| 1 | 010 | 7 | 0001000 |
| 2 | 011 | 8 | 0001001 |
| 3 | 00100 | 9 | 0001010 |
| 4 | 00101 | | |
| 5 | 00110 | | |

每个码字由3个部分组成，即Codeword=[M个0][1][INFO]。例如表7.1中code_num 等于“9”的码字为“0001010”，中间为“1”，左边是3个0，即

$M=3$ ，右边3位“010”就是INFO的内容。INFO是一个携带信息的M位数据，每个0阶Golomb码字的长度为 $(2M + 1)$ 位，每个码字都可由code_num产生，具体的编码和解码过程如下。

编码时，对每个待编的code_num，根据下面的公式计算INFO和M：

$M=\text{floor}(\log_2(\text{code_num}+1))$, floor()表示舍去小数的运算，

$$\begin{aligned}\text{INFO} &= \text{code_num} + 1 - 2^M, \\ \text{code_num} &= [\text{M个}0] + [1] + [\text{INFO}].\end{aligned}$$

解码时，读出以“1”结尾的前M个“0”，

根据得到的M，读出紧接着“1”后面的M比特的INFO数据，

根据 $\text{code_num}=2^M+\text{INFO}-1$ 可以还原出code_num。

注意，对于码字0,INFO和M都等于0。

例如，将刚才用Huffman编码的的符号集 $\{y_0, y_1, y_2, y_3, y_4, y_5\}$ 用0阶Golomb方法编码。把信源符号 y_i 映射到Golomb方法的“编号”code_num_i，按照上述规则编出的Golomb码字长度如表7.2所示。

表7.2 符号、code_num和对应Golomb码长

| 符号值 | y_0 | y_1 | y_2 | y_3 | y_4 | y_5 |
|--------------|-------|-------|-------|-------|-------|-------|
| 概率 P_i | 0.32 | 0.22 | 0.18 | 0.16 | 0.08 | 0.04 |
| code_num i | 0 | 1 | 2 | 3 | 4 | 5 |
| 码字长 l_i | 1 | 2 | 2 | 5 | 5 | 5 |

可以计算出这组码字的平均码长为 $L = \sum_{i=0}^5 P(y_i)l_i = 2.62 \text{ bit}$ ，如果按

Huffman编码方法，可得出的平均码长为2.4bit，信源熵为2.35bit。比较而言，Huffman编码的平均码长比Golomb编码的平均码长更接近于信源熵。但是，Golomb码字结构明确，能够用计算方式算出code_num，解码也比 Huffman码表简单。

3. 算术编码

算术编码(AC)是一种常用的变字长编码(VLC)方法，是一种信息保持型编码，与Huffman编码的理论基础一致，也是对出现概率大的符号序列赋予短码，对概率小的符号序列赋予长码。但算术编码不像Huffman编码，无须为一个符号设定一个码字。它将不同信源符号序列的表示为与其概率成正比的一维空间的长度，每个符号序列对应一个子空间(长度)，互不重叠，总和归一化等于1。随着编码序列的编码符号的到来，不断地找到对应概率空间的某一段，最终的那一段就唯一表示了这个序列的所有符号及排序。

算术编码有固定方式的编码，也有自适应方式的编码。采用自适应算术编码的方式，无需先定义概率模型，对无法进行概率统计的信源比较合适，在这点上优于Huffman编码。同时，在信源符号概率比较接近时，算术编码比Huffman编码效率要高，在图像压缩中常用它来取代Huffman编码。但算术编码的算法的实现要比Huffman编码复杂。

7.2 算术编码

从理论上讲，对信源数据采用Huffman熵编码方法可以获得最佳编码效果，但是实际中，由于在计算机中存储和处理的最小数据单位是1“bit”，无法表示小数比特。因此在很多情况下，实际的压缩编码效果往往达不到理论的压缩比。如以一个两符号信源{x,y}为例，其对应的概率为{2/3,1/3}，则根据理论计算，符号x、y的最佳码长及平均码长分别为：“x”的最佳码长=- $\log_2 (2/3)$ bit=0.588bit，“y”的最佳码长=- $\log_2 (1/3)$ bit=1.588bit，平均码长=0.588×(2/3)+1.588×(1/3)=0.916bit。

这表明，要获得最佳效果，符号{x,y}的码字长度应分别是0.588bit和1.588bit。而计算机不可能有非整数位出现，只能按整数位进行，即采用Huffman方法对{x,y}编码，得到{x,y}的码字分别为0和1，也就是两个符号信息的编码长度都为1bit，平均码长也为1bit。可见，对于出现概率大的符号x，并未能赋予较短的码字。这就是实际的编码效果往往不能达到理论编码要求的原因之一。为了解决计算机中必须以整数位进行编码的问题，人们提出了算术编码（Arithmetic Coding, AC）的方法。

7.2.1 一般算术编码

1. 编码过程

设图像信源编码产生4个符号a、b、c、d，它们出现的概率依次是1/2、1/4、1/8和1/8，则信源符号集所有符号的概率之和组成了一个完整的概率空间，可用单位长度（0~1）的矩形来表示它，如图7.1所示。在此长度为1的单位矩形中，各个符号依次排列，所占宽度和它的概率大小成正比。各个符号的左边（低端）的分界线称为该符号的“端点”，每个端点值

是它前面所有符号的概率累积之和。第一个端点的值为0，因为在它之前没有码字；由于d出现的概率是 $1/8$ ，故第二端点值为0.001（二进制小数，以下同）；由于b出现的概率为 $1/4$ ，再加上d出现的概率为 $1/8$ ，所以第三个端点值为两者之和 $3/8$ ，即0.011，依此类推。这样形成了最初的符号空间分割，a的端点为0.011,b的端点为0.001,c的端点为0.111,d的端点为0.0。

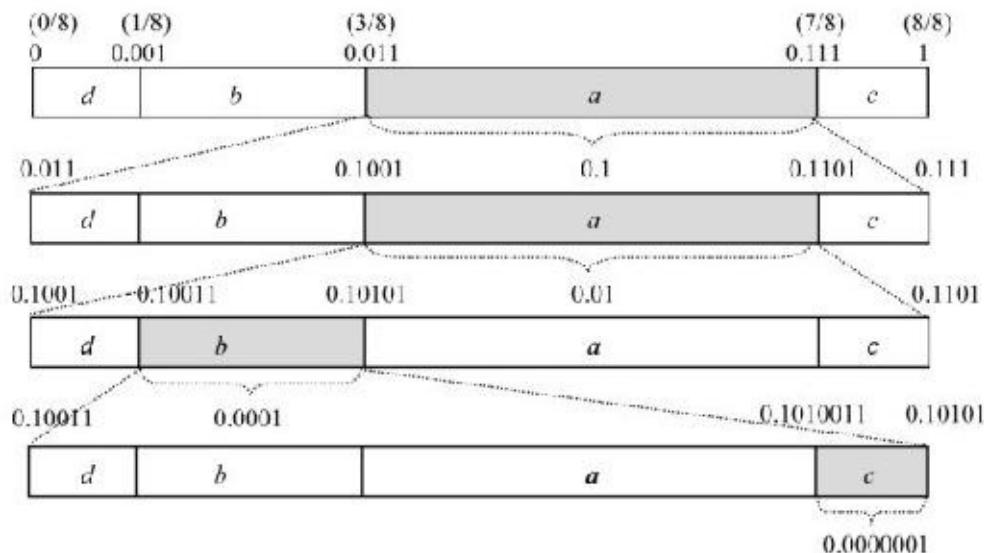


图7.1 符号序列“aab...”算术编码的子分过程

算术编码的过程实质上是对此单位区间不断“子分”(Subdivision)的过程。可以设想有一个编码“指针”，随着所编码字的进行，指针就不停地在单位区间内进行划分，指向新划分的端点。例如，我们对“a a b c ...”进行算术编码，其过程如下。

(1) 编码前，指针指向端点“0”，指针活动区间为“1”，即从0到1。

(2) 编码“a”，指针指向新端点： $0+1\times0.011=0.011$ (前端点+前活动区间 \times “a”的端点)；指针有效活动区间为： $1\times0.1=0.1$ (前活动区间 \times “a”的概率)。

(3) 编码“a”，指针指向新端点： $0.011+0.1\times0.011=0.1001$ (前端点+前活动区间 \times “a”的端点)；指针有效活动区间为： $0.1\times0.1=0.01$ (前活动区间 \times “a”的概率)。

(4) 编码“b”，指针指向新端点： $0.1001+0.01\times0.001=0.10011$ (前端点+前活动区间 \times “b”的端点)；指针有效活动区间为： $0.01\times0.01=0.0001$ (前活动区间 \times “b”的概率)。

(5) 编码“c”，指针指向新端点：

$0.10011+0.0001\times0.111=0.1010011$ (前端点+前活动区间 \times “c”的端点)；指针有效活动区间为： $0.0001\times0.001=0.0000001$ (前活动区间 \times “c”的概率)。

最后所得到“c”的编码区间：0.1010011 ~ 0.10101中任意值就是对“aabc”进行算术编码的结果，例如取1010011 (忽视小数点)。如果所给的码字数目更多，还可以以此类推。随着所编码字的增加，指针的活动范围越来越小，越来越精确，所编出的二进制码字位数越来越多。在上述运算中，尽管含有乘法运算，但它可以用右移来实现，因此在算法中只有加法和移位运算。这正是将这种算法叫做算术编码的原因。

2.解码过程

算术解码过程和编码过程相反，它是将算术编码的码字序列值通过逐次比较而逐步在单位概率空间逐渐“定位”的过程。下面以算术编码得到的“0.1010011”码序列的解码过程为例来说明。

(1) 在0 ~ 1空间里定位，即检查该码字值落在单位概率空间的哪一个

码字区间，则为得到多一个解码符号。由于 $0.011 < 0.1010011 < 0.111$ ，解得第一个码字为“a”。

(2) 由码字序列值(0.1010011)减去前端点值(0.011)得：

$0.1010011 - 0.011 = 0.0100011$ ，这是因为在编码过程中第二次子分区间的新端点的值是和0.011相加的，所以在解码时要减去它。再将得到的值0.0100011乘2: $0.0100011 \times 2 = 0.100011$ 。这是因为在编码过程中，曾将子分区间宽度乘以“a”的概率($0.1 = 1/2$)。而 $0.011 < 0.100011 < 0.111$ ，所以解得第二个码字为“a”。

(3) 由码字序列值(0.100011)减去前端点值(0.011)得：

$0.100011 - 0.011 = 0.001011$ 。这是因为在编码过程中第三次子分区间的新端点的值是和0.011相加的，所以在解码时要减去它。再将得到的0.001011乘2: $0.001011 \times 2 = 0.01011$ 。这是因为在编码过程中，曾将子分区间宽度乘以“a”的概率($0.1 = 1/2$)。而 $0.001 < 0.01011 < 0.011$ ，所以解得第三个码字为“b”。

(4) 由码字序列值(0.01011)减去前端点值(0.001)得： $0.01011 - 0.001 = 0.00111$ 。这是因为在编码过程中第四次子分区间的新端点得值是和0.001相加的，所以在解码时要减去它。再将得到的0.00111乘4: $0.00111 \times 4 = 0.111$ 。这是因为在编码过程中，曾将子分区间宽度乘以“b”的概率($0.01 = 1/4$)。而0.111恰好是“c”的端点，所以解得第四个码字为“c”。

从上述的实例中可以看到算术编码的大致过程。对“aabc”算术编码的结果为“1010011”，共7比特。如果采用Huffman编码，“a”为“0”，“b”为“10”，“c”为“110”，“d”为“111”，则“aabc”编码的结果为“0010110”，共7比特。这里两者编码长度相同，是因为此例算术编码的序列较短，如果序列较长，则可显示更高的效率，算术编码的效率一般要比Huffman编码高5%~10%。

7.2.2 自适应算术编码

在算术编码中，我们把信源的统计特性看作固定不变的，这与实际情况往往并不相符。为使编码技术适应信源统计特性的变化，提出了自适应算术编码（Adaptive AC,AAC）方法。自适应算术编码在一个符号编码中完成两个过程，即概率模型的建立和算术编码。

自适应算术编码在具体编码前并不知道各符号的统计概率，而是在编码开始时假定每个符号的概率相等，并平均分配概率区间[0,1]，然后在编码符号序列的过程中不断调整各个符号的概率。

例如：a、b、c 三种符号的初始概率暂定为 $P_a=1/3, P_b=1/3, P_c=1/3$ 。以此概率分配来划分[0,1]区间，a从0到1/3,b从1/3到2/3,c从2/3到1。设待编码符号序列为“b c c b”，下面具体说明在编码过程中符号概率自适应更新的过程。

(1) 初始假设 a、b、c 三种符号已经各出现了一次，各自的概率都为 1/3。由于第一个符号b的出现，导致a、b、c 三种符号的概率分布发生了变化，相当于总共出现的4个符号中，b占了2个，因此调整后的概率分配为 $P_a=1/4, P_b=2/4, P_c=1/4$ 。接着以调整后的概率根据上一节介绍的算术编码方法进行子分区间编码。

(2) 第二个符号c 出现后，调整后的概率分配为
 $P_a=1/5, P_b=2/5, P_c=2/5$ 。

(3) 第三个符号c 出现后，调整后的概率分配为
 $P_a=1/6, P_b=2/6, P_c=3/6$ 。

(4) 第四个符号b 出现后，调整后的概率分配为
 $P_a=1/7, P_b=3/7, P_c=3/7$ 。

解码时，每解出一个符号便进行一次概率分布的更新，以调整后的概率分布进行下一步区间划分，重复操作，完成解码。可见，自适应算术编码方式通常无需预先定义概率模型，假定所有符号的初始概率相等，均为 $1/N$ (N 为符号种类数)，然后根据符号出现的情况进行自适应概率更新。随着编、解码过程的进行，概率分布将逐渐趋于信源的实际概率分布。这

种方法对很多实际中无法进行概率统计的信源比较适合。

7.2.3 二进制算术编码

二进制算术编码的输入的信源符号只有“0”和“1”两种，实际上是算术编码的特殊情况，其编码原理和编码过程与一般算术编码基本原理相同，仍然是不断划分概率子区间的递归过程。如果信源符号集内包含有多个符号，则需先将这些符号经过“预编码”或“二进制化”，将符号流转换为二进制比特流，再对比特流进行二进制算术编码。

在两个输入符号中，出现概率较大的称为大概率符号（More Probable Symbol,MPS），MPS的概率为 P_e ；出现概率较小的称为小概率符号（Less Probable Symbol,LPS），LPS 的概率为 Q_e ，显然有 $P_e+Q_e=1$ 。编码初始化子区间为[0,1],MPS与LPS分配如图7.2所示。



图7.2 二进制算术编码的符号概率分配

1. 编码过程

设置两个专用寄存器：C和A，用[*]标识*的内容。寄存器C的值[C]=c为编端点（指针所指处），初时化为0；寄存器A的值[A]=a为指针活动子区间的宽度（该宽度恰好是已输入符号的概率），初始为整个概率空间，等于1。随着被编码比特流输入，C和A的内容按照算术编码的规则不断被修正：当小概率符号LPS到来时，[C]=c,[A]=aQ_e；当高概率符号MPS到来时，[C]=c+aQ_e,[A]=aP_e=a(1-Q_e)。下面通过一个实例来说明二进制算术编码的过程。

设二进制源符号序列为“11011111”，其中，“0”为LPS，小概率Q_e=(1/8)_{十进制}=(0.001)_{二进制}；“1”为MPS，大概率P_e=(7/8)_{十进制}=(0.111)_{二进制}。初始状态：子区间起始位置[C]=0，子区间宽度[A]=1。

(1) 编码第1个符号“1”(MPS)，

$$[C]=c+aQ_e=0+1\times 0.001=0.001, [A]=aP_e=1\times 0.111=0.111.$$

(2) 编码第2个符号“1”(MPS)，

$$[C]=c+aQ_e=0.001+0.111\times 0.001=0.001111,$$

$$[A]=aP_e=0.111\times 0.111=0.110001.$$

(3) 编码第3个符号“0”(LPS)，

$$[C]=c=0.001111, [A]=aQ_e=0.110001\times 0.001=0.000110001.$$

(4) 继续编码下去……最后得到[C]=0.01000111110111100000001,

[A]=0.000011 001001000010111111...此时区间的尾为
c+a=0.01010100011111111000000，编码区间为[c,c+a)，编码输出可以是最后一个编码区间中的任意一个小数值，但为了取得最好的编码效率，应在[c,c+a)区间中选择的最短的二进制数作为编码结果。上面区间我们可取0.0101，这个数大于c，小于c+a，即二进制算术编码的输出为0101，将原来8比特的比特流“11011111”压缩为4比特输出。

2. 解码过程

二进制算术编码的解码过程和编码过程相反，在 Q_e 、 P_e 形成的两个概率子区间，逐步判断被解码的码字落在哪个区间，并赋予对应符号。设接收到的二进制码字为 $v=0.0101$ ，初始值 $[A]=a=1, Q_e=0.001, P_e=0.111$ 。当 v 落在 $0 \sim a Q_e$ ，解码符号为 $d=0$ ，则 $v \Leftarrow v, [A]=a Q_e$ ；当 v 落在 $a Q_e \sim a$ ，解码符号为 $d=1$ ，则 $v \Leftarrow (v-a Q_e), a \Leftarrow a (1-Q_e)$ 。“ \Leftarrow ”为赋值符号。下面对 $v=0.0101$ 开始解码：

(1) $v=0.0101$ ，落在 $a Q_e \sim a$ ，解得第一个符号为 $d=1, v \Leftarrow (v-a Q_e) = 0.0101 - 0.001 = 0.0011, a \Leftarrow a (1-Q_e) = 0.111$ ；

(2) $v=0.0011$ ，落在 $a Q_e \sim a$ ，解得第二个符号为 $d=1, v \Leftarrow (v-a Q_e) = 0.0011 - 0.00011 = 0.000101, a \Leftarrow a (1-Q_e) = 0.111 \times 0.111 = 0.110001$ ；

(3) $v=0.000101$ ，落在 $0 \sim a Q_e$ ，解得第三个符号为 $d=0, v \Leftarrow v=0.000101, a \Leftarrow a Q_e = 0.110001 \times 0.001 = 0.000110001$ 。

此时已解得码字“0.110”，以此类推，就可以解得全部码字“0.11011111”。

7.2.4 自适应二进制算术编码

如果在二进制算术编码的基础上引入上下文模型（Context Model）的概念，通过预测信源符号分布来调整并适应信源统计特性的变化，这就形成了基于上下文的自适应二进制算术编码（CABAC）。

上下文模型可以根据和当前符号有关的已编码符号的情况来构造，每个上下文模型的概率分布随着符号编码的进行而自适应地更新。对于不同的上下文模型，其概率分布也是不同的，参考上下文模型的概率即为条件概率。在熵编码中，同一信源的条件熵总是低于其独立熵（将信源符号看成是相互独立的）的，因此，利用恰当的上下文模型所提供的信息，就可以向条件熵逼近，达到进一步去除符号间冗余，提高编码效率的目的。

将基于上下文模型、自适应概率更新、二进制算术编码等方法结合起来就形成了目前在视频编码领域广泛应用的CABAC熵编码方法。

7.3 HEVC的算术编码

由于视频内容和获取过程的变化，视频信号具有很明显的非平稳统计特性，而且这一特性一直会影响到混合压缩编码后产生的语法元素，这些语法元素正是熵编码的对象。以往的熵编码方法大多忽视（或难以实现）语法元素这种统计特性，将它们当作平稳的独立信源处置，使得熵编码的效率尚存在可提升余地。H.264/AVC为了改善以往标准中熵编码存在的编码效率低、重建效果差等不足，同时考虑到当时实现的复杂性，设计了两种可选用的熵编码的方法，即效率稍低、实现较易的基于上下文的自适应可变长编码（CAVLC）和编码效率较高、实现较难的基于上下文的自适应二进制算术编码（CABAC）。在编码性能方面，CABAC和CAVLC相比大致类似于算术编码和Huffman编码相比。据统计，CABAC能在各种不同码率情况下，较CAVLC节省5%~10%的码率。但在提高编码效率的同时，CABAC要比CAVLC多20%~30%的运算量，这也是CABAC付出的代价。

HEVC基本上是沿用了H.264/AVC的CABAC方法，改变的地方并不多。与H.264/AVC其他编码工具相比较，CABAC是HEVC中改动相对较少的一个部分。CABAC将二进制算术编码与自适应上下文模型结合起来，很好地利用了语法元素数值之间的条件信息，使得熵编码的效率得到了进一步提高，其主要特点为。

（1）采用二进制算术编码。将所有的语法元素转化为二进制符号串，消除了乘法运算操作，降低了计算复杂度，提高了编码效率。

（2）充分利用符号间相关性。根据已编码的语法元素为待编码的语法元素建立了概率模型（上下文建模），并根据当前的统计特性自适应地进行概率估计（模型更新），进一步提高了编码效率。

实际上，HEVC并不是对所有的语法元素都采用算术编码的方法，而

是对于不同的编码信息采用不同的熵编码方法。如对表示比特流高层特性的语法元素，本身的信息量比较小，则长采用定长编码，放置在比特流的显著位置，便于其他应用提取。因为这些信息只关乎高层特性，也仅仅是提供给解码器使用，还可为其他应用提供方便，不涉及一般的比特流细节。而对比特流中比例较大的残差系数等信息，HEVC采用的高效的CABAC编码方法。

7.3.1 CABAC框架

由视频编解码过程可知，如不考虑信道传输，CABAC 的编码是视频编码的最后一步，CABAC 的解码则对应的是视频解码的最先一步。原始视频经编码以后产生的是一系列语法元素，因此输入到 CABAC 的符号就是连续不断的语法元素，编码一个语法元素的流程如图7.3所示。大部分情况下，CABAC编码要经过如下的3个步骤：

- (1) 二进制化 (Binarization)；
- (2) 上下文建模 (Context modeling)；
- (3) 二进制算术编码 (Binary arithmetic coding)。

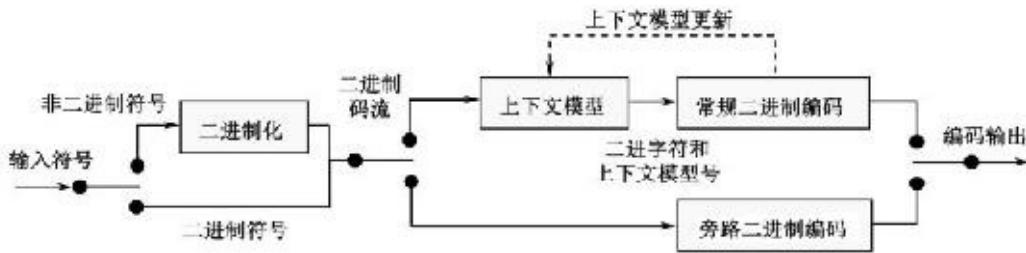


图7.3 上下文自适应二进制算术编码流程

如图7.3所示，CABAC首先对输入的非二进制值语法元素进行二进制化处理，唯一地转换为一个二进制序列，即二进制串。当然，如果输入的语法元素本身就是二进制值，那么这一步可以跳过。

经二进制化产生的二进制串随后进入算术编码阶段，这里有两种方法可以选择。

一种是常规编码模式（Regular Coding Mode），如图7.3中右上分支所示，包括上下文建模和算术编码两个部分。在算术编码之前，二进制值进入上下文建模步骤，在这里为它选择一个概率模型。模型的选择可能依赖于先前已编码的语法元素或二进制串。接着，在上下文模型确定以后，二进制值及其相应的模型一起送往常规算术编码模块进行编码，随时输出编码结果，并根据编码结果对相应的上下文模型进行更新。

另一种是旁路编码模式（Bypass Coding Mode），在该模式简化了算术编码过程，没有为每个二进制值分配一个特定的概率模型，如图7.3中右下分支所示。旁路编码模式可以看作常规编码模式编码的一种特殊情况，即二进制的0和1为等概分布。这种模式的好处是降低了实现的难度，加快了

编码（以及解码）的速度，当然其编码效率一般比常规模式略低。

7.3.2 二进制化

CABAC只对二进制符号（0或1）进行编码，因此非二进制符号（例如转换后的系数或者运动矢量等）在编码前先要进行二进制转换，实际上也是一种变长编码（VLC）。它是CABAC编码过程的前提，也可以说二进制化是对语法元素的预编码过程。如何在众多的二进制化方法中选择合适方法，应该综合考虑以下三点。

（1）形成的二进制码的信息冗余要求尽量小，或用尽量最短的二进数表示。但并非所有的VLC都适用，如Huffman编码虽然可以达到接近最小冗余编码，但它的编码、解码方法复杂，需要事先统计符号的概率，需要存储和发送码表，因而在CABAC中并不适用。

（2）形成的二进制码的统计特性要尽量有利于高效的CABAC需要。例如，两个符号的概率分布尽量相差较大，以期得到尽可能高的压缩效率。这一措施的依据是Shannon信息论，即信源符号概率分布相差越大则其变长编码的效率就越高，二进制化实质上是人为改变了信源符号的概率分布，应使其尽量地不均匀。

（3）二进制化的实现方法要尽量简单，特别是要方便译码。

根据上述要求，在实际的CABAC应用中一般并不采用Huffman之类的编码方法，而是采用实现简单、码型较规则的一类二进制化的编码方法，如一元码（Unary Code），截断一元码（Truncated Unary Code）， k 阶指数哥伦布码（ k th Order Exp-Golomb Code）和定长编码（Fixed-Length Code）等。

1.一元码（U,Unary）

对于无正负号整数值符号 $x \geq 0$ ，一元码由 x 个“1”（或“0”）加一个终结比特“0”（或“1”）组成，长度为 $x+1$ 。一元码码表见表7.3。例如，输入的语法元素值为 $x=4$ ，其一元码二值化结果为11110。解码器靠搜索一个“0”来

判断何时语法元素的结束。

2. 截断一元码 (TU, Truncated Unary)

给定一个最大可能值 $cMax$ ，对于编码一个非二进制的无正负号整数符号 x ，如果 $x > cMax$ ，则取 $x=cMax$ 进行编码；如果 $x < cMax$ ，二进制串由一元码方法给出；如果 $x=cMax$ ，一元码方法中的那个终结符号“0”被忽略，此时输出二进制串为 x 个“1”。例如，当截断值 $cMax=4$ 时，语法元素值 x 为3的二值化结果为1110，而句法元素值 x 为4的二值化结果为1111， x 值为5或更大的值，其二值化结果皆为1111，如表7.3所示。解码器靠搜索从0到 $cMax$ 比特来判断何时语法元素的结束。

表7.3 一元码和截断一元码 ($cMax=4$)

| x | $U(x)$ | $TU(x)$ |
|-----|--------|---------|
| 0 | 1 | 1 |
| 1 | 1 0 | 1 0 |
| 2 | 1 1 0 | 1 1 0 |

续表

| x | $U(x)$ | $TU(x)$ |
|-----|-------------|---------|
| 3 | 1 1 1 0 | 1 1 1 0 |
| 4 | 1 1 1 1 0 | 1 1 1 1 |
| 5 | 1 1 1 1 1 0 | 1 1 1 1 |
| ... | ... | ... |

3. 截断Rice码 (TR, Truncated-Rice)

k 等级的截断Rice码由一元码前缀和 k 比特后缀两部分构成， k 为参数。

对语法元素符号值 x ，前缀为一元码方式，其比特数为 $l_p(x) = 1 + x/2^k$ ；后缀为 k 位自然二进制数 $x_{k-1}, x_{k-2}, \dots, x_1, x_0$ 。符号“a”对 a 取其整数部分。

符号 x 的值如下：

$$x = 2^k(l_p - 1) + \sum_{i=0}^{k-1} x_i 2^i \quad (7.3)$$

当k=4时的一个截断Rice码如表7.4所示，后缀的长度为4比特，表示0 ~ 15共 $2^4=16$ 个值。

表7.4 截断Rice码一例 (k=4)

| x | TR(x) |
|-------------|---|
| 0, ..., 15 | 0 x ₃ x ₂ x ₁ x ₀ |
| 16, ..., 31 | 1 0 x ₃ x ₂ x ₁ x ₀ |
| 32, ..., 47 | 1 1 0 x ₃ x ₂ x ₁ x ₀ |
| 48, ..., 63 | 1 1 1 0 x ₃ x ₂ x ₁ x ₀ |
| | |

4.k阶指数哥伦布码 (EG_{k,kth order Exp-Golomb})

Truncated-Rice码的后缀的长度是固定的，如果后缀的长度不固定，由前缀长度来决定，这就是k阶指数Golomb码。对于给定的语法元素值x，它的取值范围为：

$$2^k(2^{l_p-1} - 1) \leq x \leq 2^k(2^{l_p} - 1) \quad (7.4)$$

EGk也是由一个前缀和一个后缀码字串接而成，EGk码字的前缀部分采用一元码方法，如 $l_p(x)$ 个“1”加1个“0”，其比特数也由式(7.4)来确定，结果为：

$$l_p(x) = \left\lfloor \log_2 \left(\frac{x}{2^k} + 1 \right) \right\rfloor \quad (7.5)$$

EGk的后缀部分的值为

$$x_s = x + 2^k (1 - 2^{l_p(x)}) \quad (7.6)$$

与这个十进制值对应的二进制串组成了EGk的后缀。当k=1时的EGk码表如表7.5所示。主要用于残差系数类语法元素的二进制化，如coeff_abs_level_minus1等。

表7.5 k阶指数Golomb码一例 (k=1)

| x | $EGk(x)$ |
|-------------|---------------------------------|
| 0, 1 | 0 x_0 |
| 2, ..., 5 | 1 0 $x_1 x_0$ |
| 6, ..., 13 | 1 1 0 $x_2 x_1 x_0$ |
| 14, ..., 29 | 1 1 1 0 $x_3 x_2 x_1 x_0$ |
| 30, ..., 61 | 1 1 1 1 0 $x_4 x_3 x_2 x_1 x_0$ |
| | |

5.定长码 (FL,Fixed-Length)

给定一个参数cMax，对于编码语法元值x，必须满足 $0 \leq x < cMax$ ，输出二进制串为十进制值 x 对应的二进制数，其最小长度固定为 $l_{FL} = \lfloor \log_2(cMax) \rfloor$ ，最高位比特 (MSB) 标注在最低位比特之前 (LSB)。一个3比特长的定长码的码表见表7.6。

表7.6 定长码一例 (cMax=8)

| x | $FL(x)$ | x | $FL(x)$ |
|-----|---------|-----|---------|
| 0 | 0 0 0 | 4 | 1 0 0 |
| 1 | 0 0 1 | 5 | 1 0 1 |
| 2 | 0 1 0 | 6 | 1 1 0 |
| 3 | 0 1 1 | 7 | 1 1 1 |

在HEVC中，不同的语法元素采用了不同的二进制化方法，具体的規定可在参考HEVC (第2版) 文档的表9.38。从该表中可以看出，主要使用了FL、TR和EGk三种二进制化方法。这里表7.7给出的是HEVC文档中表9.38的一部分。同时，对于某些语法元素，HEVC为它们制定了特别的二进制化方法，主要包括：

part_mode,intra_chroma_pred_mode,inter_pred_idc[x0]
[y0],cu_qp_delta_abs,coeff_abs_level_remaining[]。

表7.7 语法元素的二进制化方法 (部分)

| 语法结构 | 语法元素 | 进制化 | |
|----------------------|---------------------------|-------|---------------------------------------|
| | | 方法 | 输入参数 |
| slice_segment_data() | end_of_slice_segment_flag | FL | cMax=1 |
| | end_ofsubset one_bit | FL | cMax=1 |
| san() | sao_merge_left_flag | FT | cMax=1 |
| | sao_type_idx_luma | TR | cMax=2, cRiceParam=0 |
| | | | |
| prediction_unit() | merge_flag[] | FL | cMax=1 |
| | merge_idx[][] | TR | cMax=cMaxNumMergeCand-1, cRiceParam=0 |
| | | | |

续表

| 语法结构 | 语法元素 | 二进制化 | |
|--------------|-------------------------|-------|--------|
| | | 方法 | 输入参数 |
| mvd_coding() | abs_mvd_greater0_flag[] | FL | cMax=1 |
| | abs_mvd_greater1_flag[] | FL | cMax=1 |
| | abs_mvd_minus2[] | EGI | |
| | abs_sign_flag[] | FL | cMax=1 |
| | | | |

7.3.3 上下文模型

如前所述，视频编码的语法元素之间或多或少都存在一定的相关性，在熵编码时，充分利用已编码符号的概率分布信息来对当前符号进行编码，可降低编码码字的平均码长，提高编码效率，逼近符号集的条件熵。对于有记忆（相关）信源而言，条件熵总是小于将信源当作无记忆（独立）信源时得到的信源熵。在算术编码中，已编码语法元素的符号信息又称为上下文，充分准确地利用上下文信息显然有利于提高熵编码的效率。合适的上下文信息的统计特性（主要是概率分布）就是上下文模型。

通常情况下，上下文信息的统计特性是随视频内容变化的，因此一个有效的上下文模型一定要随内容而改变，也就是说上下文模型具有自适应改变的能力。不但要建立准确的上下文模型，而且需要即时地更新上下文

模型，才能够使 CABAC 获得较高的编码效率。可见，上下文建模和自适应更新是 CABAC 的关键所在。当然，上下文建模，包括自适应地估计条件概率，其计算代价是不小的，故而也是CABAC的难点所在。

为了适当地选择上下文模型，提高CABAC编码效率，在HEVC中，除使用H.264/AVC中空间邻域的语法元素外，编码树、变换树的划分深度也用于不同语法元素的上下文模型索引，例如，将语法元素skip_flag（定义CB是否要编码成帧间预测跳过模式）、split_coding_unit_flag（定义CB是否要进一步分裂）等用基于空间邻域信息的上下文模型编码。尽管用于HEVC中的上下文模型的数目要少于H.264/AVC，但熵编码设计实际上提供了比直接扩展H.264/AVC方法更好的压缩。

鉴于CABAC中的上下文建模的重要性，该部分内容在下一节介绍，主要包括上下文关系确定、初始模型的建立、编码中模型的即时更新等内容。

7.3.4 常规编码模式

CABAC中的二进制算术编码技术是基于区间的递归划分原理的，编码区间长度R的取值位于 $(2^8, 2^9)$ ，编码区间的下限用L表示，小概率符号（LPS）的概率记为 P_{LPS} 。

然后根据当前编码符号是小概率符号（LPS）或大概率符号（MPS）来确定新的编码区间。若当前编码符号是LPS，小概率符号区间 $R_{LPS}=R \cdot P_{LPS}$ 为新的编码区间；反之，当前编码符号是MPS，大概率符号区间 $R_{MPS}=R-R_{LPS}$ 为新的编码区间。

新的编码区间长度如果超出了 $(2^8, 2^9)$ 的范围，则要对R进行“重归一化”，使R的长度保持在 $(2^8, 2^9)$ 的范围内。“重归一化”的过程也包括对R的下限L值进行调整，使其与“重归一化”后的R相匹配。“重归一化”过程的同时可能会视情况输出一个或多个“0”、“1”比特，它们是已经确定的L的最

终值（二进制表示）的高位。经过反复的区间划分和“重归一化”，最终完成二进制算术编码。

参照图7.3的CABAC结构框图，二进制算术编码可采用两种模式来完成：一种是常规编码模式，它包括自适应概率模型的利用；另一种是用于快速符号编码的旁路模式，它假设数据为近似的均匀概率分布。

用常规编码模式比较复杂，图7.4出示了这种编码模式对一个二进制值binVal的算术编码过程，主要包括以下4个基本步骤：

- (1) 子分当前概率区间；
- (2) 确定新的概率区间和范围；
- (3) 上下文模型概率状态的更新；
- (4) 重归一化处理。

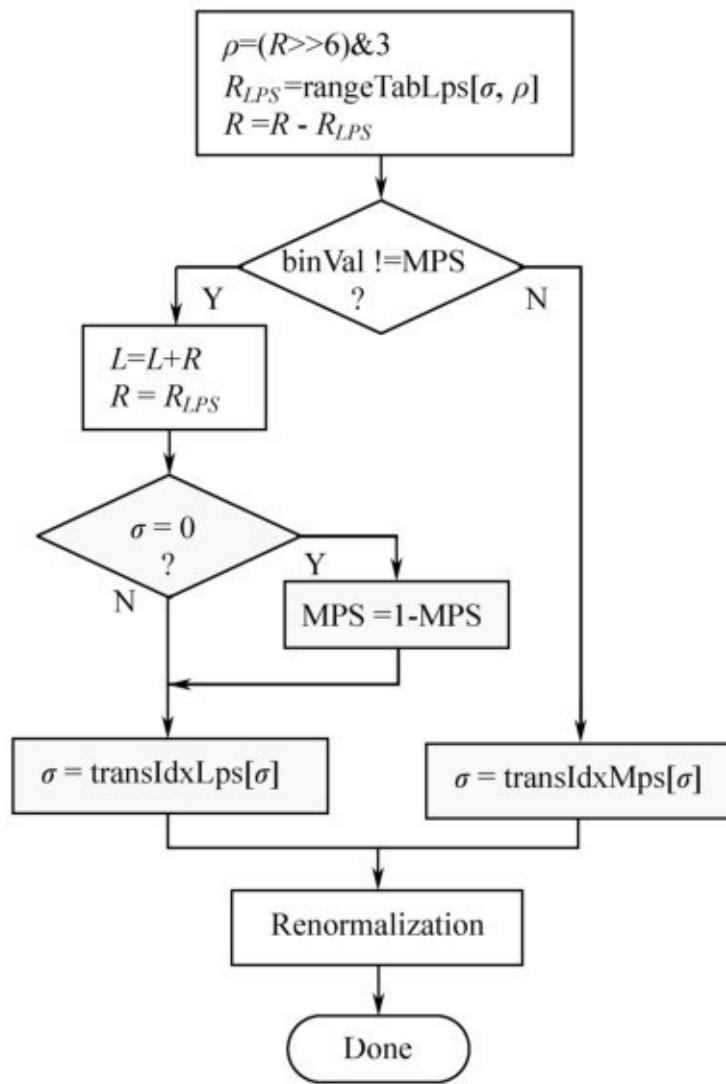


图7.4 常规编码模式流程

1.子分当前的概率区间

(1) 区间子分操作

这是第一步，也是最主要的第一步，按照给定的概率估计来子分当前的概率区间。这个区间子分的过程涉及3个基本的操作，如流程图7.4的顶端框所示。

首先，将当前的区间范围R由量化值Q(R)来近似，使用相等的分割把整个范围 $2^8 < R < 2^9$ 分为4个部分，由量化索引 ρ 表示，它可以通过联合移位和比特屏蔽来获得：

$$\rho = (R \gg 6) \& 3$$

上式表示由R得到量化索引 ρ 的过程：将R的值表示成9位二进制数，把R值的二进制形式右移6位（高位补“0”），然后和“3”的二进制表示形式“11”进行“位与”运算，所得的结果就是 ρ 的取值。

然后，这个范围索引 ρ 和概率状态索引 σ 作为2维表7.8的rangeTabLps的表项，用于决定近似子区间范围 R_{LPS} 和LPS。这里，表格rangeTabLps包括所有 64×4 个预先计算好的乘积值 $p_\sigma \cdot Q_\rho$ ，其中 $0 \leq \sigma \leq 63, 0 \leq \rho \leq 3$ ，精度为8比特。

在CABAC实现过程中，通常R、L分别用9和10比特表示，R的初始值定为Ox1FE,1 1111 1110（二进制）=510（十进制），L的初始值定为Ox00 0000 0000（二进制）=0（十进制）。

(2) 二维表格的形成

在二进制算术编码区间递归划分时，采用MPS的区间在前，LPS区间在后的顺序，如图7.5所示。设经过若干次划分以后，上一次的概率区间范围为R，区间低端为L。当前给定小概率符号（LPS）的概率估计是 p_{LPS} ，大概率符号（MPS）的概率估计为 $p_{MPS} = 1 - p_{LPS}$ 。

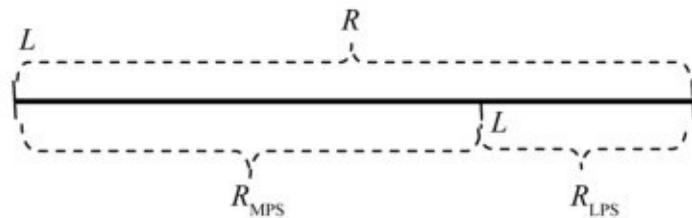


图7.5 概率区间 R_S 的子分示意

在此设置下，如果下一个编码的符号是LPS，则将给定的区间R分为两个子区间，其中表示LPS区间的宽度将作为编码下一符号的新概率区间，具体值为：

$$R_{\text{LPS}} = R \cdot p_{\text{LPS}} \quad (7.7)$$

如果下一个编码的是MPS，也是将给定的区间R分为两个子区间，表示MPS区间的宽度就成为下一个编码概率区间，具体值为：

$$R_{\text{MPS}} = R - R_{\text{LPS}} \quad (7.8)$$

在上述的二进制算术编码中，完成间隔子分所需要的乘法式（7.9）是主要的计算瓶颈所在，有必要对区间R或概率 p_{LPS} 进行某些近似来避免乘法，降低计算复杂度。H.264/AVC采取的方法就是将式（7.7）复杂的乘法运行简化为简便的查表操作。

将式（7.7）中区间R可能的值通过一组规定的量化操作，形成4个量化值 $Q_\rho, \rho=0,1,2,3$ ，作为间隔区间的近似。将式（7.7）中LPS可能的概率 p_{LPS} 离散化为64个值 $p_\sigma, \sigma=0,\dots,63$ 。式（7.7）的 $R \cdot p_{\text{LPS}}$ 就近似为 $p_\sigma \cdot Q_\rho$ 。预先将所有可能的乘法算好，其结果储存为一个二维表格rangeTabLps，表格包含所有 64×4 个预先计算好的值 $R_{\text{LPS}} = p_\sigma \cdot Q_\rho$ 。编码时只要以 σ 和 ρ 为地址，通过快速查表就可得到LPS新的子区间范围 R_{LPS} ，这样就可以免除算术编码中的乘法运算。 R_{LPS} 确定之后，根据出现的符号是MPS还是LPS就可确定新

的编码区间长度R和区间下限L。区间子分表格rangeTabLps参看表7.8，表中值表示乘积R_{LPS}的值，8位精度，σ 对应pStateIdx, ρ 对应qRangeIdx。

表7.8 区间子分表格rangeTabLps (HEVC v.2表9-46)

| $\frac{\rho}{\sigma}$ | 0 | 1 | 2 | 3 | $\frac{\rho}{\sigma}$ | 0 | 1 | 2 | 3 | $\frac{\rho}{\sigma}$ | 0 | 1 | 2 | 3 |
|-----------------------|-----|-----|-----|-----|-----------------------|----|----|----|----|-----------------------|----|----|----|----|
| 0 | 128 | 176 | 208 | 240 | 21 | 48 | 59 | 69 | 80 | 42 | 16 | 20 | 23 | 27 |
| 1 | 128 | 167 | 197 | 227 | 22 | 46 | 56 | 66 | 76 | 43 | 15 | 19 | 22 | 25 |
| 2 | 128 | 158 | 187 | 216 | 23 | 43 | 53 | 63 | 72 | 44 | 14 | 18 | 21 | 24 |
| 3 | 123 | 150 | 178 | 205 | 24 | 41 | 50 | 59 | 69 | 45 | 14 | 17 | 20 | 23 |
| 4 | 116 | 142 | 159 | 195 | 25 | 39 | 48 | 56 | 65 | 46 | 13 | 16 | 19 | 22 |
| 5 | 111 | 135 | 160 | 185 | 26 | 37 | 45 | 54 | 62 | 47 | 12 | 15 | 18 | 21 |
| 6 | 105 | 128 | 152 | 175 | 27 | 35 | 42 | 51 | 59 | 48 | 12 | 14 | 17 | 20 |
| 7 | 100 | 122 | 144 | 166 | 28 | 33 | 41 | 48 | 56 | 49 | 11 | 14 | 16 | 19 |
| 8 | 95 | 116 | 137 | 158 | 29 | 32 | 39 | 46 | 53 | 50 | 11 | 13 | 15 | 18 |
| 9 | 90 | 110 | 130 | 150 | 30 | 30 | 37 | 43 | 50 | 51 | 10 | 12 | 15 | 17 |
| 10 | 85 | 104 | 123 | 142 | 31 | 29 | 35 | 41 | 48 | 52 | 10 | 12 | 14 | 16 |
| 11 | 81 | 99 | 117 | 135 | 32 | 27 | 33 | 39 | 45 | 53 | 9 | 11 | 13 | 15 |
| 12 | 77 | 94 | 111 | 128 | 33 | 26 | 31 | 37 | 43 | 54 | 9 | 11 | 12 | 14 |
| 13 | 73 | 89 | 105 | 122 | 34 | 24 | 20 | 35 | 41 | 55 | 8 | 10 | 12 | 14 |
| 14 | 69 | 85 | 100 | 116 | 35 | 23 | 28 | 33 | 39 | 56 | 8 | 9 | 11 | 13 |
| 15 | 66 | 80 | 95 | 110 | 36 | 22 | 27 | 32 | 37 | 57 | 7 | 9 | 11 | 12 |
| 16 | 62 | 76 | 90 | 104 | 37 | 21 | 26 | 30 | 35 | 58 | 7 | 9 | 10 | 12 |
| 17 | 59 | 72 | 86 | 99 | 38 | 20 | 24 | 29 | 33 | 59 | 7 | 8 | 10 | 11 |
| 18 | 56 | 69 | 81 | 94 | 39 | 19 | 23 | 27 | 31 | 60 | 6 | 8 | 9 | 11 |
| 19 | 53 | 65 | 77 | 89 | 40 | 18 | 22 | 26 | 30 | 61 | 6 | 7 | 9 | 10 |
| 20 | 51 | 62 | 73 | 85 | 41 | 17 | 21 | 25 | 28 | 62 | 6 | 7 | 8 | 9 |
| | | | | | | | | | | 63 | 3 | 2 | 2 | 2 |

2.确定新的概率区间和范围

完成区间划分后，开始判断当前二进制符号 bibVal 是大概率符号 (MPS) 还是小概率符 (LPS)，见图7.4中上边那个菱形框，以确定哪一个子区间 (LPS的编码区间或MPS的编码区间) 作为下一个二进制符号的编码区间。

(1) 如果当前二进制符号是MPS，则MPS的编码区间作为下一个二进制符号的编码区间，区间下限L保持不变 (图7.4中右边分支)。

(2) 如果当前二进制符号是LPS，则LPS的编码区间作为下一个二进制符号的编码区间，区间下限L要增加MPS的编码区间的长度 (图7.4中左边分支)。

3.上下文模型概率状态的更新

常规算术编码过程的第三步，完成概率状态的更新（见图7.4中灰色框图部分），本部分内容在下一节描述。

4.重归一化处理

最后第四步是寄存器L和R的重归一化（图7.4中“Renormalization”框）。常规编码器在编码过程中对区间不断地细分，导致细分精度不断降低，新的编码区间长度R有可能小于 2^8 ，而“区间划分”过程要求R属于 $(2^8, 2^9)$ 的范围，因此需要经过“重归一化”过程，把R的取值恢复到 $(2^8, 2^9)$ 范围内。使得9位编码的区间值R和10位编码的下限值L保持一定精度。重归一化过程有可能输出一个或多个“0”、“1”比特作为算术编码的输出。CABAC具体的重归一化过程如图7.6所示。

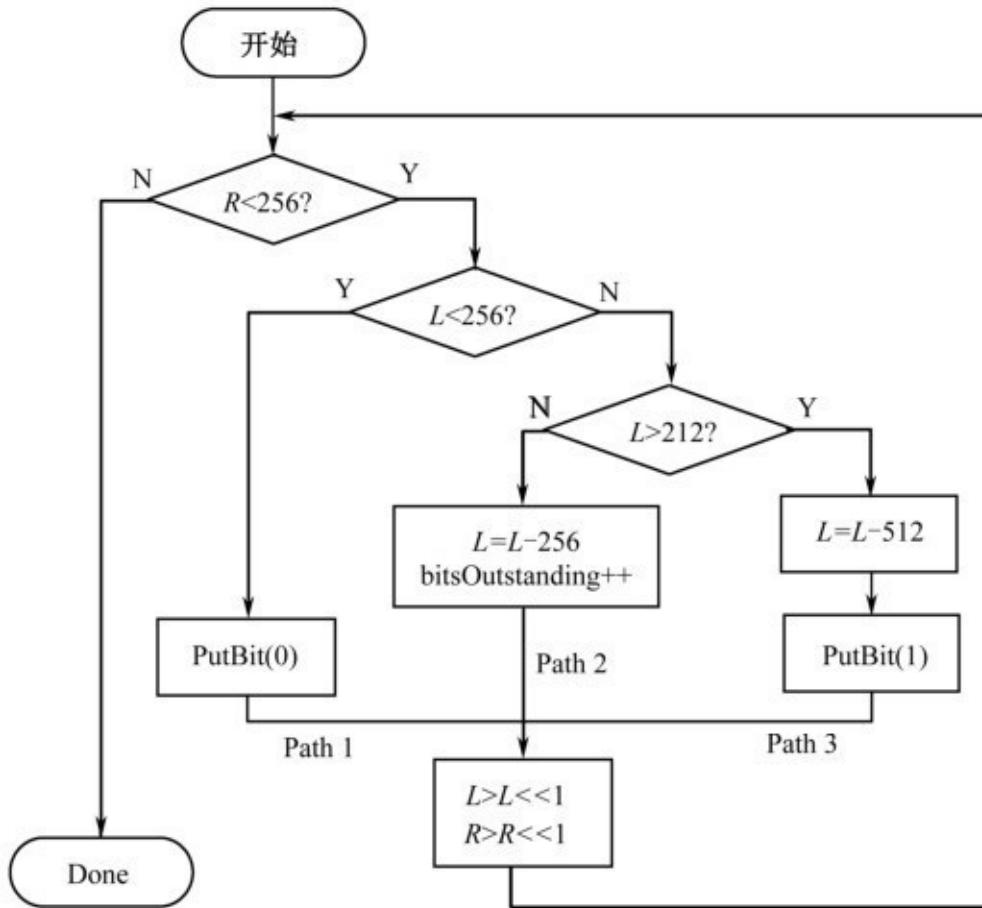


图7.6 重归一化流程

从图7.6可以看到，每次重归一化过程都要考虑R和L的大小。重归一化的过程基本思路是：通过重归一化操作将R和L限制在规定的范围内，同时通过编码区间的高位字节控制编码流的输出。重归一化算法的基本步骤如下。

Step1：根据R大小判断是否进行归一化处理，如果R大于 2^8 , ($R+L$)

的值可以落在 $(2^8, 2^9)$ 的任何位置，无法确定编码的高位值，因此无须进行重归一化操作，即重归一化过程完成；如果 R 小于 2^8 ，则执行 Step2。

Step2：根据 L 的值，执行 L 和 R 的进位输出的过程，同时要保证 L 进位后小于 512。为了提高编码器的利用率，在重归一化的过程中，如果确定了编码区间的高位值，就要输出高位，来保证递进计算的同时不断地输出码流，算术编码器将 2^8 作为重归一化的闭值，该过程根据 L 和 编码区间高位字节的值分成 3 种情况。

(1) R 小于 2^8 , L 小于 2^8 ，则 $(R+L)$ 必定小于 2^9 。可以确定编码区间的最高位为 0，可以输出这 0 比特。寄存器通过左移 1 位的操作来将 0 移出去，即图中的 Path 1 所示，执行完后将 L 和 R 倍乘来保持足够的编码精度，返回 Step1。

(2) R 小于 2^8 , L 大于 2^9 , $(R+L)$ 必定大于 2^9 。同样可以确定编码区间的最高位为 1，可以输出这个 1 比特，即图中的 Path 3 所示，执行完后将 L 和 R 倍乘，返回 Step 1。

(3) R 小于 2^8 , L 介于 2^8 与 2^9 之间。 $(R+L)$ 将有可能落在 $(2^8, 2^8+2^9)$ 和 $(2^8, 2^9)$ 两个区间内，即高两位可能是 01，也可能是 10，因此无法确定最高有效位为 1 还是 0。此时可以通过移出高位，缩小区间，判断次高位符号的方法来确定移出的高位符号值，即图中 Path 2 所示。若次高位值又落在 Path 2，将延迟移出次高位。图中的 bitOutstanding 代表非确定的延迟移出的高位值，通过继续判断当前位的值来输出编码区间的高位值。当落在 Path 1 或 Path 3，都将输出唯一确定的值而移出高位比特串，执行完后倍乘 L 和 R 返回 Step1。

7.3.5 旁路编码模式

为加快符号编码（解码）的速度，在 HEVC 中使用了 CABAC 操作的旁

路模式。这种模式假定二进制符号 0 和 1 发生的概率都固定是 0.5，使得图 7.3 中表示的常规算术编码过程得到了大大简化。首先，概率估计和更新过程被旁路；其次，区间子分执行简单，只要对区间等分就行。因此旁路编码模式相当于加速了编码过程。CABAC 编码过程中，旁路编码模式适合对大致等概率输入符号进行编码。

区间划分约定 0 区间在前面，1 区间在后面。为了使区间划分更简单，不采用直接对区间长度 R 进行二等分的方法，而是采用保持编码区间长度 R 不变，使区间下限 L 的值加倍，可通过对 L 的值左移一位实现。这种方法既可以达到同样的效果，又省去了归一化过程中要同时对 R 和 L 进行加倍的操作。图 7.7 所示为旁路模式下的编码流程，由图可知，归一化操作仅对 L 进行一次左移，而 R 保持不变。

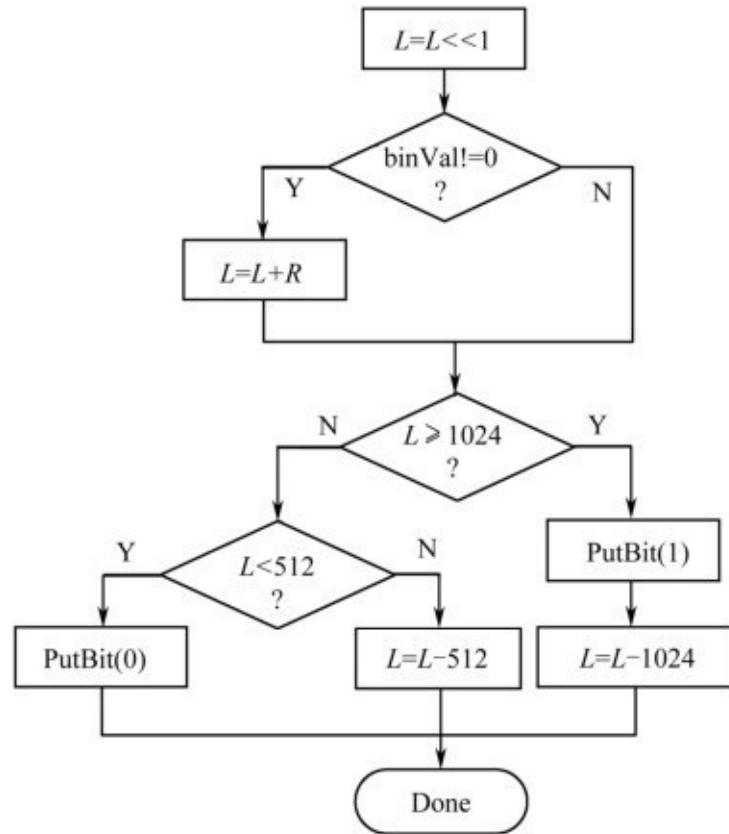


图7.7 旁路编码模式流程

7.4 上下文建模和更新

7.4.1 上下文关系

CABAC中上下文模型中的上下文关系大概可以分成4种情况。

第一种上下文模型根据它相邻的已编码相邻块的语法元素构成，一般是用其左边和上边块的对应语法元素来建立相应的概率模型，对当前语法元素进行模型预测，如图7.8所示。其中C为当前编码块的语法元素，A 和 B 为其左边和上面已编码块的语法元素。

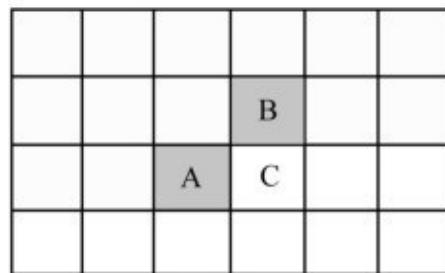


图7.8 依据相邻块的建模方法

第二种上下文模型仅局限于对块类型和子块类型的的应用，其中第 n bit 的概率模型的选定要参考前面已编码的 $n-1$ (b_1, \dots, b_{n-1}) bit 所采用的模型。

第三种和第四种上下文模型仅用于残差数据的编码，都依赖编码块的类型。其中第三种模型依赖的不是已编码的系数，而是待编码系数在扫描路径中的位置。第四种模型则是根据前面已编码系数幅度中某个特定幅度值出现的总次数，来为当前变换系数中的二进值确定上下文模型。

除这些基于条件概率的上下文模型之外，还有固定概率模型，它们对待编码的比特提供固定概率预测，已编码比特的模型不加以利用。

7.4.2 上下文模型的初始化

自适应算术编码的一个优点就是可以事先不需要知道编码符号统计特性，由后来的编码过程自适应环节逐步获得。那么在编码一开始如何处理呢？一种简单的方法就是把各类符号的概率初始值定为 $1/N$ (N 为符号类数)。对应自适应二进制算术编码的初始化问题，则是把LPS与MPS 的概率初始值均定为 $1/2$ ，这种概率的初始化方法虽然算法简便，但没有利用先验知识，会影响数据压缩效率。

CABAC是以条 (Slice) 或块 (Tile) 为独立单位进行视频编码的，所谓独立就是不同的条之间相互不借用编码数据。但具体语法元素的编码却是发生在编码块级，在同一个块中，不同的语法元素是独立编码。但在不同的编码块中，相邻块的语法元素的上下文信息（概率状态索引 σ 与最大概率符号MPS的值 ω ）可用于当前编码块的同一语法元素。在开始编码一个新的条时，需要对上下文模型进行初始化工作，一是将概率区间复原到单位区间 $(0,1)$ ，二是为每个上下文模型确定一对初始值 (ω, σ) 。

初始化的重要依据之一是量化参数 (QP)，各slice的QP不同，则在初始化时各概率状态值有可能不同。这样把每个上下文模型的“0”“1”概率分

布的经验值作为其初始概率，以此推算出该模型下的概率状态的初值，同时也确定了MPS和LPS的初值。

1. 确立上下文索引值ctxIdx和初始值initValue

在CABAC中，上下文模型是为每个二进制符号所建立的，二进制符号所属的语法元素和编码类型不同，其对应的上下文模型也不同。而且在同一种语法元素的二进制序列中，不同位置的二进制符号对应的上下文模型也不尽相同。在 CABAC 中，对不同的上下文模型进行顺序编号，称之为“上下文索引”语法元素ctxIdx，使得每一个模型都能够由唯一的索引号所标注。

1) H.264/AVC的ctxIdx范围

H.264/AVC（第1版）为各类语法元素定义了399种上下文模型索引值ctxIdx的范围，如表7.9所示。2012年版的H.264/AVC扩充了不同类型语法元素的上下文模型的索引值范围，总数达到1024种。表7.9可以帮助我们由编码类型和语法元素来获得它的模型索引值的范围。如B类编码的宏块类型（mb_type）语法元素的上下文索引值 ctxIdx 的范围为27 ~ 35。具体到编码某个语法元素时，还需要确定该语法元素的每个二进制位确切的概率模型的索引值ctxIdx底到底是多少。

表7.9 H.264/AVC语法元素的上下文索引值ctxIdx的范围

| syntax element | slice type | | |
|------------------------------|--------------------|--------------------|--------------------|
| | S/I | P/S/P | B |
| mb_type | 0/3-16 | 14-20 | 27-35 |
| mb_skip_type | | 11-13 | 21-26 |
| sub_mb_type | | 21-23 | 36-39 |
| mvd(horizontal) | | 40-46 | 40-46 |
| mvt(vertical) | | 47-53 | 47-53 |
| ref_idx | | 54-59 | 54-59 |
| mb_qp_delta | 60-63 | 60-63 | 60-63 |
| intra_chroma_pred_mode | 64-67 | 64-67 | 64-67 |
| prev_intra4x4_pred_mode_flag | 68 | 68 | 68 |
| rem_intra4x4_pred_mode | 69 | 69 | 69 |
| mb field decoding flag | 70-72 | 70-72 | 70-72 |
| coded_block_pattern | 73-84 | 73-84 | 73-84 |
| coded_block_flag | 85-104 | 85-104 | 85-104 |
| significant_coeff_flag | 105-165 277-337 | 105-165 277-337 | 105-165 277-337 |
| last_significant_coeff_flag | 166-226 338-398 | 166-226 338-398 | 166-226 338-398 |
| coeff_abs_level_minus1 | 227-275 | 227-275 | 227-275 |
| end of slice flag | 276 | 276 | 276 |

2) HEVC的ctxIdx范围

在HEVC中，上下文索引ctxIdx的确定方法和H.264/AVC不同。

(1) HEVC分别为每个语法元素给出一个上下文索引值ctxIdx的表格，由索引值决定该语法元素的初始值initValue。为此，HEVC列了一个表格(HEVC第2版文档中的表9-4)，注明了每个语法元素表格编号和不同起始类型(initType)所对应的上下文索引值ctxIdx，表7.10是其中的部分内容。如sao()的4项语法元素所对应的initValue如表7.11、表7.12所示；coding_unit()的语法元素part_mode对应的ctxIdx如表7.13所示；mvd_coding()的2个语法元素对应的ctxIdx如表7.14所示。这样的表格在HEVC中共有33个，即从HEVC第2版文档中的表9-5到表9-37。

(2) 为每个语法元素根据不同类别给出具体的是上下文索引值ctxIdx，不再是索引值范围，不同的语法元素有可能有相同的ctxIdx值。由上下文索引值ctxIdx也就唯一确定了上下文模型的起始值initValue。

表7.10 语法元素的初始值表格号及其不同初始化类型的ctxIdx值

| 语义结构 | 语义元素 | HEVC 的 表编号 | ctxIdx | | |
|---------------------|------------------------------|---------------|------------|------------|------------|
| | | | initType=0 | initType=1 | initType=2 |
| sao() | sao_merge_left_flag | Table 9-5 | 0 | 1 | 2 |
| | sao_merge_up_flag | | | | |
| coding_quadtrees() | sao_type_idx_luma | Table 9-6 | 0 | 1 | 2 |
| | sao_type_idx_chroma | | | | |
| coding_unit() | split_cu_flag[1] | Table 9-7 | 0…2 | 3…5 | 6…8 |
| coding_unit() | cu_transient_bypass_flag | Table 9-8 | 0 | 1 | 2 |
| | cu_skip_flag | Table 9-9 | | 0…2 | 3…5 |
| | pred_mode_flag | Table 9-10 | | 0 | 1 |
| | part_mode | Table 9-11 | 0 | 1…4 | 5…8 |
| | prev_intra_luma_pred_flag[1] | Table 9-12 | 0 | 1 | 2 |
| | intra_chroma_pred_mode[1] | Table 9-13 | 0 | 1 | 2 |
| | rqt_root_cbf | Table 9-14 | | 0 | 1 |
| | | | | | |
| mvd_coding() | abs_mvd_greater0_flag[] | Table 9-23 | | 0 | 2 |
| | abs_mvd_greater1_flag[] | Table 9-23 | | 1 | 3 |
| | | | | | |
| cross_comp_pred() | log2_res_scale_abs_flag[] | Table 9-36 | 0…7 | 8…15 | 16…23 |
| | res_scale_sign_flag[] | Table 9-37 | 0…1 | 2…3 | 4…5 |
| | | | | | |

表7.11 sao_merge_left_flag和sao_merge_up_flag的初始值initValue (HEVC表9-5)

| initialValue | ctxIdx | | |
|--------------|--------|-----|-----|
| | 0 | 1 | 2 |
| initialValue | 153 | 153 | 153 |

表7.12 sao_type_idx_luma和sao_type_idx_chroma的初始值initValue (HEVC表9-6)

| initialValue | ctxIdx | | |
|--------------|--------|-----|-----|
| | 0 | 1 | 2 |
| initialValue | 200 | 185 | 160 |

表7.13 part_mode的初始值initValue (HEVC表9-11)

| initialValue | ctxIdx | | | | | | | | |
|--------------|--------|-----|-----|-----|-----|-----|-----|-----|-----|
| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
| initialValue | 184 | 154 | 139 | 184 | 154 | 154 | 139 | 154 | 154 |

表7.14 abs_mvd_greater0_flag和abs_mvd_greater1_flag的初始值initValue (HEVC表9-23)

| 初值 | ctxIdx | | | |
|-----------|--------|-----|-----|-----|
| | 0 | 1 | 2 | 3 |
| initValue | 140 | 198 | 169 | 198 |

2. 确立概率状态索引pStateIdx和大概率符号值valMps

从上面表格中的8bit initValue的值导出下面两个4比特的变量slopeIdx和offsetIdx：

slopeIdx=initValue $\gg 4$ (右移4位，缩小16倍，取高4位)

offsetIdx=initValue & 15 (取低4位)

由slopeIdx和offsetIdx得到2个中间变量m和n：

m=slopeIdx $\times 5 - 45$

n= (offsetIdx $\ll 3$) - 16

再考虑编码slice的量化参数 (QP) 值：

SliceQp_Y=26 init_qp_minus26+slice_qp_delta

由上述的m、n和SliceQp_Y可得到上下文模型的2个变量，概率状态索引pStateIdx (即 σ) 和大概率符号值valMps (即 ω)：

preCtxState=Clip3 (1,126, ((m \times Clip3 (0,51,SliceQp_Y)) $\gg 4$) +n)

valMps= (preCtxState ≤ 63) ? 0 : 1

pStateIdx=valMps ? (preCtxState - 64) : (63 - preCtxState)

其中，Clip3 (a,b,c) 是一种限幅函数，表示将c的值限制在a和b之间。表达式“a ? b : c”的含义为：如果a为真（或不等于0），则a=b，否则，a=c。

在上述的推导中，还需要确定表格7.10中列出的3种初始化类型变量initType的值，导出方法如下：

```

if ( slice_type==I )
    initType=0
else if ( slice_type==P )
    initType=cabac_init_flag ? 2 : 1 slse
    initType=cabac_init_flag ? 1 : 2

```

可以看出，如果是I slice，则initType的值为0，但对于P slice和B slice,initType的值还取决于语法元素cabac_init_flag。

至此，得到了上下文模型的概率状态索引和大概率符号值，初始化处理就结束了。

7.4.3 上下文模型的更新

上下文模型主要包含两个参数，一个是概率状态索引 pStateIdx，即 LPS 的概率，简记为 $p_{\text{StateIdx}} = \sigma$ ；另一个是大概率符号值 valMps，即 MPS 的值，简记为 $\text{valMps} = \omega$ 。很显然，这两个值都会随着编码的进行而发生改变，即上下文模型发生改变，及时反映这种改变就是上下文模型更新。

1.LSP的概率

为了简化上下文模型的概率估计，用有限的 64个索引值 σ 来表征 LPS 的概率值 $p_\sigma \in [0.01875, 0.5]$ ，可由下式导出。

$$p_\sigma = \alpha \cdot p_{\sigma-1}, \quad \alpha = \left(\frac{0.01875}{0.5} \right)^{\frac{1}{63}} \approx 0.95 \quad (7.9)$$

其中， $\sigma=1,\dots,63$, $p_0=0.5$ ，概率集合的伸缩因子 $\alpha \approx 0.95$ 和索引数 $N=64$ 是一种折中的选择。一方面，如果想要获得快速的自适应，希望 $\alpha \rightarrow 0, N$ 要小；另一方面，如果想获得更加稳定、更加精确的估计，则希望 $\alpha \rightarrow 1, N$ 要大。在算术编码中，将这种概率及其索引之间的关系固化为一个便于处理的表格，每一个概率 p_σ 用其相关的索引 σ 作为其地址。

这样，CABAC中的每一个上下文模型可以由两个参数完全决定：当前LPS概率估计值的索引 σ （0,1,...,63）和MPS的值 ω （0或1）。 σ 可用6比特数表示， ω 可用1比特数表示，因此上下文模型的每一个状态仅用一个7比特数就可表示。

2. 概率状态的更新

在一个二进制符号（MPS或LPS）编码完毕输出的同时，将由此获得的数据对原有的上下文概率模型进行更新，为下一个符号的编码提供更加准确的统计模型。这种将概率估计从一个状态转移到另一个状态更新是自动的，也是和上下文有关的，因而称为“上下文自适应”。实际上，对于一个给定概率状态，概率的更新取决于状态索引和已经编码的符号（MPS或LPS）。作为更新处理的结果，导出了一个新的概率状态，如果有必要，还要修改MPS的值。

图7.9表示了LPS估计的概率值在状态索引更新时的转移规律。如果出现一个MPS事件，只要将原来的状态索引简单地加1即可，除非在状态索引为62时出现MPS，状态索引值62保持不变，因为LPS概率已经达到最小（MPS概率已经达到最大）。如果出现一个LPS，状态索引就会有一个减少量，如图7.9中大弧线所示。这一规则对LPS的每次出现都是适用的，但有个例外。假设LPS在索引 $\sigma=0$ 的状态已经被编码，这对应与等概率情况（LPS的概率已经达到了最大值0.5，输入一个小概率符号LPS后就会变成大概率），状态索引保持不变，但MPS和LPS的值需要互换。

图7.9所示的概率索引值 σ 自适应更新规则，也可用表格7.15来定义。

（1）若MPS出现， σ 的更新按照表7.15中“MPS出现的 σ 值”增加1，直

至 $\sigma=62$; $\sigma=62$ 后，若再出现MPS，则 σ 保持62不变。

(2) 若LPS出现， σ 的更新按照表7.15中“LPS出现的 σ 值”的对应关系进行。需要指出的是，当 $\sigma=0$ 时，若LPS出现，更新后的 σ 值仍为0，但LPS与MPS的值互换。

(3) $\sigma=63$ 被固定用来对算术编码结束判断(MPS固定)进行编码，不参与自适应。

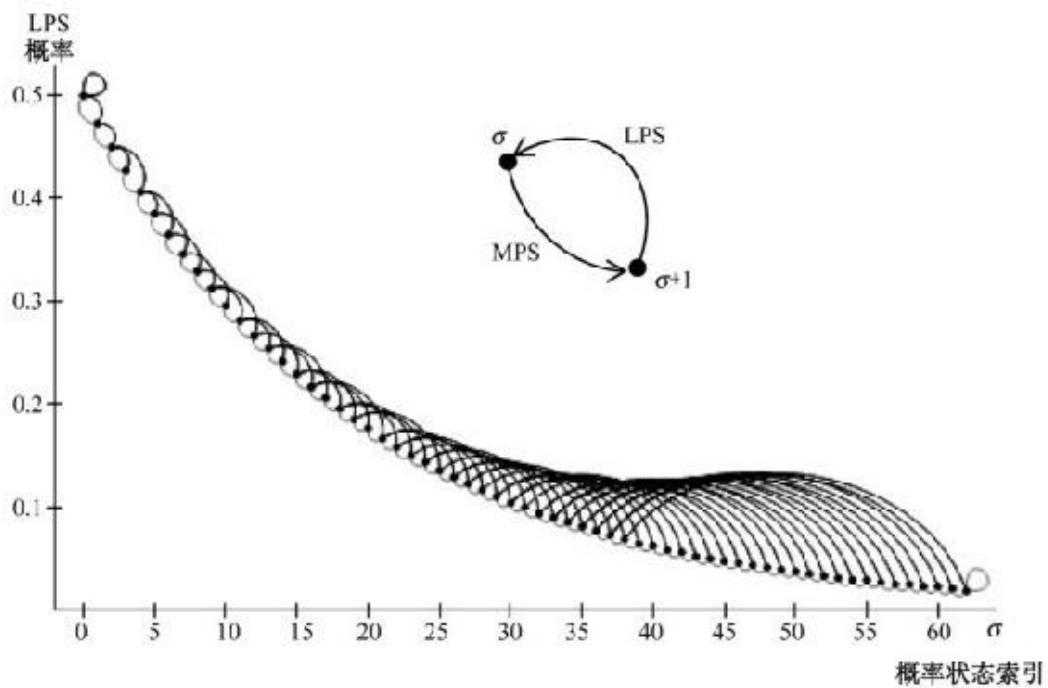


图7.9 概率状态转移示意

表7.15 状态转移表 (HEVC表格9-47)

| 原来的 σ 值 | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
|--------------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
| LPS 出现的 σ 值 | 0 | 0 | 1 | 2 | 2 | 4 | 4 | 5 | 6 | 7 | 8 | 9 | 9 | 11 | 11 | 12 |
| MPS 出现的 σ 值 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 |
| 原来的 σ 值 | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 | 29 | 30 | 31 |
| LPS 出现的 σ 值 | 13 | 13 | 15 | 15 | 16 | 16 | 18 | 18 | 19 | 19 | 21 | 21 | 22 | 22 | 23 | 24 |
| MPS 出现的 σ 值 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 | 29 | 30 | 31 | 32 |
| 原来的 σ 值 | 32 | 33 | 34 | 35 | 36 | 37 | 38 | 39 | 40 | 41 | 42 | 43 | 44 | 45 | 46 | 47 |
| LPS 出现的 σ 值 | 24 | 25 | 26 | 26 | 27 | 27 | 28 | 29 | 29 | 30 | 30 | 30 | 31 | 32 | 32 | 33 |
| MPS 出现的 σ 值 | 33 | 34 | 35 | 36 | 37 | 38 | 39 | 40 | 41 | 43 | 43 | 44 | 45 | 46 | 47 | 48 |
| 原来的 σ 值 | 48 | 49 | 50 | 51 | 52 | 53 | 54 | 55 | 56 | 57 | 58 | 59 | 60 | 61 | 62 | 63 |
| LPS 出现的 σ 值 | 33 | 33 | 34 | 34 | 35 | 35 | 35 | 36 | 36 | 36 | 37 | 37 | 37 | 38 | 38 | 63 |
| MPS 出现的 σ 值 | 49 | 50 | 51 | 52 | 53 | 54 | 55 | 56 | 57 | 58 | 59 | 60 | 61 | 62 | 62 | 63 |

本章参考文献

[1]Vivienne Sze,Madhukar Budagavi.High throughput CABAC entropy coding in HEVC[J].IEEE Trans.on Circuits and Systems for Video Technology,Dec.,2012,22 (12) : 1778-1791.

[2]Tung Nguyen,Philipp Helle,Martin Winken,et al.Transform coding techniques in HEVC[J].IEEE Journal of Selected Topics in Signal Processing,Dec.2013,7 (6) : 978-989.

[3]Vivienne Sze,Madhukar Budagavi.A comparison of CABAC throughput for HEVC/H.265 vs AVC/H.264[C].2013 IEEE Workshop on Signal Processing Systems,165-170.

[4]Mathias Wien.High Efficiency Video Coding: Coding Tools and Specification[M].Springer-Verlag Berlin Heidelberg,2015.

[5]High Efficiency Video Coding[S],ITU-T Rec.H.265 and ISO/IEC 23008-2,ITU-T and ISO/IEC JCT-VC,Mach 2013 (v.1),Oct.2014 (v.2),Apr.2015 (v.3).

[6]Wei Li,Peng Ren.Fast CABAC rate estimation for HEVC mode decision[C].2015 IEEE International Conference on Multimedia Big Data,171-175.

[7]D.Marpe,H.Schwarz,T.Wiegand.Context-based adaptive binary arithmetic coding in the H.264/AVC video compression standard[J].IEEE Trans.on Circuits and Systems for Video Technology,2003,13 (7) : 620-636.

[8]Advanced Video Coding for generic audio-visual services[S].ITU-T Rec.H.264 and ISO/IEC 14496-10 (AVC),ITU-T and ISO/IEC JTC 1,May 2012.

[9]J.Sole,R.Joshi,N.Nguyen,et al.Transform coefficient coding in HEVC[J].IEEE Trans.on Circuits and Systems for Video Technology,Dec.,2012,22 (12) : 1765-1777.

[10]Vivienne Sze, Madhukar Budagavi, Gary J.Sullivan.High Efficiency Video Coding (HEVC) :Algorithms and Architectures[M].Springer International Publishing Switzerland 2014.

[11]Bong-Hee Bae,Jin-Hyeung Kong.A design of pipelined-parallel CABAC decoder adaptive to HEVC syntax elements[C].IEEE International Symposium on Consumer Electronics (ISCE 2014) ,1-2.

[12]D.Marpe,H.Schwarz, and T.Wiegand,Context-based adaptive binary arithmetic coding in the H.264/AVC video compression standard[J],IEEE Transactions on Circuits and Systems for Video Technology,2003,13 (7) : 620-636.

第8章 HEVC的环路滤波

视频压缩编码是一种有损压缩方式，编码中的量化和计算误差会给重建图像带来不可恢复的失真，这里尚未考虑传输或存储等因素引起的图像损伤。解码重建图像的失真有多方面的表现，如方块效应（ Blocking Artifacts ）、亮度的失真（亮度变化）、彩色的漂移（色彩发生偏差、变色）等。其中方块效应是基于块的图像压缩方法中最明显一类可视缺陷，因此早在 H.263、MPEG-2 标准中，后来在 H.264/AVC 标准中，以及目前在 HEVC 中都采用了适当滤波的方法来消除方块效应，使解码重建图像的主观质量和客观质量都有较大提高，同时也提高了编码效率。

本章在简单阐述视频编码中方块效应的由来，减少方块效应的去方块滤波（ De-Blocking Filter,DBF ）的基础上，参照 H.264/AVC 的去方块效应滤波的机理，介绍了 HEVC 的环路滤波的原理和方法，着重分析了其中的类似 H.264/AVC 的去方块滤波和样值自适应补偿（ Sample Adaptive Offset,SAO ）两种滤波方法。

8.1 环路滤波

8.1.1 方块效应的产生

1. 方块效应

基于分块的视频编码机制，其处理过程往往是以块为单位独立进行的，其结果使得重建图像的块边界处的灰度值（彩色值）产生了不连续性。当一个边界两侧的图像相关性强，并且图像较平滑的时候，这种不连续性就在画面上形成了“方块效应”，人的视觉系统很容易察觉到隐隐约约的块边界。图8.1显示了典型的一幅图像经解码重建后的方块效应，（a）为原始图像，（b）为经反量化、反变换重建的图像，其方块效应非常明显。



(a) 原图像



(b) 压缩图像

图8.1 压缩图像的方块效应

2.产生原因

方块效应产生的原因是多方面的，但主要来自以下两个方面。

一是由变换、量化的误差引起的。在视频编码中，编码器基本上是逐块地对预测后的残差进行变换和量化，而量化过程是一个有损压缩的过程，因此在反换量化、反变换后重建的图像会出现误差，由于块之间的处理不一致，而且在方块边界处的误差格外大一些，造成在图像块边界上的视觉不连续。

另一个原因来自帧间预测的运动补偿过程。运动补偿的预测数据通常来自同一帧的不同位置上的内插点，也有可能是不同帧不同位置的内插点，因此运动补偿块不会绝对匹配，从而也会产生误差，引起图像在复制的边界上的不连续现象。而编码过程中的预测参考帧通常都来自这些重建

图像，这又导致了待预测图像失真。

3.块边界误差

块效应实际上是分块编码的重建块的误差，这些误差在整个块中都存在，只不过在边界处比较突出，易于观察到。当对像素或残差用方块变换进行编码时，反变换后重建方块的边界比内部像素的编码误差大。产生这个现象的原因是反变换块中的任意一个像素都是此块的DCT系数的加权和，内部点的重建是包含了四周围点的进行加权平均得到的，也比较充分地利用了相邻像素的相关性。而边界点所用的加权平均点的分布比较偏，靠近边界但在相邻块内的系数不能利用，丢失了一些周围像素的相关性，所以重建的误差相对较大。这种误差分布不均匀是形成方块效应，尤其是边界效应的一个重要原因。

4.环路滤波

为了消除或减轻块效应，我们可以用环路滤波（Loop Filter）的方法对重建图像进行滤波，降低重建误差。在环路滤波中，去方块滤波（DBF）是其中最重要的一类改善重建图像质量的方法。去方块滤波器的作用是通过“修正”重建块的像素值，尤其是边界附近的像素值，从而达到消除编解码算法带来的方块效应的目的。在H.264/AVC标准中就采用了去方块滤波的方法处理边界处的像素。而在HEVC的环路滤波中，除去方块滤波外，还增加了样点自适应补偿（SAO）滤波方法对所有的像素进行修正处理，进一步降低了重建图像的误差。

5.真伪边界

在进行环路滤波消除方块效应时，应该先判断该边界是图像的真实边界还是方块效应所形成的边界（伪边界）。对真实边界不应进行滤波处理，如在DCT边界上，正好是图像的边界，如物体的边缘等，若不加以判断而误认为是方块效应，则可能造成新的误差。而对假边界则要根据周围图像块的性质和编码方法采用不同程度的滤波。

8.1.2 环内滤波和环外滤波

1. 编码环路

变换编码和预测编码是两类不同的压缩编码方法，如果将这两种方法组合在一起，就构成目前经典的视频压缩混合编码方案，从H.261到HEVC基本如此。这种方案通常将图像分成一系列独立的小块，对它们分别使用DCT等变换进行空间冗余度的压缩，用帧间预测或运动补偿预测进行时间冗余度的压缩，以获得对视频图像更高的压缩效率。

混合编码器压缩处理的最本质的结构如图8.2所示，图中，T、IT表示正、反变换，Q、IQ表示正、反量化。由于混合编码把变换部分（T）放在预测环内，因此预测环本身工作在图像域内，便于使用性能优良、带有运动补偿的帧间预测。这一方案具有压缩性能高、编码技术成熟，以及编码延迟较短的等特点，成为活动图像压缩的主流方案。在视频压缩编码国际标准和建议中，例如，H.261、H.263、MPEG-1、MPEG-2、MPEG-4、H.264/AVC，以及最新的国际标准HEVC中都采用了这一混合编码方案。

如图8.2所示，混合编码器中预测残差信号经T、Q和VLC后形成压缩码流送往解码端的同时，还有一路经IQ、IT和前面的重建信号相加形成当前的重建信号。这样，就形成了编码器中的编码环路。在编码环路中实际上还包含一个完整的解码器，如图中的虚线框所标，它必须和远端的解码器一致，它的重建图像和解码端的解码器获得的重建图像是相同的。如果需要的话，在这里就可以看到重建图像，也能看到重建图像中的方块效应。

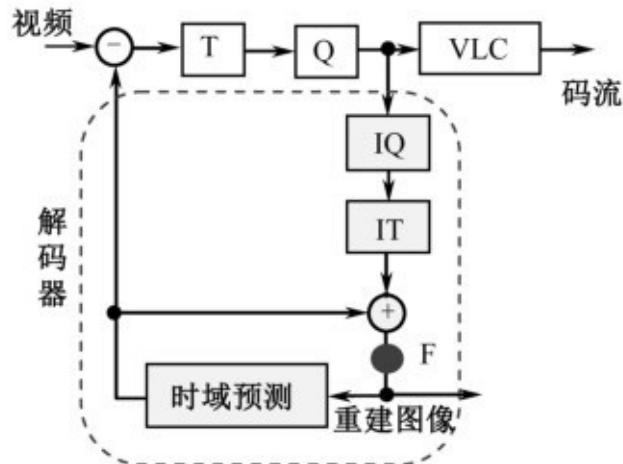


图8.2 混合编码器中的解码环路

消除块效应的环路滤波，它位于编码器预测环路中的反量化/反变换单元之后、重建的运动补偿预测参考帧之前，如图8.2中F点所示。因而，环路滤波是预测环路的一部分，属于环内处理，而不是环外的后处理。环路滤波的目标就是消除编码过程中预测、变换和量化等环节引入的失真。由于滤波是在预测环路内进行的，减少了失真，存储后为运动补偿预测提供了较高质量的参考帧。而且，主流视频标准的环路滤波都是安排在环内，因此，除非另加说明，现在的环路滤波基本上都是指环内滤波。

2. 环路滤波

在视频编码器中加入去方块滤波的方法有两种：一种是上述的环内滤波（或称环路滤波）的方法，另一种是环外滤波（后置滤波器）的方法。

（1）环外滤波

环外滤波器不在编解码环路内部出现，只是对解码端显示器缓存中的

重建图像进行去方块滤波运算。后置滤波器的方便之处在于不受标准的约束，可以比较自由地设计，但是后置滤波对图像质量的改善并不明显，而且由于滤波器位于编码环路以外，不参加预测，因而改进后的图像也不可能作为后续图像的参考帧。因此，后置滤波器一般不作为视频压缩标准中的内容，而是作为可选项。

(2) 环内滤波

环内滤波器处理编码环路中的数据。在编码器中，滤波后的重建图像帧作为后续编码帧的运动补偿参考帧；在解码器中，滤波后的图像输出显示。这要求所有与标准一致的解码器采用同一个滤波器，保持与编码器同步，才能保证解码器中的正确解码过程。当然，如果有必要，解码器还可以在使用环路滤波器的同时使用后置滤波器。

在编码环内使用滤波器比后置滤波器的优越之处在于：首先，环内滤波具有自适应性，能够适应不同边界，获得不同水平的图像质量。其次，由于在环内，滤波后的高质量重建图像可为后续编码帧提供更精确的预测，有利于提高编码效率。最后，在解码器中没有必要再为滤波器准备额外的帧缓存（后置滤波需要），可简化设备，减小误差扩散。

实验和工程实现已显示，环路滤波比后置滤波更能提高重建视频的主要客观评价的质量，同时还能有效降低解码器的复杂度。因此，环内滤波器被纳入编解码标准，作为标准的一部分。

尽管有以上这些优点，环路滤波器的复杂度还是较高的。即使经过很大的努力进行滤波算法的优化，如去除其中的乘法等措施，滤波器也常占到解码器计算复杂度的 $1/3$ 。

8.2 H.264/AVC的去方块滤波

由于HEVC的去方块滤波和H.264/AVC的去方块滤波原理相同，方法类似，因此这里先对H.264/AVC的去方块滤波作一简要介绍，这将非常有助于我们对HEVC的去方块滤波的理解。

8.2.1 自适应去方块滤波

H264/AVC中采用了去方块滤波技术后，在相同的峰值信噪比（PSNR）下，即相同图像质量，可以节省的码率超过9%。但去方块滤波给编码器带来了计算复杂度的显著增加，主要原因是它具有高度的自适应性，需要对所有的 4×4 小方块的边界及样点值进行边界强度判断和自适应滤波处理。对 4×4 的方块边界两边的各2个点进行计算、判断和修正，几乎涉及重建图像中的每个像素点。

根据去方块滤波的原理，去方块滤波在H.264/AVC编码器和解码器中是一致的，主要包括两个主要步骤，即对块边界强度的判定和对像素的滤波。自适应去方块滤波器对边界强度的判定，实际上是通过当前块的编码模式和参数来确定边界误差的等级，为接下来的滤波强度决策提供定量依据。去方块滤波过程实际上就是根据测定的边界强度，自适应地采用不同的滤波强度，对所涉及重建图像的边界周围的像素值进行修正，使其更接近原始视频的像素值。

8.2.2 边界强度测定

1. 块边界强度

编码块的边界强度 (Boundary Strength,BS) 决定去方块滤波器选择的滤波参数，并控制去方块效应的程度。对所有 4×4 亮度块的边界，边界强度的参数值在0~4。表8.1列出了BS与相邻图像块的模式及编码条件的关系。表中的条件是从上至下逐条判断的，某一条满足，即给BS赋相应的值。

表8.1 DBF的边界强度的等级

| 4×4 块模式与条件 | 边界强度 BS |
|-------------------------|---------|
| 边界两边有一个为帧内预测块 & 边界为宏块边界 | 4 |
| 边界两边有一个为帧内预测块 | 3 |
| 边界两边有一个块为残差块编码 | 2 |
| 边界两边块的运动矢量之差不小于1个亮度像素距离 | 1 |
| 边界两边块的运动补偿参考帧不同 | 1 |
| 其他 | 0 |

从表8.1可以看出，边界强度的大小主要取决于编码中产生的残差大小。帧内编码和帧间编码相比，参考样点相对较少，因此产生的残差相对来说比较大，所以如果相邻两块有帧内编码，则边界强度较大，BS为3或4。如果相邻两块都是帧间编码，并且其中有一块为残差编码，说明此时残差比较大，BS为2。当运动补偿找到一样的参考块时，不需对残差编码，但是相邻两块所使用的参考块不在同一帧内，或者运动矢量有一定差距，BS为1。其他情况说明边界两边差别很小，BS为0，不需要滤波。

在实际滤波算法中，BS决定对边界的滤波强度。当BS值为4时，表示要用较强的滤波模式；当BS值为0时，表示不需要对边界进行滤波；当BS值为1~3时，表示可采用普通滤波模式。BS的值影响滤波器的强度，实质上是影响对边界旁样点的最大修正程度。

上面的BS值是针对亮度信号块而言的，色度块边界滤波的BS值不另外单独计算，简单地从相应的亮度块边界的BS值复制即可。

2.边界特征

自适应去方块滤波器的主要难点在于判断是否要对一个具体的块边界进行滤波，同时确定其滤波强度。过强的滤波可能会导致图像细节区过度平滑，而滤波强度不够又会让方块效应降低主观质量。确定一个块边界两

边的像素是否需要滤波，主要取决于块边界两边的重建像素值的差值特征，自适应决定是否进行滤波操作，如果需要滤波，则要安排适当的滤波强度和滤波深度。

在去方块滤波中，非常重要的是区分图像中的真实边界和由 DCT 变换系数量化而造成的虚假边界。为保持图像的逼真度，应该尽量在滤除假边界的同时，保持图像真实边界不被滤波。一般来说，真实边界两侧像素的梯度差要比量化误差造成的虚假边界两侧像素值梯度差大。

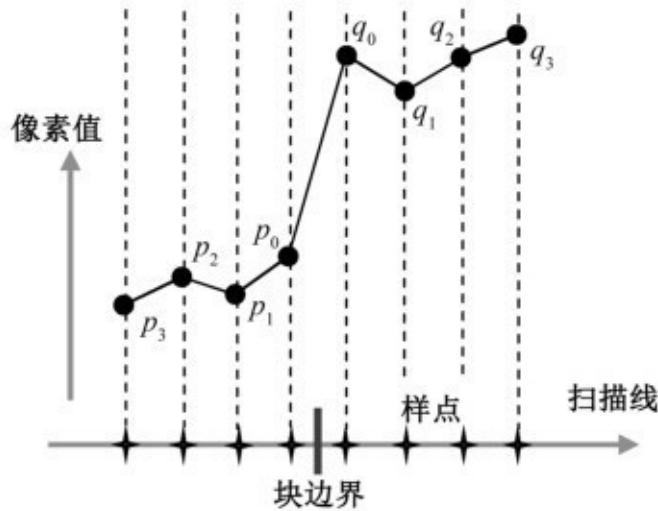


图8.3 相邻块边界两边样点的灰度值

为了区分这两种情况，需要分析每个被滤波的边界两边的样点值。这里参考图8.3来进行简单的分析，定义跨越两相邻 4×4 块边界的一条直线上的样点（如图中“+”所标）的亮度值为 p_3 、 p_2 、 p_1 、 p_0 、 q_0 、 q_1 、 q_2 、 q_3 ，左边为P块，右边为Q块。根据它们的关系，可以大致判定在 p_0 和 q_0 之间是否为真边界，是否需要进行去方块滤波。

对于BS值为非0的边界，为区分上述真假两种边界，定义一对与量化参数QP有关的阈值参数 α 和 β ，用来检查图像内容，以决定每个样点是否要滤波。只有下述三个条件同时满足，边界两边的样点才需要进行滤波：

$$|p_0 - q_0| < \alpha(\text{Index}_A), \quad |p_1 - p_0| < \beta(\text{Index}_R), \quad |q_1 - q_0| < \beta(\text{Index}_B) \quad (8.1)$$

其中， α 和 β 值由经验函数关系得到：

$$\alpha(\text{Index}_A) = 0.8(2^{\text{Index}_A/6} - 1), \quad \beta(\text{Index}_B) = 0.5\text{Index}_B - 7 \quad (8.2)$$

其中， Index_A 和 Index_B 由下式决定：

$$\text{Index}_A = \text{Clip3}(0, 51, \text{QP} + \text{Offset}_A), \quad \text{Index}_B = \text{Clip3}(0, 51, \text{QP} + \text{Offset}_B) \quad (8.3)$$

其中， $\text{Clip3}(\)$ 是限幅函数，范围为0~51。QP为相邻两块的平均量化参数， Offset_A 和 Offset_B 是在编码器中选择的偏移，以在slice级范围内控制去方块滤波的性能。

由于量化参数QP的大小基本上决定了编码图像的质量，阈值 α 、 β 与QP的关系将滤波强度与滤波前重建图像的质量联系起来。因为阈值随着QP增加而增加，增加了同时满足的3个阈值关系的可能。当QP较大时，

编码误差一般也较大，方块效应也越严重，产生虚假边界的可能性越大，边界也越需要滤波。

阈值 α 和 β 的大小值受偏移值 $Offset_A$ 和 $Offset_B$ 的影响。因此，编码器可以通过选择不同的偏移值（ $Offset_A$ 和 $Offset_B$ ）来调整滤波器阈值 α 和 β ，相对于0偏移增加或减少滤波强度。负的偏移值可以减少滤波强度，有助于改善小的空间细节的逼真度；正的偏移可以增加滤波强度，消除较强方块效应，改善重建图像的主观质量。

8.2.3 去方块滤波过程

为保证编码器和解码器的滤波过程完全一致，对每个编码图像的滤波运算必须按规定顺序进行。同时，滤波应该在适当的位置上进行，这样边界两边修改过的样点值作为后续运算的输入值而不至于引入误差。

滤波是基于宏块基础上的，先对垂直边界进行水平滤波，再对水平边界进行垂直滤波。对当前宏块的两个方向上的滤波都完成后，才能进行后面宏块的滤波。对图像中宏块的滤波按光栅扫描的方式进行。对一个 16×16 亮度宏块，包含16个 4×4 子块的边界，形成4条垂直边界和4条水平边界。先从左到右依次对宏块的垂直边界滤波，再从上到下依次对水平边界滤波。和亮度宏块对应的 8×8 色度宏块滤波次序和亮度滤波类似。

1. 普通滤波

边缘强度BS值为1、2、3时，采用普通滤波方式，包括滤波和限幅两部分。只需根据输入 p_1 、 p_0 、 q_0 、 q_1 的值来修改紧邻边界的 p_0 、 q_0 值为 p'_0 、 q'_0 。但如果边界像素差值小于 β 阈值，满足式（8.4）和式（8.5），说明很可能存在虚假边界，则还需对 p_1 、 q_1 作修改。

$$|p_2 - p_0| < \beta(\text{Index}_B) \quad (8.4)$$

$$|q_2 - q_0| < \beta(\text{Index}_B) \quad (8.5)$$

(1) p_0 、 q_0 的滤波和限幅：

先根据边界两边的样点值计算初始值 Δ_{0i} ，

$$\Delta_{0i} = (4(q_0 - p_0) + (p_1 - q_1) + 4) \gg 3 \quad (8.6)$$

再对这个初始值进行限幅后得到 Δ_0 ，

$$\Delta_0 = \text{Clip3}(-c_0, c_0, \Delta_{0i}) \quad (8.7)$$

其中，Clip3为限幅函数，限幅的容限值在 $-c_0$ 和 c_0 之间。由 Δ_0 得到 p_0 、 q_0 的滤波值，

$$p'_0 = p_0 + \Delta_0, \quad q'_0 = q_0 + \Delta_0 \quad (8.8)$$

(2) p_1 、 q_1 的滤波和限幅：

如果式(8.4)成立， p_1 滤波后的值按下式计算：

$$p'_1 = p_1 + \Delta_{p1} \quad (8.9)$$

同样，如果式(8.5)成立， q_1 滤波后的值按下式计算：其中， Δ_{q1} 、 Δ_{q1i} 分别由初始值 Δ_{p1i} 、 Δ_{q1i} 限幅得到：

$$q'_1 = q_1 + \Delta_{q1} \quad (8.10)$$

$$\Delta_{p1} = \text{Clip3}(-c_1, c_1, \Delta_{p1i}), \quad \Delta_{q1} = \text{Clip3}(-c_1, c_1, \Delta_{q1i}) \quad (8.11)$$

而 Δ_{p1i} 、 Δ_{q1i} 分别为

$$\Delta_{p1i} = (p_2 + (p_0 + q_0 + 1) \gg 1 - 2p_1) \gg 1 \quad (8.12)$$

$$\Delta_{q1i} = (q_2 + (q_0 + p_0 + 1) \gg 1 - 2q_1) \gg 1 \quad (8.13)$$

(3) 限幅的范围

如果上述的初始值 Δ_{0i} 、 Δ_{p1i} 和 Δ_{q1i} 直接应用在滤波计算中，则可能导致滤波截止频率过低，出现图像模糊。自适应滤波器的一个重要作用是限制 Δ 的值，这个过程称为限幅。对于边界两边不同的样点，限幅的过程不同。

对于内部样点， p_1 和 q_1 ，用于滤波的 Δ 值被限制在 $-c_1 \sim c_1$ 的范围内， c_1 是根据不同的 $Index_A$ 和BS，从一个二维表中查找出的参数。 $Index_A$ 和BS越大，则 c_1 也越大，允许更强的滤波。

对于边界样点， p_0 和 q_0 ，现将用于滤波的 Δ_{0i} 的限幅 c_0 定为 c_1 ，如果式(8.4)和式(8.5)都成立，说明边界内部的变化强度小于 β 阈值，需要对边界进行更强的滤波，将 c_0 增加1。

2. 强滤波

BS值为4时，对亮度值进行强滤波，涉及修改的边界两边的像素数比普通滤波要多，根据图像内容判断选择较强的4抽头或5抽头滤波器，或者是较弱的3抽头滤波器。4抽头或5抽头滤波器对边界两边的边界点及两个内部点进行修正，而3抽头滤波器仅改变边界点。

如果下面的跨边界差异的约束条件成立，则使用较强的滤波器：

$$|p_0 - q_0| < (\alpha >> 2) + 2 \quad (8.14)$$

对于亮度滤波，当式(8.4)和式(8.14)都成立时，根据下式计算滤

波后的样点值：

$$\begin{aligned} p'_0 &= (p_2 + 2p_1 + 2p_0 + 2q_0 + q_1 + 4) \gg 3 \\ p'_1 &= (p_2 + p_1 + p_0 + q_0 + 2) \gg 2 \\ p'_2 &= (2p_3 + 3p_2 + p_1 + p_0 + q_0 + 4) \gg 3 \end{aligned} \quad (8.15)$$

以上是边界一侧p点值的修改方法，另一侧的q点值的修正方法与此相同，只是在选择亮度滤波器时，用式(8.5)代替式(8.4)。

8.3 HVEC的环路滤波

由于HEVC依旧沿用的是基于块的帧内/帧间预测以及变换编码过程，因此重建图像中相邻块的不连续感依然存在，这些不连续的表现形式有很多，如振铃效应、块效应、物体模糊等。为解决这些问题，和H264/AVC一样，HEVC在重建图像被写入解码器缓存之前需要进行环路滤波（In-loop Filtering）处理。

HEVC的环路滤波比H.264/AVC增加了一项，包括去方块滤波（De-Blocking Filter,DBF）和样点自适应补偿（Sample Adaptive Offset,SAO）两项，它们在编码环路中的先后顺序如图8.4所示。

DBF与H.264/AVC的DBF作用类似，旨在减小图像编码过程中由于块分割而造成的块效应。SAO则是HEVC新引入的对重建图像的误差补偿机制。DBF仅用于位于块边界处的像素，而SAO滤波器自适应地用于满足某种条件的所有像素，如处于一定梯度结构中的像素。

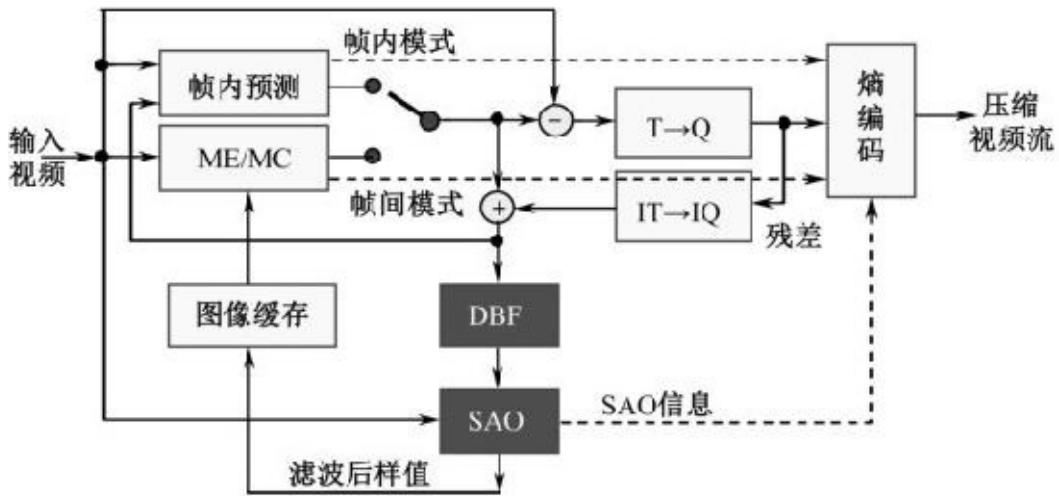


图8.4 HEVC编码器的环路滤波

8.3.1 自适应去方块滤波

HEVC 标准的去方块滤波器（DBF），针对不同的边界类型（3种）决定是采用强滤波处理、弱滤波处理还是不处理。这种判决是由特定的阈值对边界两边像素的梯度和两边块中量化参数的情况来而定的。因此去方块滤波处理作用于邻近PU或TU边界的像素，但要排除块边界同时也是图像边界的情况，排除去方块滤波不能够穿过的条或片的边界（这是一个选项，可由编码器标注）。必须注意，TU和PU边界两者都需考虑，因为在帧内预测CU的某些情况下，PU边界并不总是和TU边界排列在一条线上的。序列参数集（SPS）中的语法元素和条的头信息控制着去方块滤波器是否要穿过条和片的边界。

在H.264/AVC中，去方块滤波用于 4×4 取样栅格基线，和H.264/AVC不同，HEVC的去方块滤波仅限用于排列在 8×8 取样栅格边缘的亮度和色度像素。在没有显眼的可视质量的下降的情况下，这一限制减少了滤波计算复杂度。由于阻止了邻近滤波操作的串行交互，从而为DBF的快速并行处理带来了可能。

HEVC去方块滤波的强度由若干类似H.264/AVC的语法元素的值来控制，但只用了3个强度等级，即0、1、2，而不是如H.264/AVC中多达的5个等级。

在HEVC中，去方块滤波处理的顺序规定为首先对整个图像的垂直边缘进行水平滤波，然后对水平边缘进行垂直滤波。这种特殊的顺序保证了不管是多个水平滤波或者是垂直滤波处理都可被并行处理，或者还可一个CTB接一个CTB地实现，仅有很短的处理延时。

8.3.2 样点自适应补偿

位于DBF后的样点自适应补偿（SAO）技术是HEVC采纳的一种新环内滤波工具。不同于去方块滤波只对边界像素进行，样点自适应补偿原则上面对块中所有的样点。SAO对图像块内像素的特征进行分类，属于同一类的像素点将获得相同的补偿值，补偿的方向是让像素值向原始像素值靠拢，从而达到降低图像失真的目的。

SAO是一种修改解码像素值的处理，即在去方块滤波后对每个像素值有条件地增加一个补偿值，补偿值是查找由编码器传来的表格而获得的。

SAO滤波按照语法元素sao_type_ idx表示的每个CTB区域所选的滤波类型。sao_type_idx的0值表示SAO滤波器没有用于CTB，而值1和值2分别表示带补偿（Band Offset,BO）和边缘补偿（Edge Offset,EO）滤波类型。BO和EO是两类常用的高效SAO方法，可以选择使用其中的某一种。

8.4 HEVC的去方块滤波

去方块效应滤波器比一般低通类滤波器要复杂得多，因为它在滤波过程中要确定块边缘间不连续的程度，判别这种不连续是否为真实边界等操作，使得去方块滤波计算相当量大，在视频解码总计算量中的占比有可能高达三分之一左右。为此，HEVC 中的去方块滤波虽然原理和H.264/AVC 标准一样，但与H.264/AVC的去块滤波器相比，HEVC采取了精简措施，有效降低了DBF的计算复杂度，而且保证了良好的滤波效果。

HEVC的去方块滤波主要包括三个步骤：首先根据编码参数和边界强度（BS）判断目标边界是否要滤波；如果需要，然后再根据阈值判定需要进行普通滤波还是强滤波操作；最后则是具体的滤波处理。

8.4.1 去方块滤波单元

1.8×8的滤波处理块

与H.264/AVC相比，去方块滤波的复杂度在HEVC中有明显的下降。HEVC中去方块滤波的设计防止图像块的空间独立性，对于一条边界，同时考虑两边的块的相关性，而不进行重复的滤波操作。这个过程改变了边界两边至少处理6个样点的值，判定过程至少涉及边界两边4个样点值的要求。任何垂直边界都可以与其他垂直边界并行进行滤波运算，而且经过垂直滤波后的样点值会作为输入值参与水平边界滤波的运算。

HEVC 的去方块滤波通常用于预测单元（PU）和变换单元（TU）边界周边的点，如果这些边界同时又是图像边界时则不进行去方块滤波操作。H.264/AVC中去方块滤波的处理单元为 4×4 采样点大小，HEVC则不同，无论是亮度块还是色度块，滤波器大小都是 8×8 个采样点，且相互不

重叠，如图8.5所示。图中虚细线表示待滤波的 8×8 大小的块边界，粗线表示进行DBF的 8×8 大小的处理块。每个处理块都跨越4个 8×8 待滤波块，包含一个“+”字形的边缘。这样一来，每个处理块的目标就是对“+”字形的边界进行滤波，且包含了自身所需的所有滤波数据，这就使得滤波操作可以对每个 8×8 处理块单独进行操作，有利于HEVC对DBF进行并行滤波操作。HEVC的DBF在降低滤波计算复杂度的同时，图像的视觉质量也不会出现明显下降。

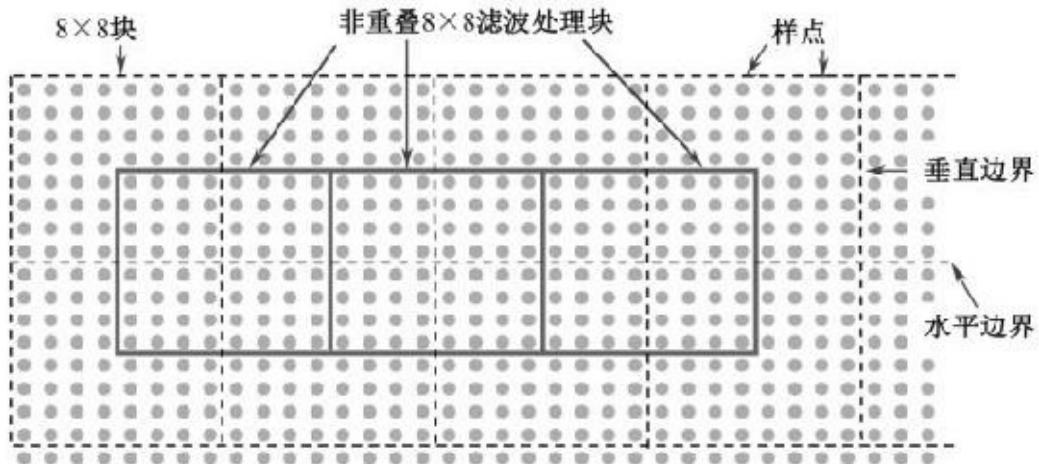


图8.5 块的垂直边界和水平边界

2.利于并行操作的滤波顺序

HEVC中的垂直边界与水平边界的滤波顺序也与H.264/AVC不同，在H.264/AVC中按照宏块来操作，而HEVC的滤波顺序与块的位置无关，并且在解码端滤波顺序也不改变，这样的设计降低了实现的复杂度。为保证编码器和解码器中的滤波过程完全一致，对每幅编码图像的滤波运算必须按照规定的顺序执行。滤波是基于 8×8 块的，边界类型edgeType分为两种，即垂直类型边界（edgeType=0）和水平类型边界（edgeType=1）。对于需要滤波的边界按照以下顺序依次进行处理：①亮度分量的垂直边界；②亮度分量的水平边界；③色度分量的垂直边界；④色度分量的水平边界。

HEVC 的去方块滤波是以条（slice）为单位进行的，编码器和解码器都是首先对条内部进行滤波运算，再对条的边界滤波，这样的设计可实现环路滤波中条级结构的并行计算过程。

8.4.2 边界强度的判定

滤波的边界强度（BS）判定，主要依据是预测单元边界以及变换单元边界的编码模式和编码参数。HEVC中边界强度有3种，分别对应BS=0、1、2三个等级。边界强度的确定流程如图8.6所示，设边界两边的图像块分别为P和Q，根据它们帧内编码模式、非零变换系数是否存在以及运动信息、参考图像等因素进行判决。

从图8.6中可以看出，当边界两边只要有一个块是帧内模式编码，则边界强度BS=2，边界强度的值最大，说明方块效应有可能最为严重，也说明在混合视频编码框架中，帧内编码是方块效应产生的主要根源。

如果P和Q都不是帧内模式，则接着顺序检查：

- (1) P或者Q中是否有非0的变换系数？
- (2) P和Q的运动补偿的参考图像是否不同？
- (3) P和Q的运动矢量的个数是否不同？

(4) P和Q的运动矢量的分量之差是否大于或大于一个整像素间隔？

只要遇到一个肯定的答复，这时的边界强度 $BS=1$ 。如果4项检测全部为否定答复，这时的边界强度 $BS=0$ 。当边界强度为0时不需进行去方块滤波运算，边界强度为1或2时才需要进行滤波运算。

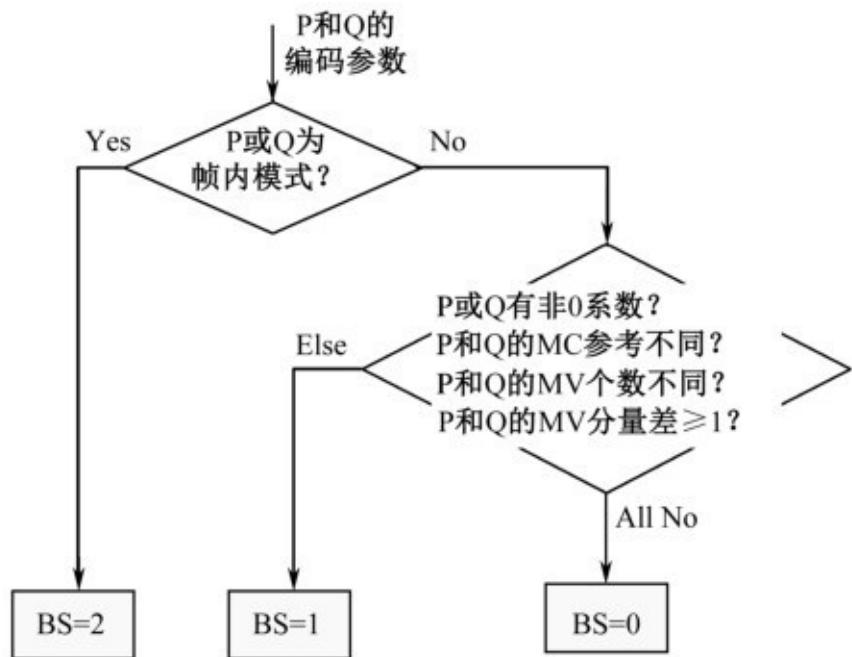


图8.6 边界强度计算流程

8.4.3 滤波强度的判定

如果边界强度BS大于0，就要对亮度块边界进行滤波。由于图像局部特性的不同，产生方块效应的程度也不同。因此，针对不同程度的方块效应，其滤波强度也应有所不同。在HEVC中，为去方块滤波强度设置了两种模式，一种是普通滤波强度的普通模式，另一种是较强滤波强度的加强模式。对于具体的边界，到底应采用哪种滤波模式，需要根据边界两边图像的特征来决定，实际上就是根据边界附近像素值的情况，通过一定的数值判断，来确定当前边界是需要进行普通滤波还是强滤波。具体的判断方法如下。

1. 滤波强度阈值

如图8.7所示，块边界由左右两个 8×8 块为P、Q，分为上下两部分，各为4行。上方两个框标识的两行（第1和第4行）样点值确定前4行的滤波强度，其结论用于前4行的滤波。在参与计算的每一行中，选取边界周围个8点（一边4个）参与滤波强度判决运算。同理，在同一边界块中，下方2个框决定后4行的滤波强度。可见，虽然HEVC的去方块滤波是针对 8×8 块进行的，但实质上是细分为两个 8×4 块分别进行的。

上述情况是针对垂直边界而言的，其实，所有的方法对于水平边界的滤波强度判决也是成立的，只要将图8.7旋转90度，将像素行改为列即可。

以上方的前4行样点值为例，在前面判决边界强度 $BS > 0$ 的条件下，仍需要满足式（8.16）条件，才会对上方4行样点值形成的边界滤波强度进行判断并执行滤波操作。

$$|P_{2,6} - 2P_{1,6} + P_{0,6}| + |P_{2,3} - 2P_{1,3} + P_{0,3}| + |q_{2,9} - 2q_{1,9} + q_{0,9}| + |q_{2,3} - 2q_{1,3} + q_{0,3}| < \beta \quad (8.16)$$

上式中，第1、2项绝对值表示P块的第0、3行像素的2阶变化率，第3、4项绝对值表示Q块的第0、3行像素的2阶变化率。整个4项之和表示边界区域的纹理程度，纹理程度越大，表示该区域变化越大，越不需要滤波。相反，当纹理程度小于某一阈值 β 时，则需要进行去方块滤波。

β 值取决于量化预测残差系数的QP值。实际应用中，将 β 的值制成为由QP为索引查找的表格，以供使用，免除临时计算。 β 的值和QP之间基本上呈正变的关系，随着QP的增加， β 的值也逐渐增加，如图8.8所示。

| | | | | | | | |
|-----------|-----------|-----------|-----------|-----------|-----------|-----------|-----------|
| $p_{3,0}$ | $p_{2,0}$ | $p_{1,0}$ | $p_{0,0}$ | $q_{0,0}$ | $q_{1,0}$ | $q_{2,0}$ | $q_{3,0}$ |
| $p_{3,1}$ | $p_{2,1}$ | $p_{1,1}$ | $p_{0,1}$ | $q_{0,1}$ | $q_{1,1}$ | $q_{2,1}$ | $q_{3,1}$ |
| $p_{3,2}$ | $p_{2,2}$ | $p_{1,2}$ | $p_{0,2}$ | $q_{0,2}$ | $q_{1,2}$ | $q_{2,2}$ | $q_{3,2}$ |
| $p_{3,3}$ | $p_{2,3}$ | $p_{1,3}$ | $p_{0,3}$ | $q_{0,3}$ | $q_{1,3}$ | $q_{2,3}$ | $q_{3,3}$ |
| $p_{3,4}$ | $p_{2,4}$ | $p_{1,4}$ | $p_{0,4}$ | $q_{0,4}$ | $q_{1,4}$ | $q_{2,4}$ | $q_{3,4}$ |
| $p_{3,5}$ | $p_{2,5}$ | $p_{1,5}$ | $p_{0,5}$ | $q_{0,5}$ | $q_{1,5}$ | $q_{2,5}$ | $q_{3,5}$ |
| $p_{3,6}$ | $p_{2,6}$ | $p_{1,6}$ | $p_{0,6}$ | $q_{0,6}$ | $q_{1,6}$ | $q_{2,6}$ | $q_{3,6}$ |
| $p_{3,7}$ | $p_{2,7}$ | $p_{1,7}$ | $p_{0,7}$ | $q_{0,7}$ | $q_{1,7}$ | $q_{2,7}$ | $q_{3,7}$ |

图8.7 决定滤波模式的样点值

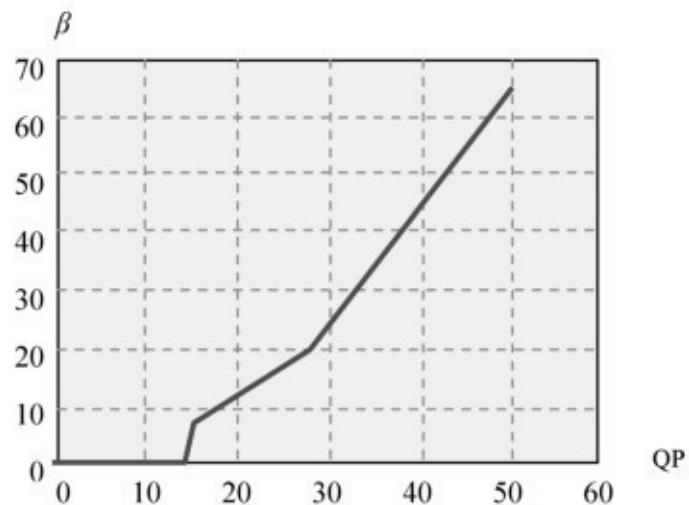


图8.8 QP- β 关系曲线

在阈值式(8.16)成立的块边界区域才需要进行去方块滤波强度判决和滤波操作。类似，对下方4行样点值，需要满足类似式(8.16)的条件，才会对下方4行样点值进行滤波强度判决和滤波操作。

同样将上述判断用于垂直方向滤波，只需将行、列的下标调换位置。一旦满足式(8.16)条件，且确定边界强度大于0，就需执行去方块滤波过程。

2. 阈值参数的导出

如果式(8.16)成立，确定执行去方块滤波过程后，就要考虑选取何种滤波模式，是普通滤波模式还是加强滤波模式。每个块边界都要根据自身的信号特征选择相应的滤波模式。可用式(8.17)、式(8.18)和式(8.19)进行两种滤波模式的判定：

$$|p_{2,i} - 2p_{1,i} + p_{0,i}| + |q_{2,i} - 2q_{1,i} + q_{0,i}| < \beta/8 \quad (8.17)$$

$$|p_{3,i} - p_{0,i}| + |q_{3,i} - q_{0,i}| < \beta/8 \quad (8.18)$$

$$|p_{0,i} - q_{0,i}| < 2.5t_c \quad (8.19)$$

上面3式中， $i=0,3$ 。和阈值 β 类似，阈值参数 t_c 也取决于QP的值，和QP之间基本上呈近似的指数关系，随着QP的增加， t_c 的值会迅速增加，如图8.9所示，实际应用中也是由预制的表格提供。

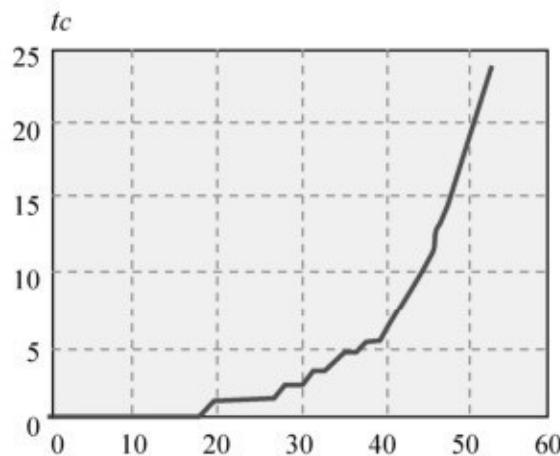


图8.9 t_c -QP关系曲线

实际上，阈值参数 β 、 t_c 的值和边界两边的QP值（即P块的 QP_P 和Q块的 QP_Q ）的平均 QP_L 有关：

$$QP_L = (QP_P + \tilde{QP}_Q + 1) \gg 1 \quad (8.20)$$

由 QP_L 限幅导出参数Q如下：

$$Q = \text{Clip3}(0, 51, QP_L + (\text{slice_beta_offset_div2} \ll 1)) \quad (8.21)$$

其中， $\text{slice_beta_offset_div2}$ 是针对不同条（slice）的补偿值。 β 、 t_c 的值除和 QP 有关外，还和像素的比特深度 BitDepth_Y 有关：

$$\beta = \beta' \cdot (1 << \text{BitDepth}_Y - 8) \quad (8.22)$$

$$t_c' = t_c' \cdot (1 << \text{BitDepth}_Y - 1) \quad (8.23)$$

HEVC提供了由参数Q导出阈值 β' 和 t_c' 的关系，具体见表8.2。

表8.2 由Q导出阈值变量 β' 和 t_c'

| Q | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 |
|----------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
| β' | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 6 | 7 |
| t_c' | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Q | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 | 29 | 30 | 31 | 32 | 33 | 34 | 35 |
| β' | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 20 | 22 | 24 | 26 | 28 | 30 | 32 |
| t_c' | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 2 | 2 | 2 | 2 | 3 | 3 | 3 | 3 | 4 | |
| Q | 36 | 37 | 38 | 39 | 40 | 41 | 42 | 43 | 44 | 45 | 46 | 47 | 48 | 49 | 50 | 51 | 52 | 53 |
| β' | 34 | 36 | 38 | 40 | 42 | 44 | 46 | 48 | 50 | 52 | 54 | 56 | 58 | 60 | 62 | 64 | - | - |
| t_c' | 4 | 4 | 5 | 5 | 6 | 6 | 7 | 8 | 9 | 10 | 11 | 13 | 14 | 16 | 18 | 20 | 22 | 24 |

3.滤波强度的判定

如果对于P、Q块的第0行和第3行，式(8.17)、式(8.18)和式(8.19)都成立的话，选择强滤波模式，否则为普通滤波模式。条件式(8.17)用来检测块边界两边样点值的变化程度，条件式(8.18)用来检测块边界两侧是否平滑，条件式(8.19)检查在块边界两侧样点值的强度差异是否超过特定阈值。

普通滤波模式又分为两种情况，这两种情况的不同在于边界两边需要改变的像素数的不同，两种情况的选择依据下面的两个公式：

$$|p_{2,0} - 2p_{1,0} + p_{0,0}| + |p_{2,3} - 2p_{1,3} + p_{0,3}| < 3\beta/16 \quad (8.24)$$

$$|q_{2,0} - 2q_{1,0} + q_{0,0}| + |q_{2,3} - 2q_{1,3} + q_{0,3}| < 3\beta/16 \quad (8.25)$$

式(8.24)用于P块的判断，式(8.25)用于Q块的判断。如果式(8.24)或式(8.25)成立，则在P块和Q块中，靠近块边界的一个像素值改变，否则只改变最靠近边界的一个像素值。式(8.24)和式(8.25)中的阈值比式(8.16)中的阈值小，而比式(8.18)中的阈值大。

整个滤波强度判定的流程如图8.10所示。首先是边界的判定过程，主要是根据边界类型来决定。滤波过程可以针对三种边界执行滤波操作：CU边界、TU边界和PU边界。通常，CU边界一定会进行滤波运算，这是因为CU边界往往也是TU和PU的边界。当PU边界包含在TU内时，该PU边界则不执行滤波。接着是边界强度BS判定过程，只有当BS为1或2时才需要滤波。然后是判定式(8.16)的阈值条件是否满足，只有当阈值条件满足，也就是边界附近的像素波动较大时需要加以滤波。最后判定是否全部满足式(8.17)、式(8.18)和式(8.19)，如果全部满足，则需要进行强滤波，否则需要进一步检查式(8.26)。

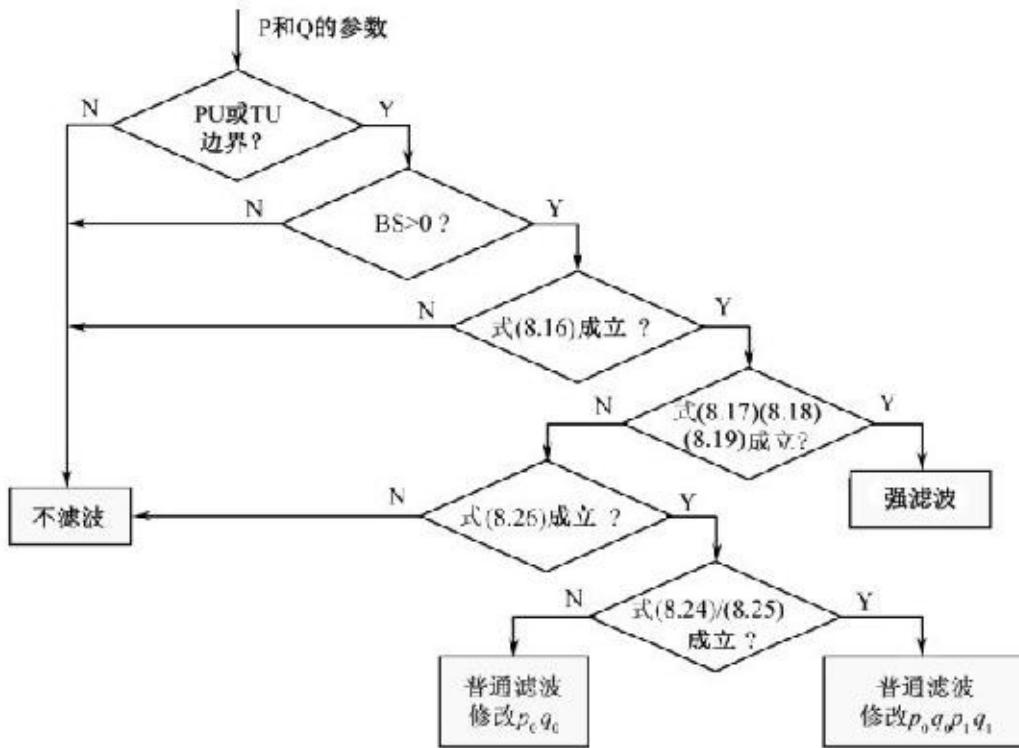


图8.10 滤波强度计算流程

如果式 (8.26) 不成立，说明当前边界大多是自然边界，无须滤波。如果成立，则需进行普通滤波。普通滤波又分两种情况，看它是否满足式 (8.24) 或式 (8.25)，判定在普通滤波时处理边界滤波的几个像素。

上面判断过程中的式 (8.26) 为：

$$|\delta_0| < 10t_c \quad (8.26)$$

其中， δ_0 表示的是滤波时像素最大的修正量。

8.4.4 去方块滤波过程

HEVC的去方块滤波的过程基本和H.264/AVC类似，且总体说来要比H.264/AVC精简。

1. 普通滤波过程

普通滤波运算每边修改2个像素，每个像素包含滤波运算和限幅运算两步操作。

(1) 滤波 p_0 和 q_0

对紧贴边界两边的点 p_0 和 q_0 进行滤波，基本滤波运算实质上就是按照一定规律修正该像素的值，滤波后的 p'_0 和 q'_0 值按下式计算：

$$p'_0 = p_0 + \Delta_0, \quad q'_0 = q_0 + \Delta_0 \quad (8.27)$$

其中， Δ_0 为修正量，即“滤波”。

上式中修正量 Δ_0 是 δ_0 限幅（clipping）以后的值，由限幅运算完成：

$$\Delta_0 = \text{Clip3}(-c, c, \delta_0) \quad (8.28)$$

其中，Clip3为限幅函数，正数c为限幅的范围。采取限幅操作的目的和H.264/AVC的去方块滤波中的限幅是一样的，也是限制 Δ_0 的值在一定范围内，以防滤波的 Δ_0 值改变太大而导致滤波截止频率过低，出现图像模糊的现象。

δ_0 的计算公式如下：

$$\delta_0 = (9(q_0 - p_0) - 3(q_1 - p_1) + 9) \gg 4 \quad (8.29)$$

(2) 濾波 p_1 和 q_1

对边界两边的点 p_1 和 q_1 进行濾波，濾波后的 p_1' 和 q_1' 值为，其濾波和

$$p_1' = p_1 + \Delta_{p1}, \quad q_1' = q_1 + \Delta_{q1} \quad (8.30)$$

限幅过程如下：

$$\Delta_{p1} = \text{Clip}(\delta_{p1}, c), \quad \Delta_{q1} = \text{Clip}(\delta_{q1}, c) \quad (8.31)$$

δ_{p1} 、 δ_{q1} 的计算公式如下：

$$\delta_{p1} = ((p_2 + p_0 + 1) \gg 1 - p_1 + \Delta_0) \gg 1, \quad \delta_{q1} = ((q_2 + q_0 + 1) \gg 1 - q_1 + \Delta_0) \gg 1 \quad (8.32)$$

2. 强滤波过程

强滤波比普通滤波运算要影响边界侧更多的像素：每边修改3个像素。
P块边的三个像素 p_0 、 p_1 、 p_2 修正后分别为：

$$p'_0 = p_0 + A_{0s}, \quad p'_1 = p_1 + A_{1s}, \quad p'_2 = p_2 + A_{2s} \quad (8.33)$$

$$A_{0s} = \text{Clip3}(-c, c, \delta_{0s}), \quad A_{1s} = \text{Clip3}(-c, c, \delta_{1s}), \quad A_{2s} = \text{Clip3}(-c, c, \delta_{2s}) \quad (8.34)$$

其中， δ_{0s} 、 δ_{1s} 、 δ_{2s} 分别按以下3式计算：

$$\delta_{0s} = (p_2 + 2p_1 - 6p_0 + 2q_0 + q_1 + 4) \gg 3 \quad (8.35)$$

$$\delta_{1s} = (p_2 - 3p_1 + p_0 + q_0 + 2) \gg 2$$

$$\delta_{2s} = (2p_3 - 5p_2 + p_1 + p_0 + q_0 + 4) \gg 3$$

Q块边的三个像素 q_0 、 q_1 、 q_2 的修正方法和P块一样，只要将上面诸式中的字母“p”换为字母“q”即可。

3. 色度滤波过程

一般，当 $BS=2$ 时，才需要对像素的色度分量进行滤波，对色度信号的去方块滤波只有一种方式，且只需要紧贴边界的一个像素修正即可。见式 (8.36)，修改后的色度信号为：

$$p'_0 = p_0 + \Delta_c, \quad q'_0 = q_0 + \Delta_c \quad (8.36)$$

$$\Delta_c = \text{Clip3}(-c, c, \delta_c) \quad (8.37)$$

其中 δ_c 的计算公式如下：

$$\delta_c = ((p_0 - q_0) \ll 2 + p_1 - q_1 + 4) \gg 3 \quad (8.38)$$

4.限幅范围

在前面所涉及的限幅函数中，所有的限幅范围都是用字母c表示的，实际上，不同的像素点，不同的滤波强度，滤波时的限幅范围是不同的。

在普通滤波模式中，对于 p_0 和 q_0 ，定义 $c=t_c(n)$ ，对于 p_1 和 q_1 ，定义 $c=t_c(n)/2$ ；在强滤波模式中，对于所有的点，定义 $c=2t_c(n)$ 。

上述定义中的n，当边界两边都是帧间预测编码，则n和当前块的QP相等，即 $n=QP$ ；当两边至少有一个块为帧内预测编码（BS=2）时，则 $n=QP+2$ 。从这里可以看出，限幅范围c实际上是QP和 t_c 之间的关系，如图8.9所示。

5.去方块滤波一例

最后，给出一个HEVC去方块滤波的实际效果图，如图8.11所示。图8.11(a)所示为未采用去方块滤波的编码器输出的重建图像。正如上述编码器方块效应成因分析，图中在DCT变换边界上有明显的边界痕迹，呈现出方块形状。图8.11(b)所示则是采用去方块滤波后的编解码器对同一幅图像的处理效果图，相对于图8.11(a)，则图8.11(b)的方块已不明显了。



(a) 去方块效应滤波前

(b) 去方块效应滤波后

图8.11 去方块效应滤波前后对比

8.5 HEVC的样值自适应补偿

样值自适应补偿（ SAO ）技术是 HEVC的一种新的编码工具，在去方块滤波后实现，也属于环路内部滤波。去方块滤波的操作只在块边界处起作用，而样点自适应补偿则对块中所有的样点值进行处理。SAO的主要思想是对照原始图像，通过对重建图像块的像素特点进行合适的分类，对不同的分类施加不同的补偿值，使其更接近原图像，从而达到减少图像失真的效果。

8.5.1 信号失真及补偿

在信号处理中，跳变信号通过系统输出后产生失真的吉布斯（ Gibbs ）现象可以说明大多数情况下经过视频压缩后的块效应，特别是振铃效应。如图8.12所示，图中没有画出的横坐标表示扫描线方向，纵坐标表示扫描线上样点的灰度值。图中虚线表示原始样点值，实线表示失掉一些高频信息后的重建样点值。由于预测、变换、量化等过程的影响，重建样点存在一些局部峰值、凸拐点、凹拐点以及局部谷点，图中用实心的小圆点表示。而非这一类点则用空心小圆点表示。如果能够对于局部峰点、凸拐点施加适当的负值补偿，对凹拐点、局部谷点施加适当的正值补偿，抵消图像编码失真的影响，将重建样点与原始样点值的差距缩小，就可以在一定程度上减小图像的失真。

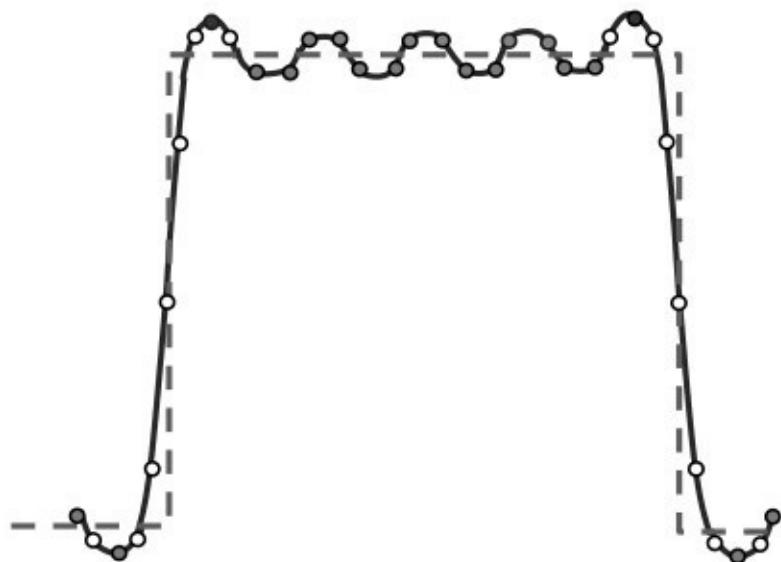


图8.12 Gibbs现象

我们会很自然地想到，这样的补偿值可以来自每个重建图像的像素值与对应原始图像的像素值的差值。这样做虽然很理想，但是要传输每个像素点的差值会增加大量的编码码字，与图像压缩的目标相违背。实际上，我们采用了比较折中的办法：将重建图像的局部特征结构分类，对不同的类赋予不同的补偿值以纠正编码带来的偏差，这样虽不能保证百分之百地覆盖和准确，但可在很大程度上补偿编码的差错，改进重建图像的质量，而且并没有增加编码比特数，甚至还可能有所减少。这就是SAO技术的基本出发点。

8.5.2 SAO的两种模式

由前面的HEVC编码端框图（见图8.4）可以看出，SAO位于编码器环路内，紧接着去方块滤波器后面。SAO是一种自适应选择的处理过程，它的输入信号是原始图像数据和去方块滤波后的重建图像数据，SAO参照原始图像数据，对重建图像数据进行补偿，输出是SAO纠正后的图像数据和SAO的补偿数据。SAO一边将纠正后的图像数据送至图像缓存，为编码环路提供质量更高的重建图像参考，另一边将产生的补偿参数送至熵编码，为解码器的SAO提供同样的补偿操作参考。显然，SAO是一种非线性滤波操作，它能使重建信号更加逼近原始图像，提高图像在平滑区域和边缘区域的观赏效果。

SAO采用的是一种分类补偿的方法，通过对重建图像样点值进行分类，为每一类像素值添加一个补偿值，以达到减小失真的目的，从而也可以提高压缩率，减少码流。从减少复杂度的角度考虑，要求SAO中的类别划分操作简单，因为这个过程需要在编码端对每个像素都要进行判断，在编、解码端都要进行补偿操作。

针对不同的重建图像特点，SAO的滤波类型有两种，边缘补偿（EO）和带补偿（BO）。由于SAO滤波器是根据区域特点实现的，其参数是可按CTU自适应调整的。因此在编码中依据每个CTU的自身特点SAO会选择不同的滤波类型。当然，如果某些CTU像素不满足特定条件，也可以选择不使用SAO滤波器。

对于SAO的BO类型和EO类型，需为每个CTB发送总共4个幅度补偿值。对于BO类型，符号也要编码。补偿值和相关的语法元素，如sao_type_idx和sao_eo_class由编码器决定，典型的方法是使用率失真性能优化准则判断。

大量资料和测试结果显示，SAO平均可以节约2%~6%的码率，而编解码的复杂度只增加了2%左右。从SAO的语法结构上看，由于多了SAO信息的编码和传输，码率应该处于增加的状态。但实际却相反，虽然当前帧的码率传输增加了字节，但却使原图像与重建图像之间的失真缩小，接下来的预测残差会变小，因此反而有可能降低码率。

8.5.3 带补偿（BO）模式

由语法元素sao_type_idx=1定义SAO的带补偿（BO）模式的应用。

1. BO补偿原理

在带补偿模式中，所选的补偿值直接取决于像素的幅度值。编码器把图像像素值范围划分为32个等份，每个小等份称为“带”（band），如图8.13所示。一个像素一般有3个值：亮度Y和色度U、V，处理方法都相同。现以亮度为例，一个8位表示的亮度样点值的大小范围为0~255，所以每个带的范围（宽度）为 $256/32=8$ 个连续的像素值。从小到大，第0个带的灰度范围为0~7，第1个带为8~15，…，第31个带为248~255。一般，第k个带中样点值的取值范围可以表示为 $8k \sim 8k+7$ ，其中k的取值范围为0~31。

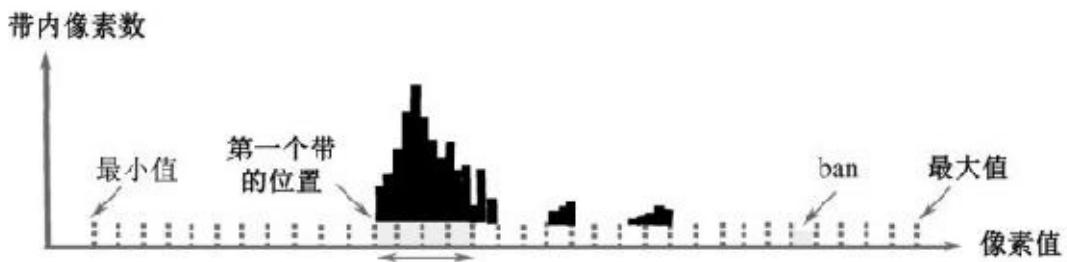


图8.13 BO模式中band的划分和CTB像素分布一例

计算每个带中原始样点和重建样点的平均值之差（即补偿值），在编码端将此补偿值加到每个重建像素上，使得此带中的原始样点和补偿后的重建样点的平均值相等。与此同时，还需将这一补偿值传输至解码端，使得解码器也能够完成同样的补偿。在解码端，也为每一个处于该带内的去方块滤波后的值加上相应的补偿值，这样可以保证在该带出现的重构样点值和原始图像样点值的均值是相等的。

在BO模式中，对补偿值的符号没有明确限制，可正可负。还有，BO模式并非对CTB中32个带都做补偿操作，而是通过某些算法来选择其中连续的4个偏移最大的带进行补偿。

2.BO的补偿过程

可以用图8.13来说明BO的具体操作过程：在编码端，统计当前CTB中的亮度和色度的值，作32个带（band）的直方图，对每一个带中包含的Y、U、V的样值分别求均值。

假设有一个带的灰度是32~39，原始图像在该CTB中有3个亮度像素的值落在这个带中，它们分别为：32、34和36。可知原始图像该CTB中亮度样值出现在这个带中的均值为 $(32+34+36)/3=34$ 。

现在开始对这个带做SAO的BO处理。假设求出去方块滤波后的重构图像在该带的均值为32。可见，在CTB的这个带中，去方块滤波后的重构的亮度均值和原始图像的亮度均值之间的差值为 $34-32=+2$ 。因此SAO可以分配补偿band offset=+2到这个带上。在解码端，为每一个处于该带上的去方块后的像素值加上2，这样可以保证在此CTB的这个带上出现的重构像素的亮度值和原始亮度值的均值是相等的。

对32个带都做这样的处理，最后选择连续4个偏移值最大的带作为最终确定的需要补偿的带。并将起始带值和4个带补偿值写到码流中传输给解码器。通过这种BO方式的处理，可以把均值差别最大的连续4个带补偿成均值相等，来缩小重构图像亮度值和原始图像亮度值之间的差距。

为何BO模式中只是选择连续4个补偿值最大的带，这是因为一般块效应在图像平坦部分比较显眼，在纹理或变化复杂的部分由于人眼视觉的不

敏感而不显眼，因此在图像的平坦部分大部分像素的取值都集中在很少的几个带中，因此使用连续的4个带能够覆盖大部分的像素。也就是说如果对平坦区域的某一个 CTB 做直方图的话，其像素很可能集中在很少的几个取值点，因此使用4个连续带可以做到很好地覆盖。还有一个原因是接下来即将介绍的边缘补偿（EO）模式使用了4个补偿值，为了不增加码率，BO模式也可复用这4个补偿值的语法，这样不需要再增加语法来专门表示EO的补偿了。

图8.14可以解释BO在很多情况下都很适用。图中水平坐标表示样点位置，垂直坐标表示样点的值。虚线表示原始图像在4个连续带中的样值，小圆点表示重建图像的样值，它们可能因预测残差的量化误差和编码运动矢量偏离真实运动而引起误差。在此例中，重建样值偏离到原始样值的左侧，呈现了一个整体的负误差。这个误差可由BO方式对第k、k+1、k+2和k+3带加上一个正补偿来纠正重建样值的误差，其作用相当于将图中重建样点值整体向右移动一点，使它更加接近原始图像。

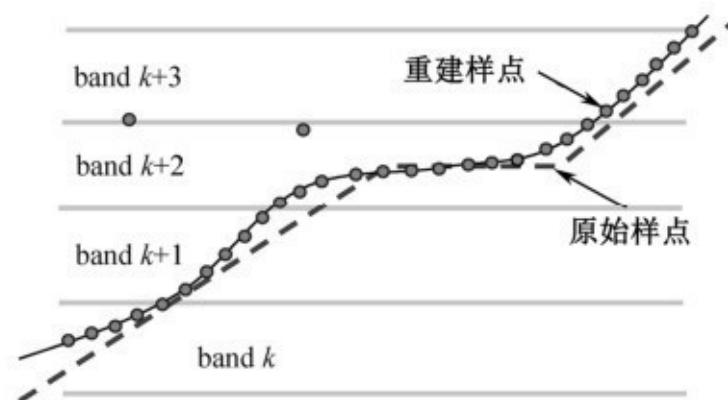


图8.14 BO像素补偿一例

8.5.4 边缘补偿（EO）模式

由语法元素sao_type_idx=2定义SAO的边缘补偿（EO）模式的应用。

1. 梯度的方向种类（class）

边缘补偿（EO）根据边界方向区分像素值，为保证复杂度与编码效率的平衡，EO采用了4种一维方向来确定该样点值所属的EO模式，这4种方向分别为水平、垂直、 135° 角以及 45° 角。由语法元素元素sao_eo_sclass的值为0~3表示这4种梯度方向，用于CTU的边缘补偿分类。图8.15出示了EO模式中sao_eo_sclass分别表示4种不同的梯度方向。

图8.15中，c点为当前样点值，a和b分别代表与之相邻的两个样点值。在编码端，每个CTU只可以选择4种EO模式中的一种，即该EO中所有的像素都为这一模式，除非该CTU判定为不进行 SAO 处理。CTU 到底选择哪种模式，则可采用率失真优化的方法来选择，将所选最优的EO类别写入码流。

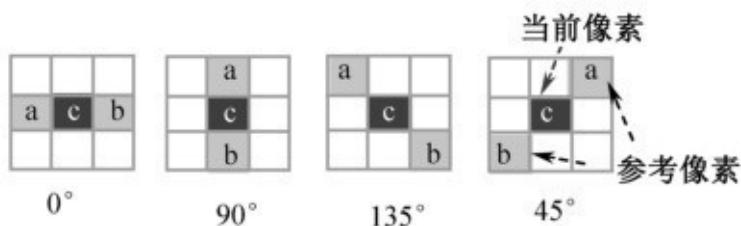


图8.15 EO中的1维梯度模式

2. 梯度的分布类型 (category)

对于一个给定的EO模式，编码器根据a、b、c值的大小关系，即当前c点像素值与周围a、b两个点的参考像素值相比，将CTU中的每个像素分为5种类型 (category) 之一，用edgeIdx值表示，从0~4，如表8.3所示。a、b、c值之间的关系还可以从图8.16中直观地看出。

表8.3 EO中样点分类规则

| 类 型 | 条 件 |
|-----|---|
| 1 | $c < a$ 和 $c < b$ |
| 2 | $(c < a \text{ 和 } c = b)$ 或 $(c = a \text{ 和 } c < b)$ |
| 3 | $(c > a \text{ 和 } c < b)$ 或 $(c = a \text{ 和 } c > b)$ |
| 4 | $c > a$ 和 $c > b$ |
| 0 | 其他 |

- (1) 当前像素值小于两个相邻像素值，该像素为谷值，属于类型1；
- (2) 当前像素值大于两个相邻像素值，该像素为峰值，属于类型4；
- (3) 当前像素值小于一个相邻像素值，等于另一个相邻像素值，该像素值为凹拐点，属于类型2；
- (4) 当前像素值大于一个相邻像素值，等于另一个相邻像素值，该像素值为凸拐点，属于类型3。

如果当前像素点均不符合1~4的情况，则该点不属于任何一个类型，属于平坦区域，也就不需要进行EO操作了。

这种对每个像素的分类是基于解码像素值进行判断得到的，因此，不需要为类型 edgeIdx 设置附加的标志，也就是说不需要外加分类标志比特。编码器根据像素的 edgeIdx 类型不同，自动在查找表选择补偿值加到这个像素值上。

3. 边缘补偿方式

从吉布斯现象出发，EO 认为像素值原来是平坦区，因为量化丢弃了部分高频分量，形成了某些像素值的偏移，使得原来相对平坦的像素值关系或多或少地形成局部峰点、谷点或拐点。那么，EO的补偿就是要人为地对这些漂移的像素值加上一个适当的相反的补偿量，以抵消编码造成的偏移。显然，对以第1类谷点和第2类凹拐点像素值，需要加上一个正补偿量，将该像素值“拉高”，和相邻像素平齐；对以第4类峰点和第3类凸拐点像素值，需要加上一个负补偿量，将该像素值“压低”，和相邻像素值平齐。EO对上述4类相邻像素值的分布情况如图8.16所示，重建图像的像素值的补偿趋势如图中的箭头所示。

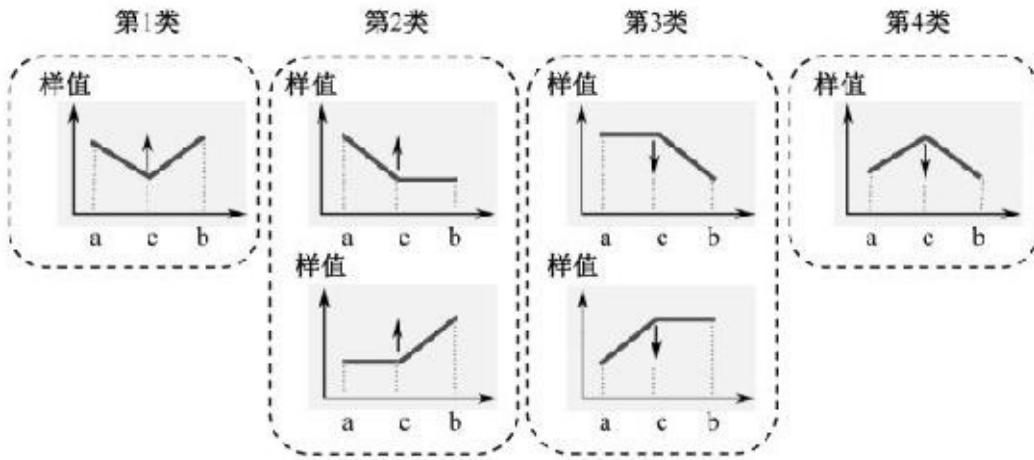


图8.16 EO模式中各类像素补偿的方向

4.EO信息的传送

在HEVC编码端，EO以CTU为单位进行的，CTU中每个像素都有4种梯度方向，通过率失真优化（RDO）选择一种EO种类，作为边信息（SI）加入到码流中。种类确定以后，检查邻近3个像素的值，判定该像素属于5种梯度值分布的哪一种类型，用语法元素类型edgeIdx来表示。

对于edgeIdx为0的图像平坦区域，可以不做任何操作。对于edgeIdx为1~4，SAO为每一个edgeIdx分配了一个补偿值，这个补偿值会加到重构像素上。实际上，SAO不是为每一个像素分配一个补偿值，而是先确定像素位置结构的方向模式，再作edgeIdx的分类，对每一个edgeIdx类分配一个补偿值（对每一个CTU有4个补偿值就足够），这样可以减少码率的消耗。

按理说，EO处理后有3种信息需传送至解码器，即反映像素梯度方向的种类，反映像素值分布的类型和具体的补偿量。但我们知道，在HEVC

中，SAO的EO模式的计算是针对去方块滤波后的重建图像进行的，编码器和解码器使用相同的方法来计算，因此不需要传输类型信息给解码器，而是由解码器自己来计算，这样虽然增加了计算量，但是可以降低码率。另外补偿量只要传送其绝对值就可以了，不需要传送它的正负号，因为解码器根据像素方向模式可以自动判别出来，在 $\text{edgeIdx}=1、2$ 这两类情况下，补偿量值必须是正数；对于 $\text{edgeIdx}=3、4$ 这两类，补偿量必须是负数。

8.5.5 SAO的模式选择和参数共享

1.SAO的模式选择

在SAO处理中，EO模式主要用于对图形的轮廓进行补偿，其分类依据是当前样点值与相邻位置样点值的差距；而对于 BO 模式来说，样点值的分类是基于样点值的大小来确定的。需要注意的是，不同颜色分量都有自己的SAO参数。为了达到低编码复杂度，减小缓存需要的目的，区域划分的尺寸设置为一个CTU大小，同时为了减小额外的信息量，多个CTU可以共享相同的SAO参数。

根据不同位置CTU图像内容的特点，编码器的SAO对不同CTU进行局部特征信息的补偿。每个区域被标记为BO 或 EO 类型，这种局部自适应方法可以在高效编码的同时快速得到局部补偿值。

如图8.17所示，虚线表示CTU边界，对于每一个CTU都有三种SAO模式可供选择：EO、BO和OFF。从图中可以看出相邻的CTU通常选择相同的补偿模式，实线表示同一模式的区域（模式合并），这是因为相邻区域的图像属性通常比较相似。图8.17中“OFF”表示当前区域既没有选取EO，也没有选取BO。至于如何为CTU选择合适的模式，HEVC并没有规定，因为这不涉及标准兼容性问题，对开发者是开放的。

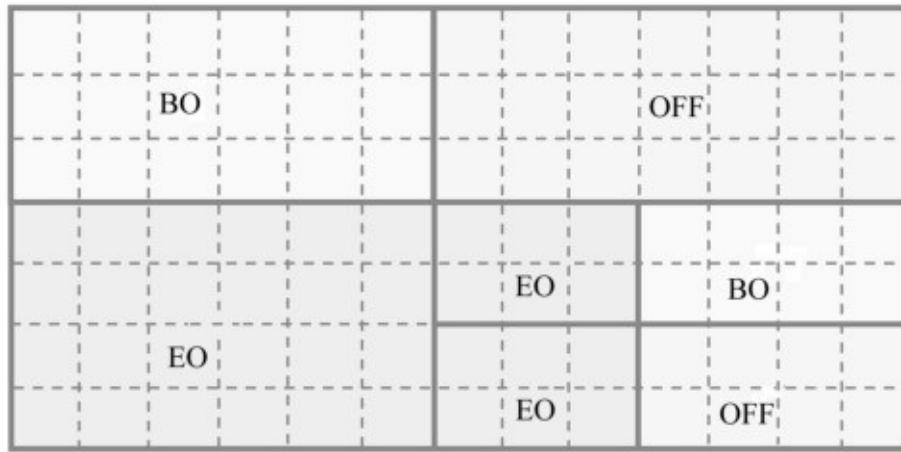


图8.17 选取不同SAO模式的CTU

由于SAO的处理位于去方块滤波后，所以除滤波后的图像数据外，还可以将去方块滤波过程中得到的统计信息也保留下来，供SAO利用这些信息对参数进行判定，如失真估计、率失真优化等，可避免不必要的重复计算。

2.SAO的参数共享

在SAO模式（包括参数）共享中，对于每一个当前的CTU来说都有三种可能的选择：共享已编码的左邻块CTU的SAO参数，共享已编码的上邻块CTU的SAO参数，计算新的SAO参数。选取上邻块的 SAO 参数通过 sao_merge_up_flag 标识，左邻块通过 sao_merge_left_flag 标识，如图8.18所示，一个CTU信息包含了相应的亮度CTB信息、Cb色度CTB信息和Cr色度CTB信息，一旦当前CTU选择了使用与左邻块或者上邻块相同SAO信息，则该CTU的亮度、色度的SAO参数只需要复制上邻块或左邻块的相应参数

即可，不再需要发送其他任何信息，以便有效减少边信息（IS）。

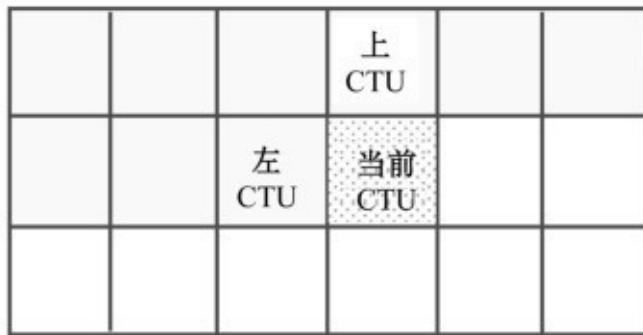


图8.18 SAO的CTU共享参考

如果当前CTU的SAO模式与相邻块皆不同，则对当前CTU编码、传输自己的参数，首先发送是亮度分量的语法元素，其次是色度分量，其过程如图8.19所示。

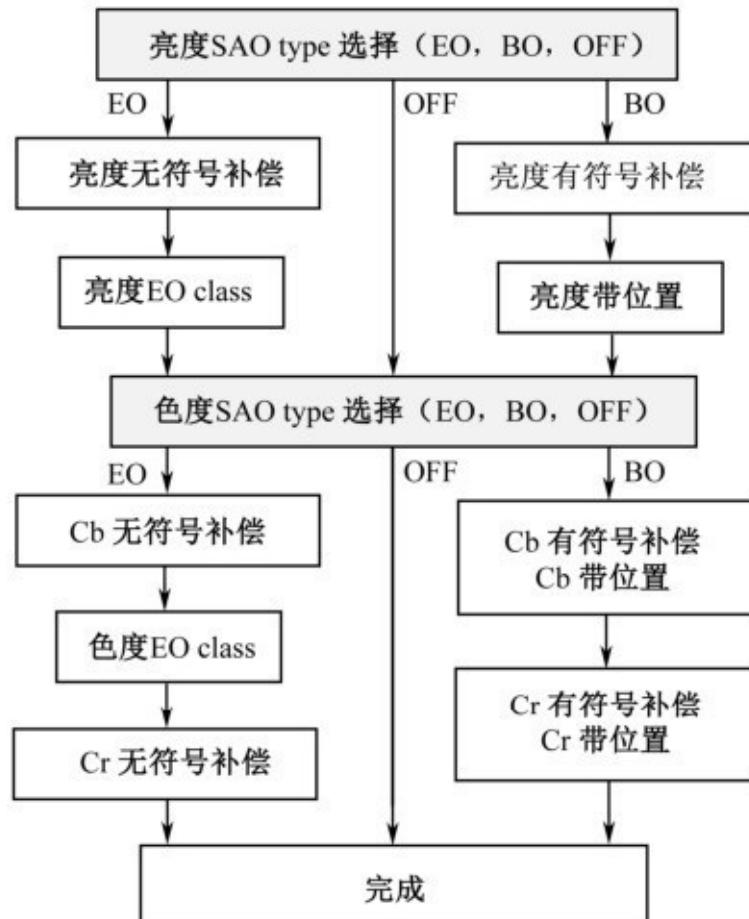


图8.19 CTU的SAO参数选择流程

SAO的类型由sao_type_ edx_luma和sao_type_edx_chroma标识，说明当前CTU选取EO模式、BO模式或者OFF（不适用SAO补偿）。如果当前CTU以BO模式作为补偿方式，则需传输补偿值所在起始带的位置标识sao_band_position；如果以EO作为补偿方式，则需传输EO所属种类的标识

sao_eo_class_luma或sao_eo_class_chroma。无论是BO还是EO，在这些标识以后需发送的就是4个补偿量。需要注意，Cb和Cr共享SAO类型信息（sao_type_idx_chroma）和EO种类信息（sao_eo_class_chroma），可进一步减少传输的边信息。

3.SAO的性能

SAO工具的主要作用就是纠正由于大尺寸的变换、量化引起振铃效应，补偿在某些区域因运动补偿的误差引起的像素值偏移。

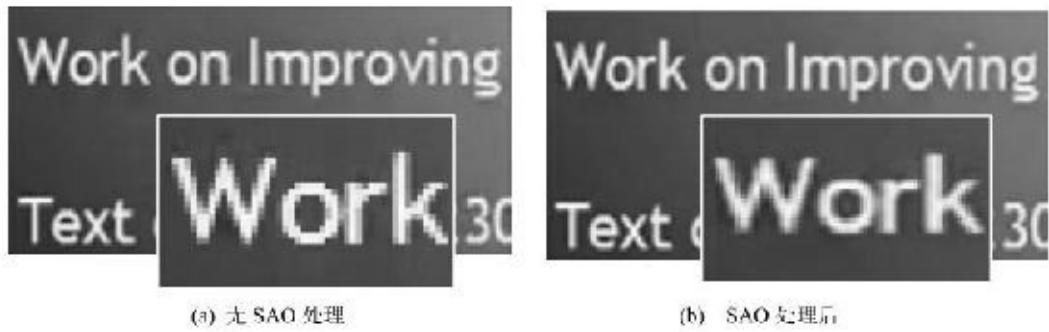


图8.20 测试序列SlideEditing某一帧的SAO处理对比

大量测试序列亮度信号的SAO实验结果说明，在CTU的尺寸为 64×64 情况下，平均编码增益大致在5%。最好的情况，如对低时延（无双向预

测) 标准测试序列BQTerrace的编码 , SAO的编码增益高达18.9%。SAO在性能提升方面还具有3个特点 : 对包含计算机图形和屏幕内容的图像更加有效 , 胜过对自然图像的质量提升 ; 在低时延单向帧间预测编码的情况下 , SAO可提供比较高的编码增益 ; SAO的计算复杂度比较低 , 平均增加的解码时间小于2% ~ 3%。图8.20所示为SAO处理的一个示例 , 抑制了邻近边缘的振铃效应 , 使文字显得更加清楚。

本章参考文献

- [1]Chih-Ming Fu,Elena Alshina,Alexander Alshin,et al.Sample adaptive offset in the HEVC standard[J].IEEE Trans.on Circuits and Systems for Video Technology,2012,22 (12) :1755-1764.
- [2]Andrey Norkin,Gisle Bjøntegaard,Arild Fuldseth,et al.HEVC deblocking filter[J].IEEE Trans.on Circuits and Systems for Video Technology,2012,22 (12) : 1746-1754.
- [3]Vivienne Sze,Madhukar Budagavi,Gary J.Sullivan.High Efficiency Video Coding (HEVC) :Algorithms and Architectures[M].Switzerland Springer International Publishing,2014.
- [4]Chia-Yang Tsai,Ching-Yeh Chen,Tomoo Yamakage,et al.Adaptive loop filtering for video coding[J].IEEE Journal of Selected Topics in Signal Processing,2013,7 (6) : 934-945.
- [5]Wei Cheng,Yibo Fan,YanHeng Lu,et al.A high-throughput HEVC deblocking filter VLSI architecture for 8Kx4K application[C].IEEE International Symposium on Circuits and Systems (ISCAS 2015) ,605-608.
- [6]High Efficiency Video Coding[S],ITU-T Rec.H.265 and ISO/IEC 23008-2,ITU-T and ISO/IEC JCT-VC,Mach 2013 (v.1) ,Oct.2014 (v.2) ,Apr.2015 (v.3) .
- [7]Mathias Wien.High Efficiency Video Coding: Coding Tools and Specification[M].Springer-Verlag Berlin Heidelberg,2015.
- [8]Kang Runlong,Zhou Wei,Huang Xiaodong ,et al.An efficient deblocking filter algorithm for HEVC[C].IEEE China Summit & International Conference on Signal and Information Processing (China SIP 2014) ,379-383.

[9]Qin Yu,Ying Chen,Li Zhang,et al.Sample edge offset compensation for HEVC based 3D video coding[C].IEEE International Conference on Multimedia and Expo Workshops (ICMEW 2014) ,1-6.

[10]E.Alshina,A.Alshin,J.H.Park.Encoder modification for SAO.Doc.JCTVC-J0044.10th Meeting: Joint Collaborative Team on Video Coding (JCT-VC) of ITU-T SG 16 WP 3 and ISO/IEC JTC 1/SC29/WG 11,Stockholm,Sweden (2012) .

[11]JCT-VC.HEVC Reference Software.http://hevc.hhi.fraunhofer.de/svn/svn_HEVCSoftware/ (2014) .

[12]G.J.Sullivan et al,Overview of the High Efficiency Video Coding (HEVC) standard[J].IEEE Transactions on Circuits and Systems for Video Technology,2012,22 (12) : 1649-1668.

第9章 HEVC的并行处理

并行处理机制是现代数字信号处理中应对日益增长的海量信息的重要方法之一，因此在HEVC 开发全过程中，都贯穿了必须充分利用并行处理结构的理念。HEVC 所提供的多项新的编码结构和编码工具都体现了它对并行处理实现的“友好”，对每个设计元素都要检查它的潜在的串行瓶颈，并予以尽可能的避免。本章先介绍了视频编码中并行处理的几种主要方式，在此基础上介绍HEVC的几个并行处理新工具，最后对HEVC的各级并行处理方式进行了简要分析。

9.1 视频编码的并行处理

与以往的视频编码标准相比，HEVC面临着计算复杂性迅速增加的问题，直接影响了它的运行和实现。其主要原因之一是高清视频的应用，如今常见的1080p视频约为200万像素/帧，而以往的CIF视频约10万像素/帧，数据量增加了几十倍。这里尚未考虑数据量更大的3D视频、多视点视频以及超高清清晰度视频等。对于单处理器系统，提高对视频数据的处理速度意味着需要更高的时钟频率，更高的功率消耗。但时钟频率的增加和功耗的提升是有限度的，对于目前许多感兴趣的应用领域，这些指标几乎已到达极限阶段。

提高处理速度，增强计算能力的有效办法之一就是并行化处理。视频系统中的并行化处理并非是一个新的概念，以往的标准在这方面已进行过多种尝试。例如，在先前的MPEG-2视频编码中，分割图像帧为多个 slices，为每个 slice 分配多核处理器中的一个核来实现一幅图像的并行处理。再如，H.264/AVC视频编码器和解码器以独立的方式并行处理视频图像的不同部分，如条（slice），实现了1080p（ 1920×1080 ）每秒60帧的实时视频传输。

以下从并行处理的基本方式出发，介绍视频编码中几种常见的并行处理方式，如功能并行、数据并行、流水线并行等。

9.1.1 并行处理的主要方式

1. 视频信号的并行处理

什么是信号处理中的并行处理？一般来说，给定一个信号处理问题，并行处理就是先把这个问题分解成若干子问题（任务），同时处理（并行

处理)这些子问题,最后把各个子问题处理的结果合并得到原信号处理问题所需要的结果。

视频编码也是一个信号处理问题,就是将原始的数字视频信号处理成一种简洁的数字信号表达方式,要求处理前后信号所包含的信息量保持不变或变化很小。根据并行处理的要求,把每一个数据量非常大的视频编码问题分解多个子问题是不容易的,这是由于子问题之间可能存在或多或少的数据相关性。由于数据相互关联,程序(或处理器)之间必然相互依存,必须相互通信,通常两个处理器之间的通信时间与处理时间相比是很高的。因此,视频数据的并行处理面临着两个重要的挑战。

(1)合理的子问题划分,这是并行处理中遇到的最大问题。在视频编码中就是如何尽量减少子问题之间的数据相关性,将编码问题分解为多个能够并行执行的子问题。

(2)合理的子问题处理的调度,尽量降低子问题之间通信的开销。

2.并行处理的性能

一个串行算法的优劣通常按其执行时间来评估。一个并行算法的优劣除考虑执行时间外,还有更多的因素需要顾及。并行算法的执行时间不仅依赖于输入数据量大小,还依赖于所用处理核(线程)的数目,及它们相关的计算速度和处理核(线程)间的通信速度。对于一个给定的问题,并行算法的性能主要通过下列三个因素评价。

(1)并行处理所需要的时间(时间复杂度);

(2)并行处理所需要处理器的个数和能力(处理器复杂度);

(3)加速比,或并行度,即同时在工作的处理器的个数。

3.常见的并行处理方式

在视频编码的并行处理中,根据子问题划分方法的不同,常见的有三类并行处理方法。

(1)功能并行处理:将总问题划分为若干不同的功能,每个处理器(线程)完成一个功能,所有处理器(线程)同时工作。

(2)数据并行处理:将总问题的数据分解为若干部分,每个处理器

(线程)处理一部分数据，所有的处理器(线程)同时工作。

(3)流水线并行处理：是一种准并行处理技术，将数量众多的任务的每一个都同样划分为一些列子任务，每个处理器(线程)处理所有任务的某个子任务，所有处理器(线程)几乎在同时工作。

9.1.2 功能并行

功能并行方法是实现视频编解码并行执行的一种最为简单的方法。根据这种方法，我们需要先对视频编码系统的功能进行划分，划分的原则是使这些功能要尽可能独立，相互之间的依赖关系尽可能少。然后将这些功能分别分配给不同的线程或不同的处理核执行，实现并行处理。在实际运行的过程中，需要对多个功能模块之间进行有效的任务调度，避免模块之间的需求或控制发生冲突或死锁。

下面以视频编码为例来说明的功能并行处理的方法。参考第3章的图3.5，对它进行简化后表示为一系列的编码功能模块，如图9.1所示，其中灰色模块表示HEVC的主要功能。从图9.1的编码功能框图中可以看出，视频序列中的每一帧都需要经过几个不同功能模块的处理，主要包括帧间/帧内预测、运动估计和补偿、变换、量化、熵编码、去方块滤波、样值自适应补偿、反量化、反变换等功能模块。为了使各个功能可并行执行，需为每一个功能独立创建一个线程或分配一个处理核，然后数据依次通过每一个线程(或处理核)，这也是一种流水线方式，这样就会有多个线程(或处理核)同时在运行，每个线程都在处理一帧的全部数据某一功能。

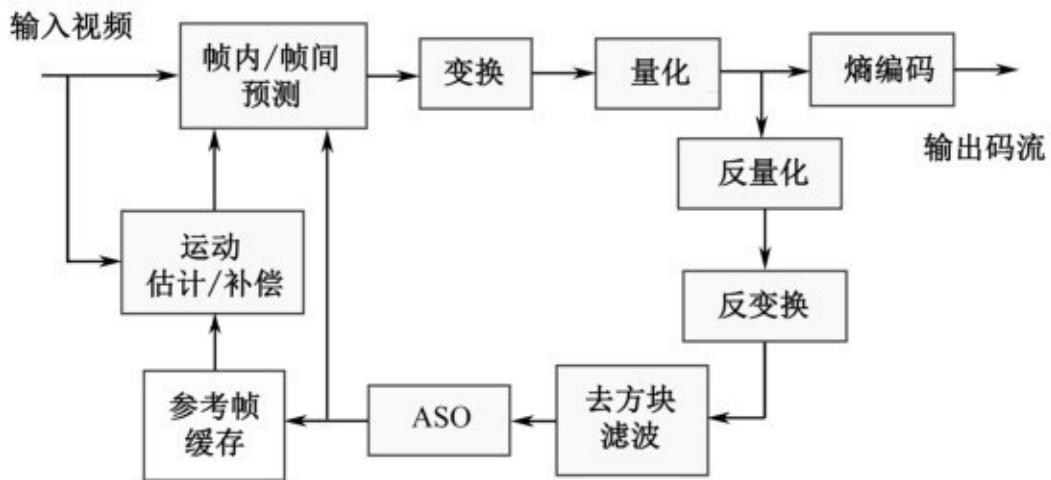


图9.1 HEVC编码功能图

采用功能级并行的方式加速典型的视频编码任务，包括HEVC编码算法，效果并不理想。其原因大致如下。

(1) 混合编码过程的各个功能比较固定，这使得任务划分方式也被固定了，这样决定并行性能的所能创建的最大线程数或最大的多核数也基本固定了。

(2) 完成各个功能所耗费的时间相差很大，如运动补偿耗费的时间就远远大于整数变换耗费的时间，这导致进行整数变换的线程（或处理核）要长时间地等待计算运动补偿的线程传数据过来。

(3) 由于采用的是多线程或多核方式并行处理各个功能，这导致功能处理模块之间的数据和控制信息的交互和同步尤为频繁，开销较大。

所以，采用任务级并行对HEVC进行加速的方法一般情况下并不可行。

9.1.3 数据并行

与根据任务的天然特性进行功能划分的并行处理方式不同，数据并行处理实行的一种“化大为小”的策略，即合理地将被处理的较大的数据划分为若干个较小的数据部分，使得每个线程或处理核独立承担一个小数据部分处理的任务，从而形成并行处理的方式。显然，采用数据并行方式使得能处理的计算规模可以随着线程（或处理核）数量的增加而不断增大，在相同的时间内能够完成更多的工作量。

视频编码的数据并行概念相对简单，将一帧图像划分为若干部分，如若干片（tile）或若干条（slice），每个部分由一个线程（或处理核）程承担。例如，将一幅图像数据划分为4个部分（slice）进行数据并行处理如图9.2所示。

可以看出，在数据并行的情况下，每个线程只处理图像帧中的一部分数据，但是要完成这一部分数据的所有编码功能。对比上述的功能并行方式，每个线程只完成一个功能，但要处理图像帧中所有的数据。因此，在数据并行中，由于数据划分的自由度远远大于功能划分的自由度，而且数据划分后的每个部分的数据量和数据特性大体相当，再加上每个线程（或处理核）的处理过程和处理负担大体一样，所以，数据并行的方式的处理速度和编码效率都比功能并行方式高，在视频编码中得到了广泛的应用。如基于GOP、基于图像、基于slice或tile等并行处理都属于数据并行的方式，HEVC中新引入的基于CTB行的波前并行处理（WPP）也属于数据并行机制。

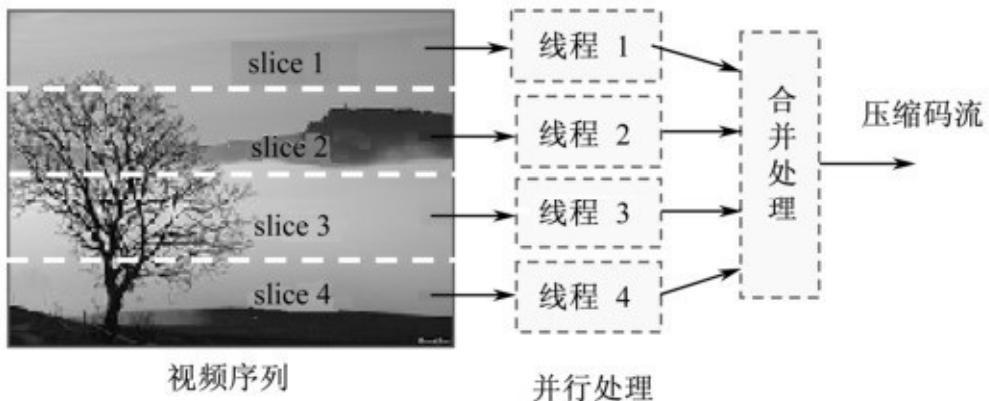


图9.2 Slice级并行处理示例

当然数据并行机制最大的不足之处在于：由于人为地将相互关联的图像数据划分为若干相互独立的部分处理，丢失了一部分图像数据之间的相关性，会给编码效率带来一些不利的影响。为克服数据并行处理的这一不利因素，人们采取了多种弥补措施。例如，在WPP方式中，采取的逐行延迟处理就是一种充分利用CTU行间的数据相关性的措施。

9.1.4 流水线并行

1. 流水线处理方法

在信号处理中，模仿工业生产装配流水线的工作方式，流水线信号处理是一种准并行处理技术，它把一个数据处理过程分解为若干个子过程，每个子过程由一个专门的功能部件来实现，每个子过程可以与其他子过程同时进行。

如一项信号处理任务必须顺序进行a、b、c、d 共4个步骤处理，每个步骤需1s，则完成一项任务需要4s，如不停息地顺序（串行）处理同样的n项任务就需要4ns。如果采用流水线方式工作，当第一项任务完成a，进入b处理阶段时，第二项任务就可开始进入a处理阶段，下面的任务也如此进行，直到所有的n项任务完成为止。这就是一个典型的流水线工作方式，所需要的时间只比n s多一点。

为了方便对这样的流水线处理进行分析，我们对上述的实例采用时空图来描述，从时间和空间两个方面描述流水线的工作过程，如图9.3所示，其中纵坐标为空间（即各个处理步骤，又称为“流水级”），即流水线的各个子过程；横坐标为时间（流水节拍），即各个任务在流水线中所经过的时间。方格中的数字说明在特定的时间、空间的任务号。从图中可以看出，采用流水线工作方式，完成n项任务仅需要 $(n+3)$ s，和串行处理4ns比较，大大缩短了处理时间。

最常见的计算机指令级流水线处理和这里介绍的流水线处理是有所不同的，在本章的最后将简要介绍。

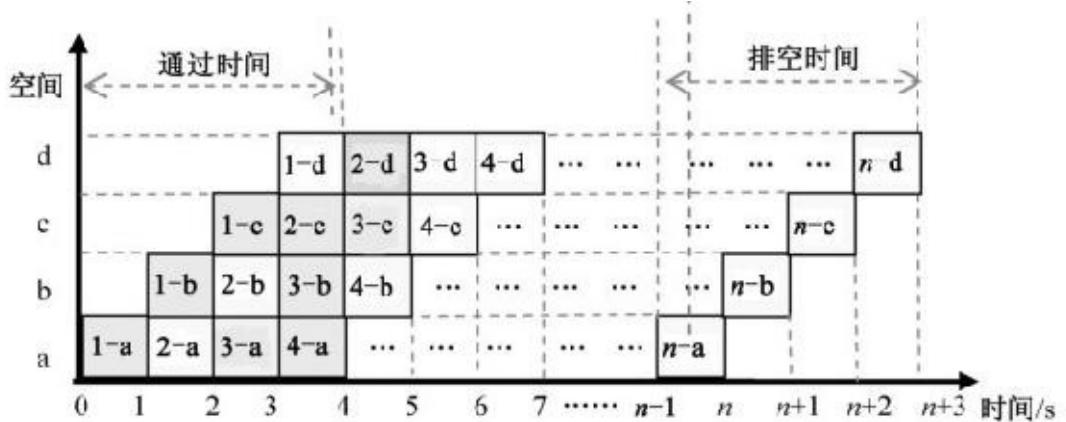


图9.3 流水线处理时空图示意

2. 流水线处理的特点

流水线处理实际上把一个任务分成了多个子任务，每个子任务由一个子功能部件完成，如上面的例子中，a、b、c、d四项任务分别由4个部件完成，依靠多个子功能部件并行工作来缩短整个任务的执行时间，具有以下特点。

(1) 流水线处理有助于提高所有任务（整个程序）的吞吐率，但并没有减少每个任务（指令）的执行时间。

(2) 流水线处理的各个功能段所需时间应尽量相等。否则，时间长的功能段将成为流水线的“瓶颈”，会造成流水线的“阻塞”。

(3) 如图9.3所示流水线处理开始需要“通过时间”和最后需要“排空时间”。

(4) 流水技术适合于大量重复的处理任务，只有在输入端不断地提供任务，才能充分发挥流水线的效率。

3.流水线处理的性能

流水线并行处理技术的性能可从以下3个指标来衡量。

- (1) 吞吐率，单位时间内流水线所完成的任务数或输出结果的数量。
- (2) 加速比，完成一批任务，使用非流水线执行时间与使用流水线执行时间之比。
- (3) 效率，指流水线的部件的利用率。从时空图看，就是n个任务占用的时空区（时空图9.3中着色的面积）和m个段总的时空区（时空图9.3中4行乘以n+3列的面积）之比。

9.2 HEVC的并行处理工具

由并行处理的介绍可知，并行处理能够保证很多系统中视频编解码的实时工作，这些系统在非并行方式执行时是不能支持编解码器的实时方式工作的。支持并行处理的两大技术基础就是硬件的多核处理器技术和软件的多线程技术。HEVC引入的并行处理新工具，如tile和slice的划分、波前处理等，都是为了适应这两项技术发展的需要而作出的努力。因此，在具体说明HEVC 的并行工具前，先简单地说明在硬件的多核技术和软件领域的多线程技术，它们都是视频并行处理的技术基础。

（1）硬件的多核技术

在硬件领域，众多的信号处理器厂家，在十多年前（甚至更早），当他们试图通过提升时钟速率来提升CPU处理能力变得越发困难时，开始将目光转向多处理器和多核处理器，形成了处理器领域的一场多处理器和多核处理器的革命。多处理器一般指一个系统中包含多个高性能的处理芯片，它们以并行方式工作，常常是比较大型的处理系统，不在本书的关注范围内。本书关注的是目前使用最为普遍的多核处理器，即一个处理器内包含多个处理核单元，可以工作在并行状态。

（2）软件的多线程技术

在软件领域中，“多线程”是指一种编程模式，在这种模式里，计算或处理工作被指定到多个独立的程序单元中进行，这些单元就称为“线程”，它们共享某些资源，特别是存储器资源。这些线程能够工作在时分方式或并行方式，这取决于实现和底层硬件。为了保证共享资源的合理使用和满足程序执行的时序限制，线程之间一般需要有通信功能，保证必要的数据交互和操作同步。例如可以通过共享存储器来实现线程间的通信，通过时钟、信号灯、障碍、条件变量等实现线程的控制和同步。

多线程技术可以工作在单核处理器上，也可以工作在多核处理器上。多线程取得的性能增益用“加速”来衡量，加速倍数就等于程序代码在单个处理单元中的执行时间除以程序代码在p个并行处理单元中的执行时间。这样，使用p个处理单元的典型的最大加速因子为p。由于实际的加速是受多种因素影响的，如线程之间的同步、存储器的存取或存储器的性能等因素的影响，一般达不到理想的水平，甚至相差不少。

9.2.1 片并行处理

HEVC 在条（slice）的基础上又引入片（tile）的概念。如第3章所述，tile 被定义为由图像中垂直和水平边界的交点所分割得到的矩形区域。在同一幅图像中，slice和tile可以单独使用，也可以同时使用，且两者的分割互不干扰。一个slice里可以有若干个tile，一个Tile里也可以有若干个slice。一帧图像中的各个tile在编码时共享头信息，每个tile都是独立可解码的。在支持tile并行处理的同时，HEVC同样支持slice的并行处理。一般的基于tile的并行处理实际上就是简单地对每个tile同时、独立地处理，即数据并行处理方式。

1. 标志tile的语法元素

在图像参数集（PPS）中，语法元素tiles_enable_flag标明tile的使用。tiles的数目和它们界线的位置可以为整个序列定义，或者逐帧改变。这个可在PPS中通过某些信令参数来取得，如 num_tile_columns_minus1、num_tile_rows_minus1、uniform_spacing_flag、column_width_minus1和 row_height_minus1等。利用这些语法元素，就可以定义tiles的数目以及图像到tiles的划分。由于tile信令包含在PPS内，允许tile结构在每帧图像之间改变，有利于编码器根据不同的图像内容采用不同的tiles结构，以控制用于编码器多核之间的负载平衡。例如，一幅图像的某一区域需要更多的处理资源，这样的区域就可以划分为比其他区域更多的 tiles，其他区域可能需要较少的编码资源。但所需的编码资源的管理需要在实际的编码处理之

前确定，以此决定tile的结构。

对一幅图像中每个slice分割（SS）和tile，至少需满足下列限制中的一个：一个slice分割中的所有CTUs属于同一tile，或者一个tile中的所有CTUs属于同一个SS。注意，作为这些限制的结果，起始点和tile的起始点不一致的slice或SS，它们的不能覆盖多个tiles。

2.tile的边界处理

类似于slice的边界，tiles的边界打破了解析和预测的依赖性，使得tile能够以独立的方式被处理，但是环路滤波器（去方块滤波和SAO）为了更好地消除边界效应，还是允许穿越边界。这项功能由PPS中的loop_filter_across_tiles_enabled_flag语法元素来控制。如果一幅图像的tiles在不同的包中传输，一幅图像中的一个tile能够独立于其他tiles而被独立处理，这也非常适合于误码多发传输环境。

tiles改变了常规的CTUs的扫描次序，从基于图像的光栅扫描顺序改变为基于tile的光栅扫描顺序。如第3章中所述，在一帧中，所有tiles按照光栅扫描的顺序处理；在一个tile中，所有CTUs也按照光栅扫描的顺序处理。为了利用tile边界处的上下文信息，当处理从一个CTU行移动到另一个CTU行时，可以通过为每个在tile边界处的tile存储和再加载所有CABAC上下文变量的值来获得并行信息。

Tile不需要在CTU层熵解码和重建之间的通信，但如果在环路滤波中操作在穿越tile边界滤波的情况下，通信是需要的。如果为了避免处理器之间的数据交换，可以关闭穿越 tile 边界的滤波模式，但在tile边界处就会产生较大的缺陷，需要慎重对待。相比slice，tile通常编码效率更高，因为在 tiles 中减少了 CTU 之间的空间距离，导致在一个 tile 中样点之间的潜在的空间相关性较高。进一步，tile 的使用可以减少 slice 的头信息负担，至少在每个 tile 不一定使用一个 slice 的地方是如此。但是，类似于 slice，编码效率的损失一般随着 tile 的数目而增加，这是由于沿着 tile 边界的依赖性被打破，以及在每个 tile 的起始所有的 CABAC 上下文变量的重初始化。

9.2.2 波前并行处理

考虑到高清、超高清视频编码的巨大运算量，HEVC提供了基于条和基于片的便于并行编码和解码处理的机制。然而，这样又会引起编码性能的降低，因为这些条和片是独立预测的，打破了穿越边界的预测相关性，每个条或片的用于熵编码的统计必须从头开始。如前所述，为了避免这个问题，HEVC 提出了一种波前并行处理（WPP）的熵编码技术，是一种将图像划分为CTU行、按照波前调度原理进行的一种的并行处理方法，对预测编码和熵编码而言，在熵编码时不需要打破CTUs之间的预测的依赖性，能尽可能多地利用上下文信息。

1.WPP方法

波前并行处理（WPP）提供了一种在条中的更细粒度的并行机。由PPS中的语法元素entropy_coding_sync_enabled_flag 来开启或关闭 WPP 功能。当 WPP 开启时，一幅图像被分成若干个CTU行，每个CTU行构成了可并行处理的一部分，如图9.4所示。第一行以正常方式处理，第二行在第一行处理完前2个CTU后开始处理，第三行在第二行处理完前2个CTU后开始处理，以此类推，每一行中熵编码上下文模式相对前一行都有2个CTU的处理延迟，其过程如图9.4中箭头所示。

之所以需要比上一行延迟两个 CTU，是因为帧内预测和运动矢量预测需要依赖当前 CTU上边和右上边的 CTU 的数据。WPP 熵编码初始化的参数可来自当前行上边的两个完全编码的 CTU 获得的信息，这样就可在新的编码线程中使用尽可能多的上下文信息。因此，WPP 常可提供比条或片更好的压缩性能，而且可避免一些因采用片结构引入的块效应。

在WPP处理过程中，在任何时刻，较当前CTU行正在处理的CTU，先前的CTU行必须已经处理完多于两个连续的CTUs，这就形成了CTUs的“波前”，从图像的右上角波动到左下角，如图9.4所示。这就是波前处理为保持CTU之间的依赖性而不允许所有的CTU行处理同时开始编码的原因。相应地，行方式CTU处理也不能在相同的时刻完成每一行的解码。这一因素

就影响了并行机制的效率，称为斜面低效，它随着使用的线程增加而变得更加显著。

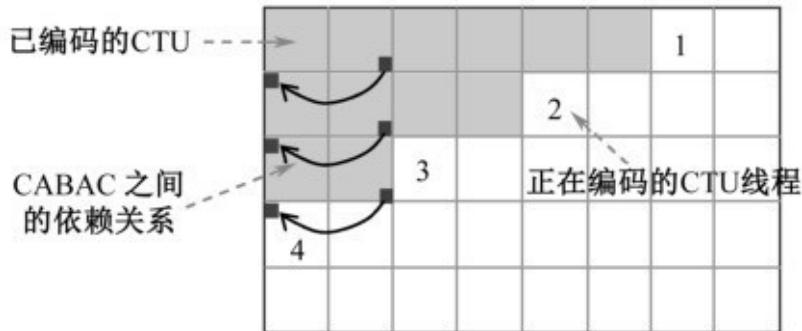


图9.4 4个线程的WPP处理示意图

在WPP方式中的CABAC上下文变量，除在每个CTU行的末尾外，相继的CTU行之间的依赖性在分界线处并未被打断。相比使用其他相同设置的图像编码，由WPP引入的效率损失是相对小的。此外，WPP并不改变CTUs的普通光栅扫描顺序。

2.WPP的多线程实现

WPP常采用多线程方式实现，一个线程负责一行CTU。高达一幅图像

中CTUs行数的线程数都是可以并行工作，同时处理各个CTUs行，这里，CTUs的行数取决于图像亮度样点的高度和亮度CTBs尺寸高度的比例。

在WPP处理中，一个线程负责一行CTU数据的熵编码，会出现线程频繁地创建和销毁。为减少创建和销毁线程的次数，可采用线程池技术，完成任务的线程并不销毁，而是暂时放入线程池，新任务来了可由线程池管理迅速从线程池调用。利用线程池技术可以使每个工作线程都可以被重复利用，可执行多个任务。每一CTU行中，解码CTU的顺序为从左向右，第N行CTU进行解码的触发条件为第N-1行已经解码完2个CTU。然后该行CTU便进入任务队列，等待工作线程对其解码。由于数据依赖性的特点，线程解码第N行第M个CTU的触发条件为第N-1行第M+2个CTU已经解码完毕。线程解码完一行CTU后，即转入线程池，等待任务队列中的其他行CTU进行解码。

3.WPP和tile/slice的比较

WPP一般适于编码和解码的并行机制，因为它允许图像划分较多的部分，带来的编码效率的损失相对较低。此外，与slice及tile比较，WPP不需要环路滤波的附加通道。WPP还可以用于低时延应用，特别是那些需要低延时或实时视频交互的应用场合。

tile也非常适合编码器和解码器的并行机制。像WPP那样，并行机制的并行量是不固定的，允许编码器根据自己的计算资源来调节tile的数量。因为tile能够用于图像划分出的多个水平和垂直覆盖的矩形，因此他们也能够比slice更好地适应感兴趣区间（ROI）编码。在一般的应用中，例如，视频目标跟踪算法的tiles能够被用于动态调节ROI的尺寸和差错保护。tile还能很好地用于不同硬件系统的编码任务，因为，假设穿越tile界线的滤波可分开完成，每个tile的处理相对独立于图像中其他tile，因此在这种情况下，需要在处理各个tile的线程之间的通信量很小。

为简化标准的实现，HEVC不允许在相同压缩视频序列中同时使用tile和WPP。然而，据估计这些工具的某些组合在将来有可能被允许使用。例如，为了保证实时编解码，使用tile 方式划分一个超高分辨率视频

(UHDV) 信号为若干子图像，在每个子图像内部使用WPP。

9.3 HEVC的各级并行处理

9.3.1 GOP级并行处理

视频是由连续的图像帧组成的，在H.264/AVC标准中，这些连续的图像可以分成若干个图像组（GOP）。一个GOP一般包含1~16帧图像，也是H.264/AVC的基本存储单位。在一个GOP中，第一帧图像称IDR帧（立即刷新帧），IDR帧的出现会先将前面GOP中尚未编码完的B帧进行编码，清空B帧队列和参考帧队列，然后产生新的序列参数集（SPS），才开始当前GOP的IDR帧的编码。IDR帧本身是一个I帧，即它采用帧内预测方式，不需要参考其他帧。所以从这里可以看出，每个GOP都是一个独立的编码单元，在进行编码时并不要参考前面的GOP，从而使得一个GOP可以和其他GOP同时进行编码，这就是GOP层面的并行处理。

但由于GOP包含十多帧图像，数据量太大，一般不适合在普通计算机或其他便携式设备上作为基本单元进行整体处理。再者，GOP级的并行也不能用在实时领域，因为在进行实时视频采集时，一段时间内需要采集的视频非常有限，一般每秒25~60帧，不可能等到采集完若干个GOP后才进行编码工作。因此，GOP级的并行非常适合于非实时视频编码的应用，例如视频数据的压缩存储备份，分布式视频处理系统等。图9.5是一种GOP级并行处理方式的示意图。和具有n个处理器的多处理器系统相对应，将视频序列分成一个个GOP，每n个GOP分别作为n个处理器输入，每个处理器完成一个GOP的编码，一次并行处理完成n个GOP。然后将每个处理器输出的编码码流数据进行汇总，形成最终码流。

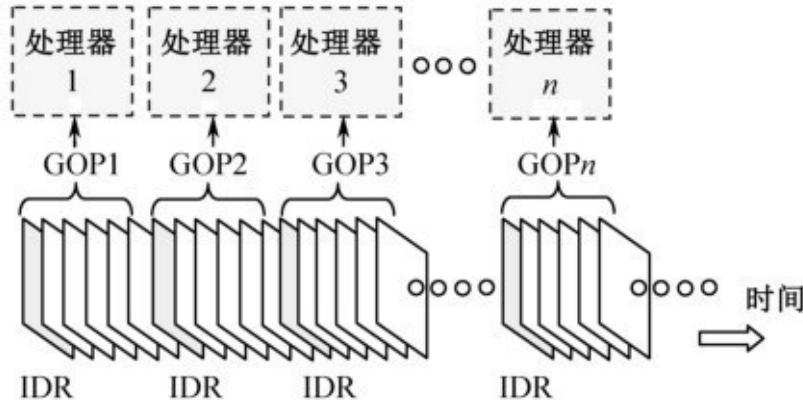


图9.5 GOP级并行编码

9.3.2 图像级并行处理

在图像级并行机制中，可同时处理多幅图像，且提供满足运动补偿预测的时间依赖。图像级并行机制对具有多核处理器的系统比较合适，因为这种方式的实现相对比较简单，不会遭遇效率损失。图像级并行机制也存在有若干限制。例如，并行化的分级受到运动矢量长度或GOP尺寸的限制，每个核的工作负担不平衡，并行机制增加了处理帧率而没有缩短延时。

在视频编码中，存在三种类型的编码帧，即I帧、P帧和B帧。I帧是采用帧内预测的方式编码，并不参考其他的帧，而P帧和B帧则是需要参考帧的，它们根据参考帧进行帧间预测，编码残差信息（当前图像减去预测图像）。由于这些帧之间存在相互参考的关系，所以不能简单对每个帧独立并行编码，而是应该按照它们之间的参考关系，使得相互没有依赖的帧能

同时进行编码。

下面，以多线程并行处理方式和最常用的IBBPBBPBB...结构（简称PBB结构）的视频编码为例来说明如何实现图像级的并行处理。按照上述结构，在每个视频序列中，I帧与P帧之间，或P帧与P帧之间都会有两个B帧。I帧和P帧是参考帧，其他的P帧或B帧都依赖于它们。B帧虽然参考（依赖）P帧或I帧，但B帧本身不作为参考帧，所以两个B帧之间不会存在参考的关系，图9.6显示了一个图像级并行处理以及帧之间的参考关系（或依赖性）的示例。

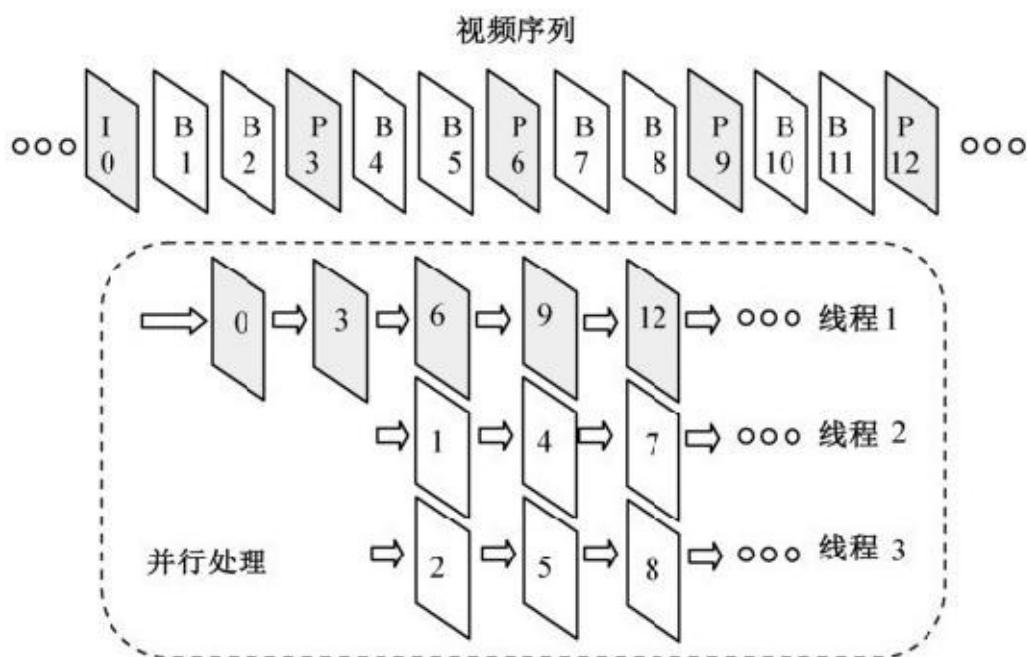


图9.6 图像级并行编码

图9.6的上半部为一按0~12编号的自然顺序排列的视频序列，根据PBB编码结构，各帧的编码I、B、P种类也标注其中。下半部为图像级并行处理的时序图。按照I、P、B帧之间的依赖关系，首先必须对开头的第0帧(I)进行编码，为后面的P帧提供参考。紧接着编码I帧后的第3帧(P)。第3帧编码完成后，第1、2两个双向预测B帧的前后参考第0帧和第3帧都具备了，前向预测第6帧(P)的参考第3帧也具备了，下面就可以开始并行处理了，即同时对第6帧(P)、第1帧(B)和第2帧(B)进行编码，形成3条编码线程。按照同样的方法，下一个节拍三条线程分别处理第9、第4和第5帧。再下一个节拍……如此处理下去，直至GOP结尾处。

从上面的示例可以看出，如果视频序列中B帧占的比例越大，则就有越多的帧可以同时进行编码，便可以获得更高的并行性。因此，P帧是编码器中的关键点，加速P帧的编码，可以使得有更多的帧能够准备好进行编码，从而避免了出现空闲线程的可能。

在具体的实现中，应该首先编码I帧和P帧，然后再处理B帧。图像级并行编码的思路是在编码完一个P帧后，使得紧接着的后一个P帧和依赖于该帧的几个B帧同时编码。这样由于同时编码B帧带来的并行处理的加速比理论上等于B帧数加一，不过由于创建线程、分配内存、线程间通信都需要开销，而且同时运行的线程数受限于CPU的核数，所以实际加速比会小于理论加速比。上一示例中的并行处理加速近似为3，即为串行处理的3倍，因为编码视频序列中每2个P帧之间就有2个B帧。

9.3.3 条、片级并行处理

1.Slice级并行机制

先前的视频标准，如H.264/AVC，已经在slice和块层面上有了并行机制。在HEVC中，也可以采取基于slice的并行处理机制。Slice的并行机制属于前面所述的数据并行处理方式，概念清楚，方法简单，就是将一帧图像划分为若干个包含整数个CTU的slice。除考虑环路滤波中穿越slice边界

的潜在的依赖性外，图像中所有常规slice相互之间是独立的。对一帧中多个slice采用多线程的方法并行处理，可以增加编码吞吐量，减少帧编码延时。但是，将一帧分为过多的slice会降低编码效率，原因有三：第一，由于打破了上下文模式的训练，不能穿越slice边界选择上下文，使熵编码的效率会变低；第二，在预测阶段不允许使用邻近slice的像素，有可能降低预测的精度；最后，需要在比特流中为每个slice附加头信息和起始码，增加了码流容量。

图9.2就是一种slice级并行化的方法，将一帧划分为4个slice，为一帧内的每个slice创建一个线程进行并行的编码。为编码每个 slice 传输的压缩视频数据加上适当的 slice 头信息，所最终将各个线程编码输出的数据组合起来，形成了该帧的最终码流。

从图9.2可以看出，slice级并行化的程度是由一帧所划分的slice的个数决定的，划分slice的个数越多，能同时运行的线程数就越多，从而加速比也就越大。但划分的树slice的个数并不是可以随便增加的，因为slice的划分会切断边界处CTU之间的相关性，当一个slice中的CTU无法利用另一个slice中CTU的相关性进行压缩时，压缩效率就会降低。实验证明，随着划分的slice数的增加，视频失真率会相应的增加，例如当一帧被划分为9个slice且维持相同级别的视频质量时，编码结果的平均码率增长约在10%以上。

2.Tile级并行机制

tile 是一种类似于 slice 的图像的划分机制，它将图像灵活地划分为多个包含整数个 CTUs的矩形区域，使得不同区域的CTUs之间的依赖性被禁止。以上所述是slice级的并行处理，如果将其中的“slice”换成“tile”也是成立的，这就是tile级的并行处理。

9.3.4 块级并行处理

宏块（MB）级并行处理在H.264/AVC中就有使用，例如，在基于硬件

实现的H.264/AVC中，普遍使用宏块级流水线处理。这一类MB级并行化技术是基于多核处理器的，一个核专用于熵编码，一个用于环路滤波，一个用于帧内预测，等等。在这种方式中，一个宏块将在不同的核中被处理，完成不同的压缩功能。因此这种并行处理也是如前所述的一种功能并行处理机制。但由于宏块间的多种依赖关系，宏块的有效并行处理需要采用比较复杂的调度方法来决定宏块处理的顺序。

在HEVC中，采用帧内波前方式的并行机制来编码图像帧，以满足预测和滤波时的块间依赖的要求。这实际上也是一种编码树块（CTB）层面的并行机制，不依赖于一帧中的多个 slice或tile，也没有相应的编码损失。

在解码时，熵解码在CTB层面不能并行化处理，必须在完整的一帧中顺序地完成。因此在多帧的情况下，熵解码是可以并行进行的。然而，这种方法为保留熵解码的语法元素而需要增加帧缓存器，且不能够减少重建和滤波过程中的帧时延。

9.3.5 指令级并行处理

1. 多媒体指令集

现代通用型的CPU一般都集成有多媒体扩展（Multi Media eXtension,MMX）指令集、单指令多数据流扩展（Streaming SIMD Extensions,SSE）指令集等多种扩展指令集，以用于加速多媒体和浮点的运算速度。一般普通指令在一个CPU时钟周期内，只能操作两个32位的寄存器，然而扩展的多媒体指令集，通过利用一种类似向量运算的方式，一条指令在一个CPU时钟周期内可以操作两个128位或256位的寄存器，这种技术被称为单指令多数据流（Single Instruction Multiple Data,SIMD）技术。采用这种技术使得同一个时钟周期所能运算的数据量增大了4倍或8倍，从而采用扩展指令集优化，也能大大提高编码速度。如图9.7所示是一个简单的SIMD的执行两个操作数的示意图，同时可以执行4对8比特数的操作。

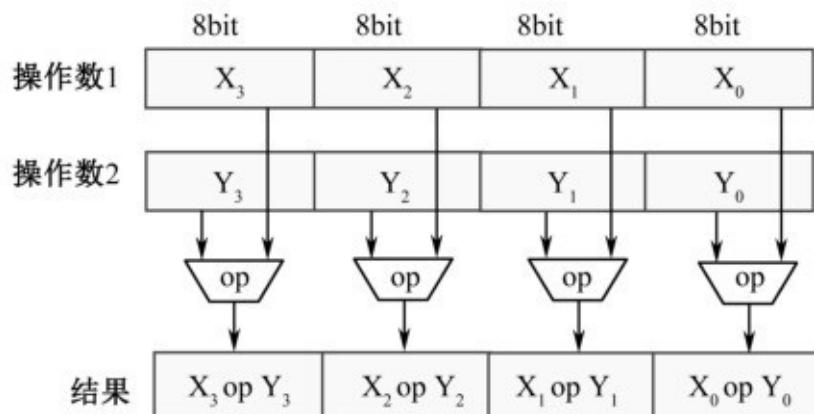


图9.7 SIMD的4字节并行操作

在视频编码中，可利用SIMD的扩展指令进行指令级并行化处理，如对4×4整数DCT变换与反变换，扩展8比特数据和16比特数据之间的转换，SAD计算，4×4量化与反量化等。采用扩展指令进行并行化加速还有一个好处就是，由于它用的是CPU内部的指令，所以并不会额外占用多核处理器的资源，这样相当于节约了处理器的使用。一般情况下，采用扩展指令并行化处理相比不采用指令级并行处理的速度提高了4倍。所以采用指令级的优化是一种最为廉价有效的方式，现在广泛用于音视频处理中。

2. 指令流水线

除计算机多媒体指令外，还可以在普通计算机中通过指令流水线，在程序执行时实现多条指令重叠进行操作，完成对数据的准并行处理。这种技术和前面介绍的流水线并行处理有所区别，流水线是在顺序指令流计算机中实现处理时间重叠的技术，因此流水线的并行处理是指完成一条指令的各个部件在时间上是可以同时重叠工作，通过提高各部件的利用率来提

高指令的平均执行速度。

指令流水线技术通过在复杂操作过程中插入寄存器，将其拆分成多个基本平衡的简单操作，每个简单操作称为一个流水线。如计算机的一个指令要完成对一组输入数据的4个操作，每个操作由不同的计算单元完成。在一般顺序处理的情况下，输入一组数据，执行完对此数据的4个操作，输出处理完的这组数据，而后再输入第二组数据，再进行4个操作处理，如此顺序执行下去。显然每组数据的需要4个操作周期才能完成。

在指令流水线结构中，如图9.8所示是一个常见的3个数据流通过4操作步骤的指令流水线结构的示例。每个时间周期都可以接受一组待处理的数据，如 a, b, c, \dots 相应地完成一组数据的处理，输出 A, B, C, \dots 计算单元不需要等待当前这一组数据的所有操作都完成后才开始处理下一组数据。每个操作周期内不同的计算单元对不同的数据进行不同的流水线操作，所有计算单元同时都在工作，避免了计算单元空闲等待，提高了系统的处理速度。区别于前述的并行处理技术，这种指令流水线技术并不需要增加额外的计算单元，只需要在不同流水线间增加用于缓存数据的寄存器。



图9.8 流水线指令处理示意图

9.4 去方块滤波的并行处理

去块效应滤波（DBF）的作用是消除重建图像的块效应，与H.264/AVC中 4×4 大小的滤波块不同，HEVC中滤波块的大小为 8×8 ，滤波操作包含对垂直边界的水平滤波和对水平边界的垂直滤波两部分。在去方块滤波中，可先将一帧图像划分为 8×8 的滤波栅格，栅格之间不重叠，覆盖整个画面，如图9.9中粗虚线所示。为了保持插图的简洁，图中只给出了4个 32×32 的CTU，只标注了一个栅格，实际上栅格应该有多个，覆盖全部CTU。栅格的水平/垂直边缘和CTU的水平/垂直边缘均有4个像素的偏移，以保证所有的PU或TU边缘（除了 4×4 的块）都处于滤波栅格的中间。然后可分别进行水平和垂直滤波，或者逐栅格滤波。

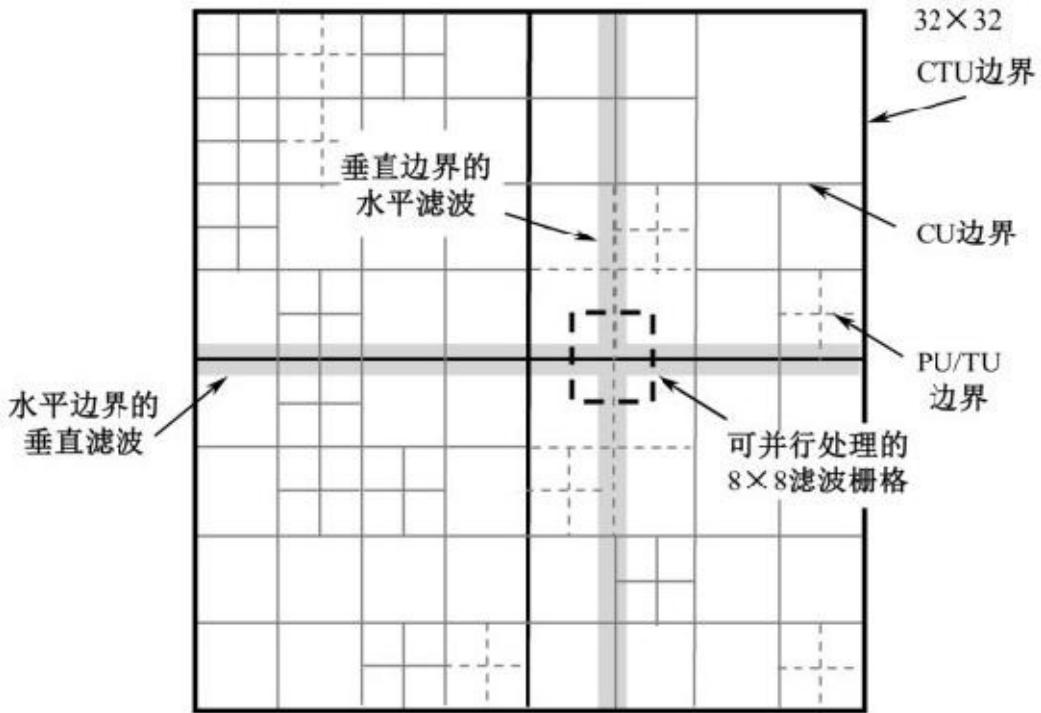


图9.9 DBF并行处理示意图

1. 边界的并行滤波

这里主要说明对垂直边界的水平方向滤波，对水平边界的垂直方向的滤波过程与水平滤波类似，只是方向转90°。

首先根据边界两边块的编码参数计算 8×8 滤波栅格中所有PU和TU的垂直边界的边界强度（BS）；如果 $BS > 0$ ，则通过靠近边界的4个像素值的特征判断其滤波模式，滤波模式有普通滤波、强滤波或无滤波3种；然后根据相应的滤波模式对垂直边界两边的像素进行滤波。

在HEVC的DBF滤波中，所有的垂直边界的滤波都是独立的，可以与

其他垂直边界并行进行滤波运算，而且经过垂直滤波后的样点值会作为输入值参与水平边界滤波的运算。因此可以设置多个垂直滤波线程进行并行处理。

2. 棚格的并行滤波

根据DBF滤波规律，对PU/TU垂直边界和水平边界交叉点为中心的 8×8 滤波栅格中“+”字形边界的DBF滤波只和本块内数据有关，因此棚格中就可以独立完成BDF滤波，不需要借助其他块的数据。每个棚格内部滤波时，先对“+”字形的垂直线（边界）进行水平滤波，再对滤波后数据的水平线（边界）进行垂直滤波。因此，也可以设置多个线程，为不同的 tile 或棚格行同时进行DBF滤波。在每个棚格行或tile中，DBF是逐栅格进行的。

本章参考文献

[1]High Efficiency Video Coding[S],ITU-T Rec.H.265 and ISO/IEC

23008-2,ITU-T and ISO/IEC JCT-VC,Mach

2013 (v.1) ,Oct.2014 (v.2) ,Apr.2015 (v.3) .

[2]Mauricio Alvarez-Mesa,Chi Ching Chi,Ben Juurlink,et al.Parallel video decoding in the emerging HEVC Standard[C].IEEE International Conference on Acoustic,Speech and Signal Processing (ICASSP) ,2012: 1545-1548.

[3]P.Piñol,H.Migallón,O.López-Granado,et al.Slice-based parallel approach for HEVC encoder[J].Supercomput (Published on line 30 Dec.2014) ,Springer,1882-1892.

[4]Mathias Wien.High Efficiency Video Coding:Coding Tools and Specification[M].Springer-Verlag Berlin Heidelberg,2015.

[5]Vivienne Sze,Madhukar Budagavi,Gary J.Sullivan.High Efficiency Video Coding (HEVC) :Algorithms and Architectures[M].Switzerland Springer International Publishing,2014.

[6]Hyunmi Kim,Seunghyun Cho,Kyungjin Byun,et al.Multi-core based HEVC hardware decofing system[C].IEEE International Conference on Multimedia and Expo Workshops (ICMEW 2014) ,1-2.

[7]Chenggang Yan,Yongdong Zhang,Feng Dai,et al.Parallel deblocking filter for HEVC on many-core processor[J].Electronics Letters,2014,50 (5) : 367-368.

[8]Seunghyun Cho,HyunMi Kim,Hui Yong Kim,et al.Efficient in-loop filtering across tile boundaries for multi-core HEVC hardware decoders with 4 K/8 K-UHD video applications[J].IEEE Trans.on

Multimedia,2015,17 (6) :778-791.

[9]Xiantao Jiang,Tian Song,Takashi Shimamoto,et al.Temporal prediction improvement for parallel processing of HEVC[C].IEEE Asia Pacific Conference on Circuits and Systems (APCCAS 2014) ,515-518.

[10]Wassim Hamidouche,Mickael Raulet,Olivier Deforge.Multi-core software architecture for the scalable HEVC decoder[C].2014 IEEE International Conference on Acoustic,Speech and Signal Processing (ICASSP) ,2014: 7545-7549.

[11]Junsoo Jeong,Junchul Choi,Soonhoi Ha.Parallelization and performance prediction for HEVC UHD real-time software decoding[C].IEEE 12th Symposium on Embedded Systems for Real-time Multimedia (ESTIMedia 2014) ,40-49.

[12]Damien de Saint Jorre,Claudio Alberti,Marco Mattavelli,et al.Exploring MPEG HEVC decoder parallelism for the efficient porting onto many-core platforms[C].IEEE International Conference on Image Processing (ICIP 2014) ,2115-2119.

[13]Georgios Georgakarakos,Leonidas Tsiopoulos,Johan Lillius,et al.Performance evaluation of parallel HEVC strategies[C].2015 23rd Euromicro International Conference on Parallel,Distributed,and Network-Based Processing,137-144.

[14]Shaobo Zhang,Xiaoyun Zhang,Zhiyong Gao.Implementation and improvement of waveform parallel processing for HEVC encoding on many-core platform[C].IEEE International Conference on Multimedia and Expo Workshops (ICMEW 2014) ,1-6.

[15]S.Radicke,J.Hahn,C.Grecos,Q.Wang.A highly-parallel approach on motion estimation for High Efficiency Video Coding (HEVC) [C].2014 IEEE International Conference on Consumer Electronics (ICCE) ,187-188.

[16]Alvarez-Mesa M,George V,Chi CC,et al.Improving parallelization

efficiency of WPP using overlapped waveform[R],Document JCTVC-J0425,Joint Collaborative Team on Video Coding (JCT-VC) ,Stockholm,July 2012.

第10章 HEVC的高层语法

HEVC视频编码器将输入视频变为压缩的码流输出，码流由串行的二进制比特流组成。这些二进制比特实际上表示的是一系列的语法元素（syntax element），按照HEVC的规定，若干二进制比特表示一个语法元素，因此也可以说压缩码流是由一连串的语法元素组成的。当然，不同的语法元素表示不同的物理意义，如slice的分割、宏块的类型、图像参数等。在HEVC标准中，语法元素的具体含义是由语义（semantics）来详细阐述的。

在以往的视频编码标准中，都是只对解码器进行规范，要求声称符合某视频编码标准的设备（或软件）必须能够解码该标准的压缩视频流，实际上也要求该设备的编码压缩视频输出要符合该标准的视频流规范，即要能够被该标准的解码器正确解码。这种方式给编码器的设计以较大的自由度，通过竞争的方式获得高性能的视频压缩。HEVC 也不例外，也只是对解码器进行规范的，就是通过定义语法和语义来规范编解码器的工作流程。因此，语法和语义是这些标准的核心部分，是具体实现标准编解码操作的守则。

由于HEVC的语法和语义部分的内容十分具体，篇幅很大，这里不可能、也没有必要全面介绍。本章将在介绍有关语法的一些基本概念的基础上，着重分析HEVC的高层语法。

描述比特流结构的语法元素，或提供用于一幅图像中多个编码块或多幅图像的信息，例如参数集、参考图像管理语法、SEI消息等，就是HEVC的“高层语法”（High-level syntax）部分。高层语法的设计扩大了标准的应用范围和灵活性，影响和系统的接口，提高了数据丢失的鲁棒性，有利于

提供新功能。可见，高层语法实际上就是不属于表示具体视频内容编码的那部分语法元素，也就是在视频编码层语法基础上的语法，为编码视频数据的进一步处理提供封装。HEVC 的高层语法的体系基本上是从 H.264/AVC 继承而来，进行了一些修改和扩展，以适应HEVC不同的需求，保证HEVC能力的提升。

10.1 HEVC语法特点

HEVC继承了不少H.264/AVC的结构和高层语法特点，如NAL单元、参数集、图像顺序计数（Picture Order Count,POC）、补充增强数据（Supplemental Enhancement Information,SEI）消息等。同时也有不少H.264/AVC中的高层特点并没有被在HEVC采用，如灵活宏块顺序（FMO）、冗余条、任意条顺序（ASO）、数据分割和SP/SI帧等。更重要的是HEVC引入了一系列新的高层语法结构，如视频参数集（VPS），纯随机接入（CRA）图像，断点连接接入（Broken Link Access,BLA）图像，时域子层接入（Temporal Sub-layer Access,TSA）图像，提供并行机制的片和波前工具，用于减少延时的关联条（dependent slice）工具，用于对参考图像管理的参考图像集（RPS）概念。

10.1.1 新增语法结构和元素

HEVC在继承H.264/AVC高层语法结构的基础上，为进一步增加在不同应用和网络环境中的操作灵活性，增强对丢失数据的鲁棒性，增加了不少新的语法结构和语法元素。

1.随机接入单元

良好的随机接入特性支持对信道切换、搜寻操作和动态流服务是十分关键的。为保证不同压缩视频流之间的顺利切换，或随机在视频流中某处开始进行正确解码，往往在压缩码流中设置若干特殊的帧，这些帧也称为随机接入点（Random Access Point,RAP）。在比特流的RAP位置，解码器可以开始顺利地解码图像，而不需要解码任何比特流中更早时刻出现的图像。

即时解码刷新 (IDR) 单元是码流中的RAP，它包含一幅独立的编码图像。从IDR图像开始，可以独立解码该图像以及在它后面的图像，无须依赖码流中的任何先于IDR的图像。在编码流中，IDR图像的出现既标志前面图像组 (GOP) 的结束，也标志新的GOP的开始。

除IDR单元以外，HEVC的RAP引入了2种新的随机接入单元：纯随机接入 (CRA) 图像和断点连接接入 (BLA) 图像。CRA允许它后面图像的解码可以参考解码顺序在CRA后、显示顺序CRA前的图像。BLA本质上也是CRA，是用于视频流切换的CRA。当前视频流在某个 RAP 处要求切换到另一个视频流继续解码，则直接将该 CRA 同另一个视频流中的接入处CRA连接，后者便是BLA。可见，HEVC的RAP帧可以是IDR帧、CRA帧和BLA帧。

2.时域子层

类似H.264/AVC可分级编码 (SVC) 扩展中的时域可分级特性，HEVC在NAL的头信息中定义了一个时域标志，以明确时域分层预测结构中的层。用这一标志在获取时间可分级信息时，只需要解析NAL单元的头，不需要拆解比特流的部分。

某些情况下，在对一个编码视频序列进行解码处理期间，解码时域子层的数量可适当调整。在比特流中，时域子层接入 (Temporal Sublayer Access,TSA) 帧和分步时域子层接入 (Step-wise TSA,STSA) 帧的出现标明，在此位置允许子层切换，可以开始解码某一时域高层图像。在TSA 帧的位置，允许从解码一个较低时域子层切换到解码任何一个较高时域子层；而在 STAS帧位置，只允许从解码一个较低时域子层切换到解码高一层的时域子层，不能高更多的层，除非这些层包含STSA或TSA帧。

3.视频参数集 (VPS)

在H.264/AVC中，参数集机制将序列层和图像层的公共信息集中存放，供解码时共享，提供了一种高效编码、加强对重要数据保护的机制。HEVC 除继续保留 H.264/AVC 序列参数集 (SPS) 和图像参数集 (PPS) 外，还在序列层上面增加了新的视频参数集 (Video Parameter Set,VPS) 结

构，作为元数据来描述编码视频的总体特征，包括时域子层间关联在内。其主要目的是借助于系统层的信令，确保HEVC标准的可兼容扩展性。例如，一个普通解码器要解码扩展的可分级编码或多视点编码的比特流时，只需要比特流中的基本层信息，而那些有关特比流结构的附加信息将被忽略，因为它们只和扩展解码器有关。

4.参考图像集和参考图像列表

HEVC 为了管理解码多参考帧，需将有关的先前已解码图像保留在解码图像缓存（DPB）中，用作码流中后续图像解码的参考。保留下来的参考图像称为参考图像集（Reference Picture Set,RPS）。为了识别这些图像，需在每个条（slice）头部发送一个图像顺序计数（POC）标志符列表。

如同H.264/AVC,HEVC也有两个参考图像列表，形成了解码图像缓存中图像的列表，分别为参考图像列表0（List0）和参考图像列表1（List1）。同时还设立了参考图像索引用于识别某一列表中的某一具体图像。单向预测时，可以从任意一个列表中选出一帧。双向预测时，则可从两个列表中各选出一帧。如果某一列表中只有一帧图像，则意味参考图像索引值为0，不必在码流中传送。

5.NAL单元头信息扩充

视频编码的VCL数据或non-VCL数据都需放入网络提取层（NAL）单元的数据包中。HEVC的NAL单元的类型较H.264/AVC有所增加，NAL单元头信息也从1个字节增加为2个字节，通过NAL单元的2个字节的头信息，可以充分地标识NAL单元所装载数据的类型和用途。

6.灵活的条语法

条是一种数据结构，它可以不依赖同帧中的其他条被而独立编码，如信号预测、残差的变换、系数量化、熵编码等。一个条可以是一个完全的帧，也可以只是一帧中的一部分。条的主要作用之一是在数据丢失时用作重同步。在打包传输的情况下，一个条可装载的最大比特数是被严格限制的，而条内包含的CTU数量通常是可以改变的，在这个限制内调整每个包

的大小，以使打包的额外开销达到最小。

7.补充增强信息（SEI）和视频可用信息（VUI）

HEVC语法支持不同类型元数据，如补充增强信息（SEI）和视频可用信息（Video Usability Information,VUI）。这类信息主要提供和视频有关的非视频编码数据，如视频图像的定时、视频彩色空间的说明、立体帧携带的3D信息等。

10.1.2 基本语法表示

1.基本语法术语

如前所述，HEVC的码流实质上是由一系列的语法元素组成，一个实际的解码器就应该具备解析标准码流中语法元素的能力。解码器的解码过程实际上就是“翻译”码流中语法元素的过程，解码器依次从码流中依次读入一个个语法元素进行解析，直至解码完成。为了方便理解HEVC的语法元素的描述和解析过程，这里先介绍几种常用的语法术语。

(1) 语法元素：表示编码输出比特的具体含义。视频编码器输出的码流中若干二进制比特形成一个语法元素，表示编码视频的某个特定的物理意义，如NAL类型、量化参数等。码流中每个比特都隶属某个语法元素，也就是说码流是由一个个语法元素依次衔接组成的，码流中除了语法元素并不存在其他的数据。

(2) 语法：表示若干语法元素的组织结构。

(3) 语义：表示语法元素的具体含义。

(4) 码流：也称比特流，即视频编码器输出的一连串的二进制比特。

(5) 变量：对语法元素进行求值计算得出的中间数据，与计算机语言中变量的概念类似。

(6) 描述符：表示从比特流中提取语法元素所对应比特的方法，即语法元素的解析算法。

2.三种结构的描述

对比特流语法元素的结构的描述，HEVC标准和H.264/AVC一样，采用一种类似计算机C语言的描述方法，提供对陈述、判断、循环等结构的基本描述语法。

(1) 陈述结构 (statements) :

用大括号封装的一组陈述 (statements)，其功能可看作一句“陈述”。

```
{  
    statement 1  
    statement 2  
    ...  
}
```

(2) 判断结构 (if ... else) :

测试“条件”是否为真，如果为真，执行基本操作，否则执行另外的操作。如果没有另外的操作，“else”及其后可以省略。

```
if ( condition )  
    primary statement  
else  
    alternative statement
```

(3) 循环等结构：

①“while”结构：测试“条件”是否为真，如果为真，重复执行操作，直至“条件”不为真。

```
while ( condition )  
    statement
```

②“do ... while”结构：每执行一次操作后测试“条件”是否为真，如果为真，则重复执行操作，直至“条件”不为真。

```
do  
    statement  
while ( condition )
```

③“for”结构：定义一个初始值，随后是测试“条件”，如果“条件”为

真，重复执行基本操作并测试“条件”，如果“条件”不再为真，则执行后续操作。

```
for ( initial statement; condition; subsequent statement )
```

```
    primary statement
```

3.语法元素和变量名称

为了说明语法元素的具体含义，也采用类似计算机C编程语言来描述，又称伪代码（pseudo code）描述。

（1）语法元素的名称：由小写字母和一系列的下划线组成，如语法元素nal_unit_type表示NAL单元的类型，语法元素pic_width_in_luma_samples类似于一个函数，表征图像的宽度，以亮度样点为单位计算。

（2）变量名称：由大小写字母组成，中间没有下画线，如PicWidthInCtbsY 表示“图像宽度中亮度CTB数”，名称可以基本反映该变量表示的内容。

4.语法元素的描述符

与H.264/AVC一样，HEVC的每个语法元素都有相对应的描述符。描述符表示从比特流中提取语法元素的方法，即语法元素的解析方法。一种描述符可对应多种语法元素。下面说明每个描述符的作用。由于视频编码的最后一步为熵编码，码流中大部分数据是熵编码的结果，因此码流中大多数描述符是针对熵编码的解码算法。

下面的描述符定义了每个语法元素的解析过程：

ae (v) : 基于上下文的自适应二进制算术编码的语法元素。

b (8) : 读进连续的8比特串（1字节），其解析过程由函数read_bits (8) 的返回值来说明。

f (n) : 读进连续的 n 比特（从左到右）固定模式的比特串，其解析过程由函数 read_bits (n) 的返回值来说明。

i (n/v) : 读进n 比特的有符号整数，在语法表中，当n 是“v”时，读进的比特数是根据其他语法元素的值而变化的，其解析过程由函数read_bits (n) 的返回值来说明。

se (v) : 有符号整数0阶Exp-Golomb熵编码，最左边的比特为先。

u (n/v) : 读进n比特的无符号整数，最左边的比特为先。当n是“v”时，读进的比特数是根据其他语法元素的值而变化的，其解析过程由函数read_bits (n) 的返回值来说明。

ue (v) : 无符号整数的0阶Exp-Golomb熵编码，最左边的比特为先。

read_bits (n) : 从比特流中读进 n 比特，将比特流指针位置前移 n 比特；当 n=0 时，规定read_bit (n) 返回一个0值，但不前移比特流指针。

上述描述符中，b (8) 表示在码流中连续读取的8个比特。其余的描述符在括号中都带有一个参数，这个参数表示需要读取的比特数。当参数是n时，表明调用这个描述符时会指明n的值，说明对应的语法元素是定长编码。当参数是 v 时，说明对应的语法元素是变长编码。对于熵编码描述符，由它们的解码算法本身来确定当前语法元素的比特长度。

10.2 H.264/AVC语法提要

HEVC不仅在编码原理、码流结构方面和H.264/AVC一脉相承，而且基本沿用了H.264/AVC的语法规则，在具体的细节上作了一些修改和增减，从某种程度上讲HEVC的语法和语义可算作H.264/AVC的扩充版。在这里将H.264/AVC中的码流层次结构、NAL单元、条和参数集语法表示作简要介绍，可有助于进一步理解和掌握HEVC的语法和语义。

10.2.1 码流的分层结构

H.264/AVC的基本流（Elementary Stream,ES）的结构分为两层：视频编码层（Video Coding Layer,VCL）和网络提取层（Network Abstraction Layer,NAL）。具体的结构可参考第3章图3.4,VCL层负责有效表示被压缩编码后的视频序列数据。NAL层用适当的方式加一些头信息来格式化VCL的编码数据，提供了一个视频编码器和传输系统的友好接口，以适应各种不同的通信网络和存储方式的需求。

H.264/AVC视频编码器输出的码流形式上就是一连串的二进制比特流，本质上却是一连串的语法元素“流”，但这些语法元素并不是随意放置的，而是被组织成一种有层次的结构，可参考第3章图3.1。编码视频流由一系列的图像组（GOP）顺序排列而成。一个GOP的第一帧一定是帧内编码图像，I帧，又称即时解码刷新（IDR）图像，它的编码不涉及其他帧数据。但I图像不一定是IDR图像，在GOP的其他地方也允许有I帧。GOP中每一帧又可以分为若干条（slice），再以slice为单位进行编码。每个slice是由若干宏块组成（Macro Block,MB），其尺寸为 16×16 ，是H264/AVC中编码的基本单位。而MB是由4个 8×8 亮度块和2个同面积的 8×8 色差块组成。这

样就形成了H.264/AVC的5层码流结构，由上到下分别为GOP、帧、Slice、MB和B。

10.2.2 NAL单元语法

1.NAL单元结构

H.264/AVC 的 NAL 单元的具体结构如图10.1所示，包括原始字节序列载荷（Raw Byte Sequence Payload,RBSP）信息和一个字节的NAL头信息。头信息包括3个语法元素。

(1) 1比特的禁止位F, forbidden_zero_bit (1)，编码中默认为0。当网络识别此单元中存在比特错误时，可将其设置为1，以便接收方丢弃该单元，或者尝试重构这个NAL单元，主要适应误码易发或不同种类的网络环境中使用。

(2) 2比特的优先级NRI,nal_ref_idc (2)，取值范围为0~3，值越高表示当前NAL越重要，需要优先受到保护。

(3) 5比特的NAL单元的类型信息，nal_unit_type (5)，表示该单元中有效载荷RBSP的数据类型。

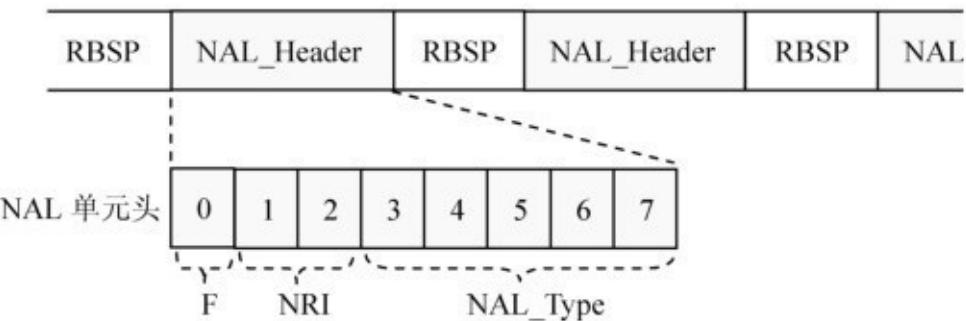


图10.1 H.264/AVC的NAL单元结构

2.封装SODB为RBSP

H.264/AVC 视频编码器刚刚生成二进制比特流称为数据比特串（String Of Data Bits, SODB），需要按一定的规则经过转换为符合NAL单元要求的RBSP的数据格式。如果SODB 内容是空的，生成的RBSP也是空的，否则，RBSP由如下方式生成。

RBSP 的第1个字节直接取自SODB的第1 ~ 第8bit，第2个字节直接取自SODB的第9 ~ 第16bit，以此类推，RBSP其余的每个字节都直接取自SODB的相应比特。RBSP的最后一个字节包含SODB的最后几个比特，如果不满足8bit，则由rbsp_trailing_bits()填充，填充的第一个bit是1，接下来是0，直到字节对齐。

3.NAL单元的类型

nal_unit_type标识NAL单元中的RBSP数据的类型，如表10.1所示。
nal_unit_type为1,2,3,4,5的NAL单元为VCL NAL单元，即和视频图像数据有关的单元，见表格中深色行。其他类型的NAL单元为non-VCL NAL单元。

表10.1 nal_unit_type语义

| nal_unit_type | NAL 类型 (RBSP 数据类型) | |
|---------------|--------------------|---|
| 0 | 未定义 | |
| 1 | 非 IDR 图像，条不分区 | slice_layer_without_partitioning_rbsp() |
| 2 | 非 IDR 图像，条分区 A | slice_data_partition_a_layer_rbsp() |
| 3 | 非 IDR 图像，条分区 B | slice_data_partition_b_layer_rbsp() |
| 4 | 非 IDR 图像，条分区 C | slice_data_partition_c_layer_rbsp() |
| 5 | IDR 图像中的条 | slice_layer_without_partitioning_rbsp() |

续表

| nal_unit_type | NAL 类型 (RBSP 数据类型) | |
|---------------|--------------------|---|
| 6 | 补充增强信息 (SEI) | sei_rbsp() |
| 7 | 序列参数集 | seq_parameter_set_rbsp() |
| 8 | 图像参数集 | pic_parameter_set_rbsp() |
| 9 | 分界符 | access_unitDelimiter_rbsp() |
| 10 | 序列结束符 | end_of_seq_rbsp() |
| 11 | 帧流结束符 | end_of_stream_rbsp() |
| 12 | 填充数据 | filler_data_rbsp() |
| 13 | 序列参数集扩展 | seq_parameter_set_extension_rbsp() |
| 14 | NAL 单元前缀 | prefix_nal_unit_rbsp() |
| 15 | 子序列参数集 | subct_seq_parameter_set_rbsp() |
| 19 | 不分区辅助编码图像条 | slice_layer_without_partitioning_rbsp() |
| 20 | 缓码条宽展 | slice_layer_extension_rbsp() |
| 16~18, 21~31 | 保留/未定义 | |

4.NAL单元的起始码

H.264/AVC编码时，在每个NAL单元前添加3字节的起始码

0x000001（16进制），解码器一旦在码流中检测到起始码，就说明刚刚检测的NAL结束，新的NAL即将开始。为防止NAL内部RBSP数据中出现“0x000001”数据，被解码器误认为起始码，H.264/AVC采取了一种防止竞争的机制。在编码完一个NAL时，编码器按字节搜索NAL单元的RBSP的数据中是否存在二进制数“00000000 00000000 0000000000”，这里00表示可能的2比特数，即00、01、10或11。如果存在，则在2个全0字节后面插入一个“0x03”字节的“添加码”来形成新的比特模式，即“00000000 00000000 00000011 000000 00”。这样，在解码时如果发现“00000000 00000000 00000011 000000 00”字节，则扣除“0x03”后还原为原来的数据。

10.2.3 Slice语法

1.宏块、条和条组

在 H.264/AVC 中，一帧视频图像可编码成一个或多个条 (Slice)，每条包含整数个宏块，即每条至少1个宏块，最多时包含整个图像。划分条的目的是为了限制误码的扩散和传输，使编码条相互间保持独立。

H.264/AVC的条共有5种类型：I条（只包含I宏块）、P条（可包含P和I宏块）、B条（可包含B、P和I宏块）、用于不同编码流之间的切换预测（Switching Predictive,SP）条和帧内切换（Switching Intra,SI）条。

条组 (slice group) 包含一个或若干个条，当然也包含更多个宏块。一般条组在编码时，每条的宏块是按扫描次序进行编码的。但是也有多种不按扫描顺序编码 MB 的，如任意条次序 (Arbitrary Slice Order,ASO) 的条组，允许一个编码帧中的条之后可以跟随任一解码图像的条。再如灵活宏块次序 (Flexible Macroblock Ordering,FMO) 条组，采用灵活的方法，把编码的宏块映射到相应的条组中，以增加编码视频的抗误码能力。

2.条 (Slice) 层语法

Slice层语法结构是H.264/AVC压缩码流中的一个核心部分，Slice语法结构主要由Slice头信息和Slice数据两部分组成，如图10.2所示。

Slice 头信息表示当前 slice 的编码特性，如 PPS 的语法元素 pic_parameter_set_id，类型 (I-Slice,P-Slice 等) 语法元素 slice_type，第一个宏块在 slice 中的位置的语法元素 first_mb_in_slice，去方块滤波操作相关的语法元素等。

Slice数据部分由多种宏块语法元素组成，如编码模式的语法元素 mb_type，色度帧内预测模式的语法元素 intra_chroma_pred_mode，编码块模式 (Coded Block Pattern,CBP) 的语法元素 coded_block_pattern，量化参数调整量的语法元素 mb_gp_delta 等。当然还包括亮度块和色度块的残差值经过变换、量化以及熵编码后的数据。

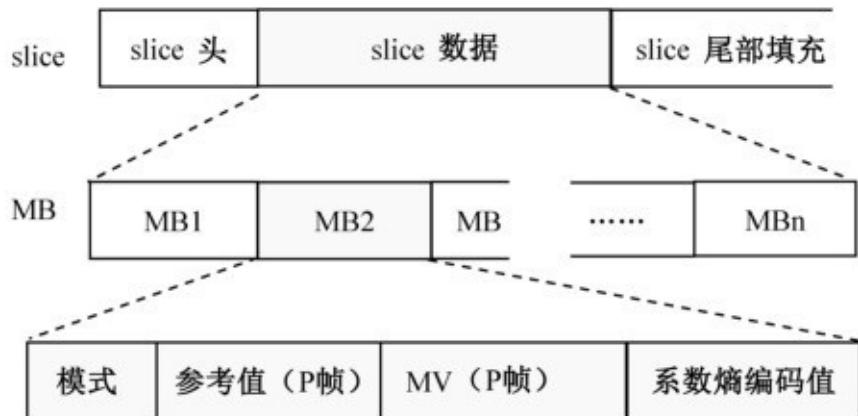


图10.2 slice语法结构

10.2.4 参数集

在以往的视频标准中，每一编码层次的头部信息都和相应的压缩数据放在一起。由于头部的语法元素是该层数据的核心，一旦头部信息丢失，数据部分的信息几乎不可能再被正确解码出来，尤其在序列层及图像层。为克服这一缺陷，H.264/AVC 中将原本属于序列和图像头部的大部分语法元素分离出来形成序列和图像两级参数集，其余的部分则放入条层，图 10.3描述了参数集与参数集外语法元素的关系。

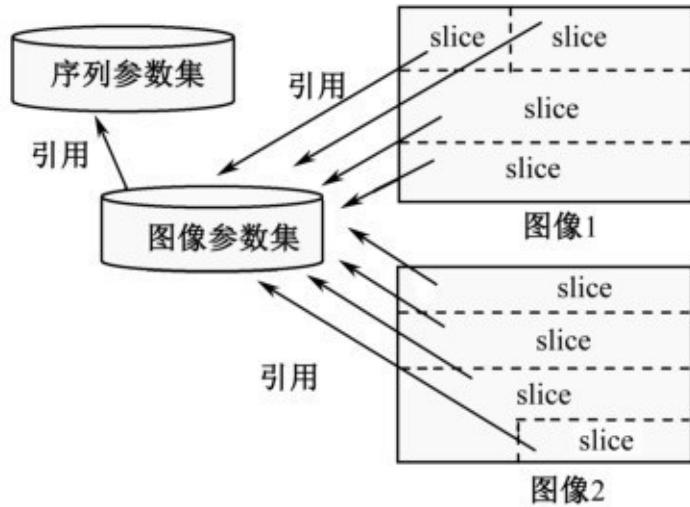


图10.3 H.264/AVC的序列参数集和图像参数集

参数集只是在条层语法元素需要的时候被引用，而且，一个参数集并不对应某个特定的图像或序列，同一个序列参数集（SPS）可以被多个序列中的图像参数集（PPS）引用，同理，同一个图像参数集也可以被多幅图像引用。只在编码器认为需要更新参数集的内容时，才会发送出新的参数。由于参数集是一个独立的数据单元，不依赖于参数集外的其他语法元素，可以被多次重发或采用特殊技术加以保护。

10.3 HEVC的NAL单元

NAL单元是HEVC封装码流的工具，也是语法元素的封装工具，编码器输出的是NAL单元，网络中传输的是NAL单元，解码器接收的也是NAL单元。因此，我们在关注高层语法时，首先需要了解的就是和NAL单元相关的结构和语法。如图10.4所示的是一个HEVC编解码器系统。输入视频图像进入编码器经编码后形成编码比特流输出。HEVC 输出的比特流包含的是NAL单元的数据序列，每个NAL单元包含整数个字节，其中前2个字节为NAL单元头，其余的字节为载荷数据。有些NAL单元携带的载荷信息是用于一幅或多幅图像控制信息的参数集，另一些NAL单元携带的是某一图像中的编码样值的数据。传送到解码端的NAL单元被解码器解码为重建图像输出显示或存储。编码器端和解码器端各有一个解码图像缓存（DPB），主要用于存储已解码图像，为当前编码图像的预测提供参考。



图10.4 HEVC的编解码系统

由图10.4可知，类似于H.264/AVC,HEVC也使用基于比特流结构的NAL单元。视频编码比特流被划分为若干NAL单元，当它在IP网络中传输时，NAL单元应当比网络允许的最大传输单元（Maximum Tansfor Unit,MTU）的尺寸小。每个NAL单元由NAL单元头和紧随其后的NAL单元载荷组成。NAL单元从携带的数据种类不同上可分为两类。一类为视频编码层NAL（VCL NAL）单元，包含编码的取样数据，如编码的slice NAL单元；另一类为非视频编码层NAL（non-VCL NAL）单元，包含通常属于多幅编码图像的控制数据等，如参数集NAL单元，或是解码过程不需要的信息，如补充增强休息SEI NAL单元。

HEVC码流与H.264/AVC码流的重要区别之一也在于NAL层。H.264/AVC中，视频参数封存于序列参数集（SPS）和图像参数集（PPS）的NAL包中。在HEVC中，除继续使用SPS和PPS封存视频参数外，还新增了一种视频参数集（VPS）的NAL单元，其中包含视频的一些全局信息，如Profile、Level等。

与H.264/AVC相比，HEVC中NAL头变成两字节长度，同时加入了该NAL所在的时间层的标识，去掉了表示是否被参考的 NRI 标识 nal_ref_idc，并将是否被参考的信息定义在了nal_unit_type中，即只有某些类型的NAL单元被参考。

由于近年来并行技术的兴起，HEVC在高层语法中加入了大量对并行技术的支持，包括传统的slice和新的tiles、WPP和关联条技术。这些技术使得一帧之内的语法元素不再是从头到尾的依赖关系，而是可以并行地进行编码和解码。较高复杂度的 CABAC 熵编码方式也能应用于实时编码的场景下，因而传统的CAVLC熵编码方式已基本被HEVC淘汰，仅在头信息编码中使用。

10.3.1 字节流格式

类似H.264/AVC,HEVC标准提供了一种自包含的字节流格式来表示 NAL单元流。这种字节流的格式定义在HEVC附录B中。例如，这种格式可以用于在流化编码视频字节流格式和视频文件（或网络视频）之间相互转换。如果传输层已经提供其他方式封装原始NAL单元数据，也可以不再需要使用字节流格式。例如，经常利用实时传输协议（Real-time Transport Protocol,RTP）的载荷格式来封装NAL单元数据。

当采用其他传输协议时，一个用户数据报协议（User Datagram Protocol,UDP）包就是一个NAL单元，解码器可以很方便地检测出NAL分界和解码。但在字节流格式中，NAL单元被编码成字节的码流，解码器将无法从数据流中分别出每个NAL单元的起始位置和终止位置。为保证解码器能够从码流中检测和提取NAL单元，字节流格式在字节流中每个NAL单元前定义了一个起同步作用的起始码前缀。起始码前缀由3个字节组成，2个“0”字节和一个“1”字节：

起始码前缀=0x00 00 01（16进制）=0000 0000 0000 0000 0001（2进制）

如果 NAL 单元是编码视频序列中的第一个接入单元的第一个 NAL 单元，或者如果 NAL 单元包含一个参数集，则要将一个附加的 0x00 字节插入到起始码之前。解码器通过扫描字节流寻找起始码前缀，就可由此确定每个 NAL 单元的开始字节的位置，确定 NAL 单元的总长度。

编码器为了防止在 NAL 单元内部 RBSP 数据流中出现起始码前缀字节 0x00 00 01 的数据，被解码器误认为起始码，引起解码同步的混乱，编码器就在这个“伪”起始码中插入一个防伪字节“0x03”，成为 0x00 00 03 01。解码器在搜寻起始码时就不会收到伪起始码的干扰，同时将遇到的 0x00 00 03 01 防伪数据中的“0x03”部分丢弃，还原为原来的数据。

实际上编码器不仅在 NAL 单元的 RBSP 中搜寻 0x00 00 01 数据，同时还搜寻 0x00 00 00、0x00 00 02 和 0x00 00 03。如果存在，则在每个码字的前 2 个全 0 字节后面插入一个“0x03”字节，形成对应的新防伪码字，0x00 00 03 00、0x00 00 03 01、0x00 00 03 02 和 0x00 00 03 03。同样在解码时如果发现这些码字，则扣除其中的“0x03”字节，还原原来的数据。这里，针对 0x01 的添加是消除数据中的伪起始码，针对 0x02 的添加是备用措施，以备将来 0x02 作为他用，针对 0x03 的添加是消除数据中的伪添加码，针对 0x00 的添加是消除数据中的长串的连零。

10.3.2 一般 NAL 单元语法

1. 语法表

与 H.264/AVC 类似，HEVC 的 VCL 也是用于规范视频数据的内容，NAL 则以适应各种不同通信网络或存储媒体要求的方式，用于规范数据格式和提供头信息。NAL 单元包含了所有的数据，每个数据包含整数个字节。NAL 单元规范了在包传输和比特流传输系统中的一般数据格式。除在附录 B 中定义的字节流格式中每个 NAL 单元的前面有前缀开始码和一些特殊的填充字节外，面向包传输和字节流传输的 NAL 单元的格式是相同的。一般的 NAL 单元的语法如表 10.2 所示。

表10.2 一般NAL单元语法

| NAL单元语义 | 描述符 |
|--|------|
| nal_unit(NumBytesInNalUnit) { | |
| nal_unit_header(); | |
| NumBytesInRbsp = 0; | |
| for(i = 2; i < NumBytesInNalUnit; i++) | |
| if(i + 2 < NumBytesInNalUnit && next_bits(24) == 0x000003) { | |
| rbsp_byte[NumBytesInRbsp++] | b(8) |
| rbsp_byte[NumBytesInRbsp++] | b(8) |
| i += 2; | |
| emulation_prevention_three_byte /* equal to 0x03 */ | f(8) |
| } else | |
| rbsp_byte[NumBytesInRbsp++] | b(8) |
| } | |

2.语义

下面对照表10.2的一般NAL单元语法表，简单介绍它的语义。

NumBytesInNalUnit，它说明了NAL单元的大小（以字节为单位），在解码NAL单元时需要使用这个值。为保证NumBytesInNalUnit的含义，一些NAL单元边界划分的方法是必需的。对于字节流格式的划分方法在附录B中说明，其他的划分方法不属于HEVC的内容。

rbsp_byte[i]是原始字节序列载荷（RBSP）中的第i个字节，RBSP是一个有序的字节序列，以下面的形式包含一个数据比特串（SODB）：

如果SODB是空的（例如0比特长度），那么RBSP也是空的；

否则，RBSP以下面的形式包含SODB：

（1）RBSP的第一个字节包含了SODB的最重要、最左边8个比特，RBSP中接下来的字节包含了SODB接下来的8个比特，如此继续，一直到SODB中的比特数少于8。

（2）在SODB之后将是rbsp_trailing_bits（），如下所示：

（a）最后的RBSP字节中的最重要、最左边的前几个比特包含SODB中剩余的比特（如果说有的话）；

（b）接下来的比特是由单个等于1的rbsp_stop_one_bit组成；

（c）当 rbsp_stop_one_bit 不是对齐字节中的最后一个比特时，那么将填充一个或多个rbsp_alignment_zero_bit进行字节对齐。

(3) 在 RBSP 结尾处的 rbsp_trailing_bits() 之后的 RBSP 中，可能出现一个或者多个等于0x0000的16比特语法元素cabac_zero_word。具有这些RBSP性质的语法结构在语法表中将用“_rbsp”后缀进行标识。这些结构将以rbsp_byte[i]数据字节的形式包含在NAL单元中。

例如，SODB的最后一个字节的只有4个比特为1001，这时 rbsp_trailing_bits填充的比特为rbsp_stop_one_bit+3个 rbsp_alignment_zero_bit=“1”+“000”=1000，如图10.5所示，形成的RBSP的最后一个字节为10011000，保证了RBSP为整数字节。图中未考虑在RBSP结尾处附加的cabac_zero_word=0x0000。



图10.5 从SODB到RBSP的转换

当知道RBSP的边界之后，解码器将通过连接RBSP字节和去除rbsp_stop_one_bit的方法从RBSP中提取出SODB,rbsp_stop_one_bit是最后一个（最不重要、最右边）等于1的bit，并且还要丢弃所有跟随在rbsp_stop_one_bit之后（次重要、次右边）的都等于0的比特。解码过程所需要的数据都包括在RBSP中的SODB部分。

10.3.3 NAL单元头语法

1.NAL头的结构

H.264/AVC定义了1字节的NAL单元头，在HEVC中定义了2字节的NAL单元头，这样的设计足以支持HEVC可分级编码、多视点编码和3D视频编码的扩展，还能支持其他的扩展。NAL单元头是一种固定格式，不包括变长编码。图10.6显示了HEVC的NAL单元的头部结构，其中各个比特的含义如下。

(1) 1bit的禁止位F,forbidden_zero_bit (1)，编码中默认为0。它的作用就是在尚存MPEG-2系统环境中，防止产生可以解释为MPEG-2起始码的比特模式。

(2) 6bit的NAL_Type，允许NAL单元的类型编码数比H.264/AVC多一倍，达到64类，其中32类用作VCL NAL单元，32类用作non-VCL NAL单元。

(3) 6bit的Layer_ID,NAL头的第1字节的最后1bit和第2字节的前5比用于HEVC的扩展层标识符。

(4) 3比特的TID,temporal_id，表示HEVC的接入单元（AU）属于那个时域子层，时域标识符的值为0到6。

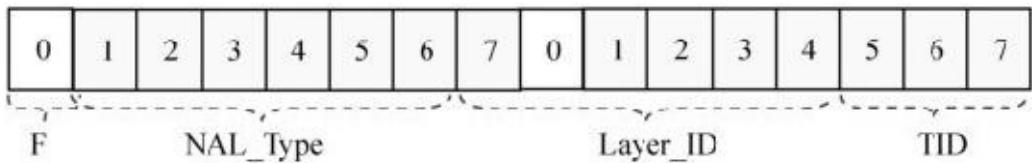


图10.6 HEVC的双字节NAL单元头结构

在HEVC的多层扩展中，Layer_ID为层识别信息，表示当前NAL单元属于哪一层。但在HEVC的第1版中，对所有的NAL单元都将layer_id设置为“000000”，兼容HEVC版本1的解码器将忽略那些layer_id不等于“000000”的NAL单元。在HEVC的可分级、3D或MV扩展中，层标识符layer_id描述除开时间维（HEVC第1版已经支持时间维分级）以外所有的多层次设置。例如在3D扩展中，layer_id将标注视点和深度，而在可分级扩展中，它将用于联合标注空间和质量可分级层。

NAL单元头中的最后3个比特是NAL单元的时域标识符，表示7种可能的情况（0~6,7禁止使用），HEVC中每个接入单元（AU）属于某个时域子层。因为每个AU都属于一个时域子层，属于同一AU的所有NAL单元在自己的头部必须标注相同的TID号。

2.NAL头的语法表

NAL单元头部的语法如表10.3所示。其中nal_unit_type说明了包含在NAL单元中的RBSP数据结构的类型，如表10.4所示，包括留待今后使用的类型在内总共有64种。nal_unit_type的值为41~63所表示的数据类型，HEVC予以保留或尚没有规范其具体语义和相应的解码过程，它们所表示的NAL单元类型将可能由不同的应用来决定，但这并不影响现在HEVC规

范的解码过程。今后可能将这些NAL单元类型用于不同的目的。如果不涉及这些应用，解码器将忽略所有使用nal_unit_type保留值的NAL单元的内容，即将它们从比特流中移出并丢弃。NAL单元头部的nuh_layer_id定义了VCL NAL单元所属的层标识符，或者non-VCL NAL单元所使用的层标识符。nuh_layer_id的范围为0 ~ 62,63保留为今后扩展使用。

表10.3 NAL单元头语法

| nal_unit_header[1] | 描 述 符 |
|-----------------------|-------|
| forbidden_zero_bit | u(1) |
| nal_unit_type | u(5) |
| nuh_layer_id | u(5) |
| nuh_temporal_id_plus1 | u(3) |
| } | |

3.NAL单元的类型

HEVC的NAL单元根据是否装载有编码视频或其他相关数据而分别形成VCL NAL单元和non-VCL NAL单元两类，每一类又包含32小类。HEVC标准中还包括了为解码器初始化和随机接入目的而识别图像分类的若干VCL NAL单元类型，保留用于今后的类型。在表10.4中列出了HEVC中NAL单元的种类、相关的内容及VCL分类。

表10.4 NAL单元的类型

| 类型 | 名 称 | 子 类 | 分 类 | |
|-------|----------------|--------------------|-------------------|--|
| 0 | TRAIL_N | non-TSA/STSA 接入图像 | VCL | |
| 1 | TRAIL_R | | | |
| 2 | TSA_N | 时域子层接入 (TSA) 图像 | 后置图像 | |
| 3 | TSA_R | | | |
| 4 | STSA_N | 分层时域子层接入 (STSA) 图像 | VCL | |
| 5 | STSA_R | | | |
| 6 | RADL_N | 随机接入或解码 (RADL) 图像 | VCL | |
| 7 | RADL_R | | | |
| 8 | RASL_N | 跳过随机接入前置 (RASL) 图像 | 前置图像 | |
| 9 | RASL_R | | | |
| 10~15 | RSV | 保留的 non-IRAP 图像 | VCL | |
| 22~23 | RSV | 保留的 IRAP 图像 | 保留 | |
| 24~31 | RSV | 保留的 non-IRAP 图像 | | |
| 16 | BLA_W_LP | 断点连接接入 (BLA) 图像 | VCL | |
| 17 | BLA_W_RADL | | | |
| 18 | BLA_N_LP | | | |
| 19 | IDR_W_RADL | 即时解码刷新 (IDR) 图像 | 帧内随机接入及 (IRAP) 图像 | |
| 20 | IDR_N_LP | | | |
| 21 | CRA | 纯随机接入 (CRA) 图像 | | |
| 32 | VPS_NUT | 视频参数集 (VPS) | non-VCL | |
| 33 | SPS_NUT | 序列参数集 (SPS) | | |
| 34 | PPS_NUT | 图像参数集 (PPS) | | |
| 35 | AUD_NUT | 接入单元限制 | | |
| 36 | EOS_NUT | 序列结尾 | | |
| 37 | EOB_NUT | 比特流结尾 | | |
| 38 | FD_NUT | 填充数据 | | |
| 39 | PREFIX_SEI_NUT | 补充增强信息 (SEI) | | |
| 40 | SUFFIX_SEI_NUT | | | |
| 41~47 | RSV | 保留 | non-VCL | |
| 48~63 | UNSPEC | 未定义 | | |

10.4 HEVC的接入图像

如前所述，视频编码层的NAL单元包括保留的在内共有64种类型，用0~63不同的值标注在NAL单元头信息中。例如，如果一个接入单元（AU）的所有VCL NAL单元的类型编号为21，那么这个接入单元就称为纯随机接入（CRA）单元，对应的编码图像就称为CRA图像。从整体编码视频序列出发，HEVC的接入图像大致可分为3种基本类型：

- (1) 帧内随机接入点（IRAP, Intra Random Access Point）图像；
- (2) 前置（leading）图像；
- (3) 后置（trailing）图像。

这三类接入图像中每一类又可分为若干子类型图像，图像类型和它们的子类型已经在HEVC的NAL单元类型中予以标识，下面予以简单介绍。

10.4.1 帧内随机接入图像

在许多视频应用中，常常需要在两个（或多个）视频序列之间进行切换或拼接。例如广播视频和网络视频，用户需要的一个重要功能就是能够在不同的码流之间切换，以最小的延时跳至视频流的特定部分。在以往的标准中，为了达到这一目标，将视频序列划分为若干GOP，每个GOP中以IDR帧开头作为视频流的切换点。HEVC这一功能是由在视频流中等间隔地插入IRAP图像来实现的，IRAP将视频流分为若干编码视频序列（Coded Video Sequence, CVS），相当于先前的GOP。在码流中，IRAP是可以开始正确解码的图像，因此可以用作切换点图像，从这一点切换，可保证随后的图像得以正确解码。参与切换的2个视频流，可以是两个实时视频流，也

可以是两个存储视频流，还可以一个是实时、另一个是存储视频流。所有的IRAP图像都属于时域子层0，而且一定是帧内图像，它的编码不需要参考任何其他图像，但帧内图像却不一定IRAP图像。

在HEVC码流中，可以有多个IRAP图像，但码流的第一幅图像必须是IRAP图像。由于IRAP图像比其他常规的编码图像耗费的比特要多，因此，插入IRAP图像的密度需要根据应用的要求来设定，总的原则是尽可能地稀疏，以增加编码效率。HEVC的IRAP图像包括IDR图像、BLA图像和CRA图像三个子类。

1.IDR图像

即时解码刷新（IDR）图像是在H.264/AVC中定义的接入点图像。在H.264/AVC中，IDR帧一定是帧内编码图像（I帧），而且一定位于图像组（GOP）的开始。但是反之却不一定成立，I帧不一定是IDR帧，一个GOP中可以有多个I帧。GOP的长度也不是一定的，在H.264/AVC的编码器中，如果判定场景发生变化，那么即使不到原定GOP的末尾，也会在这个位置加入一个IDR，作为新一个GOP的开始。H.264/AVC中所有的GOP都是独立解码的，与其他GOP无关，即它们都是“封闭”的，称为是闭合GOP。

HEVC的随机接入点沿用了H.264/AVC的IDR图像，它又可细分为两类，一类是和相应前置（leading）图像有关联的IDA图像，另一类则是和相应前置图像没有关联的IDR图像。有关leading图像随后介绍。IDR图像是帧内编码图像，切断了后续图像对此IDR前面已解码图像的联系，在解码顺序中跟随IDR图像后面的图像不能使用先于IDR图像的已解码图像作参考。这样，依赖于IDR图像作随机接入的比特流的编码效率会显著降低（有文献报道降低约为6%）。为此，HEVC不仅沿用了IDR图像用于随机接入，还增加了BLA和CRA两类接入点图像，以提高随机接入码流的编码效率。这样一来，就不像在H.264/AVC码流中必定包含IDR图像，在HEVC兼容比特流中有可能完全不包含IDR图像。

2.CRA图像

为了提高编码效率，HEVC新定义了一种位于随机接入点（RAP）图

像——纯随机接入（CRA）图像。它处于比特流GOP的第一帧，即RAP图像位置，是I帧，和IDR一样可被独立解码。但是，HEVC的GOP采用了“开放”的结构，称为开放GOP，在解码过程允许参考其他GOP中的图像数据。CRA图像都是和相应前置图像有关联的图像。

HEVC允许在解码顺序中跟随CRA图像后面，但在输出顺序中位于CRA前面的图像使用CRA图像前面的已解码图像作为参考图像，而且这样的纯随机接入（CRA）功能还可用作IDR图像。如果在CRA图像处进行随机接入，则在解码顺序和输出顺序都跟随在CRA图像之后的图像都可解码，以此可确保纯随机接入后码流的顺利解码。也就是说，解码器可以从CRA处开始顺利地解码图像，而不需要解码任何比特流中比CRA出现更早的图像。从压缩效率的角度看，它比插入IDR图像将编码视频序列分为两段编码视频序列更加有效。

围绕一幅CRA图像的一个典型的预测结构如图10.7所示。将码流顺序划分为一个个图像结构（Structure of Picture,SOP），一个SOP定义为一幅或多幅在解码顺序中的相继排列的编码图像，图像的相对解码顺序号码显示为图像内部的下标。在先前SOP中的任何一幅图像的解码顺序号都比当前SOP中任何一幅图像的解码顺序号小，以后SOP中的任何一幅图像的解码顺序号都比当前SOP中任何一幅图像的解码顺序号大。

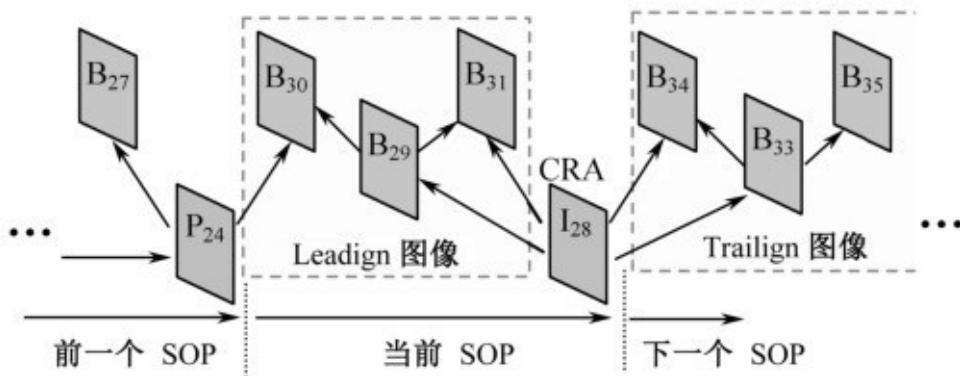


图10.7 CRA图像和leading图像

图10.7中的图像输出顺序号是按照顺序从左到右逐渐增加排列的，但没有具体标出。标注在图像中的下标数字为解码顺序的编号。一个SOP中，在解码顺序中的第一幅编码图像是一幅最低时域子层的参考图像，而且也是这个SOP中唯一的RAP图像。

CRA图像I₂₈所属的当前SOP还包含图像B₂₉、B₃₀和B₃₁，它们在解码顺序中紧随CRA图像，但是在输出顺序中却在CRA前面。因此，这些图像称为CRA图像的前置（leading）图像。由于B₂₉和B₃₁不参考解码顺序中CRA之前的任何图像，当解码从CRA图像I₂₈开始，它们能够被正确解码，因此是可解码随机接入前置（Random Access Decodable Leading,RADL）图像。

考虑到码流切换，如果解码从当前CRA图像I₂₈的前一SOP中的参考图像P₂₄开始，图像B₃₀就可被正确解码，因而是一幅RADL图像。如果随机接入从当前SOP的CRA图像I₂₈开始，B₃₀图像就不能被正确解码，因而是一幅

跳过随机接入前置（Random Access Skipped Leading, RASL）图像，在随机接入解码期间往往就被丢弃。

如果 I_{28} 是一幅 IDR 图像，解码器开始解码 I_{28} 之前就需要清空 DPB，即清除前一个 SOP，因此 B_{30} 也没有可能利用 P_{24} 作为参考图像。若使 I_{28} 成为 CRA 图像， B_{30} 就可能利用 P_{24} 。这样一对比，就会发现，CRA 方式在随机接入方面要优于 IDR，可以利用更多的 B 帧，而码流中 B 帧数量的增加就意味着编码效率的提高。

为防止来自不可能依赖于解码开始的参考图像的错误传播，在图 10.7 中下一个 SOP 的所有的在解码顺序和输出顺序中都紧随 CRA 其后的图像，将不能使用无论是解码顺序还是输出顺序在 CRA 图像之前的任何图像（包括 leading 图像）作为参考图像。

3.BLA 图像

HEVC 引入了断点连接接入（BLA）图像来完成对不同视频码流的拼接。只要简单地把后续码流中一个 IRAP 图像的 NAL 单元类型改变为 BLA 标记值，串接到前面码流中 IRAP 帧的位置成一个新的码流，就可以完成码流拼接的工作。IRAP 图像可以是 IDR、CRA、BLA 图像，CRA 和 BLA 的后面都可能跟随着 RASL 图像（取决于用于 BLA 的 NAL 单元类型的标记值）。解码器必须抛弃 BLA 图像之后的任何 RASL 图像，因为它们可能包含的码流中因拼接操作而实际不存在的参考图像。例如，希望将另一个视频流的某一 CRA 后的码流拼接到当前视频流某个 CRA 位置后继续解码，则直接将后续码流中拼接点的 CRA 标记为 BLA 接到当前码流的 CRA 位置即可。由于 BLA 之前解码到缓存的视频流与当前视频流无关，因此其特性类似于直接从该点进行随机存取后的 CRA。

当从 CRA 图像开始的比特流包含另一个比特流时，RASL 图像连同 CRA 图像都不能被解码，因为它们的某些参考图像不在整个组合比特流中。为了使这样的拼接操作更直接，可改变 CRA 图像的 NAL 单元类型，标明它为一幅 BLA 图像。这时 RASL 图像连同 BLA 图像往往不能被正确解

码，因此也不能顺利输出和显示。

10.4.2 前置图像

图10.7的一例已经给出，在当前SOP中，图像B₂₉、B₃₀和B₃₁，为CRA的前置图像（Leading Pictures,LP），一般来说，在某一GOP中，凡是解码顺序在IRAP图像前、输出顺序在IRAP图像后的那些图像都称为前置图像（一幅或多幅）。从前面那一例也可看出，前置图像根据解码过程的不同，又可包括可解码随机接入前置（RADL）图像和跳过随机接入前置（RASL）图像两类。

（1）可解码随机接入前置（RADL）图像

在SOP中，一类解码顺序在IRAP图像之后，而显示顺序在IRAP之前的前置图像（如图10.7中的B₂₉和B₃₁），叫作可解码随机接入前置（RADL）图像，因为这种图像的解码不需要参考任何解码顺序在IRAP之前的图像。

（2）跳过随机接入前置（RASL）图像

在SOP中，某些解码顺序在IRAP图像之后，而显示顺序在IRAP之前的前置图像（如图10.7中的B₃₀）有可能包含帧间预测操作，需要参考解码器中不存在的图像。于是解码器只能跳过这些无法解码的图像继续解码处理。称这些不可解码的图像为跳过随机接入前置（RASL）图像。

10.4.3 后置图像

仍然参考图10.7，在一个SOP中，解码和显示顺序都在IRAP之后的图像叫做后置图像（Trailing Pictures,TP）。后置图像必须使用后置图像NAL单元类型0~5号中的一个。在帧间预测中，IRAP图像的后置图像不允许依赖于任何前置图像和前一IRAP的后置图像作预测；而只能依赖于自己关联的IRAP图像和同一IRAP关联的后置图像。这是因为前置图像（LP）

中的RASL图像是有可能舍弃的，因此后置图像不能参考这些数据，否则万一这些图像被舍弃，将会有更多的图像受其影响而不能正常解码。另外，所有的IRAP的前置图像必须在解码顺序中超前所有的和同一IRAP关联的后置图像。这意味着，关联图像的解码顺序始终先是IRAP图像，再是关联的前置图像（如果有），最后是关联的后置图像（如果有）。

后置图像包括三类，时域子层接入（TSA）图像、分步时域子层接入（STSA）图像和一般后置（TRAIL）图像，它们都能够用于标志“时域子层的切换点”，即指不依赖于相同子层中超前（解码顺序）自己的图像。

在视频流的网络传输中，一个网络节点可以对多个子层编码的HEVC比特流进行处理。由于较低子层中的图像解码过程不依赖于较高子层中的VCL NAL单元，因此较低子层中的图像可以独立解码。基于这一原因，在比特流中任何一点，网络节点可以从此开始丢弃较高子层的NAL单元，这一操作称为“时域下切换”。时域下切换不受图像类型的限制。相反的情况是，网络节点从某一切换点开始处理较高层NAL单元的动作称为“时域上切换”。如果在切换目标层中没有任何一幅图像依赖于同一层中的、早于比特流中切换点的图像，时域的上切换才可能实现。

1.时域子层接入（TSA）图像

时域子层接入（TSA）图像是后置图像的一种，它表示一个时域子层切换点。在某一子层中，如果这幅图像是TSA图像类型，则在同一时域子层或更高的子层（时域ID大于或等于TSA的时域ID）中，没有解码顺序超前此TSA的图像需要用此TSA图像或相继的图像作帧间预测。

如图10.8所示，水平轴表示输出顺序，垂直轴表示时域子层，下标数字表示解码顺序，箭头表示帧间预测。其中，P₆图像可以用作TSA类型图像，因为只有在时域子层0（TS0）中的前一帧图像可用于TSA图像本身以及相继的图像的预测（解码顺序）。

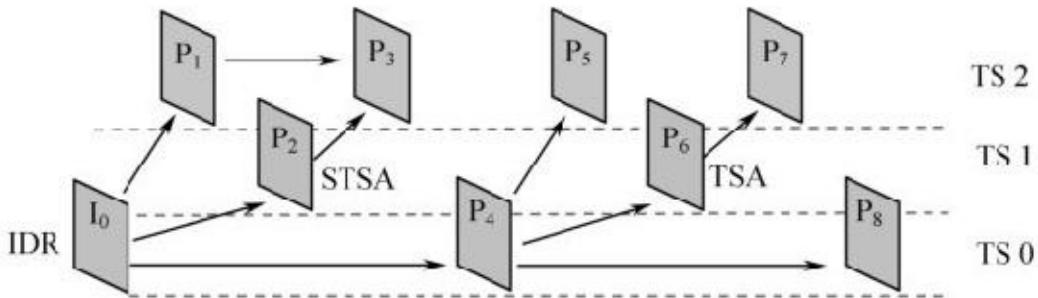


图10.8 3个时域层的编码结构一例

当解码器正在解码比特流中一个时域子层图像时，遇到其中的TSA图像，其子层值恰恰高于正在解码的最大时域子层，解码器有可能切换到并解码任何数目的附加时域子层。例如，在图10.8中，一个解码时域子层0的解码器能够从TSA图像仅保持解码时域子层0，或者决定开始解码时域子层1以及子层0，或者开始解码所有的3个子层。

对于网络节点，例如，由于先前的网络拥塞情况，解码器就可以采取类似的动作，向最低的时域子层切换。网络节点能够方便地检查和解析输入图像的NAL单元类型，并不需要很多的通信资源，因为NAL单元类型和时域ID可以在NAL单元头信息中找到。当遇到时域子层1的TSA图像，网络节点可以切换到任何时域子层中跟随TSA后的图像（解码顺序），而不用担心解码器不能正确解码它们，因为前面没有它们所依赖的参考图像。

2.分步时域子层接入（STSA）图像

分步时域子层接入（STSA）图像类型类似于TSA图像类型，但它只保证STSA图像自己、同一子层中跟随在此STSA之后图像（解码顺序）不参考相同子层超前STSA的图像。图10.8中的STSA图像的一个例子是P₂。这

幅图像不能是TSA图像，因为 P_3 要参考 P_1 。但是，图像 P_2 可以是一个STSA图像，因为 P_2 不需要参考子层1的任何图像，也不需要参考在解码顺序中跟随在 P_2 后面的任何子层1的图像。TSA和STSA图像的时域ID值都必须大于0。

需要注意的是，在比特流中标注STSA图像的位置，可以切换到和STSA相同的子层；而在比特流标注成TSA图像的位置，有可能向上切换到更高的子层。由于HEVC是禁止从高到低的时域子层的预测，因此，无论是什么类型的图像，也不管处于哪个时域子层的图像，向下切换到更低的时域子层总是允许的。

3.一般后置（trail）接入图像

“一般后置”图像可以属于任何时域子层，用TRAIL标注其类型。一般后置图像可以参考相关联的IRAP图像以及与此IRAP相关联的后置图像，但它们不能参考前置图像，或者与此IRAP相关联的其他非后置图像。它们也不能在解码顺序中下一个IRAP图像后输出。按照三种后置图像的定义，所有的TSA和STSA图像都可以标注为TRAIL图像，所有的TSA图像都能够标注为STSA图像。然而，为了标明比特流中所有可能的时域子层切换点的特点，应该严格标注后置图像的类型。

10.5 HEVC的参数集

10.5.1 三类参数集

在H.264 /AVC以前的视频编码标准中，各个层次的视频编码数据往往是和该层的管理数据（或称头数据）在码流中顺序传输的。如一幅图像第一个包丢失，不仅丢失它所携带的第一幅图像部分数据，还有图像头数据（有时还有GOP和序列头数据），即使所有其他的包都没有丢失，也将导致完全错误的重建图像（有时还影响随后的图像）。如果带有图像头信息的包丢失了，某些解码器实现将不再去解码接收到的图像数据包。

为防止这类头信息丢失对解码图像引起的严重影响，克服头信息的脆弱性，以往的有些标准，例如 H.263，引入了基于传输层的防御机制，将一幅图像划分为若干部分（如 slice），然后这些部分各自为传送单元，采用定义在RFC2429中的RTP载荷格式（如RTP包）来传输，允许在众多选定的数据包中携带一个图像头的冗余复制。

后来在H.264/AVC标准开发中，已经认识到图像头消息的脆弱性是视频编码自身的一个结构性问题，而不是一个传输问题，因此引入了参数集概念来解决这一问题。将原本属于序列和图像头部的大部分语法元素从视频编码数据中分离出来，集中组成序列参数集（SPS）和图像参数集（PPS），可对它们加以更强的传输保护，防止序列头和图像头丢失所产生的毁灭性影响。

参数集的概念有助于构建编码视频序列的简明、分等级的高层信息结构，服务于多个条分割（SS）、多幅图像、或者编码视频序列。参数集信息和编码 slice数据相分离，按照各个等级参数的需要，采用专门的保护，改进了在有损传输条件下编码视频序列的差错鲁棒性。在实际的传输方式

上，这类专门组织成的参数集信息，既可以和视频数据在同一比特流中传输，也可以通过其他方式，如可靠的带外信道、硬复制等方式传输。

1.三类参数集

HEVC在继承H.264/AVC参数集概念的基础上，做了不少修改和添加，例如，增加和修改了不同编码工具所需要的模式，新引入了视频参数集（VPS）概念。这样HEVC共计定义了3类数据集：视频参数集（VPS）、序列参数集（SPS）和图像参数集（PPS）。这些参数集中，SPS和PPS在H.264/AVC中已经存在了。

HEVC新引入视频参数集（VPS）的目的就是集中提供有关编码视频序列的不同的层和子层的信息。而对于目前定义的单层档次（Main, Main Intra, Main Still Image），这个参数集的使用是有限的，对HEVC标准的扩展却是非常有用的，例如，用于空间和质量可分级编码，用于多视点编码等。

为了增加对误码和丢包的鲁棒性，每个参数集都是自包含的，即它可以被解码而无须参考NAL单元流中的任何其他NAL单元。对于VPS和SPS，最重要的信息在开头处是定长编码的，以便直接、简单地存取和解析。为了达到差错恢复的目的，也可以在比特流中发送同一参数集的多份复制。为避免冲突，在NAL单元流中的共享同一个参数集ID的参数集需保证所有的语法元素的值相同。同时，所有已经编码的参数集必须是有效的参数集，即要服从所有的包含在规定的档次和水平内指标的限制。

每一类参数集都包含着扩展机制，允许在将来的HEVC版本中扩展这个参数集，而不会破坏后向兼容性。

2.参数集的激活

在解码过程中，三个参数集必须在解码器在解码第一个slice之前可用。PPS只能在图像的开始时激活，此后PPS在整个图像期间保持激活。VPS和SPS只在新的编码视频开始时被激活。因此，如果需要修改这些参数集中的参数，必须要在新的编码视频序列的开始时进行。在NAL单元流传输中，一个参数集NAL单元的出现标志了一个新的接入单元的开始。

参数集的激活和H.264/AVC类似，是一种层次参考过程，如图10.9所示。SS头包含一个对PPS的参考，依次而行，PPS包含一个对SPS的参考，SPS包含一个对VPS的参考。编码图像的第一个SS头激活一个PPS，随后的SS必须参考一个具有相同内容的PPS。然而，同一编码视频序列的不同的图像可以激活带有不同参数集的不同的PPS。PPS激活一个SPS，这个SPS按次序激活相应的VPS。允许多个PPS指向同一SPS，多个SPS指向一个VPS，以获得关于编码视频序列不同类型的有效、灵活的表示。

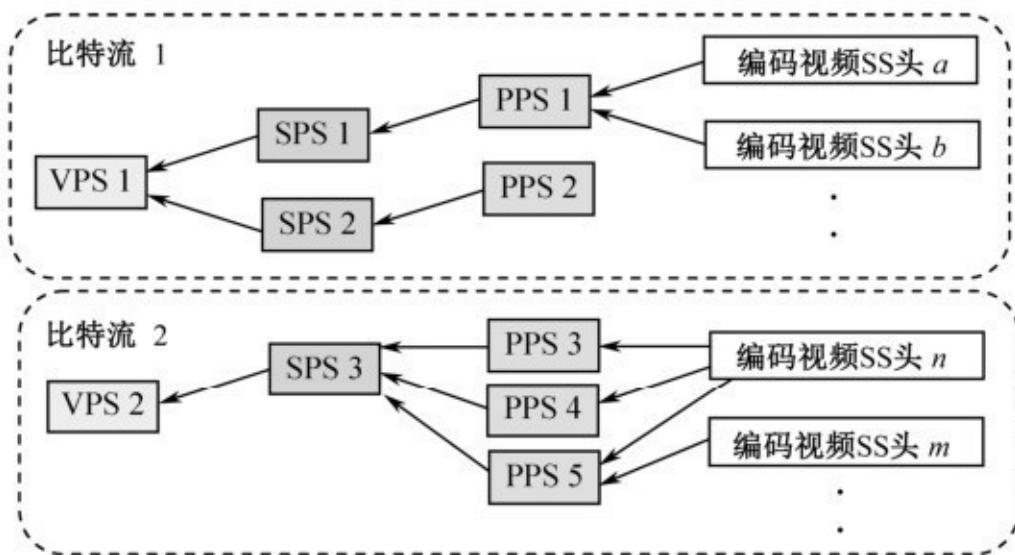


图10.9 三级参数集的激活

参数集的一个共同的实现策略就是：将一个给定类型的所有的参数集保存在相应的表格中，它们的最大尺寸由参数集ID的编号范围间接定义。在这种实现策略下，参数集的激活很简单，由 slice 头中的信息查找 PPS 表格，复制所发现的信息到相应的解码器数据结构中，按照 PPS 中的参考找到相关的SPS，再按照SPS中的参考找到相关的VPS。因为这些操作负担并不重，每一幅图像仅需要完成一次（最坏的情况）操作。

10.5.2 视频参数集（VPS）

在HEVC中，除继续使用SPS和PPS两类参数集外，还新增了视频参数集（VPS）。引入视频参数集（VPS）是为了传送应用于多层和子层视频编码所需的信息。它提供了有关编码视频序列的全局性信息，包括一个或多个层的存在、可用的操作点信息，例如在这一点可以完成从给定的比特流中提取子比特流。VPS 的引入为设计简洁、可扩展的多层视频编码器提供了方便。

由于在H.264/AVC中没有包含一个兼容的参数集，不少信息（例如 H.264/AVC 可分级扩展中的可分级性信息的大部分 SEI 消息）必须在序列参数集（SPS）中被重复，在某些应用场合需要在带外传输，因此造成了初始延时的增加，特别是在可靠性传输和带外传输中常需要重传时更是如此。相同信息的重复会带来相当大的传输负担，因为参数集在每个随机接入点为了接入和切换信道都需要重复。HEVC引入VPS后基本可以解决这些问题。

由于VPS是最高层参数集，所有直接或间接参考VPS的参数集必须满足这个VPS的约束集。一个VPS只能够在编码视频序列的开头激活，即在一个IDR或BLA接入单元，或CRA接入单元，它是编码视频序列的第一个接入单元。

1.VPS的主要信息

一个给定的视频序列的每一层都参考同一个VPS，无论它们是否有相

同或不同的SPS，这个VPS传递的信息包括：

- (1) 为避免不必要的复制，由多层或多操作点共享的共同语法元素；
- (2) 操作点之间会话谈判所需的基本信息，例如档次、水平等；
- (3) 其他操作点定义的信息，它不属于某个 SPS，例如，层或子层所需的理想参考解码器（Hypothetical Reference Decoder, HRD）参数。

2.VPS的语法表

VPS的基本语法元素如表10.5所示（HEVC版本2的7.3.2.1节），由一套（前7条）语法元素开始，共4个字节，在规定的位置上采用定长编码。因此，这些语法元素可以便捷地被解码器或外部应用存取。这一套语法元素包括VPS识别符，相关的可用层及可用时域子层的信息等。SPS用VPS识别符来激活这个VPS。在HEVC版本1中，可用层的数目被固定置为“1”表示单层序列。在HEVC的可分级和多视点扩展中，可用层的数目可分别标明可用的（空间或质量）可分级或视点的数目。可用时域子层的数目对应于时域标识符 t_{id} 的可用值。对时域子层，还需标明时域嵌套的性质。

跟随着前4个字节后的语法元素结构包括：比特流中解码器工作的可用操作点的信息。一个操作点的特性是由使用的档次、等级（tier）和水平来刻画的。它定义了解码比特流所需要的编码工具，以及对比特流尺寸或缓存容量的限制。跟在整个比特流的操作点指标后面的是比特流时域子层的操作点指标。需要再次强调，将这些语法元素定义为固定长度的码字，使得编码信息的连续字节排列可以方便解码器或外部应用存取。

表10.5 视频参数集（VPS）语法

| | |
|--|-------|
| video_parameter_set_rbsp() { | 描述符 |
| vps_video_parameter_set_id | u(4) |
| vps_base_layer_internal_flag | u(1) |
| vps_base_layer_available_flag | u(1) |
| vps_max_layers_minus1 | u(6) |
| vps_max_sub_layers_minus1 | u(3) |
| vps_temporal_id_nesting_flag | u(1) |
| vps_reserved_0xffff_16bits | u(16) |
| profile_tier_level(1,vps_max_sub_layers_minus1) | |
| vps_sub_layer_ordering_info_present_flag | u(1) |
| for(i=(vps_sub_layer_ordering_info_present_flag ? 0 : vps_max_sub_layers_minus1); | |
| i <= vps_max_sub_layers_minus1; i++) { | |
| vps_max_dec_pic_buffering_minus1[i] | ue(v) |
| vps_max_num_reorder_pics[i] | ue(v) |
| vps_max_latency_increase_plus1[i] | ue(v) |
| } | |
| vps_max_layer_id | u(6) |
| vps_num_layer_sets_minus1 | ue(v) |
| for(i = 1; i <= vps_num_layer_sets_minus1; i++) | |
| for(j = 0; j <= vps_max_layer_id; j++) | |
| layer_id_included_flag[i][j] | u(1) |
| vps_timing_info_present_flag | u(1) |
| if(vps_timing_info_present_flag) { | |
| vps_num_units_in_tick | u(32) |
| vps_time_scale | u(32) |
| vps_poc_proportional_to_timing_flag | u(1) |
| if(vps_poc_proportional_to_timing_flag) | |
| vps_num_ticks_poc_diff_one_minus1 | ue(v) |
| vps_num_hrd_parameters | ue(v) |
| for(i = 0; i < vps_num_hrd_parameters; i++) { | |
| hrd_layer_set_idx[i] | ue(v) |
| if(i > 0) | |
| cprms_present_flag[i] | u(1) |
| hrd_parameters(cprms_present_flag[i], vps_max_sub_layers_minus1) | |
| } | |
| } | |

续表

| | |
|----------------------------|------|
| vps_extension_flag | u(1) |
| if(vps_extension_flag) | |
| while(more_rbsp_data()) | |
| vps_extension_data_flag | u(1) |
| rbsp_trailing_bits() | |
| } | |

VPS中前4字节后的剩余语法元素的一部分不再是固定长度编码，而是用可变长度编码，以获得有效的表示。这些语法元素包括有关缓存和子层

延时的信息，一般定时信息，以及理想参考解码器（HRD）的参数等。

10.5.3 序列参数集（SPS）

不同于VPS关注整个比特流，描述的是编码视频序列一般特征的高层参数，SPS定义的是用于编码视频序列的特征和激活工具，其信息仅用于由层标识符Layer_ID定义的层。在HEVC第1版中，仅允许Layer_ID=0，因此在这种情况下，将VPS中有关比特流的某些全局性信息简单地复制到SPS中。如同VPS，SPS在编码视频序列的开始被激活一次，开始处可以是一个IDR、一个BLA、或一个CRA图像，它们都是随机接入单元的第一幅图像。SPS所包含的内容大致可分为以下几类。

（1）SPS自己的ID识别符，时域子层的数目，标明时域嵌套的开始标志。这些信息位于SPS的第一个字节，采用定长编码。

（2）同VPS中的描述的解码操作点相关的语法元素。如档次、水平、图像尺寸、子层数等。为可分级和多视点扩展，定义了分别用于层和视点的SPS操作点信息，用于每一层或每一视点的SPS可以被分别激活。

（3）该档次中视频特征和编码工具的参数。视频特征包括图像的彩色格式、比特深度以及编码图像的取样分辨率等。应用工具的参数包括最小和最大编码树块尺寸，即CTB参数，最小和最大TB尺寸，变换树等级、深度，帧内和帧间块等。还有包括短期和长期参考图像集，各种编码工具的激活标志等。

（4）视频可用信息（VUI）。SPS可以包含视频可用信息（VUI）。VUI参数提供了有关编码视频内容的详细信息，包括彩色空间、色度位置、高宽比等。VUI还包括HRD的参数。再次强调，对于单层编码，这些和VPS中的参数相同，但当用于HEVC扩展时则有可能不同。

具体的SPS的语法元素表格由于篇幅的关系没有列出，可参考第2版HEVC文档7.3.2.2节。

10.5.4 图像参数集 (PPS)

如同名称表示的那样，PPS包含用于一幅图像的参数。相应地，一个PPS可在图像层面激活，而不是在序列层面，而且激活的PPS可以从一幅图像到另一幅图像而改变。同一幅图像中所有的slice必须参考同一个PPS。

PPS包含的信息大致可和H.264/AVC的PPS中一部分相当，主要包括：

- (1) 用于CABAC的熵编码指标参数，环路滤波器的指标参数。
- (2) 起始图像控制信息，如初始量化参数 (QP)、偏移、伸缩列表等，在 slice 头中的特定工具或控制信息的使用标志，依赖性slice的激活和波前并行处理等。
- (3) 对当前图像的tile划分参数，由于HEVC定义的结构，序列中的tile划分可以从一幅图像到另一幅图像改变，不变的tile划分的应用可以标明在VUI中。

具体的PPS的语法表格可参考第2版HEVC文档的7.3.2.3节。

10.6 HEVC的参考图像集

在视频编码中，为了提高帧间编码的效率，必须为帧间预测提供参考图像。在早期的视频标准中，如H.261只有简单的前帧预测，因此只需参考刚解码好的前帧图像即可，无需设立专门的参考图像集。后来随着编码技术的发展，如 H.264/AVC 出现了双向预测、多达16幅参考图像的多帧参考、灵活参考图像等，参考图像的数量和类型都有所增加，对参考帧的要求也趋复杂，继而出现了对各类参考图像合理安排和管理的参考图像集概念。

与H.264/AVC类似，HEVC中的参考图像集定义了将先前已解码的图像用作参考图像，规范了如何在解码图像缓存（Decode Picture Buffer,DPB）中对它们进行管理，如取样数据预测和运动矢量预测等。

10.6.1 参考图像集

在DPB中的图像可以被标注为“用于短期参考”“用于长期参考”或“不用于参考”。一旦一幅图像标注为“不用于参考”，它就不再用于预测，如果它也不需要输出时，就可以从 DPB中删除它。HEVC对于参考图像管理不同于先前视频编码标准，它取消了对DPB中参考图像变化的标注，而是将DPB的状态标注在每个slice中。HEVC对参考图像管理的目标就是使所有兼容标准的比特流和解码器具有基本的差错鲁棒性。

1. 图像顺序

HEVC定义了一个理想参考解码器（HRD）模型，描述了在解码端的已编码图像缓存（CPB）和一个已解码图像缓存（DPB），如图10.10所示。

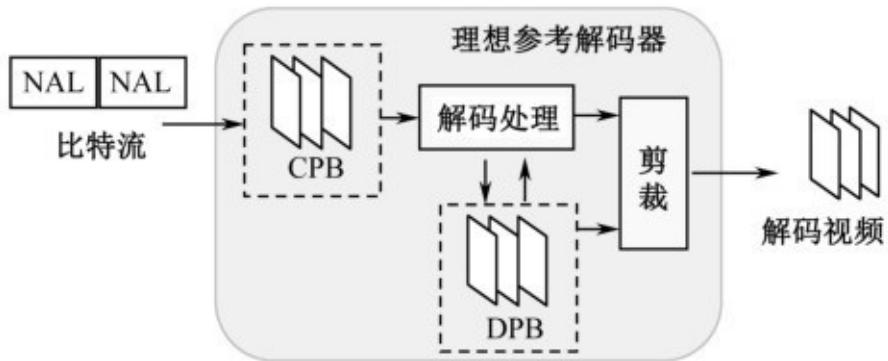


图10.10 HRD的缓存模式

在HEVC编解码系统中，已编码图像的解码顺序和已编码图像出现在比特流中的顺序相同；已解码图像的输出顺序不同于图像的解码顺序。通常，每幅图像都和一个图像顺序计数（Picture Order Count, POC）值相关联，POC表示它的输出顺序。

下面通过一个简单的示例来说明在视频编码中常见的几种图像顺序的概念，从左至右表示时间从过去到将来的进展方向，每个号码表示拍摄时的帧号。

| | | | | | | | | | | | | | | | |
|------|--------|---|-------|---|-------|---|-------|---|-------|----|--------|----|----|----|--------|
| 拍摄顺序 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | |
| 采集顺序 | 1 | | 3 | | 5 | | 7 | | 9 | | 11 | | 13 | | (隔帧采样) |
| 编码顺序 | 1 (I) | | 9 (P) | | 5 (B) | | 3 (B) | | 7 (B) | | 11 (I) | | | | |
| | 19 (P) | | | | | | | | | | | | | | |
| 码流顺序 | 1 | | 9 | | 5 | | 3 | | 7 | | 11 | | 19 | | |
| 解码顺序 | 1 | | 9 | | 5 | | 3 | | 7 | | 11 | | 19 | | |

| | | | | | | | | | | | | | | | |
|-----------|---|---|---|---|---|----|----|-------|---|----|----|----|----|----|-------|
| 输出顺序 | 1 | 3 | 5 | 7 | 9 | 11 | 13 | | | | | | | | |
| 显示顺序 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | |
| (内插偶数帧) | | | | | | | | | | | | | | | |

2.RPS及其标注

HEVC引入了一种新的参考图像管理方法，称为RPS。RPS必须为每个特定的slice提供一个完整的参考图像集，供当前图像或后续图像使用。为此，需标注一个完整的保存在DPB中的所有图像的集合。

RPS标注的一个实例如图10.11所示。在解码顺序中的第一幅图像是一幅IDR图像 I_0 ，没有给它标注RPS，因为它是解码序列中的第一幅图像，因此解码顺序在IDR前没有图像可用作IDR图像的参考，或用作解码顺序中跟随IDR图像后的任何图像的参考。在解码顺序中的第二幅图像为 P_1 ，使用 I_0 作为参考。因此，在它的RPS中必须包括 I_0 。随后的图像 B_2 使 I_0 和 P_1 作为参考，因此它们都包含在 B_2 的RPS内。

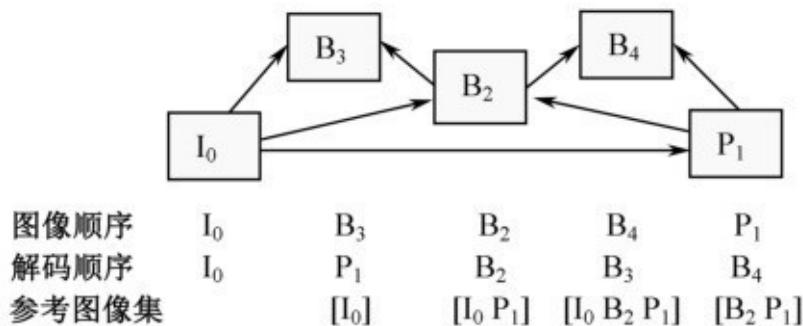


图10.11 RPS标注的实例

图像 B_3 使用 I_0 和 B_2 作为参考，所以它们都包含在 B_3 的 RPS 内。但还必须包含 P_1 ，因为这幅图像将被用于后面图像 B_4 的参考。最后，图像 B_4 将使用 B_2 和 P_1 作为预测。需注意， B_4 的 RPS 不包含 I_0 。因为 I_0 没有列入，它将被标注为“未用于参考”。这意味着 I_0 不能被用于 B_4 的参考，或用于在解码顺序中任何跟随 B_4 后面图像的参考。

3. 解码顺序和DPB操作

为了充分利用 RPS 的优点和增强差错弹性，HEVC 中的图像解码顺序及 DPB 操作和 H.264/AVC 并不相同。在 HEVC 中，首先解码来自当前图像的 slice 头的 RPS，接着在解码当前图像之前进行图像标注和缓存操作。如图 10.12 所示，由于引入了 RPS 的概念，HEVC 中基于 slice 头及以上的语法元素的解码过程也较 H.264/AVC 简单。

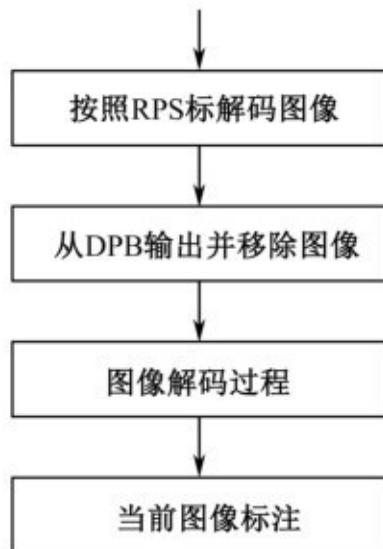


图10.12 HEVC解码顺序

对包含slice的图像，HEVC要求每个slice头必须包含标识RPS的参数。唯一例外是对IDR slices，推断RPS为空。对不属于IDR图像的I slice，即使它们属于一幅I图像，也有可能要提供一个RPS，因为在解码顺序中，可能有图像跟随在I图像后，它们采用帧间预测时要利用解码顺序中处于I图像前面的图像。在RPS中的图像数目将不超过 DPB 尺寸的限制，这个限制由 SPS 中sps_max_dec_pic_buffering语法元素确定。

每幅图像都和一个POC值相关联，它表示输出顺序。slice头包含一个定长码字，pic_order_cnt_lsb，表示完整POC值的最低比特，即熟知的POC LSB。这个码字的长度标注在SPS中，范围为4 ~ 16bit。RPS设计使用POC 来识别参考图像。除了它自己的POC值，每个slice头直接包括或者从SPS继承RPS中每个图像的POC值的编码表示。

每幅图像的RPS组成5个不同的参考图像列表，也称为5个RPS子集。

(1) RefPicSetStCurrBefore 组成所有的短期参考图像，它们在解码顺序和输出顺序中都先于当前图像，可以用于当前图像的帧间预测。

(2) RefPicSetStCurrAfter组成所有的短期参考图像，它们在解码顺序中先于当前图像，而在输出顺序中接在当前图像之后，可以用于当前图像的帧间预测。

(3) RefPicSetCurrFoll 组成所有的短期参考图像，它们可以用于在解码顺序中跟随当前图像之后的一幅或多幅图像的帧间预测，但不能用于当前图像的帧间预测。

(4) RefPicSetLtCurr组成所有的长期参考图像，它们可用于当前图像

的帧间预测。

(5) RefPicSetLtFoll 组成所有的长期参考图像，它们可以用于解码顺序中跟随在当前图像之后的一幅或多幅图像的帧间预测，但不可用于当前图像的帧间预测。

最多可使用3重不同类型的参考图像来标注RPS:POC值比当前图像低的短期参考图像，POC 值比当前图像高的短期参考图像和长期参考图像。另外，给每个参考图像一个标注 (used_by_curr_pic_X_flag)，以标明这一幅参考图像是被用作当前图像的参考（包括列表RefPicSetStCurrBefore、列表RefPicSetStCurrAfter或列表RefPicSetStCurr中的一个），或者没有用作参考（包括列表RefPicSetStFoll或列表RefPicSetLtFoll中的一个）。

在图10.13的例子中，当前图像B₁₄确实包含5个RPS子集的每个中的一幅图像：

P₈在RefPicSetStCurrBefore列表中，因为它在输出顺序中在B₁₄前，并被B₁₄使用。

P₁₂在RefPicSetStCurrAfter列表中，因为它在输出顺序中在B₁₄后，并被B₁₄使用。

P₁₃在RefPicSetStFoll列表中，因为它是短期参考图像，没有被B₁₄使用（但必须保留在DPB中，因为他被B₁₅使用）。

P₄在RefPicSetLtCurr列表中，因为它是长期参考图像，被B₁₄使用。

I₀在RefPicSetLtFoll列表中，因为它是长期参考图像，没有被当前图像使用（但需要保留在DPB中，因为它被B₁₅使用）。

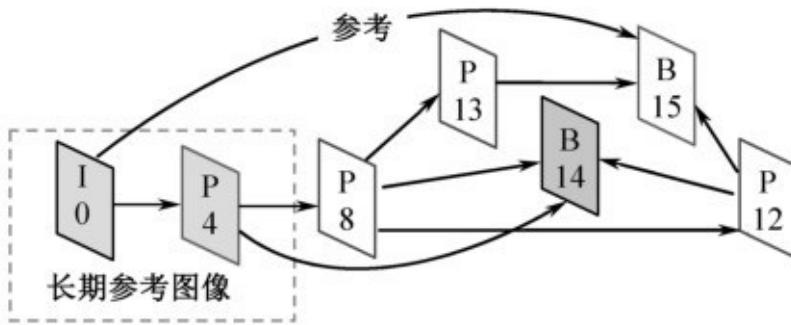


图10.13 B14的参考图像类型一例

在图像解码前，存在于DPB中的图像通常都有编号。其中一部分图像可用于预测，标注它们为“用于参考”。另一些图像是不可用于预测，只是等待输出，标注它们为“不用于参考”。当slice头已经被解析，在slice数据被解码时应实施一个图像标注过程。存在于DPB中标注为“用于参考”但不包含在RPS中的图像被标注为“不用于参考”。如果used_by_curr_pic_X_flag=0，则没有存在于DPB中，但包含在RPS中的图像将被忽略。然而，如果used_by_curr_pic_X_flag=1，则这个参考图像可用于当前图像的预测，而不是丢弃。

4.RPS的差错弹性

由于当前图像使用的所有参考图像必须包括在RPS内，因此就可以避免这样的风险：因一个或多个数据包丢失而导致不能检测到不正确的参考图像的使用。一旦在 RPS 中有 used_by_curr_pic_X_flag置1的参考图像（表示可以用于当前图像预测），但是在DPB中没有相应的参考图像，则解码器就会知道一个非故意的参考图像丢失已经发生了。在特殊的应用中，解码器端能够对丢失图像的检测有恰当的反应，如产生一个掩盖图像，或通

过反馈信道向编码端报告这个丢失。如何处置图像丢失不属HEVC规范的范围。

10.6.2 参考图像列表

在HEVC中，帧间预测表示从参考图像的数据（如取样值或运动矢量）导出的预测，而不是从当前解码图像导出。将用于帧间预测的参考图像组织在一个或多个参考图像列表中，类似H.264/AVC，一幅图像可以由多幅参考图像来预测。而参考索引则指明列表中参考图像集的哪一幅图像应当被用于产生预测信号。

单个参考图像列表List0用于P slice，而2个参考图像列表List0和List1用于B slice。与H.264/AVC类似，HEVC中的参考图像列表构建包括参考图像列表初始化和参考图像列表修改两个步骤。

1. 参考图像列表的初始化

在H.264/AVC中，List0的初始化过程是不同的，用于P slice参考图像时使用解码顺序，用于B slice的参考图像使用输出顺序。而在HEVC中，两种情况下List0都使用解码顺序。

如果是 B slice，参考图像列表初始化基于3个 RPS 子集产生默认的 List0 和 List1:RefPicSetStCurrBefore、RefPicSetStCurrAfter 和 RefPicSetLtCurr。输出顺序较早（较迟）的短期图像，按照POC距离的上升顺序（相对当前图像）首先插入List0（List1）；然后是输出顺序较迟（较早）的短期图像，按照 POC 距离的上升顺序（相对当前图像），插入到 List0（List1）；最后，将长期图像插入到List0尾部。对List0，借助于 RPS，将RefPicSetStCurr Before 中的表项插入到初始列表中，随后插入的是 RefPicSetStCurrAfter 中的项目。再后来，如果存在，再将RefPicSet LtCurr 中的表项填入。

在HEVC中，当列表中表项的编号小于激活参考图像的目标编号（标志在PPS或slice头中）时，上述过程是重复的，已经加入到参考图像列表的

参考图像再次被加入。当表项的编号大于目标编号时，则列表被截断。

2. 参考图像列表的修改

参考图像列表被初始化以后，它可以被修改，使当前图像的参考图像能以任意顺序安排。例如，一幅特别的参考图像可能出现在列表中的任意位置，甚至多个位置，这取决于参考图像列表修改指令。

当标明是否存在列表修改指令的标记被置1，则标注一个固定的指令的数目（等于参考图像列表中表项的目标数），每条指令为参考图像列表插入一个表项。由 RPS 信令中导出的当前图像的参考图像列表的索引来识别指令中的一个参考图像。这不同于H.264/AVC中参考图像列表修改：由图像的编号（从framre_num语法元素导出）或者由长期图像参考图像的索引来识别一幅图像，有可能需要很少的指令，如交换起始列表中前2项，或者在起始列表开头处插入一项并移动其他项。

10.7 HEVC的SEI和VUI

HEVC 在附录中提供了支持不同类型的元数据的补充增强信息（ SEI ）和视频可用信息（ VUI ）语法，这些数据提供了有关视频图像定时信息，用于视频信号的彩色空间的参数，3D立体帧的打包信息，其他用于显示的隐含信息等。

10.7.1 补充增强信息（ SEI ）

1.SEI消息

HEVC 的附录 D 的补充增强信息（ SEI ）机制为视频编解码器在比特流中提供了一些必要的元数据。所谓元数据不是为正确解码输出的图像取样值，而是用于其他目的数据，例如用于图像输出定时、显示、差错检测和掩盖等目的。

HEVC规范SEI消息的目的就是保证在不同的HEVC系统中对补充数据有唯一的解释。在一个接入单元中，编码器可以包含任意数目的SEI NAL单元，每个SEI NAL单元可以包含一个或多个SEI消息。SEI消息分为前缀SEI消息和后缀SEI消息两类，分别表示SEI必须在接入单元中最后VCL NAL单元前出现，或者在第一个VCL NAL单元后出现。

HEVC标准包含了若干SEI消息的语法和语义，但是并没有定义SEI消息的处理，因为它们并不影响正常的解码过程。SEI 信息对于解码器判定编码视频序列的某些特征是很有用的，否则这些信息的导出可能非常复杂。

2.SEI的主要内容

HEVC中的SEI机制和若干SEI消息是从H.264/AVC继承来的。定义在

H.264/AVC中的SEI消息的一个子集也被定义到HEVC中了。

SEI消息具有保留功能，它标明对哪些NAL单元，包含在SEI消息中的信息对它是有效的。SEI信息可以为某个接入单元而保留，也可以为全部编码视频序列保留，或者直到新的相同载荷类型的SEI消息提供为止。多个SEI消息可以组合成SEI消息NAL单元。每个SEI消息总是由整字节数组成，可由一个SEI载荷类型索引和一个SEI载荷尺寸来识别。HEVC第2版附录D中的表D.1提供了主要的SEI消息，并简要地列出了设置它们的目的。考虑到篇幅限制，这里未列出，仅给出几个SEI消息的示例：

图像结构信息的SEI消息，提供编码层视频序列中的图像集合的结构信息。

图像定时SEI消息，为隔行视频内容在HEVC中使用提供保证。

解码图像HashSEI消息，包含一个从和图像有关的解码样值导出的校验和，可以进行差错检测。

缓存周期SEI信息，提供子图像（Sub-Picture）定时SEI消息，为HRD操作中子图像提供移除时间，应用于极低延时情况，如远程屏幕共享。

激活数据集SEI消息，对激活视频和序列参数集提供的识别。

10.7.2 视频可用信息（VUI）

视频可用信息（VUI）收集了对解码视频输出和显示的有用信息，定义在HEVC的附录E中。VUI能够作为SPS的一部分，也能够通过其他方式传送到解码器（如带外传输），它是不需要解码处理的。VUI中提供的信息包括图像比例、彩色格式、图像定时信息等。在VUI语法结构中包含的信息是可选，由应用需要来决定。在未提供VUI参数的情况下，可设定参数的默认值。

1. 视频特征信息

（1）几何关系

标明的几何关系的信息包括图像样点高宽比（Sample Aspect

Ratio,SAR) 和结束扫描标记。VUI在HEVC中预定义了一套SAR表格（参见附录E的表E.1），表中不同的SAR由相应的索引值来标注，索引值为1~16的SAR具有预定义值，17~224为保留值，最大索引值225用作SAR的扩展。

由于 SAR 类型定义了样点的几何特性，它决定了编码图像的宽度和高度绝对值的几何关系。这个关系通知显示过程应当如何处理（可能要再取样）解码图像，在目标显示器件上提供无几何失真的视频表示。如果结束扫描标记出现，编码图像数据可能包括的图像边界左边（和/或右边）的区域就可不用去显示。

（2）类型和色度格式

视频信号类型消息，包括原图像格式的标识，如分量方式、PAL 制、NASC 制、SECAM制或MAC制视频等。

彩色格式信息，包括亮度及对应色度信号的格式指标，如彩色参数、转换特性、彩色转换矩阵系数等。还包括相对于亮度样点位置的彩色样点位置的标注。

（3）帧/场标识

HEVC规范在视频编码层没有提供专门用于隔行视频处理的工具，解码过程对场表示的图像和帧表示的图像没有什么不同。虽然如此，还是提供了基于场序列的处理和基于帧序列的处理。如果标明是基于场的序列，编码视频序列的每幅图像每表示为一场，需要将图像定时 SEI消息随每个接入单元发送，说明相应的图像是表示顶场还是底场。

2.显示和定时信息

（1）显示窗

SPS包括一个常规裁剪窗的规范，当从DPB中输出时，这个窗口就用于解码图像。在VUI中，一个附加的窗口可定义在常规裁剪窗口区域内，如果没有其他指示给出，该窗口就表示所建议的显示区域。

（2）定时信息

定时信息对于保证解码视频序列的正确的播放输出速率是特别重要

的。由于解码处理的指标不包括时间概念，所以这个过程本身不提供定时信息。作为VUI的一个重要部分，HRD参数的语法结构被放入到定时信息部分。

所提供的定时信息包括设置一个定时机制所必要的参数。为达此目的，定义了一个基本时钟率和时钟“滴答”(tick)的长度。在HEVC标准中给出一例，一个工作在 $f_c=27\text{MHz}$ 时钟、帧率为25Hz的视频信号，即图像帧之间的时域距离为 $t_p=0.04\text{s}$ 。这个帧率可用 $n_{tu}=1080000$ 时间单元表示，因为

$$t_p = \frac{n_{tu}}{f_c} = \frac{1080000}{27000000\text{Hz}} = 0.04\text{s}$$

实际上由上式可以看出， n_{tu} 表示在两帧的0.04s之间相当于1080000个27MHz的时钟周期（一个周期为一个tick）。

相应地，例如30Hz的序列可以由 $n_{tu}=900000$ 来表示。定时信息还包括一个标志，表明POC正比于图像的输出时间，在一定条件下，图像输出定

时也能够从POC直接导出。注意，定时信息也可以在VPS中提供，但要求在VPS和VUI中的定时信息完全相同。

3.比特流限制信息

VUI的最后一部分是比特流限制信息，具体包括以下几部分。

(1) 固定tile结构

如果UVI标明的是固定tile结构，则解码器能够在整个序列中按照此标明的tile结构处理，例如，在多线程环境中达到为tile安排线程的目的。

(2) 运动矢量的图像边界限制

如果标明运动矢量的图像边界限制，则运动矢量不引入位于图像区域外的预测块，解码过程不需要进行图像边界填充。例如，在某些应用场景，多个比较小的视频序列合并成一个大的边靠边的视频，这一限制使得在小图像的边界不需要特殊的处理。

(3) 参考图像列表限制

如果标明参考图像列表的限制，图像中的所有的slice不仅被迫使用相同的参考图像集，且还需要使用相同的参考图像列表。

(4) 最小空间分割

最小空间分割标识规定了解码器分割图像区域的尺寸，这个区域能够独立地作为一个空间分割（在一个slice中的CTU的数目）来解码。此信息可用于确定解码器对给定视频序列解码进行并行处理的数目。

(5) 图像字节数和CU比特数

每幅图像的最大字节数标识给出了编码一幅图像的VCL NAL单元的字节数的最大动态范围。每个CU的最大比特数提供了CU层面上的数据量范围。这一信息能够用于更精确地刻画编码图像缓存的操作，以及解码器中解析和解码过程的操作。

(6) 最大运动矢量长度

在水平和垂直两个方向的最大运动矢量长度的标识，用于对最大运动矢量长度值（定义在所用的档次、水平和等级中）的限制。这一信息可用于估计解码器所需要的计算负担。

本章参考文献

- [1]Rickard Sjöberg,Ying Chen,Akira Fujibayashi,et al.Overview of HEVC high-level syntax and reference picture management[J].IEEE Trans.on Circuits and Systems for Video Technology.2012,22 (12) : 1858-1870.
- [2]James Nightingale,Qi Wang,Christos Grecos.HEVStream: A Framework for streaming and evaluation of High Efficiency Video Coding (HEVC) content in loss-prone networks[J].IEEE Tran.on Consumer Electronics,2012,8 (2) : 404-412.
- [3]Vivienne Sze,Madhukar Budagavi,Gary J.Sullivan.High Efficiency Video Coding (HEVC) :Algorithms and Architectures[M].Switzerland Springer International Publishing,2014.
- [4]Mathias Wien.High Efficiency Video Coding: Coding Tools and Specification[M].Springer-Verlag Berlin Heidelberg,2015.
- [5]Jakub Stankowski,Damian Karwowski,Tomasz Grajek,et al.Bitrate distribution of syntax elements in the HEVC encoded video[C].International Conference on Signals and Electronic Systems (ICSES 2014) ,1-4.
- [6]G.Correa,A.Assuncao,L.Volcan,et al,Fast HEVC encoding decisions using data mining[J].IEEE Trans.on Circuits and Systems for Video Technology,2015,25 (4) : 660-673.
- [7]G.J.Sullivan et al,Overview of the High Efficiency Video Coding (HEVC) standard[J].IEEE Transactions on Circuits and Systems for Video Technology,2012,22 (12) : 1649-1668.
- [8]High Efficiency Video Coding[S],ITU-T Rec.H.265 and ISO/IEC 23008-2,ITU-T and ISO/IEC JCT-VC,Mach

2013 (v.1) ,Oct.2014 (v.2) ,Apr.2015 (v.3) .

第11章 HEVC的多层和可分级编码扩展

展

目前，HEVC标准的扩展（extension）工作主要集中在以下几个方面。

（1）范围的扩展，扩大标准支持的比特深度的范围和彩色取样格式，包括对高质量编码、无损编码和屏幕内容编码等。

（2）可分级性的扩展，能够使用嵌入式比特流子集作为视频内容的不同比特率的表示。

（3）立体视频的扩展，形成3D 视频和多视点视频的表示，提供新的立体视频编码工具，如深度图编码和视点合成技术等。

（4）多层编码的扩展，将可分级编码和立体编码等统一看作普通单层视频HEVC编码的多层扩展。

第一项有关编码的范围扩展（Range Extensions, RExt）并没有形成一个独立的文档或专门的附件，而是体现在HEVC第2版多处的增加和修改中，主要包括档次种类的增加和图像格式种类的增加。如HEVC第2版的附录A中“A.3.5 Format range extensions profiles”和“A.3.6 Format range extensions high throughput profiles”等，增加了多种不同的视频格式和比特深度。我们知道，HEVC在第1版中仅定义了3种不同的档次，即“Main”“Main10”和“Main Still Picture”，目标应用场合为视频广播、网络视频、视频会议和媒体存储等。这3个档次都是对4:2:0的彩色格式进行操作，限制像素的比特深度为8bit或10bit，只支持一种时域可分级性编码。

第2版HEVC进行了扩展，定义了24种新的档次，支持更高的比特深度，如12bit、16bit，支持更高分辨率的彩色取样格式，如4:2:2和4:4:4。

以上4项标准扩展中，除第3项有关立体视频内容外，其他各项任务仍然由开发HEVC的JCT-VC完成。为了完成制定HEVC的MV扩展、3D扩展等多个立体视频标准的任务，2012年7月ITU-T的VCEG和ISO/IEC的MPEG建立了第二个联合协作小组，即JCT-3V（Joint Collaborative Team on 3D Video Coding Extension Development），此后立体视频编码扩展标准的制定就由JCT-3V承担。

以上4项扩展标准中，第2、3、4项扩展已经正式纳入HEVC第2版的附录H、附录G和附录F中，第3项扩展中的3D编码已呈现在最新2015年的第3版HEVC的附录I中。本书的第11章和第12章简要介绍这4个扩展的一些基本情况。

本章首先简要介绍HEVC视频编码扩展的进程，然后从总体上介绍多层视频编码扩展，最后介绍可分级视频编码扩展的基本原理、主要方法和语法表示。

11.1 HEVC编码扩展的进程

视频编码标准HEVC在2013年已经成为正式的国际标准，称之为HEVC第1版。和H.264/AVC相比，它在缩减50%编码率的情况下可提供相同可视质量的解码视频。目前，产业界正在致力于HEVC的实现和应用。例如对超高清电视（Ultra High Definition Television,UHDTV）编码的商用服务已经开始，今后还将对能够提供更丰富细节的4倍于UHDTV的视频采用HEVC编码。与此同时，ITU-T和ISO/IEC的有关组织JCT-VC、JCT-3V等对HEVC标准扩展的研究和开发也在加紧进行，目前已推出了以立体视频和可分级视频为代表的多项扩展标准。

2014年7月，第18次JCT-VC会议，第9次JCT-3V会议，第109次MPEG会议和ITU-T的SG16会议，完成了HEVC的第2版，获得ITU-T通过。2014年10月获得ISO/IEC的通过。第2版HEVC，即ITU-T的H.265建议，ISO/IEC的国际标准23008—2，包含4个重要的扩展。

第1个扩展是多层视频编码，在HEVC第2版中作为附录F（Annex F），即“Common specifications for multi-layer extensions”，对HEVC扩展中的共同的和“多层”编码有关的语法、语义、参数和编码工具进行定义和规范。

第2个扩展是多视点视频编码，在HEVC第2版中作为附录G（Annex G），即“Multiview high efficiency video coding”，简称MV-HEVC（Multi-View HEVC），提供高效的多视点立体视频编码处理的系统结构和编码工具。

第3个扩展是可分级视频编码，在HEVC第2版中作为附录H（Annex H），即“Syntax,semantics and decoding processes for scalable extension”，简称SHVC（Scalable HEVC）。它是为了适应IP网络的异构、波动和拥塞而

采用的一种高效、强鲁棒性的可分级性（scalability）视频编码技术，支持嵌入式可分级比特流。

第4个扩展是3D视频编码，目前尚未正式完成，估计即将包含在下一版的HEVC中，可能作为附录I（Annex I），即“3D high efficiency video coding”，简称3D-HEVC。2014年10月的第2版HEVC尚未包含整个对3D视频编码的扩展，估计3D-HEVC在2015年中期完成，目前最新的草案为“3D-HEVC Draft Text 7”，编号为JCT3V-K1001-v3。

这些扩展所提供的编码工具都在位于基本层（Base Layer, BL）之上的增强层，实际上就是BL用HEVC标准编码，而HEVC扩展则编码BL的上面的增强层（Enhancement Layer, EL），以提供更高的分辨率或质量、3D 显示或更高的比特范围。因此，一个 HEVC 兼容的解码器能够解码BL，而HEVC扩展解码器解码相应的EL。

这里有两个压缩立体视频的 HEVC 扩展，一个是 MV-HEVC，另一个是3D-HEVC。MV-HEVC采用与HEVC标准编码不同的视点，并且引入了视点间预测，以改进率失真性能。其中主视点表示为BL，用HEVC标准编码，可采用帧内或帧间预测，不采用视点间预测。其他的视点表示为EL，并且允许视频的3D 表示。除对不同视点编码外，3D-HEVC扩展采用新的工具来编码和每个视点相对应的深度图。

这4个扩展中，多层次视频扩展和SHVC扩展的开发仍然由JCT-VC承担，但MV-HEVC扩展和3D-HEVC扩展则由新成立的JCT-3V承担。

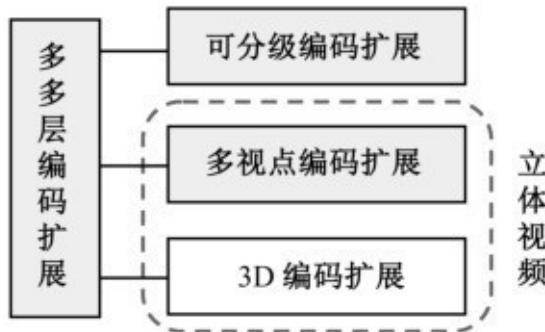


图11.1 HEVC的4个编码扩展

本章主要介绍HEVC的这4个扩展。这4个扩展之间并不完全是相互并列的关系，它们之间的联系可以用图11.1来简单表示，其中灰色部分的3个扩展已经颁布在HEVC第2版中，“3D编码扩展”也已颁布在2015年的HEVC第3版中。

按照一般的标准化思路，HEVC的每一个扩展都可以制定为一个标准，但为了防止标准数太多，JCT-VC/3V没有这么做，而是将这些扩展制定为HEVC的几个附录。打开每个附录后你都可以看到，这些附录的结构、目录和HEVC正文非常相似，一个附录本质上也是一个标准。尽管这些附录的结构、目录和HEVC正文类似，但其中大部分条款是空的，仅标注和HEVC的正文或×附录的××条款相同，说明这些扩展之间有很多共同之处，单独形成标准会大大增加标准内容的冗余。因此，只将HEVC正文中没有或不同的部分，列出相关条文，详细给出具体的规范，形成一个精练的附录，既保证“附录”起到“标准”的作用，又不失内容的简练。

除上述4个已经完成或即将完成的扩展外，HEVC对屏幕内容编码（Screen Content Coding, SCC）扩展、高动态范围（High Dynamic

Range,HDR) 扩展正在开发中。 SCC扩展的提案征集从2014年1月开始 , 工作草案第2版和测试模式第1版已经完成 , 2015年将完成提议草案和草案 , 估计在2016年早些时候可完成最后草案和表决。 SCC主要用于包括显示特性、文本、动画的视频 , 当然也包括摄像机捕获的视频场景。 HEVC的高动态范围 (HDR) 扩展目前尚处在提案征集的讨论阶段。

11.2 HEVC统一的多层编码

在多层编码扩展中，将普通的平面视频编码看作单层视频编码，进而将可分级视频编码、立体视频编码等看作单层平面视频编码基础上的扩展，即抽象为多层视频编码。例如在可分级视频编码中，将一个基本层和若干增强层看成一个多层视频系统；在多视点编码中，一个视点对应一个层，将一个独立视点和若干依赖视点看作一个多层视频系统；将3D视频编码中的纹理视频和深度图分别对应多层系统中一个层。这样，将多层编码系统中各层必须遵守的公共规范集中在一起就形成了HEVC的多层视频编码扩展。

11.2.1 多层编码的结构

1. 层标识符

在多层视频编码中，出于兼容性的考虑，每一层本身的编码都尽量沿用HEVC的编码方式，为使解码器能够识别和解码不同的层，必须在NAL单元头信息中设置不同层的标识。在编码视频码流中，每一层在NAL单元头信息中都编址有唯一的层标识符Layer_ID。NAL单元头信息中的语法元素Layer_ID长度为6bit（见第10章图10.6的6bit的层标识符Layer_ID），原则上允许编址64种不同的层。其中最小值Layer_ID=0的NAL单元用于表示兼容第1版HEVC档次的比特流。最大值Layer_ID=63保留，拟用于今后潜在的扩展，例如，如果编码的视点数超过允许数目的范围。

为规定比特流上下文中 Layer_ID的语义解释，对视频参数集（VPS）做了扩大。对于多视点视频编码，Layer_ID用于表示被编码的视点索引。对于可分级视频编码，Layer_ID用于表示层间依赖性标识，由此标识来确

定层间预测的依赖关系。VPS 语法结构还保证了今后对 Layer_ID解释的扩展，包括联合视点索引和依赖性标识的 Layer_ID。其实，这种设计还可允许更多的灵活性，例如用于标识将可分级编码和多视点编码相结合的新的多层编码方式。但这样的选择在当前的扩展标准中还没有进一步开发。

2.多视点编码的层

在多视点编码中，视频序列的不同的视点由Layer_ID的赋予的值来表示。通常，在多视点视频序列中，将多摄像机中的某个摄像机的视频序列分配给某一个视点，并赋予该视点一个标识符Layer_ID值。一般将Layer_ID=0的视点认作基本视点，或独立视点，其他视点则认作依赖视点。

3.可分级编码的层

有关时域可分级方面的内容，已经在HEVC的有关部分中提供了，这里不再多叙。对空域和保真度（质量）可分级编码，比特流包含多个可分级层，它们提供增加重建保真度，或者增加的图像分辨率所需要的信息。在可分级编码的上下文中，所有层被分为基本层（BL）和增强层（EL）两类。基本层和HEVC第1版兼容。对应于Layer_ID=0的基本层表示码流中的最低可用层，将它和多个增强层组织为等级方式，高等级的层可以依赖也可以不依赖于低等级的层。

4.参考层和目标层

一般说来，将非基本层中的那个用于预测的层认作参考层。由参考层重建的图像认作参考图像。相应的，将输出的层认作目标层。

HEVC多层编码扩展还定义了直接参考层和间接参考层之间的差别。直接参考层就是其他层可由它来预测的层。如果直接参考层自己也是从另外一个层预测的，则该层相对于当前目标层而言即为间接参考层。某一层如果作为参考层使用，必须在VPS中标识。但对于可分级编码，只允许有一个目标层（一般是最高等级层），而多个目标层在多视点编码中是可以的。

11.2.2 多层编码的工具

在HEVC的Annex F中详细介绍了多层视频编码扩展的共同的语法、语义和解码过程，包括定义多层比特流和各层含义所需的修改和扩展。这里，简要给出两种编码工具和高层语法的要点。

1. 层间预测

层间预测是多层视频编码系统中最重要的编码工具之一，也是在单层编码中所没有的工具。在多层比特流中，不同层之间的预测是通过参考图像集和参考图像列表来处理的。为便于预测，直接将层间参考图像放在目标层图像的参考图像列表中。因此，可获取重建样点阵列以及与参考层相关的运动信息、划分信息，将它们用于层间预测。参考图像列表构建过程的细节已定义在多视点和可分级编码的附录中。

多层编码扩展附录还包括CTB解码过程，相对HEVC基本层并没有变化。除对参考图像列表修改、和其他层必要的集成外，每一层都可被基本的HEVC解码器解码。这样设计是为了便于对现存HEVC解码器实现多层扩展。

层间预测所包含的参考图像是有限制的，只允许来自相同接入单元中的其他层图像，不允许来自其他接入单元的图像。这和可分级视频编码中的常见方法是一致的，相同图像比较低的层可用于目标层的预测。在多视点编码中，层间预测也有限制，所允许的是同一视点的时域预测，或者是同一输出时间的视点间预测。这种限制实际上是一种在复杂度和高效性之间的折中考虑。相同的限制实际上也已用于H.264/AVC的多视点视频编码中了。

2. 辅助图像

以常规方式编码的图像被标注为基本图像，以此区别于辅助图像。由定义可知，辅助图像对基本图像的解码过程没有影响。辅助图像的概念已经存在于先前的视频编码标准中，例如H.262/MPEG-2、MPEG-4 Visual、H.264/AVC等。如MPEG-4中的阿尔法(α)平面编码，它们可用于在混合

图像应用中标明图像的透明区域。

辅助图形也可用于立体视频的深度图编码，为基本图像（纹理图像）提供深度信息。一幅辅助图像由Layer_ID的赋值来标记，不同的赋值以及这些值的语义解释定义在VPS中。因此，辅助图像的处理和基本图像的处理可以无缝地集成在一起。

目前，多层扩展只定义了阿尔法图像和深度图为辅助图像。扩展标准允许定义总量达256种不同类型的辅助图像。而大多数的类型索引处于保留状态，留给今后ITU-T和ISO/IEC使用，还有16个类型索引处于未定义状态。这些类型的解释已超出目前规范的范围，因此可能留用于其他场合。不同类型的辅助图像并不需要和基本图像关联起来处理。例如，这样就允许对来自于不同信源的图像以辅助图像的方式进行编码。但阿尔法平面则不同，因为它一般是在基本图像上的一种范围表示信息，因此需要将阿尔法平面辅助图像和对应的基本图像联系起来处理。在3D 视频编码的环境中，深度图作为一种辅助图像可用于基本图像的预测过程。因此，这种辅助图像也需和基本图像进行关联处理，使之自适应于3D视频编码。

11.3 HEVC的多层扩展

HEVC的附录F是为多层编码扩展制定的一个共同的高层语法规范，包括可分级、多视点和深度图层，以保证扩展的语法和语义支持不同类型层的组合。附录F大量涉及的是和HEVC不同的语法和语义的细节，限于篇幅，这里仅简单提及几个和HEVC多层编码扩展密切相关的概念。

11.3.1 层和子层

HEVC的NAL单元设计规则和H.264/AVC基本一致，但具有不同的头信息长度，使用2字节的NAL单元头。HEVC扩展并未改变NAL单元头的结构，但起用了其中表示NAL单元类型的6bit信息（Layer_ID），用于表示层ID值的语法元素。

在HEVC中，定义“层”为一个NAL单元的集合，这些NAL单元的头信息中具有相同的层ID值（Layer_ID）。一个“层”可能表示某一规格（如分辨率、SNR、视角、纹理、深度）的视频，以后“层”还可能表示视频场景的某些增强特性。在HEVC第1版中，层ID值的6个比特必须等于0，表示这是基本层，其他ID值则表示各个高层。对于某一层中的不同时域帧率的帧，则定义它们为时域“子层”，同一子层使用相同的层ID。

定义在HEVC及其扩展中的子比特流提取和H.264/AVC的SVC、MVC中的子比特流提取功能类似。一般将可分级参数的目标值作为输入，而兼容的子比特流则是输出，它仅包含基于可分级参数的目标层和子层。

11.3.2 接入单元

在多层视频编码中，为了包含多层扩展，修改了原来为单层比特流定义的若干规范，具体可参见AnnexF的F.3。最关键的修改是关于编码图像和接入单元的定义。在多层编码上下文中，每一幅编码图像都具有一个给定的Layer_ID值。在HEVC扩展中，来自在同一时刻的所有层的图像组成一个接入单元（AU）。一个接入单元（AU）包含了所有具有相同输出时间的编码图像，以及所有关联的VCL和non-VCL的NAL单元。因此，多个编码图像包含在多层接入单元中。为了便于识别，在一个接入单元中的编码图像都是和相同的图像顺序号（POC）值相联系的。

11.3.3 视频参数集扩展

HEVC定义了一个新的视频参数集（VPS），用于整个视频序列中所有层的系统接口、能力交换和子比特流提取。一个VPS包括基本VPS和VPS扩展两部分。基本VPS包含和HEVC第1版兼容层的信息，同时还包含这些层对应的操作点信息。VPS扩展则包含基本层以上的增强层信息。

视频参数集（VPS）的扩展不仅包含了在解码视频序列中所有层之间的存在和相互关系的描述，还包含了每一层的高层特征描述。对于基本层，VPS需要使它的层标识符Layer_ID等于0。给顶层的属性作为最高可用参数集，VPS将它们用于组合在比特流中所有的层。对每个层，所分配的序列和图像参数集可以是动态变化的。SPS和VPS两者包含的内容都是可变的，它们用于增强层NAL单元，以保证从一个参考层导引出相关的编码参数。SPS还包含了参考层和目标层空间分辨率之间关系的描述，以及参考层图像相对于目标层图像的位置。因此，完整地定义了在不同层中图像的几何关系。这在空域可分级编码中，还必须适当地规范参考层的再取样过程。在多视点编码中，以及仅在保真度可分级编码中，图像分辨率对所有层保持不变。

HEVC对理想参考解码器（HRD）也作了扩展，为的是支持整个码流的缓存模式以及某些层的专用缓存模式。除定义整个比特流的编码图像缓

存外，还定义了仅保存比特流中部分相关层的比特流部分缓存。

11.4 可分级视频编码

网络的异构、用户终端的差异等因素，使得原来面向存储和电路交换的视频压缩算法，已经很难满足现代IP网络，特别是无线网络的实时传输要求。为了解决这些问题，人们提出了种种办法和改进措施，其中可分级视频编码（SVC）就是一种比较成熟有效的方法。为此，多年前JVT为可分级编码制定了H.264/AVC标准的SVC扩展；近年来，JCT-VC已制定和正在完善HEVC的可分级编码（SHVC）扩展。图11.2形象地展示了空间可分级编码的一种应用场景。

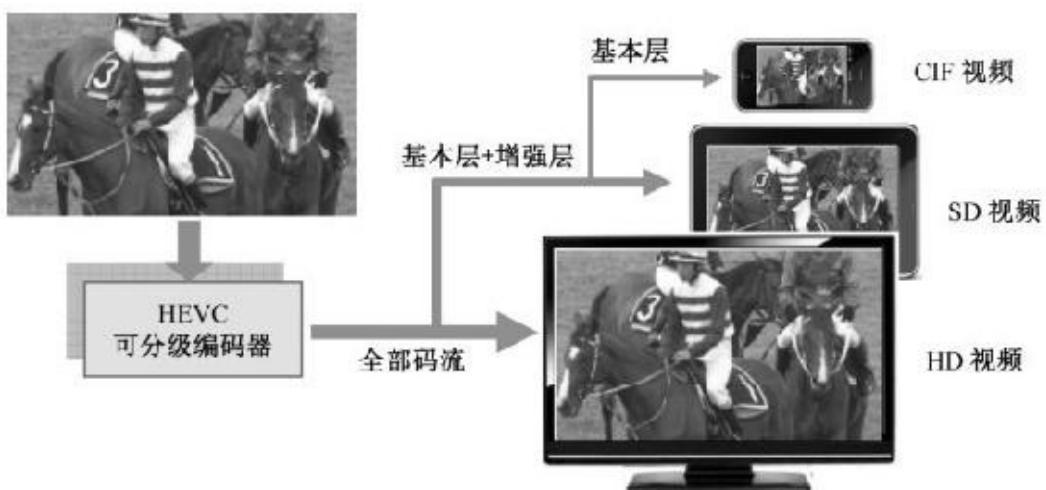


图11.2 HEVC分级视频编码示意图

从图11.2可以看到，一次编码可得到多层次的可分级码流，形成一个综合码流，包含一个基本层和若干增强层数据。用户根据不同的网络带宽或终端能力，从码流中提取适合自己分辨率部分，可解码出从CFI到HD不同质量的视频。

11.4.1 常用可分级编码方法

可分级视频编码，对单一视频序列编码产生若干个（若干层）高低有序的压缩码流。其中必须有一个为基本层（BL）码流，然后依次为第一增强层（EL）、第二增强层等。基本层提供基本图像质量的码流，增强层提供可在基本层基础上重构出更高图像质量所需的码流。视网络状况和用户能力，编码器将这些“层”的码流发送到网上，首先发送的是基本层码流，如果网络允许，再逐渐依次发送增强层码流。接收端的解码器也是首先接收、解码基本层码流，获得一个可接收质量的重建视频，如果带宽和用户终端能力允许，还能够接收并解码增强层码流，将解得的结果“加”到基本层，产生质量更好的重建视频。随着更多增强层的加入，重建视频的质量也逐渐提高，直至和原始发送视频质量相当。

目前，视频的可分级编码方法较多，最为常见的包括：

- (1) 空域可分级（spatial scalability）编码；
- (2) 质量可分级，又称保真度可分级（fidelity scalability）或信噪比可分级（SNR scalability）编码；
- (3) 时域可分级（temporal scalability）编码；
- (4) 细粒度可分级（FGS,Fine Granularity Scalability）编码等。

从可分级视频编码的功能出发，可以将传统的视频编码看成只有基本层的单级视频流。而在一般情况下，可分级编码至少包括两层，即基本层码流和增强层码流。和普通的视频编码相比较，可分级视频编码具有以下特点：

- (1) 具有可分级的视频流，能够动态地适应网络带宽的变化，重构质

量与带宽呈近似线性关系；

- (2) 具有较强的抵御数据丢失的鲁棒性；
- (3) 能够同时满足具有不同处理能力的用户终端的需求。

1. 空域可分级编码

空域可分级编码指对视频序列中的每帧图像产生多个不同空间分辨率的视频流，即基本层和增强层码流。解码基本层码流信息得到的是低分辨率图像，如同时加入增强层码流到解码器，得到的是高分辨率的图像，如果逐渐解码更多增强层的码流，图像空间分辨率也会逐渐增加，直至到达原始图像的分辨率。

空域可分级视频编码的具体过程如图11.3所示。在编码端，先对原始视频中的每帧图像进行下采样得到低分辨率图像，经DCT、量化后形成低分辨率图像的变换系数。这个系数一边送去熵编码后得到空域基本层码流输出；一边送到反量化、反DCT得到编码端的基本层低分辨率重建图像。然后上采样该低分辨率图像，和原始图像相减，对减得的差值进行DCT、量化和熵编码以后，生成空域增强层码流输出。增强层码流包含原始图像与基本层上采样后图像之间的量化差值。由于增强层使用了较小的量化参数，它可以获得比基本层更高分辨率的图像细节信息。更高增强层的编码方法与第一个增强层的编码方法类似。

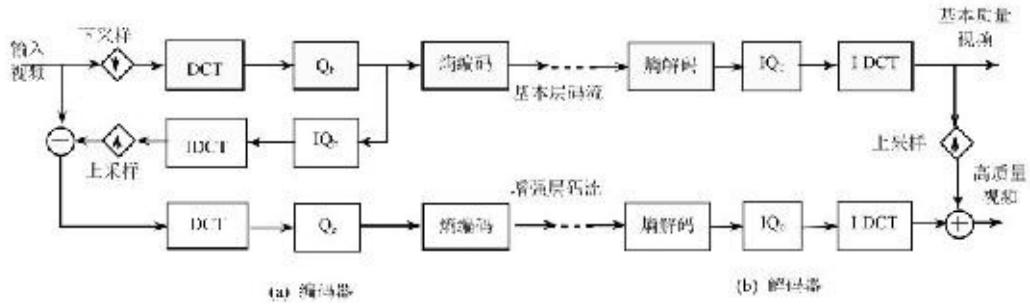


图11.3 空域可分级视频编码原理图

空域间可分级视频解码过程同编码相反。解码端对接收到的基本层码流进行普通的解码重建，得到低分辨率视频图像。如果还有增强层码流可以利用，则对此码流进行解码，得到增强层信号，将此信号和对应的基本层上采样信号相加，即可得到高分辨率的视频。用于增强层的信息近似等于原图像和基本层图像之差，因此增强层的信息和基本层相加，就可以获得比基本层更高分辨率的图像。

2.质量可分级编码

质量可分级视频编码可形成不同质量的图像，其基本层图像和原图像相比，对应像素之间的灰度（彩色）误差较大，图像质量较差。其增强层数据实际上近似等于这些误差数据，可在解码端用于“抵消”基本层的误差，获得较高的图像质量。由误差引起的图像质量可以用SNR来度量，因此质量可分级编码又称为信噪比（SNR）可分级编码或保真度可分级编码。质量可分级编码常见的方法就是用不同的量化参数对DCT系数进行量化，量化参数越小，SNR值越高，恢复出来的图像质量就越好。

质量可分级编码过程如图11.4所示，基本层编码这一路对原始图像

DCT变换后进行一次较粗糙的量化（量化参数较大），再经熵编码后形成基本层码流存储或发送。与此同时，粗糙量化后的数据经反量化后形成基本层系数，与原始图像DCT变换系数相减形成差值信号，对此差值信号再进行一次细量化和熵编码生成增强层码流。如有多个增强层则重复上述过程。

解码过程和编码完全相反。如果只收到基本层码流，则可解码得到可接受质量的视频；如果同时收到基本层和增强层码流，则可获得较高质量的视频。在多个增强层的情况下，每引入一层增强层，都会使得重建图像的信噪比提高，视频质量得到进一步改善。

从图11.4可以看出，质量可分级编码的思路和空域可分级编码很类似，区别在于质量可分级不涉及图像的空间分辨率，无须对原始视频进行下采样和上采样。

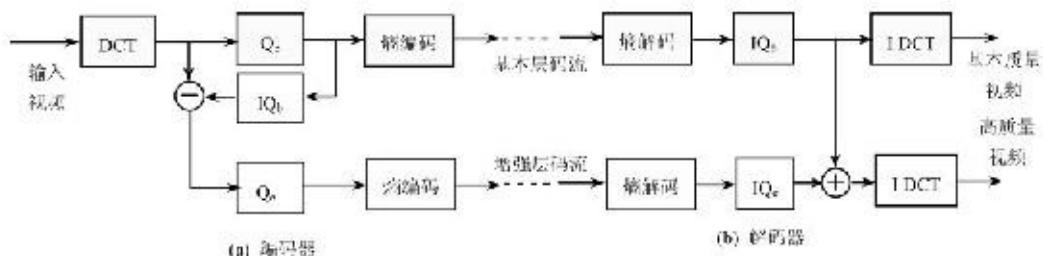


图11.4 质量可分级视频编码原理图

3.时域可分级编码

视频序列的帧率越高给人的感觉就越流畅，视觉效果就越好。但帧率的提高会增加输出码流的数据量，提高对信道带宽和用户终端处理能力的要求。因此，对于不同的信道、不同的用户，一般会对视频序列的帧率有不同的要求，这导致了时域可分级视频编码技术的产生。

时域可分级视频编码是对同一序列以不同的帧率来分别编码的一种方法。其大体的过程如图11.5所示，首先把视频序列按帧不重叠地分割成两层（或更多），如奇帧和偶帧，对基本层（奇帧）进行普通的视频编码，提供具有基本时间分辨率的基本层码流；增强层（偶帧）则是利用基本层数据对增强层偶帧进行预测编码，生成增强层数据。

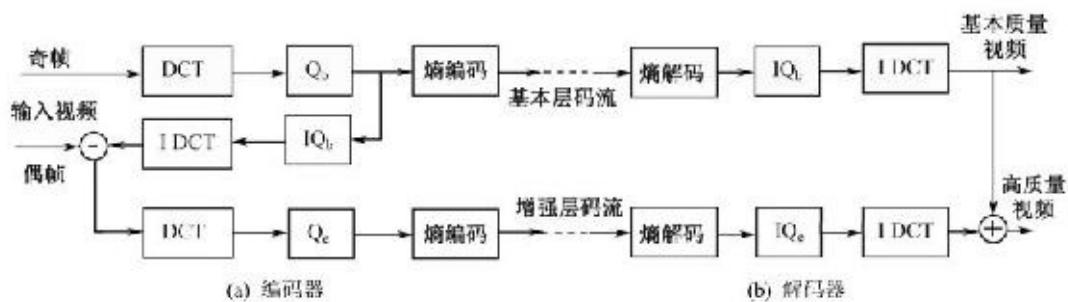


图11.5 时域可分级视频编码原理图

解码端如果只解码基本层帧数据，可以得到帧率较低的重建视频。如果有解码的增强层数据加入，随着加入的增强层的增多，解码视频的帧率越来越高，直至达到与原始视频序列相同的帧率。

4. 细粒度可分级编码

前述三种可分级编码方法中，只能提供几个等级的分级视频，各个层的码率间距较大，码率调节粒度较粗，这相对于网络带宽的变化来说太粗糙了。为了适应带宽变化范围大的网络以及对视频编码的新要求，MPEG-4制定了细粒度可分级（FGS）视频编码标准。FGS 是将视频编码成一个可以单独解码的基本层码流和一个可以在任何地方截断的增强层码流。其中基本层码流适应最低的网络带宽，而增强层码流用来满足网络带宽变化的动态范围和终端异构性的要求。

FGS编码器原理如图11.6所示，其基本层采用基于分块运动补偿和DCT变换的编码方式，达到网络传输的最低要求；增强层使用位平面编码技术对原始DCT系数与基本层粗量化系数的残差进行编码，以适应网络带宽的变化。位平面编码技术使得每一个系数的较高位（重要部分）优先得到编码。每一帧图像的增强层码流可以在任何地方截断而不影响解码的继续；收到并解码的比特数越多，解码器重建的视频质量越高，从而提供了精细可分级的特性。

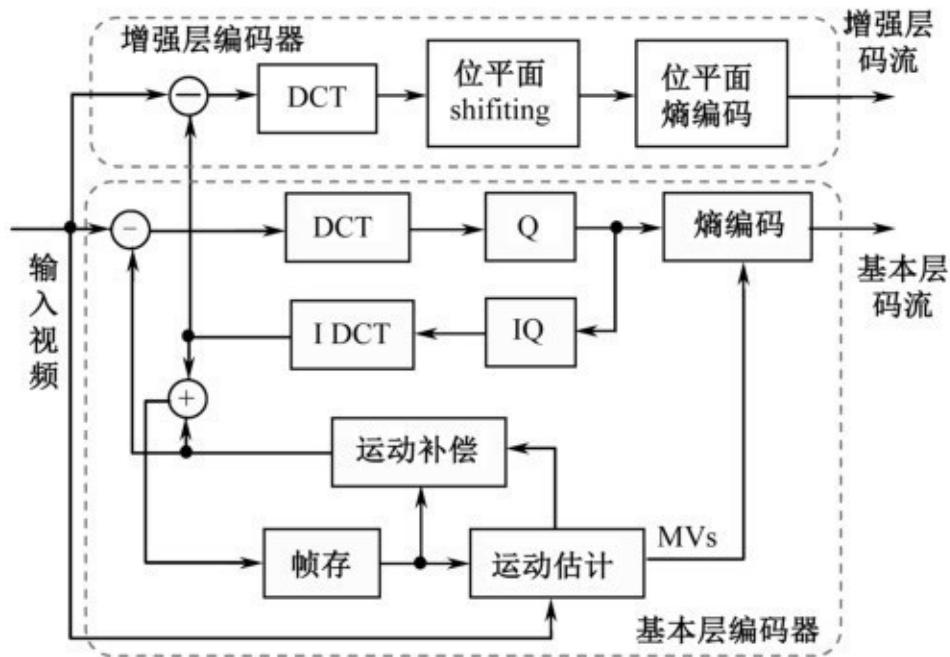


图11.6 FGS可分级视频编码原理图

11.4.2 H.264/AVC的可分级编码

HEVC的可分级扩展（SHVC）继承和改进了H.264/SVC的可分级编码方法，因此大致了解SVC的方法有助于对SHVC的理解和应用。为了应对越来越多样化的网络和应用需求，2007年JVT推出了H.264/AVC的可分级视频编码（SVC）扩展。SVC扩展是作为H.264/AVC的附录G出现的，又简称为H.264/SVC或SVC，相应的参考软件为JSVM（Joint Scalable Video Model）。

H.264/SVC通过提供不同的编码层，在空间、时间、质量上实现可分级编码，以满足不同帧率、图像分辨率和图像质量等级的自适应调整。从分层编码的角度看，H.264/SVC 属于多层编码系统，主要由基本层（BL）和若干增强层（EL）组成。SVC的基本层与H.264/AVC兼容，H.264/AVC的大部分编码工具都可用于SVC，仅有个别工具需要根据不同层的特殊结构或要求进行调整。和普通视频编码不同的是，在可分级多层视频编码中，出现了层间的信息冗余，因此如何有效压缩两个可分级的层间冗余已成为SVC 乃至其他可分级系统中一个非常关键性的问题。

1.空域可分级的层间预测

H.264/SVC中的空间可分级编码加入了层间预测机制。由于不同层中对应位置的帧是对同一时刻场景的不同分辨率表示，其纹理信息和运动信息具有很强的相关性，故可通过层间预测进一步去除运动信息和纹理信息冗余，如图11.7所示。图中层间的竖直向上的箭头表示增强层利用基本层的信息来进行预测。当使用层间预测时，运动信息和纹理信息需按比例进行缩放。

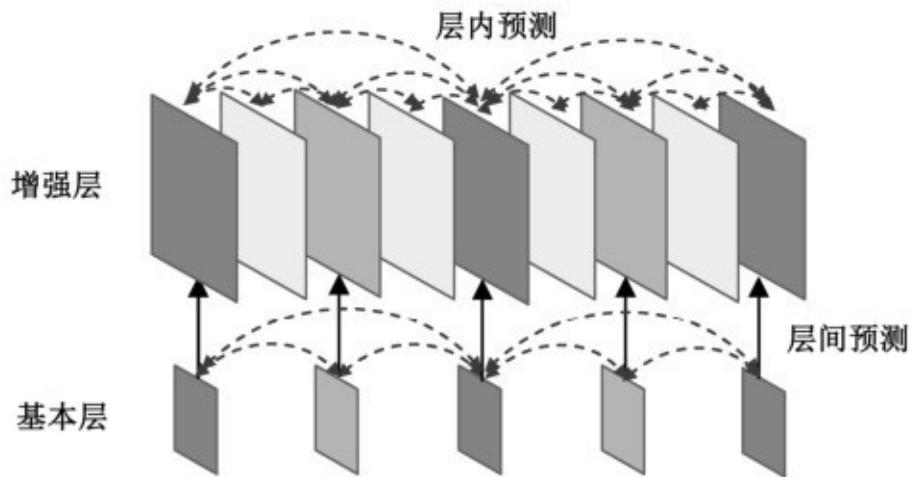


图11.7 H.264/SVC空域可分级层间预测结构

(1) 层间帧内预测

当增强层宏块使用帧内预测模式进行编码时，除H.264/AVC中定义的帧内预测模式外，SVC定义了一种帧内基本层（IntraBL）宏块模式。使用此模式时，要求低层相应位置的 8×8 子块也为帧内编码，并在预测前使用去方块滤波器对低层重建宏块进行滤波和边界扩展，以保证可利用基本层中相应的重建宏块纹理来对当前宏块的纹理进行预测。IntraBL模式使用上采样的重构BL图像来预测EL中的MB，并仅对残差进行编码。

(2) 层间运动信息的预测

当增强层宏块使用帧间预测模式进行编码时，除H.264/AVC中定义的帧间预测模式外，SVC新定义了两种宏块模式：基本层模式（BASE_LAYER_MODE）和1/4像素精细模式（QPEL_REFINEMENT_MODE），统称为基本层跳过（BLskip）模式。

在这种模式下，要求低层相应位置的宏块为帧间模式，可使用它的运动信息（运动矢量、参考帧序号和宏块分割模式）来预测当前宏块的运动信息。

使用BASE_LAYER_MODE时，需利用基本层相应宏块的运动信息（包括宏块划分）。如果基本层是空间分辨率较低的一层，就需相应地“放大”其运动矢量场。使用 QPEL_REFINEMENT_MODE 时，形式上类似于基本层模式，但是需额外增加传输一个1/4像素精度的运动矢量。对所有其他的运动补偿宏块模式（除了直接模式），运动矢量预测可以在空间运动矢量预测器（H.264/AVC定义）和相应的基本层预测器之间切换。

（3）运动信息残差的预测

由于层间运动信息的高度相关性，不同层的运动补偿预测的残差之间也有着相关性，因此使用低层运动补偿残差来对当前层运动补偿残差进行预测可进一步去除层间冗余。需要注意的是，当低层运动信息与当前层运动信息相同或近似时，残差信息相关性强，此时使用残差预测将提高编码效率；相反，当低层与当前层运动信息相差较大时，使用残差预测反而会降低编码效率。为此，SVC在宏块层增加了一个标识。如果这个标识为真，则采用相应底层（基本层）的残差信息作预测；否则，编码当前残差信号而无须从基本层预测。

2.时域可分级的等级B帧结构

在H.264/SVC扩展中，其时域可分级性是通过等级B图像编码结构来实现的，这种编码方法利用B帧双向内插的方法提供了时间可分级性。图11.8描述了一个总共有4级时间分层的等级B帧图像编码结构。

图11.8中的第0、8和16帧是关键帧，属于时间层0。两个关键帧之间的所有帧以及后面的那个关键帧合起来成为一个图像组（GOP）。关键帧可以是帧内预测帧（I帧），也可以是利用前一个关键帧作为参考的前向帧间预测帧（P帧）。图像组中的其他图像都编码为B帧，但是编码的顺序是按照金字塔分级的顺序进行的，所以简称为“等级B帧”编码结构。以图11.8中的第一个GOP为例，首先编码时间0层的第8帧（关键帧）；然后编码时间1

层的第4帧；接下来编码时间层2的第2和第6帧；最后编码时间层3的第1、3、5、7帧。这样，通过在时间上分等级的编码顺序，就在同一个码流内实现了时间可分级性。所有的关键帧组成了最粗糙时间分辨率的基本层视频序列，随着在编码顺序上的图像的递增，时间分辨率也跟着增加，最后达到完全时间分辨率的视频序列。

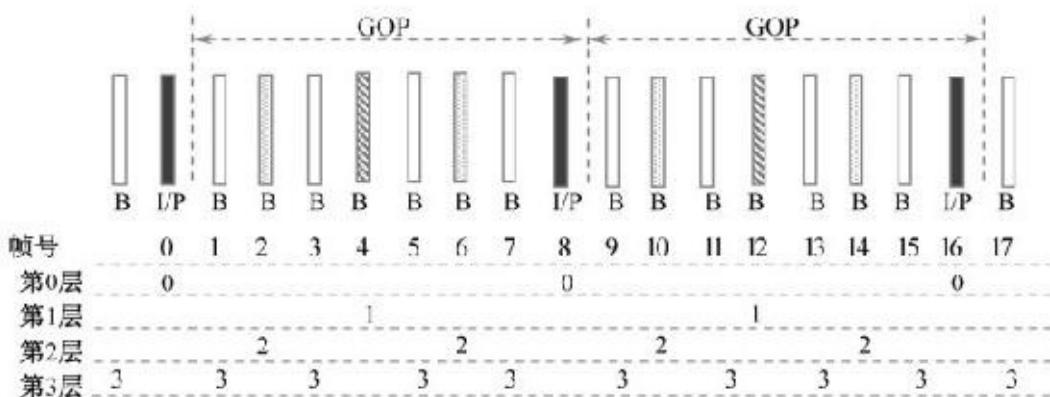


图11.8 H.264/SVC时域可分级B等级编码结构

3.质量可分级的中粒度分级

质量可分级编码可认为是空域可分级的一种特殊情况，即基本层和增

强层图像的尺寸相同时的空域可分级编码。这种情况也称为粗粒度可分级（Coarse Grain Scalability,CGS）编码。H.264/SVC 在比较了不同粒度的质量可分级方式后，为了增加比特流的灵活性以及容错能力，同时提高当 SVC 比特流需要提供多种比特率时的编码效率，确定采用中粒度可分级（Medium Grain Scalability,MGS）技术。MGS可以算是CGS的变种，能够获得性能和复杂度的折中。由于MGS基本层和增强层具有相同的图像尺寸，所以不需要将重建后的低层图像进行上采样。在 MGS 层间预测时，仅需要在高层使用一个相对于低层较小的量化因子来对残差纹理信号进行量化就可获取更多的纹理信息。与CGS不同，在MGS可分级的比特流中，增强层中的任意NAL单元可以被丢弃，因此提供了一种基于包的质量可伸缩编码。

11.5 HEVC的可分级扩展

HEVC的可分级扩展SHVC正式标准已于2014年中期完成，以HEVC第2版附录H（Annex H）的形式出现，它采用空域的和粗粒度SNR可分级的方式。与此同时，JCT-VC也推出了SHVC的测试模式（SHVC Test Model, SHM），版本在不断更新，目前是2015年2月的第9版，SHM 9。

11.5.1 SHVC的编码框架和性能

1. 多环编码框架

可分级视频编码既可以基于单环编码结构，也可以基于多环编码结构实现。在多环结构中，每个中间层需要一个完整的解码环路来解码一个目标层。在每个层中重建所有的帧内和帧间编码块，中间层的重建样点用作增强层（EL）的附加参考。多环结构中，可分级编解码器加大解码图像缓存（DPB）尺寸和用于解码端运动补偿的存储带宽，它的编码效率比单环结构的可分级编解码器要高，因为它可以充分利用 EL 图像和重建参考层图像之间的相关性，而且仅使用帧内预测，不需要限制参考层。

SHVC采用多环编码结构，解码器为了解码增强层，必须首先解码它的参考层，使之成为有效的预测参考。这和H.264/AVC的SVC扩展不同，SVC对帧间预测的宏块（MB）使用单环（single-loop）解码。所谓单环解码方案是指在编码端提取基本层的相关信息来预测增强层，在解码端只需解出目标层即可获取视频图像，也就是说它只进行一次运动补偿处理，是单层视频解码操作。

当使用两个空域或SNR层时，基本层是唯一的参考层。但对于3层或更多的空域层或SNR层，中间层也可以用作参考层。SHVC在编码端增强层

还可以执行HEVC传统帧间编码，即执行所谓双环（或多环）控制编码，在传统帧间预测编码和层间预测编码之间通过率失真优化做自适应选择。很显然，单环解码方案在解码端具有较低的复杂度和稍差的编码效率，而多环解码方案则相反，具有较高的压缩效率和复杂度。

多环编码提供了优于单环编码方式的编码效率，但使得编解码器的计算和存储总量将比单环方式要高，因此HEVC可分级扩展的高层设计采取一些措施，如充分利用现有HEVC方法、限制块级变化等，其初衷都是为了减轻SHVC的实现负担。

2. 可分级编码的性能

HEVC的可分级扩展的编码工具只限于改变slice及slice以上的内容。增强层（EL）的所有的块层面的操作保持和单层HEVC编解码的操作完全相同。参考层图像，如果有必要的话，经上采样后可用作增强层预测时选择的参考图像，提供层间纹理、运动信息和残差的预测。

多环设计在某种程度上类似于H.264/AVC和HEVC的多视点扩展，在解码依赖于视点的情况下，它们需要先完全解码基本视点。然而，在多视点情况下，按照目前的定义，所有的视点都具有相同的分辨率，因此就没有必要上采样。在SNR可分级中情况也是如此，可分级的各层表示相同空间分辨率的图像。基本层比特流应该是一个标准的比特流，可以是一个HEVC比特流，或者是一个H.264/AVC比特流。当基本层是H.264/AVC比特流时，只要完成层间纹理（帧内）预测，并不需要支持层间运动预测。研究表明，此时的压缩收益是比较小的，而且H.264/AVC的基本层运动矢量在现有的解码实现中不容易获得。

由于视频的同播编码和可分级编码的应用目标相似，都是为网络中不同类型的用户终端提供合适的编码视频流。可分级编码对各层信息进行非独立编码，将表示不同质量和分辨率的视频压缩在单一的码流中。而同播则对各层信息进行独立编码，形成多个独立的压缩码流。因此，人们常常从性能和复杂度等方面来比较可分级编码和同播编码。

从网络利用的角度来衡量，可分级视频流比同播视频传送更有效，因

为可分级视频流可以服务于所有的终端，而同播视频流则需要并行提供所有分辨率码流。在典型的可分级编码或同播的应用场合，例如在可变码率或分辨率切换编码时，往往只需输出其中的一层。但在可分级多环解码的情况下，还必须要解码所有的参考层，这就使得可分级和同播比较，其整体解码复杂度大为增加。这一影响在SNR可分级中是比较显著的，因为这里的参考层是不用亚取样的。更重要的是，可分级的所有层非独立编码方式比同播在压缩性能方面具有优势。

当采用空间可分级时，解码的参考层图像需用一个归一化定义的上采样滤波器重抽样。在目前的设计中，空间可分级的比率限制为每一维1.5倍和2倍的空间上采样因子。因为这些因子已足够覆盖通常的使用场合，而且也简化了实现。

3.SHVC的解码结构

图11.9为SHVC解码器的结构框图，为简明起见，这里只给出了基本层（BL）和1个增强层（EL），更多的增强层的情况和此类似。SHVC解码器接收到SHVC码流以后，经码流分解单元解析出基本层码流和增强层码流分别送至BL解码单元和EL解码单元。EL中所有块级以下的解码都使用HEVC解码器，但并非和基本HEVC完全一致，允许必要的块级以上的变化，因而标注为HEVC^{*}解码器。

BL的重建帧来自于BL的解码图像缓存（DPB）。如果EL需要进行层间预测处理，可用重建的BL图像作为层间参考（ILR）图像。ILR图像送至EL的DPB，作为长期参考图像和EL的时域参考图像一起用作增强层解码的参考图像。

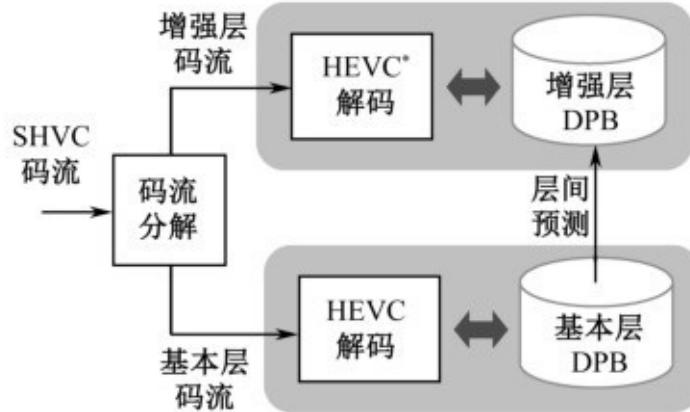


图11.9 SHVC解码结构框图

图11.9的可分级系统具有以下优点。

(1) EL 中的所有块级处理保持和单层 HEVC编解码一致，只允许在 slice以上的高层语法的改变。

(2) 由于EL只需要重建的BL图像，因此，还允许其他标准的BL编码，如H.264/AVC的BL图像的使用，甚至允许由外部方式传输基本层码流，而不是由“带内”编码产生的可分级BL码流。

(3) 图11.9所示的结构设计和多视点扩展的MV-HEVC是兼容的。尽管SHVC和MV-HEVC的制定目标是不同的应用场景，但这两个扩展可以采用统一的编码结构，为两者的合并和更多应用带来便利，如将来可设计一种结合视点和空间的可分级编码。当然，图11.9中的层间预测在目前的 MV-HEVC中还是不允许的。

4.SHVC和H.264/SVC的比较

表11.1给出了SHVC和H.264/SVC所支持的主要可分级特性的比较。

SHVC支持更加丰富的可分级特性，除支持通常的时域、空域和质量可分级以外，还支持混合编解码可分级、比特深度可分级以及彩色γ可分级（Color Gamut Scalability, CGS）。

表11.1 SHVC和SVC可分级特性比较

| 可分级特性 | 可分级标准 | | SHVC 定例 | |
|-------|-------|------|-------------|------------------|
| | SVC | SHVC | 基本层 | 增强层 |
| 时域 | ✓ | ✓ | 30 fps | 60 fps |
| 空域 | ✓ | ✓ | 720p, 1080p | 1080p, 4000×2000 |
| 质量 | ✓ | ✓ | 33dB | 36dB |
| 混合编解码 | | ✓ | H.264/AVC | H.265/HEVC |
| 比特深度 | | ✓ | 8 bit | 10 bit |
| 色彩 γ | | ✓ | BT.709 | XYZ, BT2020 |

11.5.2 上采样滤波器

在SHVC中，层间预测是通过插入层间参考（Inter Layer Reference, ILR）图像到EL解码图像缓存中来实现的，参考图像由基本层（BL）图像重建产生。增强层的高分辨率图像需要参考低分辨率的基本层图像，为此，需要对基本层图像进行插值，使其分辨率和增强层图像匹配。上采样滤波器正是用于匹配从参考层到高分辨率增强层的重建样值。它允许增强层预测中使用重建的参考层样值。在可分级扩展中，SHVC 对上采样的滤波过程作了规定，为亮度上采样定义了8抽头多相有限冲激响应（FIR）滤波器，为色度上采样定义了4抽头多相FIR滤波器。

1. 亮度上采样8抽头多相滤波器

决定上采样滤波器抽头数的因素和HEVC分数位置内插的运动补偿设计是一致的，那里也是将8抽头和4抽头FIR滤波器分别用于亮度和色度。在HEVC中，亮度运动补偿仅为1/4取样精度，4:2:0的色度为1/8精度。而SHVC中，为了适应较高的上采样倍数，相应的参考层插值的精度定义成1/16取样间隔，因此设计成带有多相移的滤波器。这样的设计保证了任意上采样率的使用，其中所有16个相位位置的滤波器是必需的，但目前的规

范仅限于1.5倍和2倍的上采样率，这种情况则需要较少的位置。

可分级的参考层偏移可被标识，以使参考层和增强层相对自由，不需要完全对应于图像的相同区域。水平和垂直方向的分级系数可用相关增强层和参考层之间各自宽度和高度之比来计算。对每个增强层样点，考虑到分级系数和分级的参考层偏移来决定相应的参考层样点的位置和1/16样点相位。对于所计算出的相位的8抽头（或4抽头）滤波器系数被用于输入参考层样点，它们是在参考样点位置的样点，它的邻近样点在参考层。

亮度上采样滤波器的滤波系数显示在表11.2中。8抽头值 T_0, T_1, \dots, T_7 的选择也是模仿HEVC的运动补偿内插过程。0、8/16相位和HEVC过程的0、1/2相位对应，抽头值一样，而且对于2倍的上采样是必需的。0、5/16、11/16相位对1.5倍是必需的，后2个相位的设计和运动补偿内插器中1/4、3/4相位并不直接对应，但设计方法相同，且满足相同的频率响应和计算精度的约束。

表11.2 亮度上采样滤波器的系数

| 相位 | T_0 | T_1 | T_2 | T_3 | T_4 | T_5 | T_6 | T_7 |
|-------|-------|-------|-------|-------|-------|-------|-------|-------|
| 0 | 0 | 0 | 0 | 64 | 0 | 0 | 0 | 0 |
| 5/16 | 1 | 4 | 11 | 52 | 26 | 8 | 3 | -1 |
| 8/16 | 1 | 4 | 11 | 40 | 40 | 11 | 4 | -1 |
| 11/16 | -1 | 3 | -8 | 26 | 52 | -11 | 4 | -1 |

2. 色度上采样4插头多相滤波器

与亮度滤波器类似，色度上采样滤波器显示在表11.3中。这里，色度上采样需要定义为9个相位的多相滤波器以支持1.5和2倍的上采样率。色度滤波必须有较多相位数的原因是在4:2:0色度中亮度和色度样点之间存在固有的相移，这在匹配基本层和增强层色度位置时必须要考虑。和亮度上采样滤波器类似，色度滤波器抽头和那些运动补偿滤波器相对应的相位具有相同的抽头值，如相位为4/16、6/16、8/16、14/16之处。而没有直接对应运动补偿滤波器相位的相位处，需满足相同的频率响应和计算精度的约束。

表11.3 色度上采样滤波器的系数

| 相位 | T ₀ | T ₁ | T ₂ | T ₃ |
|-------|----------------|----------------|----------------|----------------|
| 0 | 0 | 54 | 0 | 0 |
| 4/16 | -4 | 54 | 16 | -2 |
| 5/16 | -6 | 52 | 20 | -2 |
| 6/16 | -6 | 46 | 28 | -4 |
| 8/16 | -4 | 36 | 36 | -4 |
| 9/16 | 4 | 30 | 42 | 4 |
| 11/16 | -2 | 20 | 52 | -6 |
| 14/16 | -2 | 10 | 58 | -2 |
| 15/16 | 0 | 4 | 62 | -2 |

11.5.3 层间纹理预测

使用上述的上采样过程可实现从参考层重建的样点值到增强层分辨率的投影。为保证增强层预测对上采样信息的选择，可分级扩展应用了参考索引方法。从概念上说，这种方法要求增强层解码器将上采样的参考层图像插入到增强层参考图像列表（Reference Picture List, RPL）中。而后，将上采样图像标注为参考图像，和通常帧间预测方式的参考帧一样用于预测。也就是说，增强层比特流用参考索引标记了一个帧间模式CU，该索引对应于插入到增强层RPL的上采样图像，不过这个特别的参考图像的运动矢量为0。

在解码器中构建RPL的过程比较直接。首先，用和HEVC版本1相同的方式构建一个初始的RPL。也就是说，将比特流中检测到的短期参考图像和长期参考图像加到列表中。跟在这些图像后面，将上采样的基本层图像添加到初始RPL中，并标记为长期参考图像。因此，参考这些参考图像的运动矢量预测器并不需要按照时间差距离来伸缩。还有，除了现在起始列表包含了上采样基本层图像和附加的参考层图像（如果存在）外，其他都和第1版HEVC是一致的。

当比特流中存在RPL修改信息时，增强层解码器使用的实际RPL可以修改它们的初始值，这和HEVC第1版的情况一样。当修改信息不存在时，可以直接使用初始RPL。当RPL修改信息存在时，编码器在使用前可以标

注再排序起始列表，其过程和HEVC第1版中规定的相同。这种再排序允许和参考层图像对应的图像移动到列表中的其他地方。这样做的一个好处就是提高编码效率，因为处于列表末尾的图像需要更多的比特来指示。当上采样参考层样点和增强层高度相关时，有利于编码器移动上采样参考层样点到列表中一个比较前的位置。

基于参考索引标记的方法增加了编码的灵活性。例如，编码器能够使用双向预测方法来完成一种预测，它对来自不同时间位置的参考层和重建增强层图像的信息进行平均。为了限制存储器带宽和复杂性，参考索引方法也规定了比特流的限制，即当参考上采样层样点时运动矢量必须为0。这简化了解码器设计，特别是在实现中，作为预测过程的一步，上采样可能“飞过”一部分，而不是在处理步骤中上采样全部参考图像。

除上采样参考层样点的使用外，可分级扩展还支持来自解码的参考层运动信息的预测。通过联合来自参考层的运动数据和插入到增强层RPL中的上采样的参考层图像来完成上述功能。

11.5.4 层间运动预测

在可分级扩展中，当利用HEVC第1版现存的时域运动矢量预测（Temporal Motion Vector Prediction, TMVP）过程编码增强层运动矢量时，运动场匹配就是使用参考层运动信息的过程。

在HEVC中，用TMVP方法从参考图像的同位（co-located）PU来预测当前PU的运动信息。这一过程需要定义同位 PU 的预测模式、参考索引、亮度运动矢量和参考图像顺序计数（POCs）。将这一信息存储在一个 16×16 亮度基块上，它在码流中传输的较小PU尺寸情况下具有较低的分辨率。这样减少了最坏情况下存储参考层运动信息所需的容量和带宽。此后，运动场匹配处理的目标是投影这个来自参考层的运动信息到增强层分辨率，同时还要处理在参考层的 16×16 的TMVP存储单元。

运动信息匹配的第一步是为当前增强层 PU 决定在存储的参考层中运

动信息中的 colocated位置，以利于减少的运动信息的存储分辨率，减少两层之间的上采样率和参考层的偏移。一旦co-located位置确定，同位参考层PU的运动信息就可用，然后可伸缩操作按照上采样率用于这些运动矢量，因为运动矢量也是随着图像分辨率而增加的。然而，由于事实上参考层图像被标明是“长期”参考图像，不需要按照时间距离再进行伸缩处理。

在比特流中，可以开放或禁止运动匹配处理，当使用H.264/AVC基本层时就被禁止。结合参考层样点值的上采样和参考索引信令机制，运动匹配提供了一种方法来处理大量参考层信息而没有改变HEVC解码器的块层设计。

11.5.5 SHVC编码一例

SHVC 提供时域、空域和质量可分级编码。SHVC 基于环内（in-loop）层间预测来进一步提高编码效率。和同播（simulcast）编码指标相比，层间预测可增加率失真性能15% ~ 30%。图11.10显示了2层可分级层的SHVC的编码框图。

其中，SHVC编码系统应用了2个HEVC编码器，第一个编码器编码HD分辨率的基本层（BL）视频，第二个HEVC编码器使用BL中的解码帧作为层间预测的附加参考图像，编码4K分辨率的增强层视频。在空域可分级的情况下，层间参考图像需上采样，它的运动矢量也要放大到能和正在解码的EL分辨率相匹配。

本章参考文献

[1]Gary J.Sullivan,Jill M.Boyce,Ying Chen,et al.Standardized extensions of High Efficiency Video Coding (HEVC) [J].IEEE Journal of Selected Topics in Signal Processing,2013,7 (6) :1001-1016.

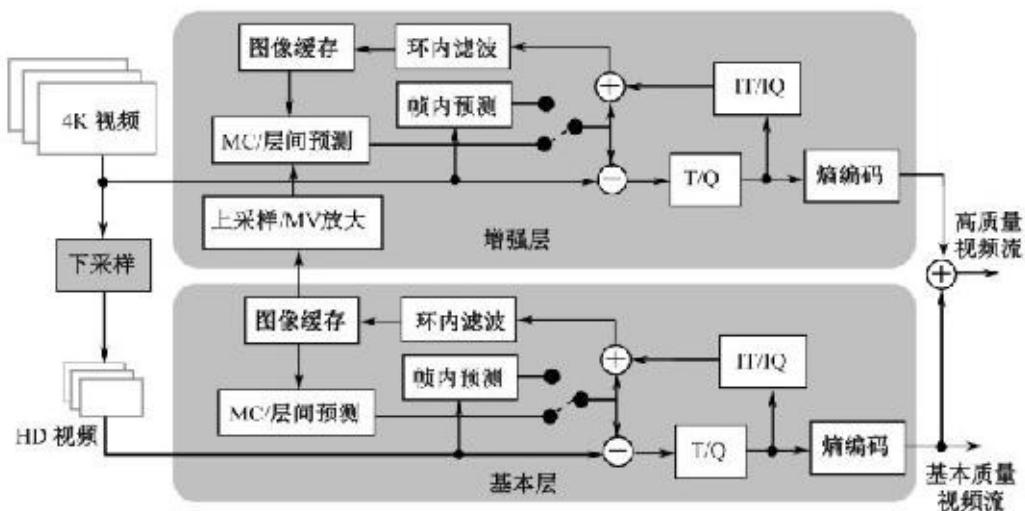


图11.10 2层空间可分级SHVC编码框图

[2] Flynn,D.,et al.High efficiency video coding (HEVC) range extensions text specification: Draft 5[R].JCTVC-O1005.15th meeting: Joint Collaborative Team on Video Coding (JCT-VC) of ITU-T SG 16 WP 3 and ISO/IEC JTC 1/SC 29/WG 11,Geneva,CH (2013) .

[3] Sullivan,G.J.,Ohm,J.-R.Meeting report of the 15th meeting of the Joint Collaborative Team on Video Coding (JCT-VC) [R],Doc.JCTVC-O1000.15th meeting: Joint Collaborative Team on Video Coding (JCT-VC) of ITU-T SG 16 WP 3 and ISO/IEC JTC 1/SC 29/WG 11,Geneva,2013.

[4] Boyce,J.Specification text and profile for SHVC with AVC base layer[R].JCTVC-O0143.15th meeting: Joint Collaborative Team on Video Coding (JCT-VC) of ITU-T SG 16 WP 3 and ISO/IEC JTC 1/SC 29/WG 11,Geneva,2013.

[5] Wien,M.,et al.JCT-VC AHG report: Single-Loop Scalability (AHG16) [R].JCTVC-O0016.15th meeting: Joint Collaborative Team on Video Coding (JCT-VC) of ITU-T SG 16 WP 3 and ISO/IEC JTC 1/SC 29/WG 11,Geneva,2013.

[6] Heiko Schwarz,Detlev Marpe,Thomas Wiegand.Overview of the scalable video coding extension of the H.264/AVC standard[J].IEEE Trans.on Circuits and Systems for Video Technology,2007,17 (9) : 1103-1120.

[7] C.Feldmann,F.Jager,J.Hsu,et al.Single-loop SNR scalability using binary residual refinement coding[C].Proceedings of IEEE International Conference on Image Processing (ICIP 2013) ,Melbourne,15-20 Sept.2013,1583-1587.

[8] Benoit Martin,Wassim Hamidouche,Jean Le Feuvre,et al.Unified real time software decoder for HEVC extension[C].Proceedings of IEEE International Conference on Image Processing (ICIP 2014) ,Paris,27-30 Oct.2014,2186-2188.

[9] Thorsten Laude,Xiaoyu Xiu,Jie Dong,et al.Jörn Ostermann.Scalable extension of HEVC using enhanced inter-layer prediction[C].Proceedings of

IEEE International Conference on Image Processing (ICIP 2014) ,Paris,27-30 Oct.2014,3739-3743.

[10]High Efficiency Video Coding[S],ITU-T Rec.H.265 and ISO/IEC 23008-2,ITU-T and ISO/IEC JCT-VC,Mach 2013 (v.1) ,Oct.2014 (v.2) ,Apr.2015 (v.3) .

[11]Yan Ye,Pierre Andrivon.The scalable extensions of HEVC for Ultra-High Definition Video delivery[J].IEEE Multi Media,July–September 2014,58-64.

[12]Mathias Wien.High Efficiency Video Coding: Coding Tools and Specification[M].Springer-Verlag Berlin Heidelberg,2015.

[13]High Efficiency Video Coding[S],ITU-T Rec.H.265 and ISO/IEC 23008-2,ITU-T and ISO/IEC JCT-VC,Mach 2013 (v.1) ,Oct.2014 (v.2) .

[14]Gary J.Sullivan,Jens-Rainer Ohm,Woo-Jin Han,Thomas Wiegand.Overview of the High Efficiency Video Coding (HEVC) Standard[J].IEEE Transaction on Circuits and Systems for Video Technology,2012,22 (12) : 1649-1668.

[15]A.Abdelazim,A.M.Hamza.Adaptive hierarchical motion estimation optimization for scalable HEVC[C].Proceedings of the 8th IEEE GCC Conference and Exhibition,Muscat,Oman,1-4 February,2015,1-5.

[16]Zhongbo Shi,Xiaoyan Sun,Feng Wu.Spatially scalable video coding for HEVC[C].2012 IEEE International Conference on Multimedia and Expo,1091-1096.

[17]H.R.Tohidypour,M.T.Pourazad,P.Nasiopoulos1,et al.A new mode for coding residual in Scalable HEVC (SHVC) [C].IEEE International Conference on Consumer Electronics (ICCE 2015) ,372-373.

[18]Do-Kyoung Kwon,Madhukar Budagavi and Minhua Zhou.Muli-loop scalable video codec based on High Efficiency Video Coding (HEVC) [C].38th IEEE International Conference on Acoustics,Speech,and Signal

Processing (ICASSP' 2013) ,1749-1753.

第12章 HEVC的多视点和3D编码扩展

展

随着计算机视觉、计算机图形学和视频处理技术的快速发展和相互融合，现有的平面视频，即二维（2D）视频业务已经不能满足人们的需求，能够为人们提供立体视觉的三维（3 Dimensions,3D）视频、多角度观察的多视点视频（Multi View Video,MVV）受到更多人的关注，已成为目前视频领域的开发和应用的热点。平面视频由于高清化的进程和日见“泛滥”的使用，其数据量已属“海量”级别，再加上如今的视频立体化进程，其数据量又较高清视频成倍、成几十倍地增长，因此对立体视频的压缩编码技术和标准的开发已到了刻不容缓的地步。

为避免类似的术语之间易于混淆，本章人为设立了一些区别，如“多视点视频”一般指对于同一场景的多个视点的摄像机同步拍摄的视频信号组，每个视点拍摄的都是双目视频；“3D视频”一般指某个视点的双目视频；“立体视频”一词则用于对“3D 视频”和“多视点视频”的统称，也用于泛指常见的双目视频，具体语义可以由上下文语境来区分。

本章重点介绍的立体视频编码标准扩展，包含对多视点视频的编码和对3D 视频的编码，3D视频是多视点视频的特殊情况，即“单视点”视频。本章对多视点视频分析的侧重点在它的系统结构和视点间的视频处理，对3D 视频分析的侧重点则在于对双目视频的编码方法，尤其是“视频+深度图”的编码方法。

本章在简要介绍立体视频编码必要内容的基础上，着重介绍多视点视

频编码系统和3D 视频编码工具，分别对应的是MV-HEVC扩展和3D-HEVC扩展。由于3D-HEVC的扩展工作还在进行中，这里所述的内容难免今后会有改变，应当以今后的正式标准的文档为准。

12.1 立体视频编码

本节首先介绍立体视频的“立体感”的来由，多视点视频的形成，立体视频图像的采集和显示；在此基础上给出立体视频压缩编码的基本方法；最后简要给出H.264/AVC的多视点视频编码的结构，因为HEVC的立体视频扩展很大程度上是扩充H.264/AVC的MVC而来。

12.1.1 立体视频和多视点视频

1. 立体视觉和视差

自然界是一个三维空间，所有的景物都是呈立体状的。和客观世界的三维空间相匹配，人类视觉系统也是一个天然的立体视觉系统，不仅能够感知外界左右的宽度、上下的高度，同时还能够感知它前后的深度。这就是说人眼在观看自然界空间景物时具有立体感，或者说人眼具有立体视觉机制。

一般来说，当人观察一个立体物体时，由于左右两眼相距58~72 mm，所以是从不同角度来观察的。具体说来，双眼在朝一个共同的方向观察物体时，每只眼睛的视网膜上各形成一个独立的视像，左眼看到物体的偏左边，形成左视像，右眼看到物体的偏右边，形成右视像，这样在两个视网膜上得到稍有不同的视像，两者之间的差别就是两眼的视差。左右视像传到大脑皮层后“结合”起来，产生一个单一的具有深度感的视像。我们正是依靠这种双眼视差来感知物体的前后深度，产生立体景象。当然，人的立体感觉的产生是一个多因素综合作用的结果，例如从所观察景物的透视情况、运动速度、光影变化、物体相互参照等，都能够产生一定的立体效果。但结合三维世界中的先验知识、双眼的视差应该是获得强烈深度

感的最主要的原因。

根据人眼视觉的立体成像机理，如果能使人的左、右眼能分别观察到目标的具有视差的左右视图，那么就能在人脑中呈现出具有真实世界立体感的视像，这就是基于双目立体视觉的立体图像采集和显示技术的基本依据。

2.多视点视频

为进一步提高视频场景真实性，需要对双目立体视频系统进行扩展，形成多视点视频系统。多视点视频（MVV）是一种新型的具有立体感和交互操作功能的视频系统，通过在场景中放置多台摄像机，记录下多个视点的数据，为用户提供视点选择和场景漫游的交互能力。

多视点视频由多个摄像机从不同角度拍摄而得，在显示时，可以根据观看者所处位置显示相应角度的图像；当观看者的头部移动时，看到的内容也会出现相应的变化，从而得到了“运动视差”和“环视”的效果。人眼生理视觉研究表明，人眼对场景画面变化的辨别能力非常强，为得到自然平滑的运动视差效果，双眼距离内需要提供超过10幅的画面内容，因此，需要使用分布比较稠密的摄像机来获得多视点视频序列。然而这样系统太复杂，实际很难做到。在实际应用中，可使用比较稀疏的摄像机阵列拍摄某场景的视频图像，然后利用视差信息和两个相邻摄像机的视频合成两摄像机之间“虚拟视点”的图像。

为同时获得水平和垂直方向的运动视差效果，多视点视频需要摄像机阵列来采集，考虑到复杂度，目前的测试序列大都只提供水平方向的运动视差效果，使用水平一维摄像机阵列获得，如图12.1所示。



图12.1 水平排列的多视点系统

12.1.2 立体视频的采集和显示

1. 双目摄像系统

当用双目立体摄像机记录3D景物时，先要把两部摄像机的光轴（摄像机镜头到景物的连线）汇聚于感兴趣的物体上，这时称两部摄像机光轴的交点为汇聚点，汇聚点到两部摄像机透镜中心连线中点的距离为汇聚距离。在数学上汇聚距离可以是有限值，也可以是无限值，当汇聚距离非常远时，两部摄像机的光轴近似平行，可认为汇聚点在无穷远处，汇聚距离为无穷大。为便于数学分析，称汇聚距离为无限远时的双目立体摄像系统为平行立体摄像系统，汇聚距离为有限值时的双目立体摄像系统为汇聚立体摄像系统。

会聚摄像系统对接近摄像机的物体能提供比平行配置更好的深度感觉。平行系统的摄像机对所摄取的左右视图可以直接呈现给人的双眼，会聚系统摄像机对所摄取的左右视图则需要经过校正才可以呈现给人的双

眼。一般立体视频系统常常默认为平行双目摄像系统。

2.立体视频的采集

立体图像或视频的采集，可以模仿人眼的立体视觉机理，采用类似于二维成像方法，利用图像采集设备（如摄像机、照相机等）来记录或生成客观场景的立体图像。立体图像（包括双视图像方式和多视图像方式）采集的必要条件包括光源、摄像机和景物，立体图像的大小和立体感的强弱主要由三者的相互位置和运动情况所决定。

最常见的立体视频的采集，是通过两台水平相隔一定距离的摄像机来对目标场景摄像而实现的。两台摄像机同步摄取两段视频，一个是左目视频，另一个是右目视频，统称双目视频。这样获取双目视频通过立体图像显示设备可为我们提供立体成像。若用多于两个摄像机在不同位置对同一场景摄像，形成同步的多段视频，就是多视点视频采集，每一段对应一个视点。

立体视频的采集还可以用带有测距功能的摄像机来拍摄，这种摄像机在和普通2D 摄像机一样拍摄视频的同时，它的红外（或其他介质）传感器也在感知并记录物体的位置，形成与二维视频相对应深度信息，或深度图。由深度信息可以用信号处理的方式得到双目视频中的另一目视频。

3.立体视频的显示

立体图像显示就是根据人眼的视觉机理，模仿视网膜成像过程，只要能够将同一场景的左右两幅视图分别投影到人的左右眼视网膜上，就可以获得该场景的立体视觉感受。做到这一点，关键要有适当装置保证每只眼睛只看到对应的视图，不允许发生混叠，否则就不能保证立体显示效果。对于3D视频显示技术而言，主要可分为两类：眼镜式3D显示技术和裸眼式3D显示技术。每一类中又有多种不同的方式，下面分别介绍一种眼镜式和一种裸眼式立体显示技术。

（1）偏振光眼镜式3D显示技术

各种眼镜式3D 显示技术的一个共同点就是需要观众在观看3D 视频时戴一副特殊的眼镜，偏振光眼镜是最为普遍的一种光分的技术，利用光线

是电磁波，可以分解为垂直振动和水平振动方向的光，将它们分别用作左右眼视图的区分。



图12.2 偏振光式3D显示示意图

偏振光式3D显示需要提供两路视频，即3D视频的左视图和右视图，分别用两台投影仪向一个屏幕投射。在两台投影仪投射光前分别加一块偏振片（允许某一方向振动的光通过），一块是垂直偏振片，另一块是水平偏振片。这样相互正交的两路视图的光就叠加投射到屏幕上。和观看普通投影仪屏幕一样，观看者看到的是屏幕上反射出来的光线。这时，如果观看

者不戴眼镜，他每只眼睛都会同时看到屏幕上的左视图和右视图放射出光线的叠加，感觉是一种没有立体感、却有些模糊的视频，模糊是因为左右视图有一定的视差。

但是，如果观看者在观看时戴上一副特制的偏振眼镜，眼镜的两个镜片分别是水平偏振片和垂直偏振片，水平偏振片只允许水平偏振光通过，垂直偏振片只允许垂直偏振光通过，这样就保证了人的左右眼看到是两组不同的画面，经过大脑“合成”了立体影像，其观察过程如图12.2所示。

(2) 视差障栅裸眼式3D显示技术

视差障栅 (barrier) 3D显示技术如图12.3 所示，其原理和偏振式3D技术较为类似，与现有的LCD液晶显示工艺兼容。

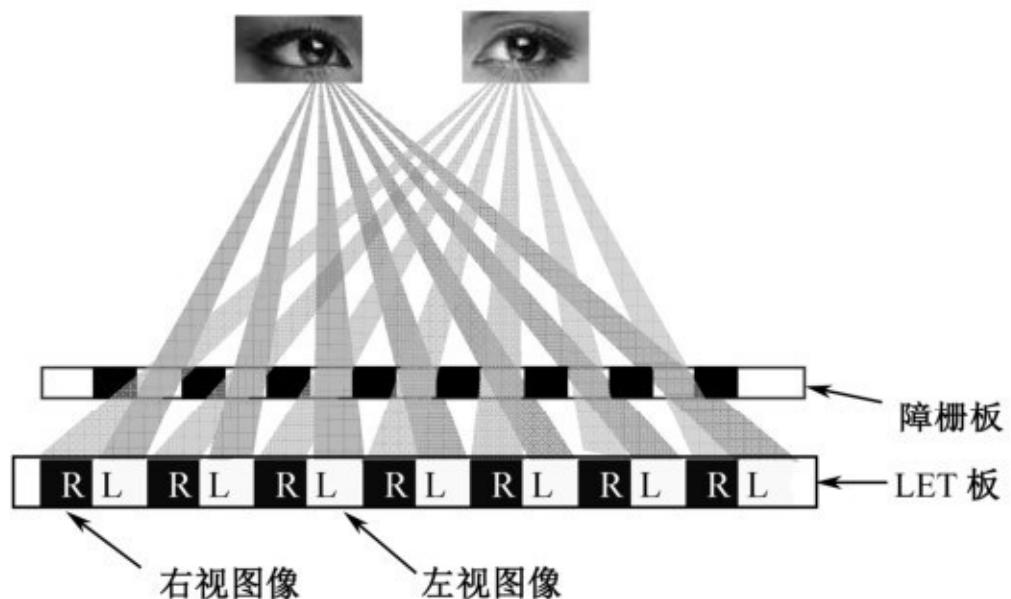


图12.3 视差障栅式显示技术示意图

视差障栅式3D 技术使用一个开关液晶屏、偏振膜和高分子液晶层，利用液晶层和偏振膜制造出一系列方向为90°的垂直条纹。这些条纹宽几十微米，通过它们的光形成了垂直的细条栅模式，称之为“视差障栅”。该技术正是利用了安置在背光模块及LCD面板间的视差障栅，在立体显示模式下，应该由左眼看到的图像显示在液晶屏上时，不透明的条纹会遮挡到右眼的光线；同理，应该由右眼看到的图像显示在液晶屏上时，不透明的条纹会遮挡到左眼的光线，通过将左眼和右眼的可视画面分开，使观者看到3D图像。

在视差障栅式3D 技术中，由于视差障栅的存在，阻挡了一半的光线，造成显示视频的分辨率和亮度都有所下降。

12.1.3 多视点视频编码

多视点视频编码（Multi-view Video Coding,MVC）系统的结构和传统的平面视频系统类似，主要包括多路视频输入、多视点编码、信道传输或存储、多视点解码、显示等几部分，如图12.4所示。n个视点的视频输入数据并行输入到多视点视频编码模块中，生成压缩后的传输或存储码流。解码器收到这些数据后，经过多视点解码（MVD），恢复出原始的多视点视频，按照观看者的要求对多路视频进行单视点或多视点显示。

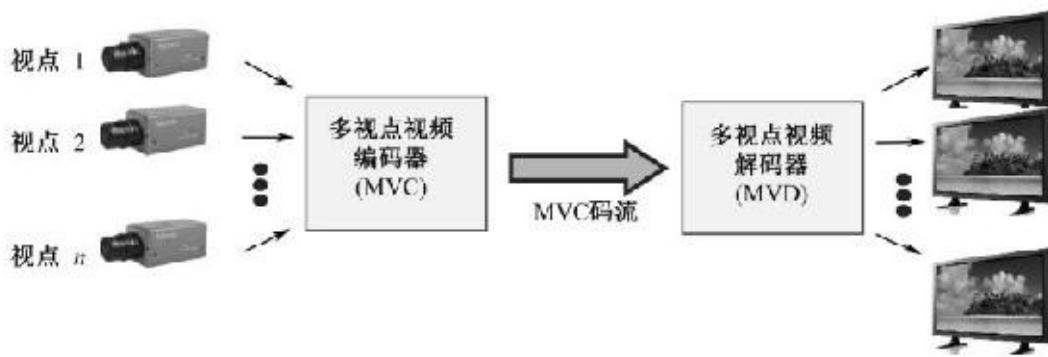


图12.4 多视点视频编解码系统结构

在多视点视频的编码系统中，除视点内视频存在的时间和空间的冗余以外，还存在相邻视点之间的冗余，因此可以在运动估计以及运动补偿的基础上，进行视点间的预测来消除视点间的“视点冗余”，达到进一步提高视频压缩效率的目的。可见，在多视点视频编码中采用什么样的预测结构是决定多视点视频编码效率的重要因素之一，下面简要介绍目前常用的2种效果较好的MVC预测结构。

1.顺序式预测结构

视点顺序预测结构（Sequential Prediction Structure, SPS）如图12.5所示，图中的箭头表示预测方向。第一个视点（视点0）的图像帧按照通常的单视点视频的时间顺序依次进行时域预测编码；第二个视点（视点1）的每一帧不仅可以利用本点视频序列的时域预测，还可以利用第一视点视频序列中相对应的帧进行视点间预测（inter-view prediction）。依次类推，其余各视点情况与第二视点类似。

图12.5的SPS结构中仅仅使用了前向预测帧（P帧），在此基础上还可

以在同一视点内引入双向预测帧（B帧）进一步改进这种基本预测结构，形成另一种编码效率更高的一种SPS预测结构，但这种SPS结构不太适合随机访问。

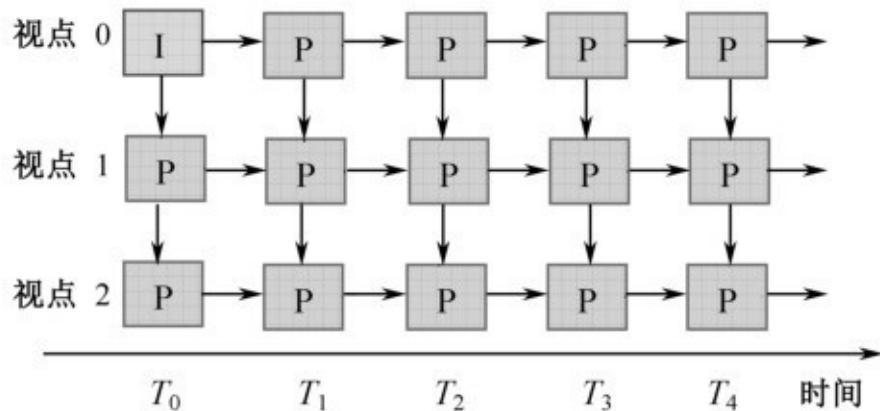


图12.5 基本顺序视点预测结构

2. 等级B帧预测结构

在SPS结构的基础上，2006年德国 HHI (Heinrich-Hertz-Institute) 提出了基于等级B帧的视点间预测和时域预测相结合的预测编码结构，因其能够获得较高的编码效率而被JVT 采纳为测试模型 JMVM 的参考预测结构。

等级B帧是在视点内的一种预测结构，每个GOP中的B帧依据预测的先后次序视作不同的等级，较高等级的B帧可以继续为较低等级的B帧提供预测参考。其具体示例如图12.6所示，以视点0为例，最高等级是GOP两端(T_0 和 T_8)的I帧或P帧，第二等级为这两帧预测得到的 T_4 时刻的B帧，第三等级为由 T_0 、 T_4 的B帧预测得到的B帧(T_2)和由 T_4 、 T_8 预测得到的B帧(T_6)，用类似的方法可得到最低等级的4个B帧(T_1 、 T_3 、 T_5 、 T_7)。其余视点内也是采取这种等级B帧的预测方法。

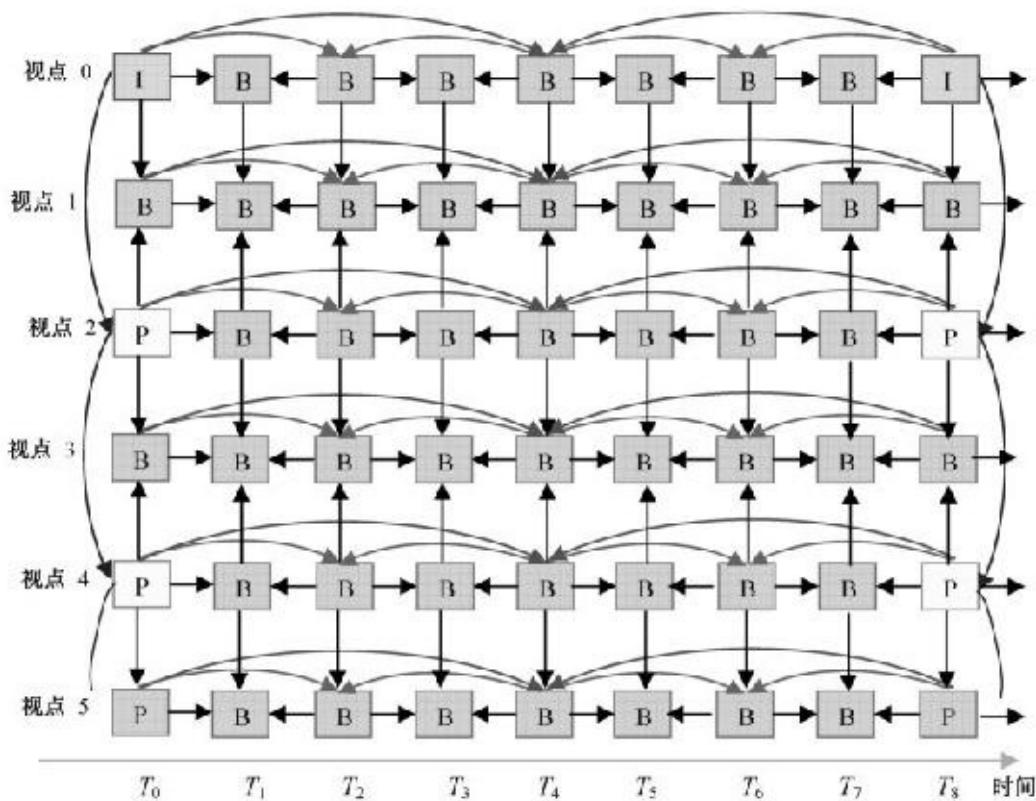


图12.6 Hierarchical B 预测结构

图12.6的预测结构中包含6个视点（视点0~视点5），为保证画面的流畅性，需定期插入关键帧（或称“锚帧”），关键帧有效的范围即是一个GOP的长度。图12.6中GOP的长度为8，其中视点0为基本视点，在编码时仅使用时域等级B帧预测而不使用视点间预测。其余视点为非基本视点，编码时同时使用时域等级B帧预测和视点间预测。图12.6结构中，除基本视点外，视点2、视点4中关键帧（如图中的T₀、T₈时刻的帧，可以是I帧或P帧）只使用了视点间预测，其余视点的所有编码帧都可以使用时域预测和视点间预测。为允许随机访问，每个GOP最好以I帧开头。

在这种预测结构中，一个GOP除了关键帧图像，剩下的都是非关键帧，也都是采用双向预测的B帧图像。等级B帧结构大大增加了B帧图像的数目，编码图像和参考重建像之间的相关性得到加强，与传统的IPPP（每帧I图像后紧接3帧P图像）或者IBBP（每帧I图像后紧接2帧B图像，再接1帧P图像）编码结构相比，在编码增益上有较大的提高。如图12.6的Hierarchical B帧结构还可同时实现时域上的可分级特性编码，其中所有的关键帧图像构成基本层，不同等级的非关键帧B图像构成各级增强层。

12.1.4 H.264/AVC的多视点编码

为满足3D电视、多视点视频电视等新兴媒体通信的要求，JVT于2006年开始制定基于H.264/AVC的多视点视频编码（MVC）的标准，简称H.264/MVC或MVC。鉴于MVC和以往的视频编码有很多相似之处，因此JVC并没有将MVC制定为一个独立的编码标准，而是将它作为H.264/AVC的一个扩展方案，即附录H，可参见2010年6月ITU-T发布的H.264/AVC标准。

MVC采用了H.264/AVC的编码工具，用于消除同一视点内和不同视点间的冗余度，对MVC的编码框架、比特流结构和高层语法操作了一系列规范。其中MVC的视点间—时间预测结构提高MVC的编码效率，同时还

具有随机接入、可分级编码等功能。为便于对各种基于H.264/AVC的MVC技术的研究和比较，JVT还提供了联合多视点视频编码模型（Joint Multi-view Video Model,JMVM）及相应的参考软件，公布了MVC的通用测试环境。

1.MVC的预测结构

如前所述，多视点视频在预测结构上与H.264/AVC 单视点编码最大的不同就是采用了视点间预测，即处于某一视点内的当前帧既可采用本视点内的帧作为参考帧，又可采用相邻视点的帧作为参考帧，从而不仅能减少时间冗余度，还能减少视点间冗余度。另外，与H.264/AVC 类似，MVC的预测过程也具有自适应性，即能根据率失真优化原理在视点内或视点间寻找到最佳的匹配数据，使得编码代价最小。

当前帧在相同视点内的参考帧称为视点内参考帧，在不同视点间的参考帧称为视点间参考帧。为防止MVC预测复杂度过高而编码增益却十分有限，MVC标准限制了视间参考帧必须和当前帧处于同一个接入单元内（AU）。对于 MVC 而言，接入单元指在某个时间点所有视点的视频集合，如图12.6等级B帧预测结构中时间点T₀处的接入单元为该时刻的视点0到视点7的所有帧的集合。

MVC没有要求固定的预测结构，但要求具有可分级性。为了在MVC语法中支持可分级性，在设计编码器时，需要考虑预测结构能否实现可分级的功能。图12.6给出的Hierarchical B是JVC接受的JMVM参考软件中一种具有可分级功能的预测结构。在这种预测结构中，视点0是基本视点，相当于基本层。B帧的分布具有层次性，对基本视点而言，图中时间点T₄的B层是第一时间层，时间点T₂、 T₆是第二时间层，时间点T₁、 T₃、 T₅、 T₇是第三时间层，这些都相当于增强层。

视点间预测的实现需要改进H.264/AVC的参考帧管理技术，即在解码图像缓存（DPB）中不仅要保存当前帧所在视点的解码帧作为参考帧，还要保存其他视点的帧作为参考帧，并在有关的参考帧列表中出现。MVC中

设定一个接入单元中所有帧可以同时输出。为有效使用DPB，在DPB管理过程，包括解码图像保存到DPB中的过程、参考帧的标志过程、解码图像从DPB中输出和移除的过程等方面，都做了进一步的修改。

2.MVC的码流兼容

在很多场合中，都需要能在MVC比特流中无障碍地提取出某个视点进行播放、观看、编辑、存储等操作。如在电视系统中，需要能够提取出基本视点数据在传统电视终端上解码和播放，而3D终端则可以结合解码非基本视点，提供立体视频服务。这就要求MVC码流结构能够向下兼容2D视频解码，即要求在MVC的比特流中，必须保证基本视点数据的存在并可独立解码性。

MVC的向下兼容性可通过使用NAL单元的类型结构来实现。基本视点可使用原有的NAL单元类型进行封装，而其余增加的视点则使用了一种新的NAL单元类型——编码条扩展（coded slice extension）。此类NAL单元在H.264/AVC的可分级视频编码（SVC）中也有使用，可用标志位（svc_extension_flag）标明对应的NAL单元属于SVC还是MVC比特流。传统的解码器不能识别这种NAL类型，因此会丢弃其所指的视点，只解码基本视点。而MVC解码器则可识别这种NAL单元的扩展类型，可根据需要对非基本视点进行解码。与H.264/AVC类似，特定的MVC比特流也属于某一档次和等级，处于对应档次范围内的解码器才能对此进行解码。

另外，基本视点不可避免地带有MVC的一些特性，使得H.264/AVC的NAL单元的语法不能对此进行描述。因此，MVC针对基本视点专门提供了一种新的NAL单元，前缀NAL单元（prefix NAL Unit），它也源自SVC语法，用于提供MVC信息。前缀NAL单元不能被传统解码识别，因此也可以被丢弃，从而不影响对基本视点的解码。不同于H.264/AVC的NAL单元头部只有一个字节长度，MVC编码条扩展NAL单元和前缀NAL单元的头部有4个字节，包括了如视点号（view_id）、时间层级（temporal_id）、视点间预测标识位（inter_view_flag）等MVC专有信息。

视点的识别、视点间的依赖、操作点的级别等信息在SPS MVC扩展（Sequence Parameter Set MVC Extension）中定义。视点识别信息包括了视点的数目和视点的解码顺序；视点依赖信息给出了任一视点所需的参考视点，结合NAL头部MVC扩展中的视点间预测标识位，可以确定某帧的参考帧列表中视点间预测参考帧所在的视点。MVC 某个接入单元的比特流结构如图12.7所示，其中SPS为序列参数集，IDR为即时解码更新帧。

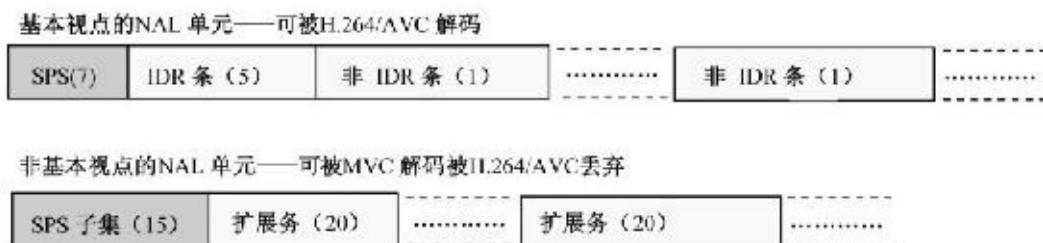


图12.7 MVC的NAL单元比特流

3.MVC的高层语法

H.264/AVC的语法和语义是针对单视点平面视频的，MVC作为H.264/AVC的扩展部分，必然要包容MVC的定义和操作，因此需要对原H.264/AVC语法做适当的补偿和修改。具体而言，H.264/AVC需要根据多

个视点联合编码的特点，扩展自己语法和语义。目前，关于MVC 扩展的高层语法主要有以下4方面：

(1) 在序列参数集 (SPS) 中添加语法元素，用来说明视点的数目以及定义各视点之间的预测关系；

(2) 扩展NUL单元的语法元素，添加当前NAL单元属于哪个视点，是否作视点间参考图像等信息；

(3) 制定适合MVC 的参考图像管理机制，包括参考图像标记、参考图像列表初始化以及参考图像重排序等；

(4) 扩展H.264/AVC补充增强信息 (SEI)，使其能够支持MVC中所定义的各项功能需求，如视点可分级性、支持观看部分视点、并行处理等。

12.2 HEVC的多视点扩展

MPEG组织于2011年在日内瓦会议上公开征集基于HEVC的3D视频编码技术的提案，德国HHI (Heinrich-Hertz-Institute) 通信技术研究所继而提出的基于HEVC的多视点加深度3D视频编码方案，经过性能比较和主观评价之后，比起其他方案效果更好，于是将该方案纳入HEVC的3D扩展部分。目前，由ITU-T和ISO/IEC组成的3D视频编码联合合作组 (JCT-3V) 一直致力于对基于HEVC的多视点视频和3D视频编码进行改进，同时，包括支持先进3D功能的深度图编码也在研究中。

JCT-3V于2012年推出了用于MV-HEVC和3D-HEVC测试模型 (MV/3D-HEVC Test Model,HTM)，此后一直在不断更新，目前是2015年2月的第11版。

1.多视点HEVC

HEVC的多视点扩展 (MV-HEVC) 使用与以往H.264/AVC的MVC扩展相同的原理。多视点视频编码的关键技术就是通过对HEVC参考图像列表 (RPL) 结构的修改来保证层间预测，以使在同一时刻来自其他视点能被用于预测。这种视点间的视差偏移在预测过程中被补偿，替代因时间差引起的运动偏移。扩展工作主要包括：适当地扩展高层语法和存储所需的其他视点的解码图像为参考图像。

对高层语法的扩展包括设置不同视点间预测关系的标志，图像所属层的标志，以及有助于提取基本视点的语法元素。这种结构一个最大的好处就是不用改变slice头层以下的单层HEVC语法或解码过程，即允许使用现有的HEVC编码器和解码器实现，对立体和多视点应用没有重大的变化。

2.MV-HEVC的档次

当前多视点扩展的仅包括立体主 (Stereo Main) 档次，它是基于

HEVC的Main档次的。因此，Stereo Main支持8比特YCbCr 4:2:0视频。这一档次允许相同空间分辨率的两个视点的编码，一个独立视点，一个依赖视点。

MV-HEVC今后设立的档次可能定义在将来修改的文档中，可允许多于2个视点的编码。进一步，辅助图像的定义将保证支持每个视点深度图的传输。要注意的是，按照辅助图像的定义，在这种情况下，由辅助图像来预测基本图像是不允许的。

12.2.1 MV-HEVC编码系统

为取得更高的压缩性能，还要保持和HEVC单视点视频编码的后向兼容性，多视点编码设计了一种改进的编码系统。在这样的系统中，为了可以提取出单视点视频，基本视点还可以完全和HEVC兼容，使得其具有只有依赖视点才需要使用附加编码的特点。

1.多视点的码流结构

基于HEVC的多视频编码编解码系统结构如图12.8所示，将视频序列、深度图和摄像机参数送入多视点HEVC视频编码器中，主视点由传统的2D编码器来编码，其他视点则使用它们之间的相关数据来编码。编码后得到多视点视频的单个比特流。码流中可以包含摄像机参数，以便用于解码端合成中间视点。编码端还允许不输入深度图，即深度序列可不参与编码。

比特流若被3D视频解码器解码，则可得到视频序列、深度图和摄像机参数。当然，若深度图未参与编码，这时仅得到视频序列和摄像机参数。若深度图参与了编码，如图12.8的右边所示，可以通过MV-HEVC解码，利用基于深度图绘制技术生成N个视点序列，即可提供多视点立体显示；也可以通过双目视频解码，利用基于深度图绘制技术生成2个左右视点图像，即可提供双目立体显示；最简单的操作可以直接抽取其中1个视频序列进行普通的2D视频解码，即可提供传统二维显示。若深度图未参与编码，之后利用图像域变形（Image Domain Warping，IDW）技术生成中间视点，也

可提供多视点立体显示、双目立体显示以及传统二维显示。

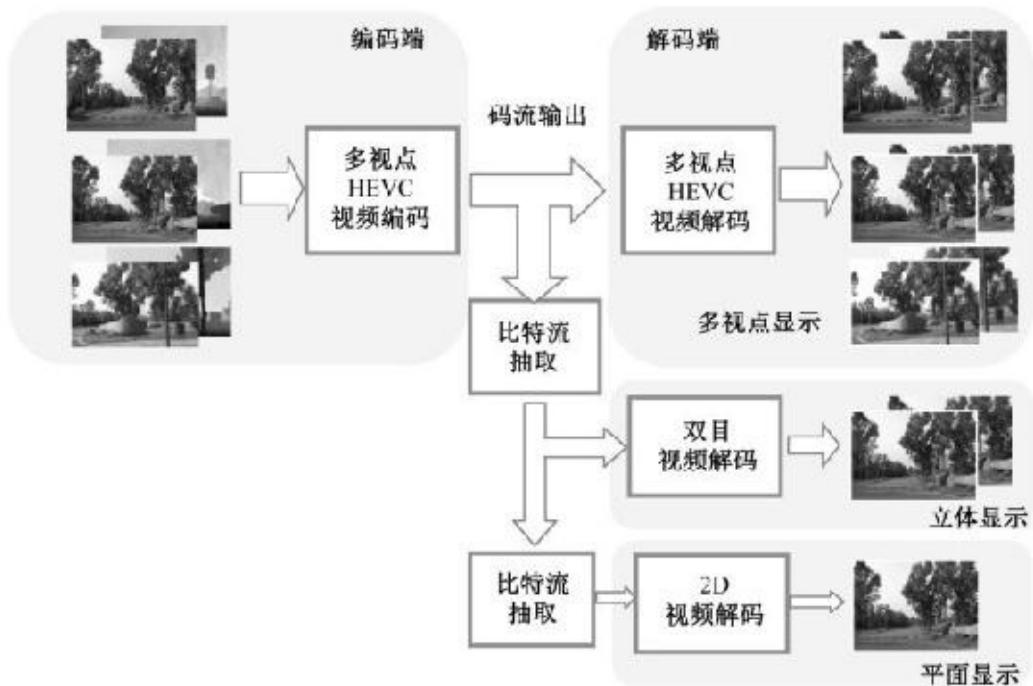


图12.8 MV-HEVC视频编解码结构

2.多视点的编码结构

基于 HEVC的3D视频编码框架如图12.9所示。同一时间 N个视点按0,1,

$\dots, N - 1$ 的顺序被依次编码，每个视点均包括视频帧和深度图，同一时间呈现同一场景的 N 个视频帧加深度图被称为一个接入单元。参考视点 0 因不参考其他视点称为独立视点，或基本视点；视点 $1, 2, \dots, N - 1$ 因参考了其他视点称其为依赖视点。每个视点先编码视频帧，再编码深度图。

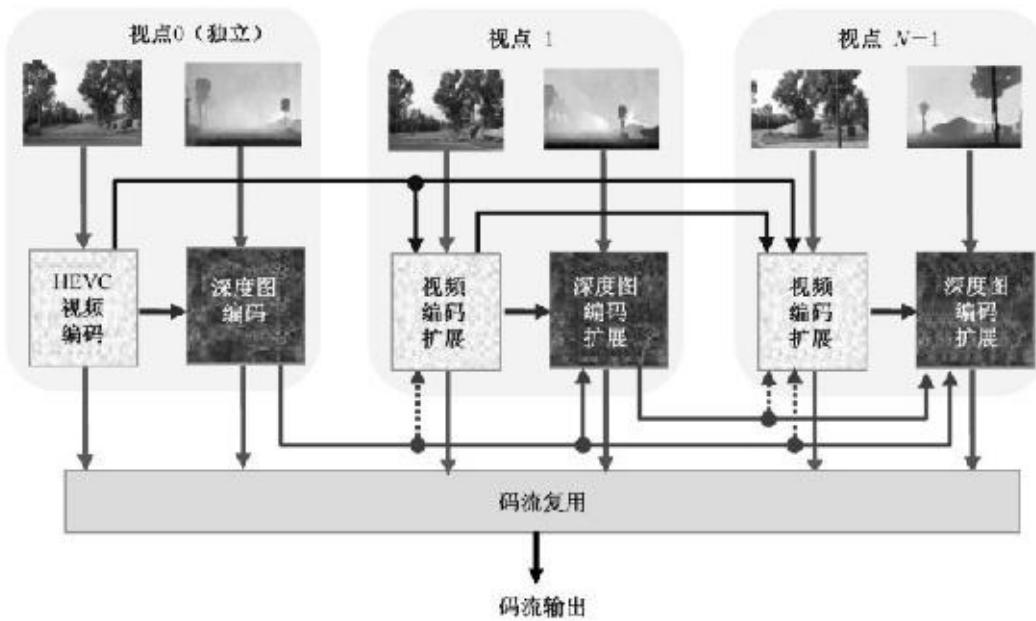


图12.9 3D-HEVC的视频编码结构

独立视点0的视频帧通过第1版HEVC 编码器编码，相应的深度图则通过改进的HEVC深度图编码器编码。依赖视点的视频帧通过改进并增加了视点间预测的3D-HEVC 编码器编码，相应的深度图序列则通过改进并增加了视点间预测的HEVC深度图编码器编码。

由图12.9可以看出，同一视点内，深度图可通过已编码的对应视频帧来预测（对象间预测）。不同视点间，未编码的视频帧既可通过参考视点的已编码视频帧来预测，也可通过参考视点的已编码深度图来预测。未编码的深度图只能通过参考视点的已编码深度图来预测。

3.多视点的预测结构

一个典型的多视点视频编码的预测结构如图12.10所示，和图12.6类似，是一种等级B帧预测结构。其中，视点0标注为基本视点（依赖视点），在非基本视点（视点1或2）中的一幅图像能够从同一时刻的基本视点图像来预测；粗线框标注的图像属于时域随机存取点；由“I”标注的图像仅用于帧内图像预测；由“P”标注的图像为使用单向帧间预测；由“B”或“b”标注的图像使用双向预测，但标注为“b”的图像不用于时域预测。预测是自适应的，因此在时域和视点间参考中（或双向预测平均，或加权预测），借助于率失真代价函数，可以获得基于块的最优预测器。

层间样值预测保证HEVC的灵活的参考图像的管理能力。编码过程中，为了用于预测过程，来自其他视点的解码图像被插入当前视点的参考图像列表（RPL）中。结果，RPL 中既有当前视点的时域参考图像，也有来自同一时刻相邻视点的视点间参考图像，它们都可能用于当前图像的预测。由于这种设计，可以保证块级的解码处理部分保持不变，基本和HEVC一致。但是，如上所述的少量变化还是需要。例如，跨视点的预测依赖性的标识等。

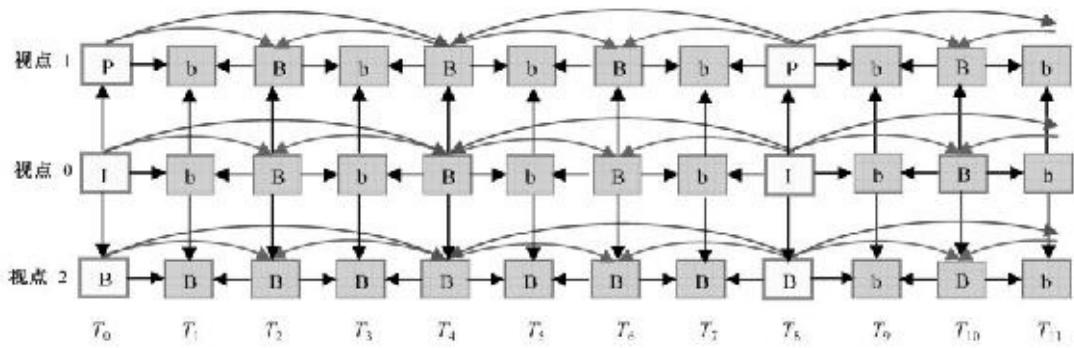


图12.10 多视点预测结构的一例

12.2.2 多视点编码工具

预测是提高视频编码效率的有效措施之一，和普通的预测视频编码类似，多视点视频编码也可采取预测方式编码。除一般二维视频中的帧内预测和帧间预测外，多视点视频帧间的相似性也给多视点视频之间提供了预测的可能性，这就是视点间预测。通过视点间预测，我们希望得到当前编码块运动参数的预测值和视频残差的预测值。视点间预测并没有改变块层语法和解码过程，仅改变了高层语法，在参考帧列表中增加了同一接入单元的其他视点的已编码视频帧。

在多视点编码中，在对独立视点进行编码时，使用传统的 HEVC 编码器，不参照其他视点的数据。在对依赖视点进行编码时，除常规的HEVC 方法外，为了更有效地利用视点间冗余，增加了多种与视点间预测有关的编码工具，如视点间运动参数的预测、视点间残差数据的预测和亮度补偿等。

1.视点间运动参数的预测

多视点视频是从不同的角度拍摄同一场景所获得的视频信号，因此，各个视点图像之间存在视差，即场景中同一点在各视点中投影位置之间的相对偏差，可具体表示为视差矢量（Disparity Vector,DV）。在多视点场合，场景中的运动物体在同一时间、不同视点间的运动特性和视差特性具有相似性，那么不同视点视频间的运动参数就存在着冗余，我们可以用同一时刻参考视点的已编码块的运动参数来预测当前依赖视点编码块的运动参数。例如，用参考视点块的运动矢量来确定当前块的运动矢量，如图12.11所示。由当前编码块B₁的视差矢量（DV），指向参考视点T₁时刻帧中的参考编码块B₀。这样，T₁时刻依赖视点1的运动矢量MV₁就可以从同一时刻参考视点0的运动矢量MV₀导出，即MV₁=MV₀。这就是一种当前编码块利用基于相邻块视差矢量（Neighboring Block based Disparity Vector,NBDV）从参考视点导入运动矢量的方法。

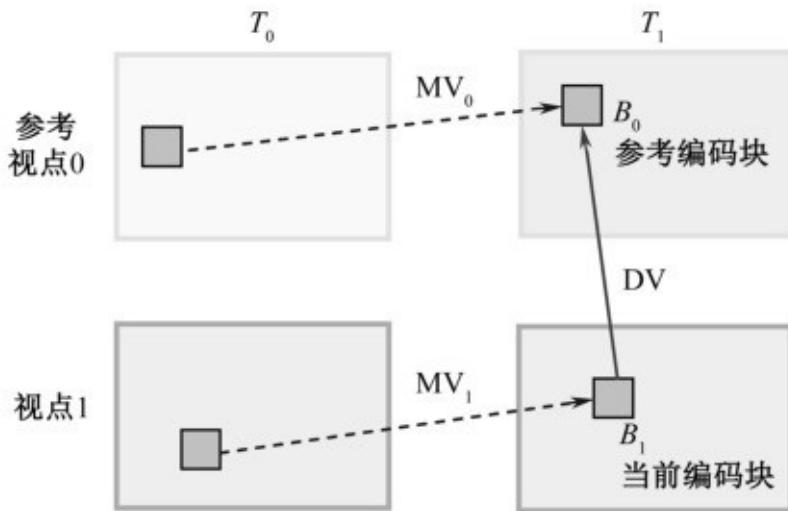


图12.11 视点间运动预测示意图

(1) 基于相邻块视差矢量 (NBDV)

类似于经典视频编码的运动估计中寻找匹配的宏块，为寻找不同视点的匹配块，引入了基于相邻块视差矢量 (NBDV)，并将它用于3D-HEVC中设计成类似于HEVC中高级运动矢量预测 (AMVP) (第6章称为Inter模式) 和Merge的方式。但从相邻块导出的视差矢量是唯一的，因此不需要花费附加比特来标志。NBDV的基本想法就是将有效的视差矢量用于空间和时间相邻块的视点间的样点预测。

空间相邻块的位置和用于HEVC中的AMVP、Merge模式的候选块是相同的，在图像中具有相同的块取出顺序： A_1, B_1, B_0, A_0 和 B_2 ，如图12.12所示。但由于它们中很可能没有一个使用视点间参考，因此就需要再检查时域相邻块。一旦识别到视差矢量，当前块的视差矢量就被导出，其值和相

邻块的视差矢量相同，全部 NBDV 过程结束。当需要时，例如视点间运动预测和视点间残差预测，就用这个视差矢量去识别视点间参考图像中的参考块。如果从邻近块中没有发现视差矢量，NBDV过程返回一个0视差矢量。

(2) 3D-HEVC的Merge/AMVP模式

3D-HEVC利用视差进行补偿预测的概念晚于HEVC的运动补偿预测（MCP），但和 MCP 非常相似，视差补偿预测是用于同一时刻的不同视点之间（在同一个接入单元内），而MCP则是用于同一视点内时间相邻的视频帧之间。它们都是在相邻的参考中选出最好的“候选”块，同时产生指向这一位置的索引。

在3D-HEVC中，视点间运动预测是通过引入附加候选到Merge模式列表中来实现的，而高级运动矢量预测（AMVP）模式则保持不变。此后 Merge列表包含了6个候选，比HEVC中5个候选多一个。AMVP候选列表的结构和HEVC的候选列表相同，2个附加候选可这样布置如下。

第一个候选是运动矢量，又称为视点间候选，对应于由NBDV在视点间参考图像中发现的块的参考图像索引，如图12.11所示。

第二个候选是NBDV用视点间参考图像索引导出的视差矢量。类似于 HEVC的Merge 过程，通过比较来自空域相邻块A₁和B₁候选，对附加候选加以删剪，如图12.12所示。

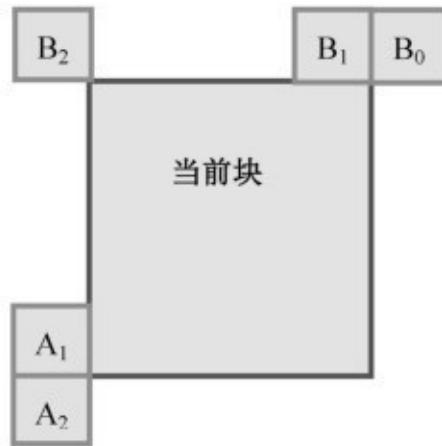


图12.12 NBDV空域相邻块的取出

2.视点间残差数据的预测

同一时间，不仅不同视点对应视频块的运动参数存在冗余，残差也存在冗余。类似于视点间运动预测，仅需对当前视点的残差块与参考视点对应残差块相减得到的差值进行变换编码。这就是高级残差预测（Advanced Residual Prediction,ARP）方法，它利用的就是两个视点间的运动补偿的残差信号间的相关性。

如图12.13所示，使用运动矢量 V_D ，在当前非基本视点中完成块 D_C 的运动补偿。首先，NBDV矢量识别出视点间预测块 B_C 。然后在基本视点的重建 B_C 和相应的重建 B_r 之间使用 V_D 进行运动补偿。再将来自 D_r 块的运动补偿的预测残差加到预测信号上。由于使用相同的运动矢量 V_D ，当前块的残差信号的预测更加精确。当ARP有效时，残差的预测可采用0.5或1的加权系数。

由于在基本视点处附加的运动补偿有可能增加存储量和计算量，为减轻这一负担，目前若干以较小的编码效率损失为代价的优化方法也已出现。例如，当视差矢量指向分数像素点时，基本视点的参考图像的残差块使用简单的双线性滤波器进行插值。

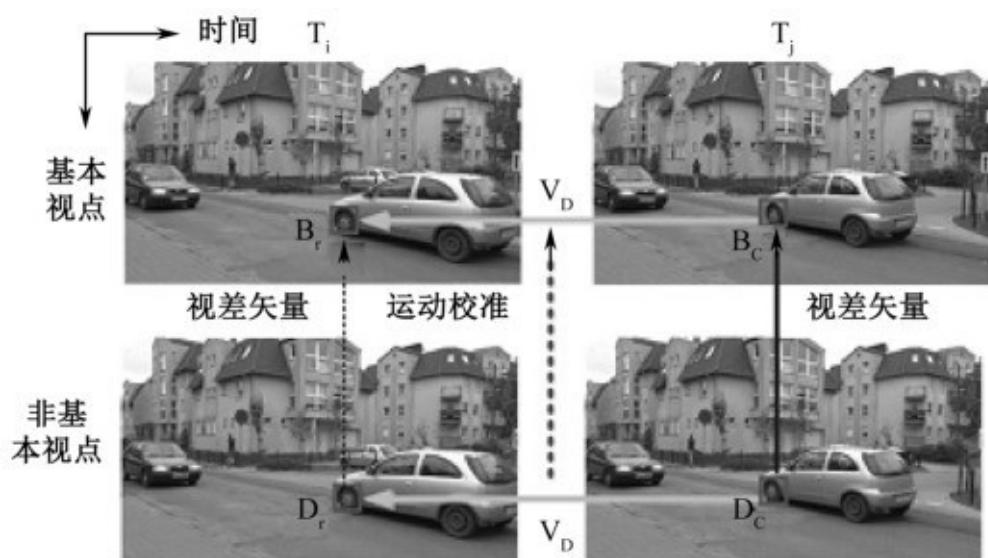


图12.13 高级残差预测的预测结构

3.亮度补偿

当捕获相同场景的不同视点的摄像机，由于彩色标度不统一或由于照明影响，预测可能不准。亮度补偿（Illumination Compensation,IC）技术就是为了处理这一问题的。这种方法仅用于视点间预测，对参考视点预测块的亮度和色度进行补偿，以匹配当前视点的光照，提高它们的预测精度。

对当前的 PU，在上面的相邻行、左边的相邻列的样点，以及视点间参考块的相应的相邻样点都是输入参数，这些参数通过标度一个伸缩因子 a 和一个偏移量 b ，可形成一个线性补偿模式。 a 和 b 的值由最小平方解来决定，同时考虑到 a 应当接近于 1 的限制。在参考视点中相应的相邻样点由当前 PU 的视差运动矢量决定，如图 12.14 所示。来自视点间参考的视差运动补偿后，增益/偏移模式应用于每个值，由 a 对它进行伸缩，并加偏移 b 。亮度补偿技术还可用于深度图。

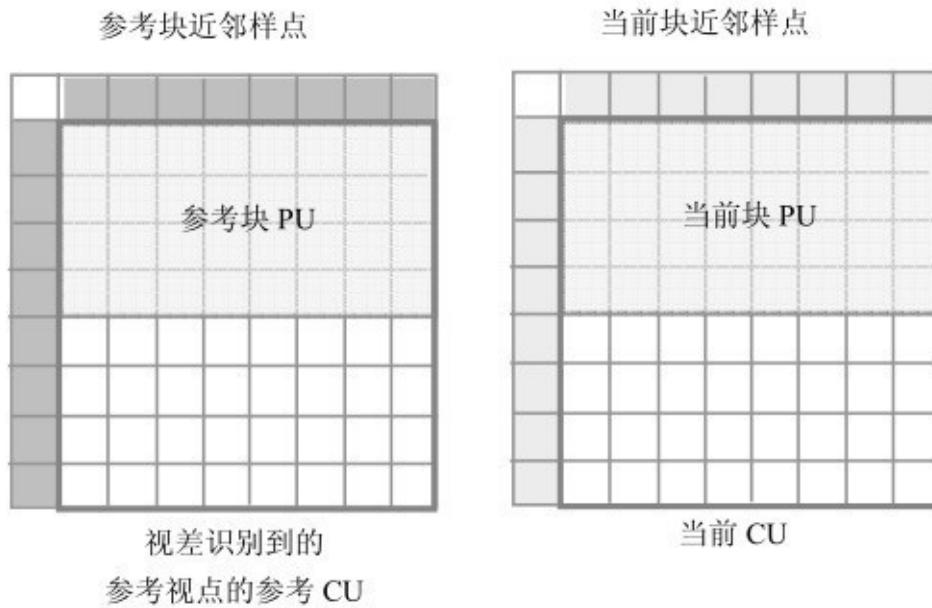


图12.14 用于亮度补偿的邻近像素

12.2.3 虚拟视点的合成

在多视点视频的应用系统中，观看者可以在一定范围内任意选择观察地点和角度观看视频内容。在多视点视频采集端，可以根据需要放置一系列摄像机。为了让观众在 360° 任何一个角度都可以选择观看场景，则需要将摄像机围绕场景摆放成一个圆圈；如果观众只需要较小角度的选择范围，则可将摄像机在场景前摆放成一个半圆或弧形甚至是直线。由于受存储和传输条件限制，不可能在所有位置都放上一个摄像机，只能在一些采样点上放置摄像机进行拍摄，然后利用两个或者多个采样点上摄像机的视频和摄像机参数信息合成中间视点的图像。如图12.15 (a) 所示，场景周

围放置了12个摄像机进行拍摄。在接收端，如果不采用任何技术，观看者只能从12个不同的地点和视角进行选择。如图12.15（b）所示，通过图像合成技术合成12个虚拟视点的图像后，观看者就可以从24个不同的地点和视点中进行选择。

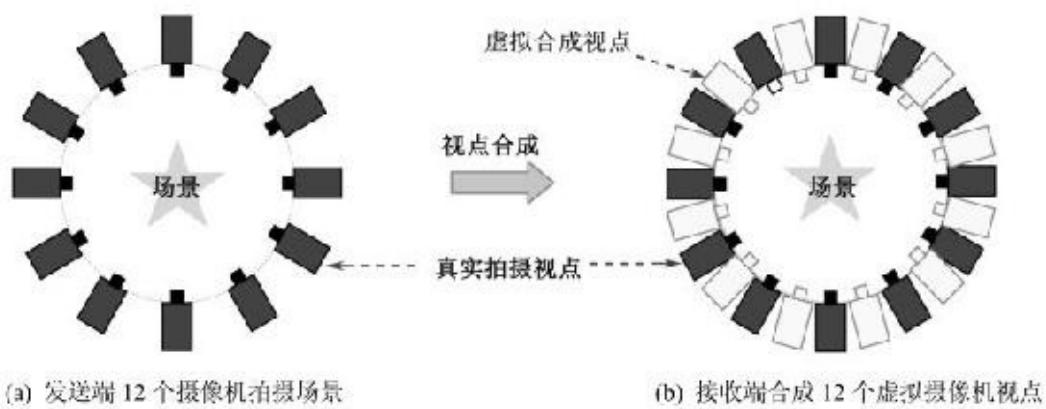


图12.15 视点图像合成技术

虚拟视点图像合成（简称虚拟视图合成或视图合成）是指依据两个或多个关于同一景物的真实视点图像，合成出的虚拟视点图像（虚拟视图）。视图合成已在多视点视频等领域中有广泛的应用，同时它可以视为

一种有效的视频压缩手段，用来实现视频帧的内插。

目前，具体的视点合成的方法可以分为两大类：基于纹理映射的三维模型的绘制（Model Based Rendering,MBR）法和基于图像的绘制（Image Based Rendering,IBR）法。

1. 基于模型的绘制方法

在基于3D模型绘制（MBR）的方法中，根据多视点系统采集的数据，使用两幅或多幅图像，首先为场景对象建立三维几何模型，可以较精确地刻画物体的形状、运动、变形等信息。在三维绘制时，通过三维模型在虚拟视点的位置，进行明暗处理、隐藏面消除等过程，往往还需要进行模型表面网格的划分和多种不同的纹理映射。这样绘制出的新视点的图像质量较高，这是 MBR 方法的优越之处。基于模型绘制方法的另一个优点是参数的存储量较低。由于事先建立了场景中对象的模型，这些已经参数化的模型能在一定范围内进行自由运动或发生位移，因此大大减小了不确定性造成的参数存储量。

尽管基于模型的绘制方法与其他绘制方法相比，绘制质量较高，数据存储量较小。但基于模型绘制的方法需要依靠传统的绘制手段来实现，包括模型变换、视点变换和消隐处理等，存在计算复杂、实时性差等缺陷，并不适合在实时多视点视频中应用。

2. 基于图像的绘制方法

IBR 是利用参考图像及其对应的每个像素的深度信息来合成场景中虚拟视点的过程。这一过程主要包括两步：利用深度信息将平面图像映射到三维空间的对应位置；将三维空间上的点映射到二维平面，形成新的视图。IBR 方法直接利用参考图像合成新的视图，避免了复杂3D模型的重建过程，能够实时合成高质量的中间视图。随着计算机视觉及图像处理技术的发展，现已提出多种IBR算法。其目的都是从采集系统获取一定数目的原始图像，然后通过所设计的合成算法生成所需位置的新视点图像。

根据采用场景几何信息的比例，可将IBR技术方法分为三类，一类是不使用几何信息的IBR方法，如光场控制的方法；另一类使用明确几何信

息的IBR方法，如基于视点的纹理形变的方法；介于两者之间的第三类是使用隐含几何信息的IBR方法，如视图插值方法和基于深度图像的绘制方法等。

基于图像绘制方法的主要优点是视图插值可直接从已知图像插值生成新的图像，因此新图像的绘制时间与场景的复杂程度无关。但这种方法也有不足之处，一般很难寻找真实图像对应关系，往往比较适用于视点比较相近的图像。如在多视点系统的IBR中，根据已知两个视点生成基线范围内新视点是相对比较简单，但若在存在较多遮挡以及在大基线情况下的视点生成仍然没有得到很好地解决，而自由视点的绘制则要更复杂一些。

12.3 HEVC的3D扩展

JCT-3V正在开发中的3D-HEVC是新的3D视频编码标准，它是HEVC标准的扩展，用于对3D视频内容的编码。3D-HEVC不但可使用所有的HEVC中的编码纹理视频的特性，还增加了改进编码性能的某些特性。

此外，3D-HEVC允许视频以多视点加深度（Multiview plus Depth,MVD）的数据方式来表示。MVD格式数据中包括来自不同视点的一系列纹理帧信息和相应的深度图信息，它们都在一个比特流中发送。而位于编码视点之间的虚拟视点可用视点合成技术自适应地产生，这就是基于深度图的渲染（Depth Image Based Rendering,DIBR）技术。然而，3D-HEVC编码器并未限制编码过程只用MVD格式，它还可用没有深度图的多视点格式。

3D-HEVC使用和HEVC相同的方法划分编码块为CTU。每个CTU最大可至 64×64 样点，CTU可分裂为更小的编码单元（CU），成为用于帧内或帧间编码的基本单元。每个CU还可以再分裂为预测单元（PU），PU可以是对称的或不对称的。对称的PU可能是方形的或非方形的，都可用于帧内（只用方形PU）和帧间预测。非对称PU仅用于帧间预测。

在3D-HEVC标准中每个纹理通道都是和深度图关联的，对于独立视点，3D-HEVC不限制它的纹理及其深度图的编码顺序。但一般情况下总是先编码一个纹理帧，然后编码和它相对应的深度图。还有，正被编码的（深度或纹理）帧可以使用同一接入单元中（如在相同的时刻）对应的先前已编码帧的信息。

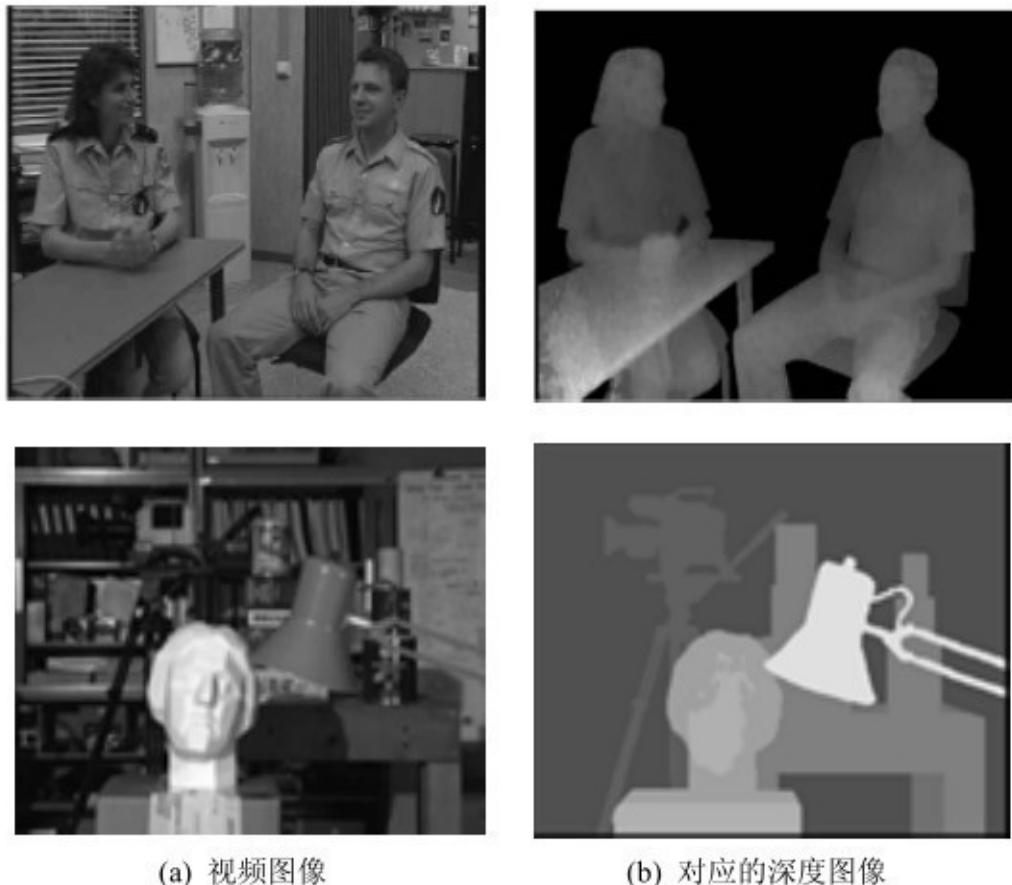
12.3.1 立体图像的深度图

1.深度图

3D视频中的深度表示物体距离摄像机的距离，深度图顾名思义就是表示图像场景中不同物体深度数据的图像。一般深度图本身也是一幅256级的灰度图像，常常用0（即黑色）代表离摄像机最远，255（白色）代表离摄像机最近。根据立体成像的几何模型，只要有一个视图及其深度图（包括摄像机参数），就可以根据它们的几何关系，推断出另一个视点图像。由此可见，深度图直观地提供了二维图中物体的深度线索。有了普通二维视频图像及对应的深度图，就能由此生成另一个视点图像，再通过立体显示器使人眼观看到三维视频了。

图12.16所示是一对典型的“二维视频+深度图”格式的图像，图12.16（a）是2幅普通的二维图像，12.16（b）图是表示各自场景深度的Z轴图像，是图12.16（a）对应的深度图。这2幅深度图是采用主动可测量目标距离的摄像机产生的，在拍摄视频的同时也将对应的深度信息做了记录。

在三维视频系统中，分别对二维图像数据和深度图数据进行压缩编码，编码后的信息可以通过传输或存储送到用户端。用户利用深度图可以重建出另一个视点图像，采用三维显示器即可渲染出三维视频内容，达到三维显示的效果。



(a) 视频图像

(b) 对应的深度图像

图12.16 视频图像和对应的深度图

2.“视频+深度图”表示

双目视频和视频加深度都可以表示场景的立体信息，前者比较直接，两路视频直接对应左右眼，后者属于间接的表示方法，需要经过转换才能形成双目视频，但两者从理论上说是等价的，完全可以相互转换的。在视频编码领域，对比场景的立体图像的双目视频表示，这种“2D视频+深度信息”的表示方式具有明显的优点。

(1) 数据量不大。由于深度图仅是灰度图像(单色)，灰度表示的是距离，并不表示物体的纹理等信息，内容相对简单。因此，这种“视频+深度图”的格式较双路视频的格式所需的数据量较少，十分有利于传输与储存，只需要附加不多的信息量就能获得三维视频效果。

(2) 和二维视频系统兼容性好。由于“2D视频+深度信息”格式数据与现行二维系统中的视频图像数据有诸多共同点，可采用相同的压缩和传输方式，不需要做特殊处理，能最大限度地与现有二维视频兼容。而且二维图像与深度图是可以分离的，接收端既能够利用深度图信息重建出三维效果，也可以单独提取二维图像信息进行平面显示。

(3) 资源利用率高。可以利用现有的大量的二维视频，近似提取它们的深度信息，形成准三维视频，大大开拓了立体电视的节目源，这是这种方式的最大优点。

当然，深度图也存在两方面不足之处。

(1) 表示能力有限。深度图是一幅灰度图像，一般情况下灰度级只有256级，要反映实际场景中物体巨大的距离差别，从几十厘米到几十千米，它对场景中物体深度的表示能力还是比较有限的。对于结构比较简单、目标运动平缓、距离摄像机较近的场面，其表示效果比较好。反之，深度图的效果就不够理想了。

(2) 深度信息获取有困难。如何获取深度信息是一项尚未完全解决的问题，获取不准也会严重影响立体显示的效果。

12.3.2 深度图的编码

单路“视频+深度”的3D视频格式，在多视点情况下就是“多视点视频+深度”(Multiview Video plus Depth,MVD)格式。深度数据表示了所捕获视频场景的基本几何结构，一般情况下需要每一个视点的相关深度数据。解码端在解码所传输的视频图像和深度图的基础上，还可以利用这些数据产生新的视点数据来改进多视点视频的整体显示效果。接收端这些附加的

视点可以使用深度图像用基于深度图的渲染（ Depth Image Based Rendering,DIBR ）技术来产生。

1.多视点数据的编码顺序

与 H.264/MVC 类似，在多视点视频场景中同一时刻所有视频图像和深度图组成了一个接入单元（ AU ），编码器按照输入的MVD信号的接入单元的顺序进行编码，实际上是按照时间顺序，一个接一个地对接入单元进行编码。

在一个接入单元内，首先编码“独立视点”视频和深度数据，然后再编码其他非独立视点的数据。

在每个视点内，视频图像首先编码发送，随后编码发送深度图，一幅视频图像后总是直接跟随其相应的深度图。

从概念上说，每个视点的视频和深度信号都可使用基于HEVC编码器来编码。多个视点的比特流被多路复用形成综合的3D视频比特流。独立视点使用标准的HEVC编码器来编码。对应的子比特流可以从3D比特流中提取出来，用HEVC解码器解码，然后在普通的2D显示器上显示。其他的分量使用扩展的HEVC编码器编码，HEVC的扩展包括附加的编码工具和视点间预测技术，这种预测使用在同一接入单元中的已经编码数据。

2.深度图的编码考虑

深度图像表示场景中对象的深度信息，可以采用一般的灰度图像压缩方式对其进行编码。但深度图是和对应的纹理图像紧密相关的，它的统计特性有别于传统的灰度图像，具有陡峭的边缘和大片的平坦区。而且在解码端获得的深度图像不是为了显示，而是为了辅助合成新的视点，因此在对深度图进行压缩编码时就要充分考虑到这种图像数据的特性。这里着重考虑的是如何利用相应的纹理图像信息来提高深度图像编码的效率。

（1）利用纹理图像的边缘信息编码

图像与传统2D 图像区别之一在于深度图像对象内部区域较平滑，对象边缘部分较锐利。在深度图像的频谱中，低频信息和高频信息占据主导。传统的视频编码方法较多保留了图像的中低频信息，压缩了图像的高频信

息。而在深度图像中，高频信息主要是对象的边缘部分，如果直接利用传统编码方法来压缩深度图像会造成边缘信息的损失，导致在利用深度图像合成视点时，合成视点像素位置较理想位置出现偏移，影响了合成视点的质量。因此，为提高合成视点的质量，在编码深度图像的时候要重点保护图像边缘信息，提高边缘信息的重建质量。

(2) 利用纹理图像的运动信息编码

在深度图像编码中，可以利用深度图像与对应纹理图像间的相关性来设计视频图像和深度图像之间共享运动信息的编码策略，提高编码效率。例如，可以利用对象运动的一致性，将视频图像中运动信息直接用于深度图像的运动补偿编码，省去了编码深度图像时运动估计，在提高编码效率的同时减少了编码时间。

(3) Skip模式编码

此外，还可以通过增加编码中的Skip模式来提高编码效率，如利用深度序列对应的纹理序列的时域相关性和视点间相关性来决定是否对当前编码的深度块采用Skip模式，在很多情况下，大部分块采用了Skip模式，省去了复杂度较高的率失真模式选择，大大降低了深度序列的编码复杂度。

12.3.3 深度图的编码工具

为有效地压缩具有多个“视频+深度”信号格式的3D视频数据，利用这些分量中的相关性，3D-HEVC开发了若干深度图编码工具。在多视点场合，假设第一个视点为独立视点，其余为依赖视点。独立视点的视频分量可以由平常的2D HEVC独立编码，提供和现存的2D视频编码的兼容性。对每个依赖视点的3D 视频分量，即相关视点的视频分量和深度图，可以应用HEVC附件的3D扩展的编码工具。这样，一个3D视频编码器能够从一套通常的2D编码工具和附加的编码工具中，为每个块选择最好的视频编码方法。

深度图的编码和视频图像的编码一样也使用常规的帧内预测、运动补

偿预测、视角补偿预测、变换编码等方法。根据深度图的统计特性，3D-HEVC对传统编码器做了一些改进，如关闭对深度图边缘有不利影响的DBF和SAO环路滤波、采用整像素精度的运动补偿和视差补偿等。此外，为有效地编码深度图，还新增了一些深度图编码工具，它们聚焦于在编码过程中尽量减少边缘信息的损失。如基于划分的帧内编码，运动参数的继承，视点合成预测等。

1. 基于划分的帧内编码模式

3D-HEVC设计中，深度块只允许非矩形划分。在所有这些模式中，每个深度PU可被划分为一个或两个部分，每个部分用一个固定值表示，即DC值 P_0 和 P_1 ，如图12.17所示。每个部分的 DC 值使用邻近参考样点来预测，形成的残差值用于编码传输，获得信息的压缩。在基于划分的深度帧内编码模式中包括深度建模模式（Depth Modeling Modes,DMM）和简化深度编码（Simplified Depth Coding,SDC）两类方法。

| | | | |
|-------|-------|-------|-------|
| P_0 | P_0 | P_0 | P_0 |
| P_0 | P_0 | P_0 | P_1 |
| P_0 | P_0 | P_1 | P_1 |
| P_0 | P_1 | P_1 | P_1 |

(a) 楔形模式

| | | | |
|-------|-------|-------|-------|
| P_0 | P_0 | P_0 | P_0 |
| P_1 | P_0 | P_0 | P_0 |
| P_1 | P_1 | P_0 | P_1 |
| P_1 | P_1 | P_1 | P_1 |

(b) 轮廓模式

图12.17 深度图编码的PU划分一例

(1) 深度建模模式 (DMM)

DMM将深度块分割成的两个区域，每个区域内深度近似为常数。需要传递两个信息：分割的模式和每个分割区域的深度取值。实际传递的是深度值的预测的差值，预测值为空间左、上邻近的块深度值的平均值。DMM采用两类分隔模式，一类是楔形模式，另一类是轮廓模式。

楔形模式用直线分隔一个深度PU，如图12.17 (a) 所示。楔形划分模式标识可采用不同方法，可以从预定义的一套楔形模式中直接选择一个楔形模式来标记，也可以从重建的同位纹理块而导出相应深度块的划分模式。

轮廓模式则用区域边界链码 (Region Boundary Chain, RBC) 方式来处理区域边界，形成块的划分。首先找出块内边界，利用链码来表示该边界，再将链码转换为比特流，最后计算划分区域的预测值并填入块。在区域边界链码 (RBC) 中，链码表示的是分隔模式，每个链就是一个样点和它的8连通样点的一个连接，索引号为0到7，因此划分界线可以不是直线，如图12.17 (b) 所示。轮廓划分模式能够支持两种不规则的划分，其中每一种都可能包含分开的子区间，如图12.18所示。深度块的轮廓 (划分的边界) 由分析与该深度块对应的纹理块来决定。

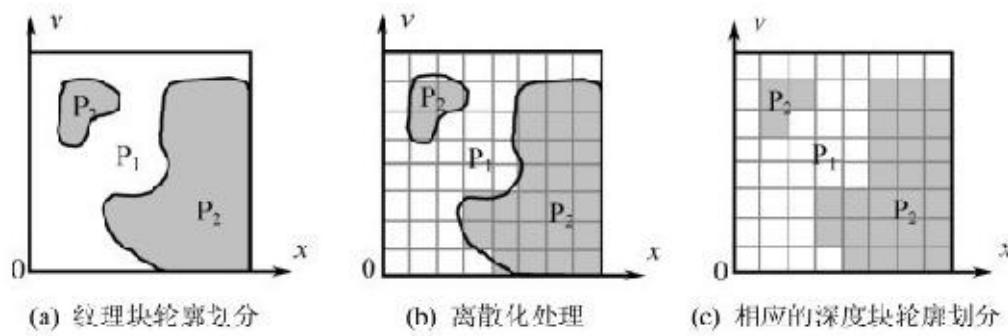


图12.18 由纹理块导出同位深度块内的轮廓划分

(2) 简化深度编码 (SDC)

用自然图像的压缩算法处理深度图像往往会产生较严重的振铃效应，在合成图像中会导致相应的几何失真。

对于深度图中较为平滑的区域，3D-HEVC采用了一种简单深度编码 (SDC) 的方法来简化深度帧内预测运算，深度帧内编码的CU的大小一般为 $2N \times 2N$ ，无须编码划分信息。当前编码CU选择SDC时，只需要传输3部分信息，且无须进行变换和量化，直接编码预测残差。

(1) 当前块的预测模式，可以是视频帧内预测的 DC 模式、Planar 模式或深度建模模式 (DMM)；

(2) 当DMM被选择时需要传输额外的划分信息；

(3) 根据深度查找表 (Depth Lookup Table, DLT) 计算出的原始像素和预测像素的索引之差并传输到解码端，当然还需要传输DLT以供解码端使用。

为减少深度表示的动态范围，建一个深度查找表 (DLT) 来转换深度值，特别适用于仅使用部分深度值时。DLT是在最初的处理过程中，由输

入序列的几帧深度图像分析构成。DLT充分利用深度图像的特性，利用不同水平的28个深度图像排列值，在编码过程中有效地降低表示索引残差的比特数。

SDC像素值的索引残差计算过程如公式所示：其中， i_{resi} 是索引残差， d_{orig} 是深度图原始像素值， d_{pred} 是深度图预测像素值， d_{rec} 是重建深度像素值。 $I(\cdot)$ 是指DLT的映射方法。编码索引残差时，只需查找在DLT上查找像素值的索引，再求索引之差；在解码时，索引残差值通过DLT映射到未压缩深度图对应的深度值。

$$i_{\text{resi}} = I(d_{\text{orig}}) - I(d_{\text{pred}}) \quad (12.1)$$

$$I(d_{\text{rec}}) = i_{\text{resi}} + I(d_{\text{pred}}) \quad (12.2)$$

2.运动参数继承

同一视点的视频帧和深度图是同一时刻目标场景的投影，具有相同的运动特性，因此，编码深度图时可以直接继承已编码对应视频帧的运动参数和分割信息。在进行运动矢量继承时，深度图的编码单元、预测单元的划分和运动矢量均可从对应视频帧的对应纹理块继承。当然，传输的运动

参数除继承得来的外，还可以是预测得出的新运动参数。

在3D-HEVC中，为达到深度数据的纹理运动参数继承的目的，除HEVC的Merge模式的空间和时间候选外，可以通过增加一个额外的Merge候选项到当前深度块的Merge列表的第一项来取得。这个额外的候选项来自同位的纹理块的运动信息。因为视频的运动矢量是1/4精度，而深度图的运动信息是整像素精度，需要对视频的运动矢量进行量化。

3.视点合成预测

视点合成预测（View Synthesis Prediction,VSP）是一种减少视点间冗余的有效方法，它利用深度信息将参考视点的纹理数据转换到当前视点，形成当前视点的预测器。

在基于深度图的显示中，视点合成操作是一种典型的前向转换，将给定视点的深度图像转换到一个合成视点。但在VSP编码的上下文中，这是不切实际的，因为它将需要首先产生一个完整的合成图像，并在编码或解码当前图像之前将它存储到参考图像缓存中，这将导致在解码端的复杂性大大增加。于是，3D-HEVC采用一种变通的方法，即基于块的后向VSP（Backward VSP,BVSP）技术，它导出当前块的深度信息，用以决定在视点间参考图像中的相应像素。

由于一般情况下纹理编码先于深度编码，当前块的深度可以用相同的先前描述的NBDV过程来估计。这种方式中，通过假设当前块和邻近块有相同的深度和视点间位移矢量，就能够推断出深度块。在参考视点中位移矢量指向的深度块能够用于当前视点中的后向转换。作为这种方法的扩展，来自这个深度块的最大深度可转变为视差矢量，然后这个提炼的视差矢量将用于运动继承并完成BVSP操作。上述BVSP过程可以用作HEVC第1版的运动补偿引擎，但适用较小的块，如每个PU为 4×4 块，每个块都带有一个不同的视差（运动）矢量。

当前的3D-HEVC设计中，在Merge列表中的一个VSP merge候选就标记了VSP模式的使用。因为有VSP merge候选选项是采用BVSP的标记（tag），因此其他正常的候选选项在标注为Merge候选选项的过程中并没有使

用BVSP。这样一个VSP的Merge 候选项包含运动矢量，它是一个视差（运动）矢量和标明视点间参考图像的参考索引，由它来预测当前块。注意，这个从NBDV导出的候选视差矢量将用于为上述PU中每个比较小的块（如 4×4 或 8×8 ）确定精准的视差矢量。

本章参考文献

[1]H.Schwarz,T.Wiegand.Inter-view prediction of motion data in multiview video coding[C],IEEE Picture Coding Symposium (PCS 2012),Kraków,Poland,7-9 May 2012: 101-104.

[2]G.Tech,H.Schwarz,K.Müller,et al.3D Video coding using the synthesized view distortion change[C],IEEE Picture Coding Symposium (PCS 2012),Kraków,Poland,7-9 May 2012:25-28

[3]M.Winken,H.Schwarz,T.Wiegand.Motion vector inheritance for high efficiency 3D video plus depth coding[C].IEEE Picture Coding Symposium (PCS 2012),Kraków,Poland,7-9 May 2012,53-56.

[4]G.Tech,K.Wegner,Y.Chen,et al.: 3D-HEVC Draft Text 7[R].Doc.JCT3V-K1001-v3.11th meeting.JCT-3V of ITU-T SG 16 WP 3 and ISO/IEC JTC 1/SC 29/WG 11,Geneva,2015.

[5]Y.Chen,G.Tech,K.Wegner,et al.Test Model 11 of 3D-HEVC and MV-HEVC[R].Doc.JCT3V-K1003.11th meeting: Joint Collaborative Team on 3D Video Coding Extension Development (JCT-3V) of ITU-T SG 16 WP 3 and ISO/IEC JTC 1/SC 29/WG 11,Geneva,Feb.2015.

[6]A.Vetro,T.Wiegand,J.Sullivan.Overview of the stereo and multiview video coding extensions of the H.264/MPEG-4 AVC standard[J].Proceedings of the IEEE,2011,99 (4) : 626-642.

[7]K.Müller,H.Schwarz,D.Marpe,et al.3D High Efficiency Video Coding for multi-view video and depth data[J].IEEE Trans.Image Processing,2013,22 (9) : 3366-3378.

[8]G.Sullivan,J.Boyce,Y.Chen,et al.Standardized extensions of High

Efficiency Video Coding (HEVC) [J].IEEE Journal of Selected Topics in Signal Processing,2013,7 (6) :1001-1016.

[9]High Efficiency Video Coding[S],ITU-T Rec.H.265 and ISO/IEC 23008-2,ITU-T and ISO/IEC JCT-VC,Mach 2013 (v.1) ,Oct.2014 (v.2) ,Apr.2015 (v.3) .

[10]G.Sullivan,J.Ohm,W.Han,et al.Overview of the High Efficiency Video Coding (HEVC) Standard[J].IEEE Trans.on Circuits and Systems for Video Technology,2012,22 (12) :1649-1668.

[11]Mathias Wien.High Efficiency Video Coding: Coding Tools and Specification[M].Springer-Verlag Berlin Heidelberg,2015.

[12]Yi-Wen Chen,Jian-Liang Lin,Yu-Wen Huang,et al.Single depth intra coding mode in 3D-HEVC[C].IEEE International Symposium on Circuits and Systems (ISCAS 2015) ,1130-1133.

[13]A.Vetro,Y.Chen,K.Mueller.HEVC-compatible extensions for advanced coding of 3D and multiview video[C].2015 Data Compression Conference,13-22.

[14]Y.Chen,Li Zhang,V.Seregin,et al.Motion hooks for the multiview extension of HEVC[J].IEEE Trans.on Circuits and Systems for Video Technology,2014,24 (12) : 2090-2098.

[15]D.Kwon and M.Budagavi.Combined scalable and multiview extension of High Efficiency Video Coding (HEVC) [C].IEEE Picture Coding Symposium (PCS 2013) ,2013,414-417.

[16]N.Ling,High efficiency video coding and its 3D extension: a research perspective[C].7th IEEE Conference on Industrial Electronics and Applications (ICIEA 2012) ,Singapore,July 2012,2150-2155J.

[17]J.Sullivan,M.Boyce,C.Yin,et al.Standardized Extensions of High Efficient Video Coding (HEVC) [J].IEEE Journal of Selected Topics in Signal Processing,2013,7 (6) : 1001-1016.

第13章 HEVC的实现

HEVC视频编码标准相对于H.264/AVC编码效率提高了一倍左右，仅用一半的比特率就能传送相同质量的视频。同时，HEVC与H.264/AVC相比，其编解码复杂度也显著增加。尽管半导体技术和计算机技术近年来在不断升级，但相对于HEVC的高复杂度，再加上分辨率和视点的增加，HEVC视频编解码器的实现，尤其是实时实现和便携式实现，无论是软件方式还是硬件方式，都存在着巨大的挑战。

本章首先简单介绍HEVC的参考软件HM，在此基础上分析HEVC的主要编码模块的复杂度，继而分别介绍HEVC的软件实现方案和硬件实现方案的设计考虑。2013年HEVC（第1版）标准公布至今已近2年，还在不断地扩展，参考软件HM也在不断地版本升级，HEVC的实现方案和产品开发尚处于初始阶段，实用、高效的HEVC视频编解码在未来数年内都会是一个热门的研究和开发领域。

13.1 HEVC的参考软件HM

视频编码国际标准为全世界从事视频压缩处理的企业和用户提供一个共同的标准。但是，制定标准不是最终的目的，其目的是应用。应用的第一步就是要能够实现符合标准的视频编码软件、硬件或设备。为缩短从标准制定到产品开发成功之间的时间延迟，ITU和ISO的视频标准的制定者JCT-VC/3V，在制定标准的同时就给出了相应的参考软件，作为该标准实现的一种典型的参考方法，又可算作对标准的补充。近20多年来，这一做法在视频标准制定过程中已经成为一个惯例。从H.261的参考模式RM8（Reference Model 8.0）、H.263的TMN2（Test Model Near-term 2.0）、MPEG-1的SM3（Simulation Model 3.0）、MPEG-2的TM5（Test Model 5.0）、MPEG-4的VM8（Verification Model 8.0）、H.264/AVC的JM等，直到现在HEVC的HM（HEVC test Model）都是如此。这些参考软件在推出以后一直都在不断更新，表现为软件的版本不断升级。目前HEVC最新的参考软件为2014年10月推出的HM16.0。

按照参考软件可以在计算机上编程实现视频序列的HEVC编码压缩，形成相应的压缩码流。但编码的速度极慢，因为参考软件的主要目的和作用是用具体的程序来“解释”标准的含义，可以解决不少技术环节难以或不能用普通语言表述清楚的问题。因此参考软件几乎并不顾及程序执行的速度，只关注清楚地表达原理和结构，经常用于学习和研究。开发人员可以借助参考软件比较快速、深入地理解标准的具体内容，验证压缩码流对标准的兼容性，并不指望将参考软件直接用于实现视频压缩的产品。但这也不是说参考软件没有一点实用价值，开发人员（尤其是初学者）可以在参考软件的基础上采取结构调整、删繁就简、模块优化、并行处理等一系列措施来提高参考软件的运行速度，达到实用的目的。

以下简要介绍HM参考软件的主要内容，如果需要具体编程应用，还需仔细研究HEVC规范和HM程序，这里的内容只是一个简单引导。

1.HM软件

H.264/AVC的参考软件JM (Joint Model) , 即JVT的 Model , 现在已更新到JM18版本。HEVC的参考软件HM的使用方法和JM类似，只是具体的程序有所不同。HM测试模型可以从多个网上下载，其中德国 HHI 研究所的链接为：[https://hevc.hhi.fraunhofer.de/svn/svn_HEVCSoftware /tags/](https://hevc.hhi.fraunhofer.de/svn/svn_HEVCSoftware/tags/)。这个网址下有多个版本的HM，到目前为止，最新版本是HM16.0，读者可以根据自己的实际需求进行选择。

下载了HM的程序后，就可以对HM工程进行编译、调试和运行了。至于运行环境，HM既可以在linux系统下使用makefile对HM进行编译，也可以在windows系统下使用集成开发环境（ Integrated Development Environment,IDE ）如VS（ Visual C++ ）对HM进行编译。

HM包括如下7个主要工程，基本对应HEVC的主要编码功能：

- (1) TAppCommon表示编码器和解码器公共的应用函数；
- (2) TAppDecoder表示解码器应用函数；
- (3) TAppEncoder表示编码器应用函数；
- (4) TLibCommon表示编码器和解码器公共的库函数；
- (5) TLibDecoder表示解码器库函数；
- (6) TLibEncoder表示编码器库函数；
- (7) TLibVideoIO表示视频输入/输出库函数。

2.HM15简介

2014年4月，在西班牙巴伦西亚（ Valencia ）的JCT-VC第17次会议上通过了HEVC测试模型（ HM15 ）的第15版。HM15的有关文档可用作有关HEVC的一般指导信息的来源，提供了HM15的编码侧描述。HM软件中定义了编码工具的两组指标：Main和Main10。这些工具的指标分别对应于HEVC规范中的Main档次和Main10档次，在Main10中亮度和色度像素的比特深度都是10bit，其主要编码工具已罗列在表13.1中。

表13.1 HM编码工具的结构

| | Main | Main10 |
|--|--|--------------------------------------|
| High-level Structure | High-level support for frame rate (temporal nesting and random access) Clean random access (CRA) support Rectangular tile-structured scanning Wavefront-structured processing dependencies for parallelism Slices with spatial granularity equal to coding tree unit Slices with independent and dependent slice segments | |
| Coding units, Prediction units, and Transform units | Coding unit quadtree structure square coding unit block sizes $2^N \times 2^N$, for $N=4, 8, 16, 32$ (i.e. up to 64×64 luma samples in size) Prediction units (for coding unit size $2^N \times 2^N$): for Inter, $2^N \times 2^N, 2^N \times N, N \times 2^N$, and, for $N > 4$, also $2^N \times (N/2+3N/2) \times 2^N$; for Intra, only $2^N \times 2^N$ and, for $N=4$, also $N \times N$) Transform unit tree structure within coding unit (maximum of 3 levels) Transform block size of 4x4 to 32x32 samples (always square) | |
| Spatial Signal Transformation and PCM Representation | DCT-like integer block transform (for Intra) also DST-based integer block transform (only for Luma 4x4) Transforms can cross prediction unit boundaries for Inter, not for Intra Skipping transform is allowed for 4x4 transform unit PCM coding with worst-case bit usage limit | |
| Intra-picture Prediction | Angular intra prediction (35 modes including DC and Planar) Planar intra prediction | |
| Inter-picture Prediction | Luma motion compensation interpolation: 1/4 sample precision, 8x8 separable with 6 bit tap values for 1/2 precision, 7x7 separable with 6 bit tap values for 1/4 precision Chroma motion compensation interpolation: 1/8 sample precision, 4x4 separable with 6 bit tap values Advanced motion vector prediction with motion vector “competition” and “merging” | |
| Entropy Coding | Context adaptive binary arithmetic entropy coding (CABAC) Rate-distortion optimized quantization (RDOQ) | |
| Picture Storage and Output Precision | 8 bit-per-sample storage and output | 10 bit-per-sample storage and output |
| In-Loop Filtering | Deblocking filter Sample-adaptive offset filter (SAO) | |

3.HM的主要功能

HM的文档提供了一个编码端HEVC测试模型（HM）的介绍，其目的是共享对HM所包含的参考编码方法的理解，提供以HM软件方式实现编码模型的指导，促进HEVC标准新技术的应用。文档还定义了实验工作所使用的公共测试条件和软件参考指标。尽管HEVC设计的简要描述有助于对HM的理解，但对任何有关标准化细节的确定，还需要同时参考HEVC规范。

的相应部分。

众所周知，HEVC标准是基于块的混合编码架构，联合运用了运动补偿的预测、变换编码和高效熵编码技术。然而，对比以往的视频编码标准，HEVC 采用了灵活的四叉树编码块划分结构，来确保多尺寸的预测编码、变换块编码的使用；采用了改进的帧内预测、运动参数预测和编码、新的环路滤波和基于上下文的自适应二进制算术编码（CABAC），采用了有利于实现高速并行处理的结构。

按照HEVC的要求，HM编码器的功能框图如图13.1所示，可归纳为以下6个部分。

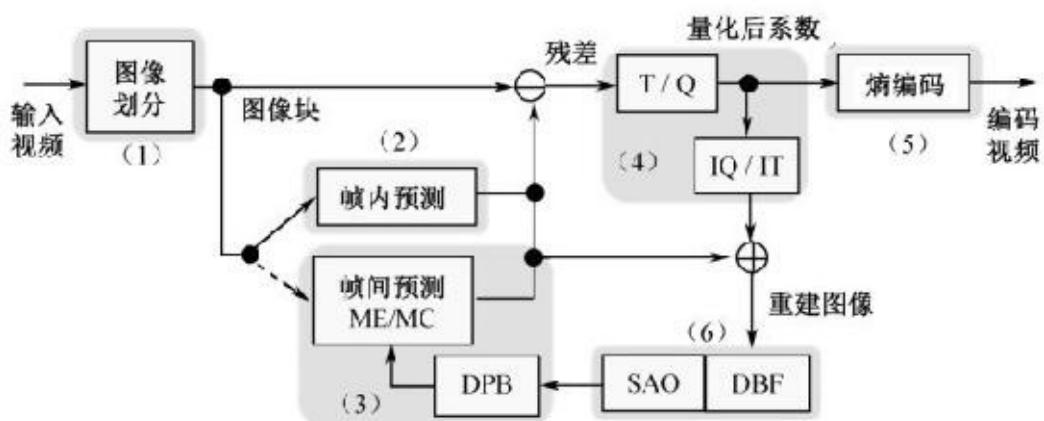


图13.1 HM编码器的功能框图

(1) 图像划分

输入视频首先被划分为编码树单元 (CTU)，是四叉树的“树根”，很大程度上类似于以往标准中“宏块”的角色。位于四叉树的“树叶”节点的是编码单元 (CU)，它是基本的编码区域。CU所包含的预测单元 (PU) 为采用相同预测模式 (帧内、帧间或跳过) 的区域。CU所包含的变换单元 (TU) 为采用相同变换和量化方式的区域，可由另外的“变换四叉树”来表示。除了保留图像的条 (slice) 划分外，还增加了有利于并行计算的图像的片 (tile) 划分，它们都包含整数个CTU。

(2) 帧内预测

在总数35种模式 (平面、直流和33个角度) 中选择最好的帧内预测模式来编码。采用了依赖于上下文参考样值的平滑滤波来增加预测效率，采用了3种最可能模式 (MPM) 来增加对模式信息编码的效率。

(3) 帧间预测

采用合并模式和自适应运动矢量预测 (AMVP) 模式来选择最好的运动参数并对其编码，即在若干候选者中选择运动预测器并对其简单地索引编码。为增加运动补偿预测的精度，采用了一维FIR滤波器的非串行的像素内插。一个8抽头或7抽头滤波器分别用于产生半像素和四分之一像素位置的亮度内插，一个4抽头滤波器用于色度内插。

(4) 变换和量化

将输入信号减预测值就得到残差，然后对残差进行空间变换和量化。在变换处理中，使用的是近似DCT矩阵。为减少计算负担，采用部分蝶形算法结构来实现变换。在 4×4 帧内预测残差的情况下，对亮度信号可采用近似离散正弦变换 (DST)。采用52级量化步长的非线性量化处理，可选用率失真优化的量化 (RDOQ) 技术。

(5) 熵编码

在编码过程中，使用基于上下文的自适应二进制算术编码 (CABAC) 来对符号及量化后的变换系数进行熵编码。

(6) 环路滤波

为获得更高的编码效率和图像可视质量，对编码端重建以后的图像采用两种环内滤波处理：去方块滤波（DBF）和紧跟其后的样值自适应补偿（SAO）。滤波后的重建图像存储在图像缓存中，用作帧间预测的参考图像。

13.2 HEVC的复杂度

为实现HEVC视频编解码，必然要考虑HEVC实现的可行性，而决定HEVC可行性的最重要的因素就是HEVC的复杂度（或计算复杂度）。HEVC标准是由若干处理模块或编码工具组成，它们形成了HEVC复杂度的主要组成部分，因此下面主要分析HEVC的主要编码模块，考虑它们在实现中的复杂性问题。这里涉及HEVC的参考软件，主要参考的是8.0版本的HM。

13.2.1 功能单元的复杂度

1. 图像划分单元

编码图像的灵活四叉树划分方式是HEVC的率失真性能显著地优于H.264/AVC的重要原因之一。尽管这种可变尺寸、自适应方法特别适用于较高的分辨率的视频，例如2K、4K视频，但它带来的复杂度增加也是不容忽视的。

解码四叉树结构没有太多的额外负担，因为四叉树结构已由编码器确定，解码器只要使用“Z”字形扫描、深度优先的方式可很容易地遍历。其中，帧间编码CU的划分模式支持非方形PU，可能需要在Z字形扫描和光栅扫描顺序之间进行多次转换，因而在解码器中需要额外的逻辑支持这些非方块形状的扫描。

图像划分的主要负担在编码器方面，如果要确定一个CTU的四叉树划分方式以及一系列编码参数，不但要遍历所有可能CU划分和CU尺寸，而且在给定CU尺寸时，要确定这个CU的PU划分方式和TU划分方式，以及最佳编码参数，如帧内/帧间预测模式、帧内预测方向、帧间预测模式、参考

索引、运动矢量等。对于一个 64×64 的 CTU，模式选择过程需要遍历各个编码深度下的CU分割，当编码深度为4时，即从一级 64×64 、二级 32×32 、三级 16×16 到四级 8×8 四叉树划分，一共需要遍历85个CU（注： $1+4+16+64=85$ ）。对于每个CU，选择预测模式时，需要遍历帧间和帧内所有的预测模式。每种预测模式在计算率失真代价过程中，由于采用RDO模型，需要做变换、量化和熵编码，其中HEVC中变换单元TU大小也是可选的，因此每种在变换、量化、熵编码模块中还需要做 TU 的划分选择，从中选择最佳的编码参数进行编码。虽然这种基于率失真优化的遍历选择方式可以得到最优的编码参数，但它的计算复杂度非常高，在普通的PC上用HM8.0的编码器做测试，发现CTU的模式选择耗费时间是整体编码时间的 $2/3$ 以上。在实际的编码器中难以实现，特别是难以实时性实现。所以，在HEVC的编码实现中，快速而准确地选择 CU 尺寸和编码模式，非常必要，已成为当前热门的研究问题之一。

2. 帧内预测单元

HEVC 的帧内预测与 H.264/AVC 颇为相似，样点由本帧中重建的邻近样点块来预测。与H.264/AVC显著的不同在于HEVC引进较大块尺寸和较多的模式总数。HEVC的帧内预测可使用35模式之一，最大块的尺寸扩大至 32×32 个样点，最小的块的尺寸和H.264/AVC一样为 4×4 。这样，模式的选择、预测值的计算，再加上帧内预测步骤需要串行进行，这就使得HEVC帧内预测的复杂度大为增加，成为编码器的复杂度瓶颈之一。

(1) 非角度模式预测

DC和Planar模式为非角度预测模式，是帧内预测中最为简单的两种模式。DC模式块中的所有的预测样点值相同，由一个求和公式就可算出。对于平面模式，预测样本值的计算比 DC模式有所增加，每个样本需要3个16bit的加法、一个16bit的位移。

(2) 角度模式预测

由于需要乘法，HEVC的角度（方向）模式的比H.264/AVC的方向模

式更复杂。每个预测样点的计算方法为 $((32-w) \cdot x_i + w \cdot x_{i+1} + 16) \gg 5$ ，其中， x_i 是参考样点， w 是其加权系数。加权系数在预测行或列时保持恒定，这有利于单指令多数据流（SIMD）的实现。一个预测公式可以用于全部33个预测的角度，从而降低了实现此功能所需的计算量。

（3）参考样点的平滑处理

HEVC在预测前还要有选择地对某些参考像素作平滑滤波处理。

3. 帧间预测单元

（1）子像素插值

HEVC中的帧间预测相比H.264/AVC增加了一些开销。亮度的1/4精度像素插值的可分离8抽头滤波器的使用，导致了内存带宽的增加和运动补偿所需乘加操作次数的增加。为了尽量减少硬件成本，滤波器系数被限制在有符号7bit的范围内。在软件实现时，一个 $N \times N$ 块的运动补偿中，每样点需 $(8+56/N)$ 次8bit乘加运算，以及每样点8个16比特乘加运算。对子像素位置的色度信号，使用一个和亮度滤波器系数限制相同的可分离的4抽头滤波器。对比H.264/AVC用于色度子像素位置的双线性内插，这种4抽头滤波器也增加了存储器带宽和运算次数。

（2）预测块尺寸限制

实现成本增加的另一个地方是中间存储器缓冲器，特别是在双向预测的情况下。事实上，HEVC需要两个16比特的缓冲器来保存数据，而在H.264/AVC中，一个8比特和一个16比特缓冲器就足够了。在HEVC实现中，并不一定需要增大这些缓冲器以适应最大 64×64 的PU尺寸，因为较大的块的运动补偿可分解成更小的块来处理，以获得内存需求和操作次数之间的理想折中。

为减少内存带宽，HEVC规定编码器必须服从一个简单的限制：最小的运动补偿亮度块的尺寸为 4×8 和 8×4 之一，因而禁止 4×4 块的帧间预测，同时限制最小运动补偿块只使用第一个参考图像列表，即没有 4×8 和 8×4 亮度块的双向预测。

HEVC引入了一种所谓的合并模式，它设置所有帧间预测块的运动参数等于合并候选者的参数。对运动矢量编码，合并模式和运动矢量预测处理允许一幅图像再使用先前图像的运动矢量，这基本类似于H.264/AVC的时域直接模式。H.264/AVC下采样运动矢量到 8×8 的水平，HEVC却是通过对每个 16×16 块只保持单个运动矢量来进一步降低存储需求。

(3) CU到PU的分割

HEVC提供了更多的方式来分割一幅图像为运动补偿的分割模式。虽然这并不显著影响解码器，但它为编码器留下了更多的选择。这种额外的自由度在充分发挥HEVC优势的同时，也会增加编码器的复杂性。

在帧间预测编码中，为了实现CTU深度选择的快速算法，可采基于skip模式提前终止CU分割的算法，也可采用自适应设置率失真门限算法。对于当前 CU，基于 skip 模式的提前终止算法是在选完PU的模式后，如果 PU 为 skip 模式，则提前终止 CU 分割，减少 CU 遍历的个数。自适应设置率失真门限的方法是对每个深度的不同预测模式的 CU 设置不同的率失真门限，如果当前深度下，最佳PU模式对应的率失真代价小于此门限，则提前终止CU的分割。

4. 变换和量化单元

H.264/AVC采用 4×4 和 8×8 的整数DCT变换，其实施开销非常低。但这种低开销的实现依赖于简单的一系列的移位和加法运算，不容易扩展到更大的变换尺寸，如 16×16 和 32×32 。从而 HEVC采取不同的方法，定义 4×4 ~ 32×32 变换为两轮一维变换，直接的定点矩阵乘法表示。如反变换的垂直和水平分量的矩阵乘法分别如式(13.1)和式(13.2)所示：

$$Y = s(C^T \cdot T) \quad (13.1)$$

$$R = Y^T \cdot T \quad (13.2)$$

其中， $s(\cdot)$ 是一个缩放和饱和函数，保证中间值Y可以使用16bit来表示。变换矩阵T中的每个元素使用有符号的8比特数表示。操作过程中，16bit有符号系数C和矩阵元素相乘，因此需要大于16比特的累加器。由于变换是离散余弦变换的整数近似，因此它们保留了对称性，从而保证了部分蝶型运算的实施。对于 4×4 点的变换，HEVC还定义了另一种近似离散正弦变换（DST）。

以Intel处理器上的变换为例来比较一下不同尺寸变换的复杂度。有实验表明，完成一个 8×8 反变换需要158个周期， 16×16 反变换需要861个周期， 32×32 反变换需要4696个周期。如果用相关块的尺寸来归一化这些周期值， 8×8 块的每个样点为 $158/64=2.47$ 周期，其他 16×16 块、 32×32 块，以此类推，每个样点分别需要3.36、4.59个周期。因此 32×32 反变换的每个样点的时间成本小于 8×8 反变换的两倍。更进一步，通过利用大多数高频系数通常为零的特点，较大变换的周期数量还可能进一步减少。

5. 熵编码单元

(1) 上下文建模和更新

HEVC规定CABAC为唯一的熵编码方法，它包含了三个阶段：语法元素的二值化、上下文建模和二进制算术编码。其中上下文建模和自适应更新是计算量最大的一部分，HEVC经过努力，大大减少了上下文模式的数量。从开始的HM1.0版超过700种的上下文，精简到到8.0版只有154种。在

4:2:0色彩格式（逐行、高档次）编码情况下，这个数目和H.264/AVC的299种相比毫不逊色。H.264/AVC使用的299种中的237种上下文用于支持残差信息编码，而HEVC使用的154种中的112种用于此目的。对比残差编码中的53%减少和其他语法元素32%的减少，很明显，与残差语法关联的上下文数量得到了更大幅度的降低。上下文模式数量的减少，尤其是和残差信息有关的上下文的减少，对熵解码器和初始化存储量的减少作出了贡献。HEVC 算术编码的状态初始值定义从H.264/AVC中每个上下文16比特减少为8比特，从而进一步降低存储需求。

H.264/AVC中确定上下文的一种广泛使用的方法是利用空间的相邻关系。例如，使用上面块和左边块的值导出当前块的上下文。在HEVC中，大部分这样的空间依赖关系已避免，目的是减少行缓存器的数量。

（2）旁路编码模式

上下文建模的熵解码随着比特率的增长，有更多的二进制符号串（bin）需要进行处理，其负担也随之增长。因此，HEVC 将大的语法元素的 bin 字符串分为前缀和后缀两部分。所有前缀bin以常规模式（使用上下文建模）编码，而所有后缀bin以旁路模式编码。由于旁路模式解码一个 bin 的耗费低于常规模式，因此应最大限度地采用旁路模式解码，尤其在高比特率情况下更是如此。

6.环路滤波单元

（1）去方块滤波

HEVC的去块滤波和H.264/AVC的去块滤波机理相同，但滤波的方式不尽相同，这对复杂性有显著的影响。在 H.264/AVC 中， 4×4 网格的每个边缘都需滤波，而 HEVC 则限制滤波 8×8 网格的边缘。这立即减少了一半需计算的滤波器模式的数目和需滤波样点的数目。为保证使并行处理，HEVC 的去块滤波修改了边缘处理的顺序。一幅图像可以被划分成可并行处理的 8×8 的块，如图13.2所示，需要去方块滤波的边缘处于这些块的内部，也就是说只有这些块的内部包含的边缘需要滤波。某个方块的滤波只和它本身有关，和其他的方块无关，这些方块自然就可以并行滤波处理

了。这些块中，有的覆盖在CTB边界上，有的覆盖在slices边界上。这一特点能使它可以以任何顺序滤波slice边界而不影响重建图像。

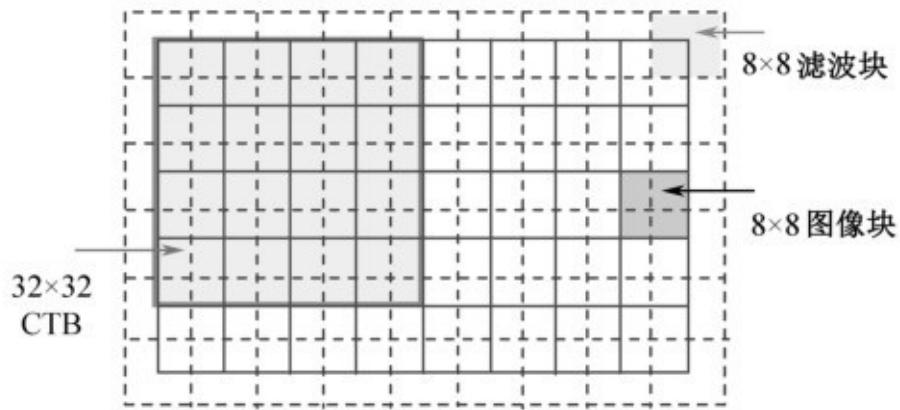


图13.2 去方块滤波的8×8块示意图

需要注意的是垂直边缘在水平边缘前滤波。因此，滤波垂直边缘形成的修改后的样点可用于水平边缘滤波。这就允许有不同的并行实现方法。一种方法是先并行滤波所有垂直边缘，然后并行滤波所有水平边缘。另一种方法则是同时对水平和垂直边缘进行并行处理，处理时设法适当延时水平边缘滤波处理，以使得被滤波的样点先被垂直边缘滤波器滤波过。

然而，HEVC也有一些措施增加了滤波器的复杂性，如除在强滤波器模式下的限幅。

（2）样点自适应补偿

HEVC标准在去块滤波器后增设了一个样点自适应补偿（SAO）滤波器，增加了环路滤波环节的复杂性。

SAO滤波器简单地对某些样值施加一个补偿值，它能够以相当直接的方式实现，通过索引一个小的查找表将补偿量添加到每个样值。查找表的索引可根据所使用的两种模式之一计算得到。其中的一种是带补偿模式，样值被量化后来索引表格。因此，所有的位于某一样值带范围的样值都使用相同的补偿。另一种为边缘补偿模式，需要更多的操作，因为它需基于当前和两个相邻的样本之间的差异计算索引。虽然SAO操作简单，但它代表一种额外的计算负担，因为它可能需要一个额外的解码步骤，还有可能要增加行缓存区。由编码器产生的SAO补偿量需在特流中传输。如果考虑所有SAO模式，编码器的模式搜索过程预计需要比SAO解码过程高约一个数量级的计算量。

7.存储容量

存储部分虽不是编解码器的功能部件，但在每一个编解码功能部分都需要一定容量的存储器。HEVC解码所需的存储总量估计和H.264/AVC的解码所需差不多。大部分的存储体用作保存多幅解码图像的缓存器。缓存器的大小由不同的水平所规定，有可能大于HEVC中给定的最大图像尺寸。这种存储需求的增加不是HEVC设计的一个基本要求，只是用来满足不同水平图像单元的缓存器尺寸的要求。

由于需支持较大尺寸的图像块，HEVC也可能需要更多的超高速缓存。在H.264/AVC中，尺寸为 16×16 的宏块的确定了存储预测和残差所需缓存区的大小。在HEVC中，帧内预测和变换的尺寸可能是 32×32 ，因此，相关缓冲区的大小往往是H.264/AVC的4倍。

13.2.2 HM的编码复杂度

用C++写成的HM编码器的目标主要是提供一个共同的HEVC编码器的参考实现，用作技术评估和独立编解码器开发的测试平台，而不在于提供一个真实的HEVC编码器的样例。HM编码器是相当慢的，使用HM一般需要一组计算机才有可能实现实时编码的目的。尽管HM的编码速度在大多数情况下并不理想，但在HM的开发过程中对某些编码时间已经予以考虑。当用HM 7.0版参考软件编码仅19秒的测试视频时，需耗费100小时以上，但到后面几个版本时，其运行时间大幅减少，降低到20小时。在引起HM编码器耗时的众多因素中，主要和频繁使用率失真优化（RDO）方法密切相关。

顺便提一下，除编解码速度以外，存储带宽也是需要考虑的一个问题。HM使用16比特数据格式存储图像样值，即使在8bit编码模式操作时也是如此，这就需要比H.264/AVC的JM（仅用8bit）更多的存储器和更宽的存储带宽。

在JCT-VC的“公共测试条件”中定义了一套用于实验的配置，包含如下几种情况：

- (1) 全帧内 (AI, All Intra)，这种情况对所有图像采用I slices编码。
- (2) 随机接入 (RA)，这种情况对随机接入的图像每一秒钟进行一次金字塔结构的重排序，比较适用于广播环境。
- (3) 低时延B slices (LB)，这种情况下不需要图像重排，只是第一帧使用I slices编码，比较适用于视频会议环境。

1. 整体编码耗时

表13.2显示了JCT-VC的B类和C类测试序列的HM 8.0的编码时间，每个序列的时长为10秒。编码结果限定使用定义在“公共测试条件”中的两个中间量化参数 (QP) 值。为标明编码耗时对实时操作比率，时间是以10秒为单位记录。从表13.2中可以看出，即使对于仅帧内编码 (AI)，编码时间有的已超过1000倍实时时间。编码时间是在一组包含Xeon服务器 (E5670，时钟频率1.93GHz，使用gcc4.4.5) 上获得的。表格中AI-27表示全帧内方式，量化参数QP=27, RA表示随机接入方式，LB表示低时延B帧方

式，其他以此类推。

表13.2 HM8.0的编码时间

| 测试序列 | 编码时间 (10秒) | | | | | |
|------------------|------------|-------|-------|-------|-------|-------|
| | AI-27 | AI-32 | RA-27 | RA-32 | LB-27 | LB-32 |
| Kimono | 392 | 357 | 1283 | 1123 | 2016 | 1739 |
| ParkScene | 462 | 395 | 1145 | 1000 | 1743 | 1501 |
| Cactus | 955 | 811 | 2590 | 2257 | 3625 | 3133 |
| Basketball Drive | 870 | 759 | 3155 | 2707 | 4417 | 3793 |
| BQTerrace | 1228 | 1043 | 2936 | 2485 | 4039 | 3315 |
| Basketball Drill | 194 | 166 | 606 | 515 | 826 | 706 |
| BQMall | 229 | 202 | 642 | 562 | 900 | 779 |
| PartyScene | 245 | 210 | 614 | 505 | 882 | 724 |
| RaceHorses | 120 | 104 | 481 | 396 | 686 | 570 |

2.单元编码耗时

视频编码器的编解码耗时一般和执行编码任务的器件有关，和完成编码操作的程序有关，因而也难有统一的测试方法和指标。这里仅以计算机上执行HM参考程序的结果来对HEVC各个编码单元的编码耗时为例，来了解各个部分占时的多少，获得一个相对的耗时比例。表13.3列出了在一个编码实例中不同的C++类所耗费的时间，编码视频序列为 BasketballDrive (1080p, QP=27) 。

表13.3 HM8.0的不同类编码时间分布

| 功能 (类) | 编码时间占比 (%) | |
|-------------------------|------------|------|
| | AI | RA |
| TEncSearch | 11.8 | 7.4 |
| TComTrQuant | 24.4 | 10.7 |
| TComRdCost | 9.8 | 38.8 |
| TComInterpolationFilter | 0.0 | 19.8 |
| TComYUV | 0.1 | 1.7 |
| partialButterfly* | 8.7 | 4.0 |
| TComDuaCU | 5.8 | 2.7 |
| TEncSbac | 8.4 | 3.5 |
| TEncEntropy | 1.2 | 0.6 |
| TEncBinCABAC** | 2.2 | 0.9 |
| TComPrediction | 10.0 | 1.1 |
| TComPattern | 6.6 | 0.4 |
| memcpy/memset | 11.0 | 7.1 |

(1) 全帧内 (AI) 方式

在AI中，大量的时间（约1/4的总时间）耗费在TComTrQuant类上，这里时间主要被率失真优化的量化（RDOQ）占据了。变换（partialButterfly*）在这里占9%。帧内图像预测（TComPrediction和TComPattern）占接近16%。在熵编码前端，CABAC核心操作的耗时量较少：TEncBinCABAC*类（包括TEncBinCABAC和TEncBinCABAC Counter类）约占2%。注意，TEncBinCABACCounter类是用于估计CABAC引擎产生的比特数，没有导出这些比特的值。更多的时间耗费在扫描和上下文引导：TEncSbac类超过8%，TComTrQuant类的getSigCtxInc约1.7%。

（2）随机接入（RA）方式

很明显，在RA中运动估计占据编码时间的主要部分。绝对差之和（SAD）及其他失真矩阵的计算发生在TComRdCost类中，约占编码时间的40%。还有，运动补偿滤波发生在TComInterpolationFilter中，约占20%的编码时间。在HM中分数像素搜索未进行加速优化，大量的时间耗费在运动估计过程中分数样点的精细搜索上。它包括对半像素点网格上的9个位移和另外的1/4像素网格上的9个位移的测试。此外，带有大量候选选项的合并模式的使用也加大了内插滤波的时间耗费，因为2-D内插过程必须为每个合并候选重复进行。在内插滤波上的时间耗费量，可以通过为1/4网格上的16个分数位移的每一个预先计算滤波的参考图像来减少。然而，这将显著增加对存储量的需求，而且也不一定适合所有的结构。如同在AI方式中一样，TComTrQuan中的大部分时间耗费可归结于RDOQ，因为实际的变换（partialButterfly*）是另外计算的，约占4%。和SAO相关的编码函数未列在表13.3中，因为其中的时间耗费量低于总编码时间的1%。

3. 快速算法

HM编码器已经出现了一些加速的技术。例如，为图像自适应划分的快速搜索算法，为帧内或帧间图像预测模式快速搜索算法，为CABAC比特计数而使用的基于表格的估计等。

表13.4出示了采取了几种快速搜索功能的HEVC随机接入（RA）和低时延（LB）方式的编码时间。所采用的这类减少搜索次数的方法都采用或

部分采用了于跳过模式，仅影响帧间图像预测模式的判决，因此表格中只给出了RA和LB的对比数据。在率失真判决中，较高QP情况下更喜欢选择这一模式。这就为什么比较快的加速总是伴随着比较高的 QP。在表13.4所显示的 QP 情况下，编码时间能减少2~3倍，其代价是编码效率的稍微下降，低于3%的 BD (Bjontegaard Delta) 率。对QP等于37，能够获得6倍的编码加速。BD率表示在相同的比特率情况下，不同编码方法重建图像的平均PSNR的dB数之差。

表13.4 有快速选择功能的HM 8.0编码时间和性能 (QP=27/32)

| 测试序列 | 编码时间 (毫秒) | | BD (%) | 编码时间 (10秒) | | BD (%) |
|------------------|-----------|-------|--------|------------|-------|--------|
| | RA-27 | RA-32 | | LB-27 | LB-32 | |
| Kineno | 628 | 416 | 1.6 | 1251 | 845 | 1.2 |
| ParkScene | 473 | 306 | 2.2 | 953 | 605 | 1.6 |
| Cactus | 1227 | 860 | 2.8 | 2115 | 1461 | 2.1 |
| Basketball Drive | 1631 | 1150 | 2.2 | 2749 | 1882 | 1.1 |
| BQTerrace | 1180 | 617 | 2.3 | 1955 | 980 | 1.2 |
| Basketball Drill | 367 | 253 | 2.8 | 581 | 402 | 0.9 |
| BQMall | 334 | 236 | 2.3 | 553 | 389 | 1.5 |
| PartyScene | 371 | 243 | 1.8 | 640 | 431 | 1.2 |
| RaceHorses | 321 | 225 | 2.5 | 535 | 381 | 1.0 |
| 平均加速倍数 | 2.0 倍 | 2.6 倍 | | 1.6 倍 | 2.1 倍 | |

13.2.3 HM的解码复杂度

与HM编码器类似，也以一个HM解码器的实例来分析HEVC解码复杂度，其结果如表13.5所示。解码器所解码的序列和前述的编码器一致，每个序列时长为10秒，只不过提供给解码器的是它们的压缩码流。解码程序以单线程的方式运行，没有使用并行化技术。在测量时，不计算码流文件的读写时间，只测量纯解码时间。例如，从表13.5可以看出，对BQTerrace (AI方式，QP=27) 码流的解码时间长达126.6秒，为了达到实时解码，需要解码加速近13倍。如果QP进一步降低至22，这个倍数将增加至17 (这一项表中没有) 。

表13.5 HM 8.0的解码时间 (QP=27/32)

| 测试序列 | 解码时间 (秒) | | | | | |
|------------------|----------|-------|-------|-------|-------|-------|
| | AI-27 | AI-32 | RA-27 | RA-32 | LB-27 | LB-32 |
| Kimono | 34.8 | 31.2 | 20.3 | 18.5 | 22.5 | 18.4 |
| ParkScene | 49.9 | 41.1 | 20.9 | 18.4 | 22.1 | 18.0 |
| Cactus | 96.8 | 83.4 | 39.8 | 24.8 | 40.4 | 32.7 |
| Basketball Drive | 83.9 | 75.7 | 45.0 | 29.0 | 50.7 | 41.6 |
| BQTerrace | 126.6 | 107.9 | 50.1 | 40.3 | 34.8 | 40.4 |
| Basketball Drill | 22.5 | 18.6 | 9.8 | 8.3 | 10.2 | 8.2 |
| BQMall | 25.5 | 22.5 | 10.7 | 9.2 | 11.2 | 9.5 |
| PartyScene | 28.8 | 24.7 | 10.9 | 8.9 | 12.5 | 9.5 |
| RaceHorses | 12.5 | 11.2 | 7.4 | 6.0 | 8.2 | 6.5 |

解码时间在很大程度上取决于所选的 QP 值。在更高的比特率（即更低的 QP 值）时，就有更多的系数要编码，CABAC 解析过程就需要更多的时间。解码时间还取决于所选的图像块尺寸，如果选择较小的块尺寸，模式判决的数量会激增，也给解码器增加额外的负担。从表13.5中可以看出，在相同QP的情况下，随机接入（RA）和低时延（LB）之间解码时间的差异是相当小的，而AI解码需要高达两倍的RA的时间。为进一步说明解码器的那部分能是较费时的，表13.6大致显示HM解码器的主要C++类的结果，解码的视频序列为BasketballDrive,1080p,QP=27。

表13.6 HM 8.0的不同类解码时间分布

| 功能 (类) | 解码时间占比 (%) | |
|--------------------------|------------|------|
| | AI | RA |
| TComTrQuant | 8.7 | 4.2 |
| TComInterpolationFilter | 0.0 | 24.8 |
| TComYUV | 0.5 | 8.2 |
| partialButterfly* | 15.9 | 7.6 |
| TComDataCU | 7.5 | 7.1 |
| TDecShac | 6.2 | 2.8 |
| TDecEntropy | 1.4 | 1.0 |
| TDecBinCABAC | 5.3 | 2.3 |
| TDecCU | 7.2 | 2.6 |
| TComPrediction | 5.1 | 2.3 |
| TComPattern | 9.4 | 2.6 |
| TComSampleAdaptiveOffset | 3.8 | 2.4 |
| TComLoopFilter | 12.9 | 12.4 |
| memcopy/memset | 6.2 | 10.1 |

在AI方式中，partialBufferfly*、 TComDataCU、 TComPattern和

TComLoopFilter占大部分时间。partialBufferfly*类表示反变换，约占解码时间的15%。虽然这一段时间占比较大，但也有望通过实施局部变换技术来显著减少。这样的局部变换技术对较大块的变换是特别有效的，因为这里的高频系数是最有可能为零。TComPattern类为帧内图像预测处理生成参考样本，这个类的花费时间较多，主要是因为这种依赖多次复制参考样本的实现方法效率不高。TComLoopFilte类是实现去方块滤波器。

TComDataCU类负责对一个CU内的大多数数据元素的管理，导出相邻块地址的功能在 TComDataCU 类中是最耗时的，某种程度上是因为取决于需要检查 slice、CTB、tile边界等状态的数量。

在随机接入方（RA）式中，解码时间以TComInterpolationFilter和TComLoopFilter为主，前者包括运动补偿处理的内插滤波的功能。为达到实时解码的程度，预计所有的HM解码器组件将需要加以改进。

13.2.4 和H.264/AVC比较

尽管在 HEVC 中一些关键模块或功能，如变换、帧内预测和运动补偿的复杂性可能比H.264/AVC 更高，但另外一些模块，如熵编码和去方块滤波的复杂度却有所降低。因此，对于同样分辨率的视频码流，HEVC的解码器的实现成本因而并不需要比H.264/AVC解码器高多少，即使加入如SAO的环路滤波器也会如此。

编码器的实现需要完成比解码器更多、更复杂的处理功能。例如，HEVC的四叉树结构划分、帧内预测、运动估计以及各种模式选择，还常常需要结合率失真性能的优化等。因此，HEVC编码器预计比H.264/AVC编码器复杂若干倍。

13.3 HEVC编码器的实现考虑

视频编码实现的方式可分为软件编码和硬件编码两类，各有优点，也各有不足之处，在实际应用中需根据不同的应用需求决定到底采用哪种方式。

13.3.1 软件实现考虑

软件视频编码一般是指在普通的计算机上，基于通用的操作系统采用软件程序的方式实现对视频序列的压缩编码处理，形成符合HEVC标准的压缩视频码流数据或文件。普通计算机可以是台式机，也可以是笔记本或平板电脑，甚至是智能手机。这些设备并不是专门用于视频处理的，视频编码只是它们的众多功能之一。编码程序赖以运行的处理器（CPU）是通用型的，并非专门用于视频处理芯片，既可以是单核芯片，也可以是目前主流的多核芯片。

除主CPU外，还允许在计算机中加装图形处理单元（Graphics Processing Unit, GPU）之类的协处理器芯片，协助主CPU承担部分工作量特别大的图像处理任务，加快编码处理速度。

视频编码软件是在计算机操作系统的控制下工作的，不同的操作系统都有其特殊的要求，因此，一般的视频编码程序并不能工作在不同的操作系统平台。由此看来，软件视频编码实际上是工作在公共硬件平台上的应用程序。有时也称这类编码程序为“纯”软件编码方式，因为它不需要考虑平台的硬件结构，但并不表示它不需要硬件支持。

软件视频编码方式的实现有两个明显的优势。

（1）软件编码借助于普通计算机成熟的处理方式，甚至于开源软件，

可以充分利用计算机的多核、多线程、协处理器等先进的处理工具，来开发视频编码程序，具有开发方便，成本较低，周期较短。

(2) 由于是软件方式，后续的改进、升级都比较方便，同时借助与通用计算机平台，视频编码程序的推广应用的阻力相对较小，便于推广，便于和其他应用相互融合。

当然，软件视频编码方式的实现也有它明显的不足之处：由于通用计算机不是专门处理视频数据的，不太适合进行数据密集型的高强度计算，因此它的处理效率并不高。即是在当今高速、多核计算机平台上，高清、超高清的视频编码也难做到实时处理（或即使做到），也几乎要竭尽计算机的全部计算能力，使得计算机很难同时完成其他任务，失去通用计算机的意义。如果让一台计算机仅用作一个视频编码器，那么其成本、体积、功耗指标等就失去了商业价值。所以在计算机平台、平板电脑或智能手机平台，常常运行的是实时视频解码，因为视频解码的复杂度要远远低于编码，或者对一些分辨率较低或帧频较低的视频进行实时编码。当然，计算机软件编码方式对高分辨率视频的非实时编码也是可以的，这就是用时间来换取速度。因此，对视频会议、电视直播等实时性要求较高的应用，对能耗限制比较严格的智能手机等移动端应用，HEVC软件编码实现方案目前还很难做到。

13.3.2 硬件实现考虑

硬件视频编码，其指导思想是“用专门的工具解决专门的问题”，即用专门针对视频编码处理设计的或特别适合视频数据处理的芯片来完成视频编码的任务。需要说明的是，我们把这种方式称为硬件编码，绝不是说不需要软件支持了，事实恰恰相反，不仅要软件支持，而且软件支持的优劣对视频编码性能的好坏有决定性的影响。

目前的硬件编码器主要有三种形式。

(1) 可编程器件实现

可编程器件中应用最广的是现场可编程门阵列（ Field-Programmable Gate Array,FPGA ） ,FPGA 内部主要由寄存器（ Register ）、查找表（ LUT ）、 Block RAM 、 DSP 以及 I/O 接口等构成。在 FPGA 上可以反复多次编程，便于对设计方案进行改进和升级。也可以在其上对电路功能进行现场验证，发现问题之后，能够很快进行调试和修改，再重新验证。因此，在 FPGA 上进行设计、开发和定型是目前比较流行的一种方法。

（ 2 ）专用集成电路实现

专用集成电路（ Application Specific Integrated Circuit,ASIC ）是为完成某种功能专门设计的一种集成电路，如视频编解码 ASIC 。由于针对性强，因而芯片的部件可以达到最少，性能最优，布线缩短，体积和功耗减小，提高了系统可靠性，可以最小的开销来充分满足用户的要求。但是， ASIC 设计周期长，工艺与测试难度高，开发成本较高，改进和升级困难，适于大批量应用。

（ 3 ）高速信号处理器实现

高速信号处理器或多媒体数字信号处理器（ DSP ），它的功能是通过面向芯片结构的指令软件编程来实现的，而且常常带有面向视频编码的硬件协处理器，使处理速度加快，功能的改变而无须更改硬件平台，可以满足高速视频信息密集、规则、重复、快速的处理的要求。对比 ASIC,DSP 并非是为某种功能所设计的芯片，所以它应用范围相对较广，可以降低芯片成本。还可以将 DSP 将和 ASIC 结合起来，即在设计 ASIC 时，将一种或几种 DSP 核心嵌入其中，形成“ DSP 内核”（ DSP Core ），增加了芯片的灵活性，加快了设计过程、降低了开发费用。

对视频编码这样的高运算量应用，通过硬件电路的优化设计，如高并行度处理结构、流水线指令结构、专用协处理器、 DSP 核 /FPGA/ASIC 的混合等，可以实现高速、大容量的视频处理。因此，硬件实现方案在视频编码领域有着非常广泛的应用，在很多情况下甚至是唯一的解决方案。但是，与软件编码实现相比，硬件实现方法的开发成本较高，开发周期较长，开发完成后一般难于进行修改与升级。

13.4 HEVC的解码实验

为了说明HEVC解码器的性能，下面介绍一组优化实现的HEVC软件解码器的实验结果，数据来自参考文献[1]。该解码器用C语言编写，对两种不同的指令集架构（Instruction Set Architecture,ISA）的x86和ARM进行了代码优化。在这两种情况下都大量使用SIMD指令，获得比HM解码器明显更快的速度。为了利用这些指令在C代码中插入了内部函数，在ARM中还使用了汇编代码、NEON技术（适用于ARM Cortex-A系列的一种128位 SIMD扩展结构）。在x86中使用单指令多数据流扩展（Streaming SIMD Extensions,SSE）4.1。

13.4.1 HEVC的测试序列

JCTVC-1100文档给出了HEVC编码后的视频测试序列，为HEVC解码器性能的测试提供了一套公用的压缩视频码流。这一套视频序列共有24个不同的视频段，它们的时间复杂度和空间复杂度处于中等偏下的水平。从A到F共6组，包含6种分辨率：

$2560 \times 1600, 1920 \times 1080, 832 \times 480, 416 \times 240, 1280 \times 720, 1024 \times 768$ 。每种分辨率包含4种不同的量化参数的码流，即 $QP=22, 27, 32, 37$ 。

按照HEVC通用测试条件，参考视频序列主要包括了三种不同特性的编码方式，第一种是适合娱乐应用的随机接入的方式，第二种是适合交互应用的低时延方式，第三种是没有特殊指向的帧内编码方式。所有6组视频都可用于帧内编码的性能测试。此外，一般A组序列（UHD内容）只用于测试随机接入方式的编码性能，E组序列（典型的视频会议 HD 内容）只用于测试低时延方式的编码性能，B、C、D和F组视频既可用于随机接入

又可用于低时延方式的编码性能测试。

表13.7中“HE10”表示*-High efficiency 的配置文件，如“All Intra-High efficiency (AI-HE10) ”的配置文件为encoder_intra_he10.cfg。“Main”表示*-Main的配置文件，如“All Intra-Main (AI-Main) ”的配置文件为encoder_intra_main.cfg。表13.7中B组的“ParkScene”序列和“BasketballDrive”序列的各一帧样例如图13.3所示。

表13.7 HEVC的测试序列

| 类型 | 序列名称 | 帧数 | 帧率 | 比特数 | 帧内 | 随机接入 | 低时延 |
|----------------|---------------------|--------------|-------|-------|-----------|-----------|-----------|
| A 2560×1600 | Traffic | 150 | 30fps | 8 | Main/HE10 | Main/HE10 | |
| | PeopleOnStreet | 150 | 30fps | 8 | | | |
| | Nobuta | 300 | 60fps | 10 | | | |
| | SteamLocomotive | 300 | 60fps | 10 | | | |
| B 1920×1080 | Kimono | 240 | 24fps | 8 | Main/HE10 | Main/HE10 | Main/HE10 |
| | ParkScene | 240 | 24fps | 8 | | | |
| | Cactus | 500 | 50fps | 8 | | | |
| | BQTerrace | 600 | 60fps | 8 | | | |
| | BasketballDrive | 500 | 50fps | 8 | | | |
| C 832×480 | RaceHorses | 300 | 30fps | 8 | Main/HE10 | Main/HE10 | Main/HE10 |
| | BQMall | 600 | 60fps | 8 | | | |
| | PartyScene | 500 | 50fps | 8 | | | |
| | BasketballDrill | 500 | 50fps | 8 | | | |
| D 416×240 | RaceHorses | 300 | 30fps | 8 | Main/HE10 | Main/HE10 | Main/HE10 |
| | BQSquare | 600 | 60fps | 8 | | | |
| | BlowingBubbles | 500 | 50fps | 8 | | | |
| | BasketballPass | 500 | 50fps | 8 | | | |
| E 1280×720 | FourPeople | 600 | 60fps | 8 | Main/HE10 | | Main/HE10 |
| | Johnny | 600 | 60fps | 8 | | | |
| | KristenAndSara | 600 | 60fps | 8 | | | |
| F 832×480 | BasketballDrillTest | 500 | 30fps | 8 | Main/HE10 | Main/HE10 | Main/HE10 |
| | 1024×768 | ChinaSpeed | 500 | 30fps | | | |
| | | SlideEditing | 300 | 30fps | | | |
| | | SlideShow | 500 | 20fps | | | |



(a) ParkScene

(b) BasketballDrive

图13.3 B组 (1920×1080) 序列的2帧样例

13.4.2 基于ARM的解码

在ARM解码实验中，C组视频序列分辨率为 832×480 ，平板电脑上运行的ARM的性能为1GHz的Cortex-A9双核处理器。使用OpenGL ES 2.0并行解码，实时显示解码视频。一个单独的线程用于解码环路，解码环路没有分成附加的线程。为了说明与每个帧相关的解码时间的变化，用于解码和显示的帧缓存器有16个通道以保证流畅播放。

表13.8显示了随机接入（RA）和帧内（AI）方式的实验解码时间，使用的QP值配置为27。对于RA，所有的测试实例均为实时解码。这些数据表明，2Mbps速率、30帧/秒的480p的视频流在平板电脑是不难达到实时解码的。虽然没有测试，在低功率设备情况下（如智能手机）这也是有可能的。与随机接入情况不同，AI的实时解码在任何测试情况下都没有实现。原因之一是执行熵解码所需的处理量太大，在I slice中，大多数比特用来表示变换系数，因而解析它们成为瓶颈之一。

表13.8 ARM的平均解码时间 (QP=27)

| 测试序列 | 帧率 (Hz) | RA方式 | | AI方式 | |
|------------------|------------|---------------|-----------|---------------|-----------|
| | | 比特率 (Mbps) | 耗时 (s) | 比特率 (Mbps) | 耗时 (s) |
| Basketball Drive | 50 | 1.66 | 7.5 | 11.31 | 19.1 |
| BQMall | 60 | 1.70 | 8.4 | 13.97 | 23.3 |
| PartyScene | 50 | 3.10 | 9.4 | 27.23 | 30.4 |
| RaceHorses | 30 | 2.03 | 6.8 | 8.99 | 12.9 |

表13.9表示解码时间在各个模块上的分布。在RA方式中，运动补偿是最耗时的，占用了接近一半的解码时间。环路滤波器（去块效应和SAO）占用约1/5的解码时间，熵解码大约为1/4的解码时间。反量化和变换仅占4%左右，占比还是比较低的，主要是采取的局部逆变换和主动代码优化的算法。可见引进较大尺寸的变换并不会出现显著增加软件的解码时间。在AI方式中，熵解码占据了一半的解码时间，其次是帧内预测平均占1/5的时间，这两项成为ARM解码耗时的瓶颈。

表13.9 ARM的平均解码时间分布

| 解码模块 | RA方式 (%) | AI方式 (%) |
|---------------------|----------|----------|
| Motion compensation | 43 | |
| Entropy decoding | 24 | 51 |
| Intra prediction | 6 | 20 |
| DBF | 17 | 13 |
| IQ&IT | 4 | 9 |
| Rest | 2 | 1 |
| SAO | 4 | 6 |

13.4.3 基于X86的解码

在X86计算机上的解码分析采用B组序列(1920×1080)。计算机为单核、单线程的英特尔酷睿 i7-3720QM 处理器，主频为2.6GHz(可增频至3.6GHz)的2012型笔记本电脑，使用4.7.1版gcc编译器。

在随机接入(RA)方式中，如表13.10所示，最高的编码视频的比特率高达约7.3Mbps(BQTerrace序列)，解码10秒序列需要的时间小于6秒，这种实时性能在60fps时也取得了较宽的余地。对AI方式解码，如表13.10

所示，并不总是能在单核上实现实时解码。在最坏的情况下（BQTerrace QP=27）需要19.1秒解码10秒的视频。剖析这种情况可见，60%的时间都花在熵解码，13%花在帧内图像预测和残差相加，9%花在去方块滤波，7%花在反量化和变换，9%花在SAO滤波。显然，熵解码成为明显的解码瓶颈，但这并不奇怪，因为此时比特率大约高达80Mbps。

表13.10 X86的平均解码时间 (QP=27)

| 解码模块 | 帧率 (Hz) | RA方式 | | AI方式 | |
|-----------------|---------|------------|--------|------------|--------|
| | | 比特率 (Mbps) | 耗时 (s) | 比特率 (Mbps) | 耗时 (s) |
| BasketballDrive | 50 | 6.01 | 4.9 | 29.1 | 10.2 |
| BQTerrace | 60 | 7.31 | 5.6 | 79.3 | 19.1 |
| Cactus | 50 | 5.72 | 4.0 | 48.7 | 13.6 |
| Kimono | 24 | 2.18 | 2.0 | 12.1 | 3.5 |
| ParkScene | 24 | 3.33 | 2.4 | 28.6 | 7.3 |

表13.11给出在了不同的模块中的解码时间分布。和前面ARM的结果相仿，RA方式中，运动补偿仍然是最耗时的模块，平均占一半的时间。

表13.11 X86平均解码时间分布

| 测试序列 | RA方式 (%) | AI方式 (%) |
|---------------------|----------|----------|
| Motion compensation | 49 | |
| Entropy coding | 21 | 60 |
| Intra prediction | 4 | 13 |
| Loop filter | 18 | 18 |
| IQ&IT | 4 | 7 |
| Rest | 4 | 2 |

13.4.4 解码性能分析

上述实验结果虽然是从不同视频序列获得的，但在ARM和X86上获得的分析结果是非常相似的。在这两种情况下，每个相关模块的解码时间比例是类似的。运动补偿的约占解码时间的50%。环路滤波器（去方块滤波和SAO）约占20%，熵解码占另外的20%。余下的10%为反变换、帧内图像预测等所占用。在其他实验中，运动补偿和熵编码的时间有一些差异，如

运动补偿时间可能占35% ~ 40%，熵解码可能占25% ~ 30%等。

在运动补偿过程中的制约因素之一是内存带宽和高速缓存未命中。在测试的解码器中，参考帧存储器的布局使用隔行色度分量光栅扫描方式，并且增加了简单有限的预读取机制。隔行色度分量方式减少了内存读取操作的工作量，也保证了内插处理的最小带宽。使用不同的内存布局或较好的预读取，可能有助于减少和运动补偿有关的解码时间。在运动补偿处理中的另一个制约因素是乘法累加运算的次数。在测试的解码器中，通用的8抽头内插滤波器用于亮度信号，其滤波器系数存储在一个表中。在实现的调试中，如利用特定滤波器系数等于0或1的知识，也可以减少和运动补偿相关的解码时间。

13.5 HEVC的编解码器简例

下面简要介绍几种不同类型的HEVC编解码器的实例。其中解码器实例有两个，一个是基于DSP的HEVC解码器，另一个是HEVC解码器芯片的设计考虑。编码器实例也是两个，一个用于8K视频的高性能HEVC编码芯片设计考虑，另一个是一台用于SHV视频的HEVC编码设备。由于HEVC的实现是一个难度很大的技术问题，尤其是编码器，各方面技术还很不成熟，尚在不断发展之中。因此这里的介绍得比较简单，仅希望从中可以看出HEVC实现的大体思路和技术瓶颈，可以看出目前的技术所达到的程度以及解决这些问题的方向。

13.5.1 基于DSP的HEVC解码器

这里简单介绍一种在Texas Instruments公司的DSP TMS320DM6437平台上运行的低成本HEVC视频解码器，它的依据是HM9.0参考软件，编程基础为OpenHEVC。

OpenHEVC (<https://github.com/openhevc/openhevc>) 是一个兼容HEVC解码的开源项目，它用C语言写成并进行了优化，比HM码字更加适用于用作起步阶段实现HEVC解码器。

OpenHEVC解码流程如图13.4所示。对图像中光栅扫描顺序排列的每个CTU树，用递归函数hls_coding_tree遍历，直至CU层。对每个CU，用hls_prediction_unit计算帧间或帧内预测。然后用递归函数hls_transform_tree寻找TUs，用hls_transform_unit对TU解码。所有的CTU已经被解码以后，如果需要，用Deblocking Filter (DF) 和 Sample Adaptive Offset (SAO) 进行滤波。

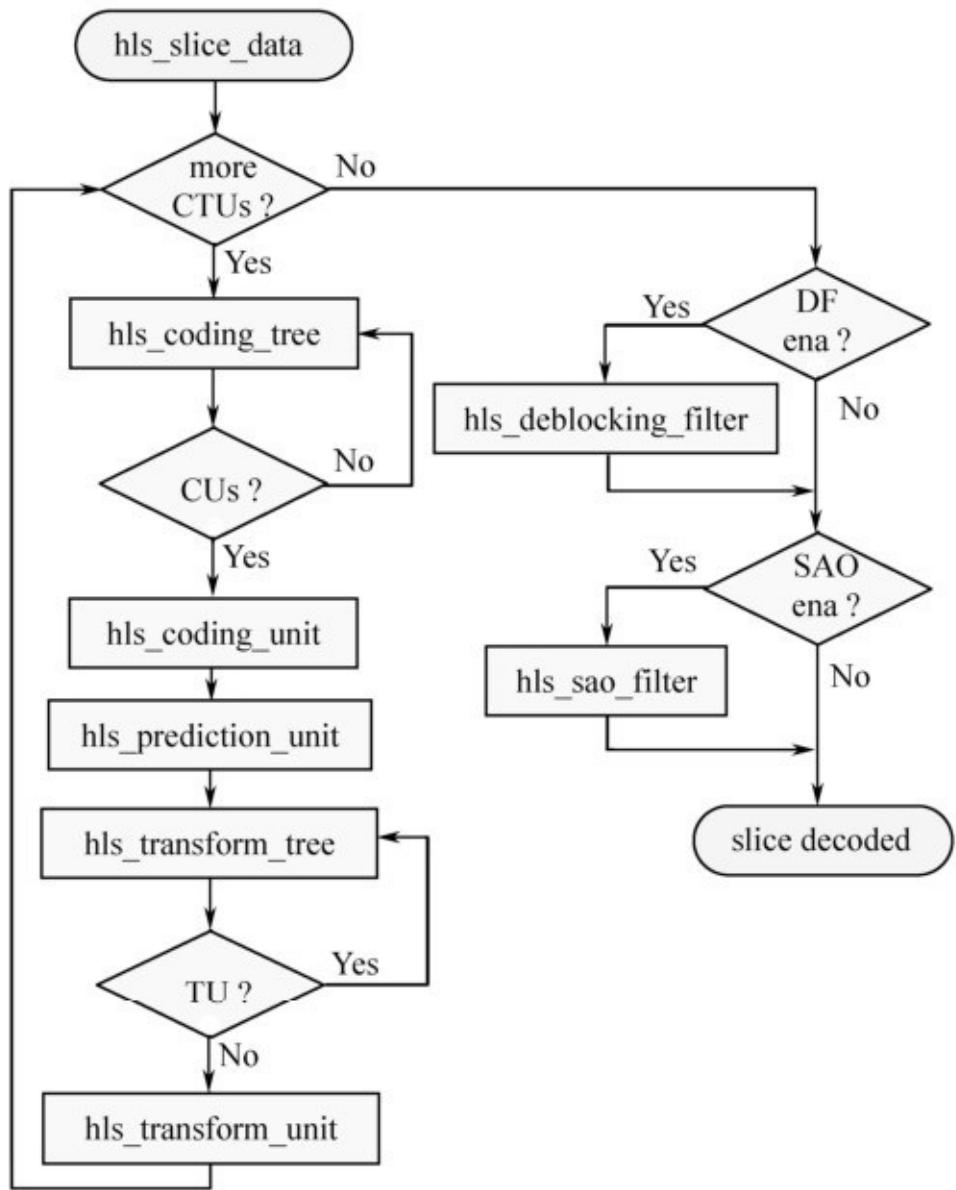


图13.4 OpenHEVC解码环路的简化流程图

在图13.5中，显示了这个DSP解码器的简化的框图，它主要由一个VLIW核、一个视频处理核和一套公用的外围接口组成。VLIW核有2层cache（L1和L2），一个内部DMA（IDMA）。视频处理（Video Processing）部分在此解码器中没有使用。L1 cache的32KB用于程序（L1P），80KB的用于数据（L1D），而128KB的L2 cache既可用于程序也可用于数据。

性能测试结果说明，在DM6437处理器上，基于OpenHEVC的解码器比基于HM9.0的解码器速度要快，反映到帧频数据上约为2.3倍。

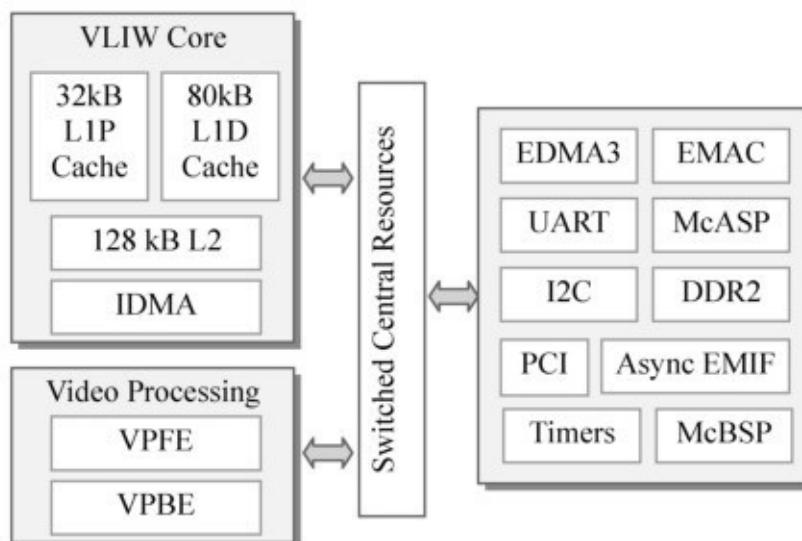


图13.5 HEVC解码器的DSP结构

13.5.2 HEVC解码器芯片

这里介绍一种HEVC视频解码集成电路的设计，它支持4倍全高清（Quad Full HD,QFHD 3840×2160 ）视频的解码，具有以下3方面的特点：

- (1) 采用系统流水线技术，自适应于可变尺寸的CTU，为存储器优化提供2级子流水线；
- (2) 采用统一的处理引擎来处理等级编码结构，以局部有效的方式处理多预测和变换块尺寸；
- (3) 采用减少CTU的DRAM带宽的运动补偿（MC）cache，以满足因长滤波器产生的高吞吐量需求。

图13.6出示了该系统框图和流水线技术，对流水线系统采用可变尺寸流水线块（Variable-size Pipeline Blocks,VPB），它们分别是 64×64 、 32×32 或 16×16 。为统一硬件处理流程以及保证在预测引擎中子流水线存储器有效，VPB尺寸可适当选择。图中粗线方框为处理引擎，细线方框为SRAM缓存器，虚线箭头为DMA数据路径。

VPB流水线缓存尺寸正比于 64×64 （比宏块大16倍），它显著增加了存储需求。为此将流水线分为两组，第一组包括熵解码处理和运动补偿信息处理，第二组包括反变换、预测、去方块和DMA重建处理。同时还作了两点改进：第一点就是减少20KB系数的SDRAM，这是通过用较小的4KB变换单元（TU）的FIFO来替换熵解码和变换引擎之间的VPB流水线来实现的。第二点就是调节运动补偿的高速缓存（MC cache），因为它对VPB流水线具有不确定的DRAM存取时延。MC cache从调度引擎接收存取需求，它的输出存储在参考像素缓存中为预测引擎所用。为了节省DRAM带宽，还使用了2行缓存器来存储顶行的像素和编码信息。

这个解码芯片采用40nm的CMOS工艺，大小为 $2.18\text{mm} \times 2.18\text{mm}$ ，处理核的尺寸为 1.77mm^2 ，包含715K个逻辑门，124KB片上SRAM。解码指标符合HM（HEVC test Model）4.0。该芯片对QFHD视频（200MHz）的

解码吞吐量为249Mpixels/s,DDR3 SDRAM 工作在400MHz。芯片的处理核在不同的工作条件下的功耗是不一样的，在常见使用情况下，如30fps帧频的QFHD视频解码，核电压为0.9V时的平均功耗约为76MW。

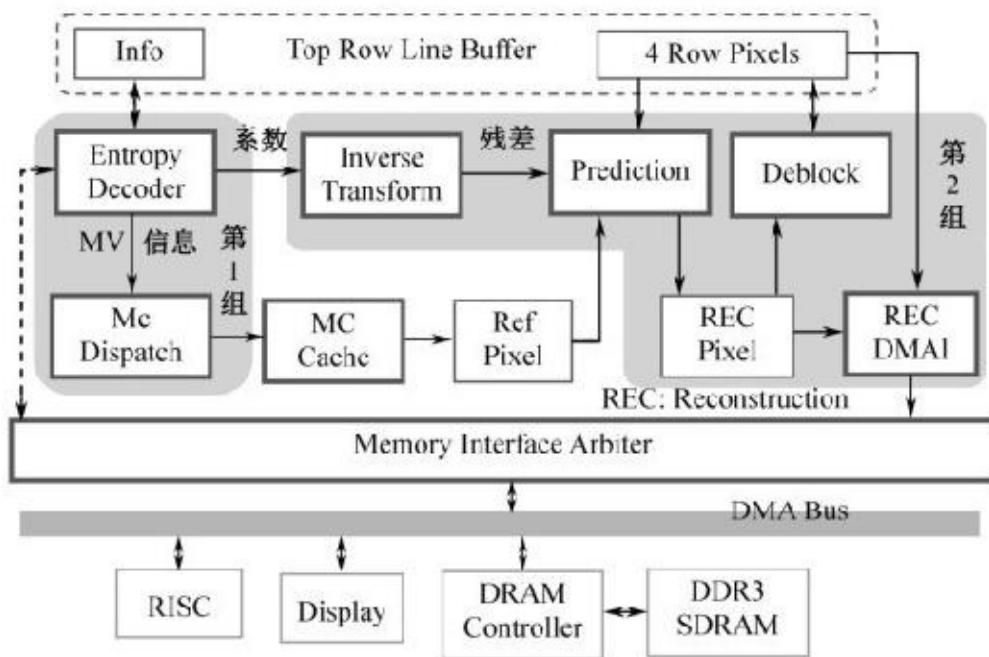


图13.6 系统框图和流水线技术

13.5.3 HEVC编码器芯片

图13.7显示了一个在 25mm^2 晶片 (die) 上实现的单片HEVC编码芯片，28nm集成工艺。这个芯片支持 $8192\times4320\text{p}/30\text{fps}$ 视频的实时编码，此时的功耗约为700MW。芯片设计面临的3个关键挑战为：

- (1) 帧级数据依赖形成巨大的外部带宽；
- (2) 参考帧的存取需要多接口 (port) 和高带宽；
- (3) 对于HEVC中复杂的预测模式需要高复杂度的判决能力。

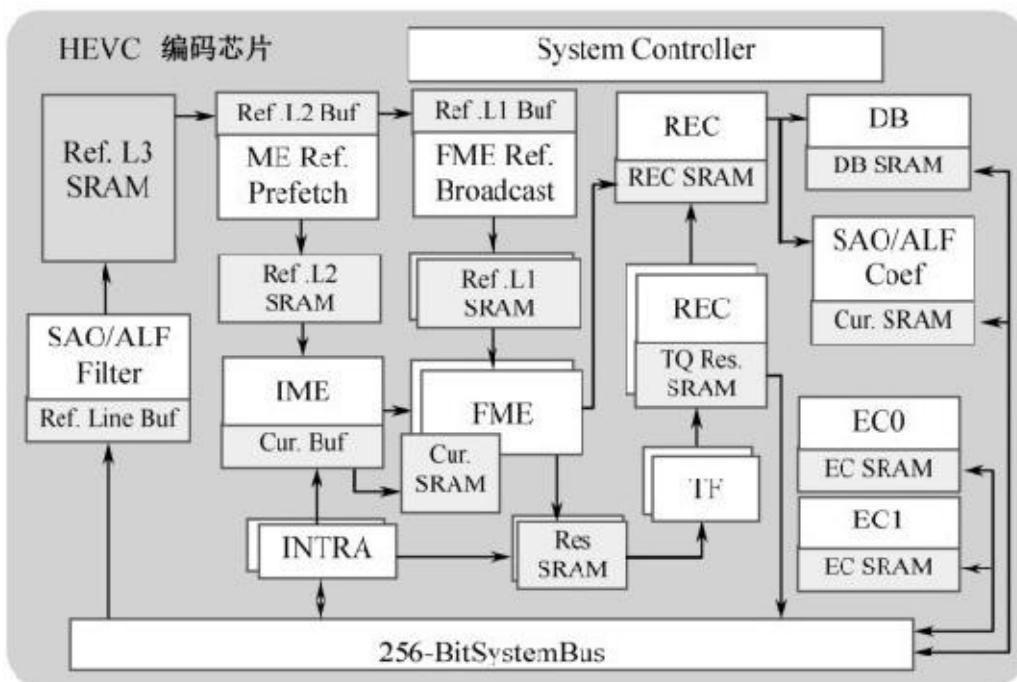


图13.7 HEVC编码芯片框图

为克服这些挑战，采取了对应的若干技术：

(1) 采用帧级流水线结构，减少8.9Gbps外部存储带宽，降低50%的CABAC计算量；

(2) 采用三等级存储结构，提供了13个口的协同存取和内部43.38Gbps的带宽，减少外部参考帧存取带宽到2.97Gbps；

(3) 采用高复杂度模式判决硬件，它带有低成本上下文固定二进制算术编码（Context Fixed BAC,CFBAC）的码率估计器。

13.5.4 HEVC编码系统

SHV (Super Hi-Vision) 视频的分辨率为 7680×4320 ,10/12比特深度，帧率可高达120帧/秒。对这种高分辨率的视频进行编码是一项十分具有挑战性的任务，这里简单介绍一台用于SHV的HEVC硬件编码设备，它基本满足HEVC编码指标。

该设备的开发分为三个步骤完成：第一步，使用双绿色通道偏移的4个800万像素 (3840×2160) Bayer 模式的系统作为输入视频。它具有每像素10比特深度，60帧/秒的帧率。每个编码部分处理大致和HDTV相同的像素数。第二步，使用全8K系统，每个R、G、B信号都具有3300万像素SHV系统 (7680×4320) ,12比特深度，59.94和60帧/秒的帧率。最后一步是实现对上述视频的编码，开发的编码器基于HEVC工作草案4，和HEVC当前标准的主要差别在于缺少SAO、AMP (Asymmetric Motion Partitioning) 和WPP。技术指标说明速率在85Mbps时的编码图像质量完全符合要求。

图13.8是硬件HEVC编码设备的外观图，它包含1个CPU部件和17个编码部件。CPU单元管理每个编码部件的目标比特率。为了降低每个编码部件的计算复杂度，将SHV图像分为17个slices分别编码，每个编码部件压缩相应的slice。CPU部件通过CPU总线负责控制编码参数，如G具有有12比特深度，经由4根光纤输入，视频格式转换和编码器之间信号则由17根电缆传送到各编码单元。

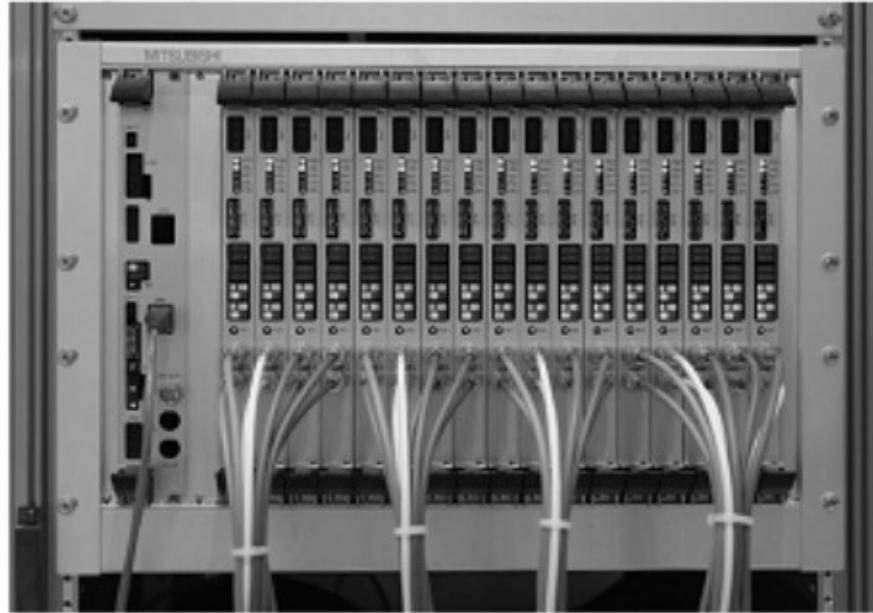


图13.8 SHV视频的HEVC编码设备

图13.9显示了slices的划分结构，将SHV图像均匀地划分为17个slice。每个slice的行数为64的整数倍，符合CTU的 64×64 的要求。每个编码单元在进行运动补偿预测时都需参考邻近的单元。这样的水平slices分割适于摇摄之类的平动，因为运动矢量的方向主要是水平的，并且只和2个相邻的单元有关。而如果采用格状（latticed）或其他方式的slice划分，常需要关联8个相邻的单元来实现运动估计和运动补偿。为了进一步降低编码计算功耗，对于 16×16 或更大的PU，只能采用部分帧内预测方向；对于 8×8 的PU，可被采所有的预测方向，以保证图像质量；不采用 4×4 的PU。

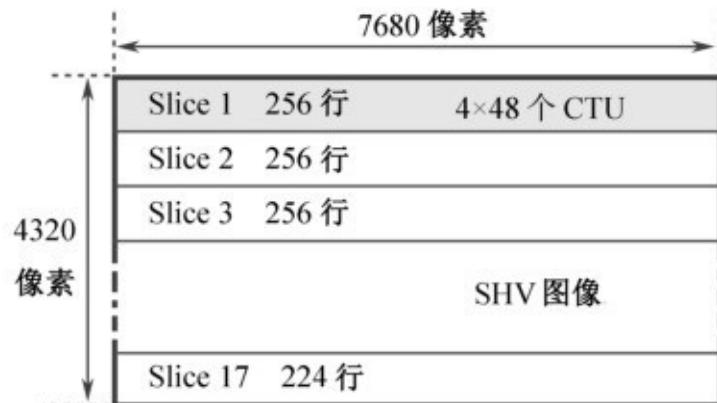


图13.9 SHV图像的slice划分

编码设备的主要指标如表13.12所示。为了保证重建图像质量，编码器使用预测残差和纹理特性来决定编码参数，如CU划分和预测模式。对一幅SHV图像，每种图像类型的划分和比特分配都需优化，特别是对低比特率情况更是如此。

表13.12 SHV编码器主要性能

| 参数 | 指标 |
|------------|---------------------------------|
| 视频编码技术 | HEVC (WD 4.0) Main10 档次, 6.1 水平 |
| 分辨率、帧率 | 7680×4320, 60fps |
| 彩色格式, 比特深度 | 4:2:0, 10 比特 |
| 最大比特率 | 340 Mbps |

本章参考文献

- [1]Frank Bossen,Benjamin Bross,Karsten Suhring,et al.HEVC complexity and implementation analysis[J].IEEE Trans.On Circuits and Systems for Video Technology,2012,22 (12) :1685-1696.
- [2]F.Bossen.On Software Complexity[R],document JCTVC-G757,JCT-VC,Geneva,Switzerland,Nov.2011.
- [3]F.Pescador,J.P.Caño,M.J.Garrido,E.Juarez,M.Raulet.A DSP HEVC decoder implementation based on OpenHEVC[C].2014 IEEE International Conference on Consumer Electronics (ICCE) ,Da Nang,Vietnam,30 July-1 August,2014: 61-62.
- [4]Chao-Tsung Huang,Mehul Tikekar,Chiraag Juvekar,Vivienne Sze,Anantha Chandrakasan.A 249 Mpixel/s HEVC Video-Decoder Chip for Quad Full HD Applications[C].2013 IEEE International Solid-State Circuits Conference,2013: 162-164.
- [5]Sung-Fang Tsai,Chung-Te Li,Hsuan-Hung Chen,et al.A 1062 M pixels/s 8192x4320p High Efficiency Video Coding (H.265) Encoder Chip[C].2013 Symposium on VLSI Circuits Digest of Technical Papers,C188-189.
- [6]Kazuhisa Iguchi,Atsuro Ichigaya,Yasuko Sugito,et al.HEVC Encoder for Super Hi-Vision[C].2014 IEEE International Conference on Consumer Electronics (ICCE) ,Da Nang,Vietnam,30 July-1 August,2014 : 57-58.
- [7]Open HEVC[EB/OL].<https://github.com/openhevc/openhevc>.
- [8]Frank Bossen,David Flynn,Karsten Sühring.HM Software Manual v9.0[R],ITU-T/IEC JVT-VC,2012.11.
- [9]Il-Koo Kim High Efficiency Video Coding (HEVC) Test Model

15 (HM15) Encoder Description[R].Doc.JCTVC-Q1002.17 th meeting: JCT-VC of ITU-T SG 16 and ISO/IEC JTC 1/SC 29,27 Mar.-4 Apr.2014,Valencia,2014.

[10]F.Bossen,Common Test Conditions and Software Reference Configurations[R],document JCTVC-H1100,JCT-VC,San Jose,CA,Feb.2012.

[11]Mathias Wien.High Efficiency Video Coding: Coding Tools and Specification[M].Springer-Verlag Berlin Heidelberg,2015.

[12]Vivienne Sze,Madhukar Budagavi,Gary J.Sullivan.High Efficiency Video Coding (HEVC) :Algorithms and Architectures[M].Switzerland Springer International Publishing,2014.

[13]M.Chavarriás,F.Pescador,M.J.Garrido,et al.A multicore DSP HEVC decoder using an actor based dataflow model and OpenMP[J].IEEE Trans.on Consumer Electronics,2015,61 (2),236-144.

[14]Jianghan Nan,Ningmei Yu.A DST hardware structure of HEVC[C].2015 2nd International Conference on Information Science and Control Engineering,546-550.

[15]Matteo Naccari,Andrea Gabriellini,Marta Mrak,et al.HEVC coding optimization for Ultra High Definition Television services[C].IEEE Picture Coding Symposium (PCS 2015) ,20-24.

[16]HEVC Software Reference Manual:
https://hevc.hhi.fraunhofer.de/svn/svn_HEVC Software/branches/HM-9.2-dev/doc/software-manual.pdf.

[17]D.Grois,D.Marpe,A.Mulayoff.Performance Comparison of H.265/MPEG-HEVC,VP9,and H.264/MPEG-AVC Encoders[C],IEEE Picture Coding Symposium (PCS 2013) ,San José,USA,Dec.8-11,2013,394-397.

[18]J.Ohm,J.Sullivan,H.Schwarz,et al.Comparison of the Coding Efficiency of Video Coding Standards-including High Efficiency Video Coding (HEVC) [J].IEEE Transactions on Circuits and Systems for Video

Technology,2012,22 (12) : 1669-1684.

[19]High Efficiency Video Coding[S],ITU-T Rec.H.265 and ISO/IEC 23008-2,ITU-T and ISO/IEC JCT-VC,Mach 2013 (v.1) ,Oct.2014 (v.2) ,Apr.2015 (v.3) .

缩略语 (Abbreviations)

3D-HEVC 3D High Efficiency Video Coding 三维高效视频编码

AAC Adaptive AC 自适应算术编码

AI All Intra 全帧内

ALF Adaptive Loop Filter 自适应环路滤波

AMP Asymmetric Motion Partitioning 非对称运动划分

AMVP Advanced Motion Vector Prediction 高级运动矢量预测

ARP Advaced Residual Predition 高级残差预测

ASO Arbitrary Slice Order 任意条次序

AU Access Unit 接入单元

AVC Advanced Video Coding 高级视频编码

BLA Broken Link Access 断点连接接入

BL Base Layer 基本层

BMA Block Matching Arithmetic 块匹配算法

BO Band Offset 带补偿

BP Baseline Profile 基本档次

CABAC Context-based Adaptive Binary Arithmetic Coding 上下文自适应

二进制算术编码

CAVLC Context Adaptive Variable Length Coding 上下文自适应变长编码

CB Coding Block 编码块

CBF Coded Block Flag 编码块标志

CBP Coded Block Pattern 编码块模板

CBR Constant Bit Rate 固定比特率
CD Committee Draft 委员会草案
CG Coefficient Group 系数组
CGS Coarse Grain Scalability 粗粒度可分级
CIF Common Intermediate Format 公共中间格式
CPB Coded Picture Buffer 编码图像缓存
CRA Clean Random Access 纯随机接入
CTB Coding Tree Block 编码树块
CTU Coding Tree Unit 编码树单元
CU Coding Unit 编码单元
CVS Coded Video Sequence 编码视频序列
DAM Draft Amendment 草案
DBF De-Blocking Filter 去方块滤波
DC Direct Current 直流
DCT Discrete Cosine Transform 离散余弦变换
DIBR Depth Image Based Rendering 基于深度图的渲染
DIS Draft International Standard 国际标准草案
DLP Decodable Leading Picture 可解码前置图像
DMM Depth Modeling Mode 深度建模模式
DPB Decoded Picture Buffer 解码图像缓存
DPCM Differential Pulse Code Modulation 差值脉冲编码调制
DSCQS Double Stimulus Continuous Quality Scale 双刺激连续质量评分
DST Discrete Sine Transform 离散正弦变换
DTV Digital TV 数字电视
DU Decoding Unit 解码单元
DV Disparity Vector 视差矢量
DVD Digital Video Disc 数字视频光盘
EC Entropy Coding 熵编码

EG Exponential-Golomb 指数哥伦布 (编码)
EL Enhancement Layer 增强层
EO Edge Offset 边缘补偿
EOB End Of Bitstream 比特流结束
EOS End Of Sequence 序列结束
EP Extended Profile 扩展档次
ES Elementary Stream 基本流
FCD Final Committee Draft 委员会最终文档
FDAM Final Draft Amendment 最后草案
FDIS Final Draft International Standard 国际标准最后草案
FGS Fine Granularity Scalability 细粒度可分级
FIFO First-In First-Out 先进先出
FIR Finite Impulse Response 有限冲激响应
FM Frame Memory 帧存
FMO Flexible Macroblock Ordering 灵活宏块顺序
FPGA Field-Programmable Gate Array 现场可编程门阵列
FS Full Search 全搜索
GOP Group Of Pictures 图像组
GPB Generalized P and B picture 普通P和B帧
GPU Graphics Processing Unit 图形处理单元
HBS Hierarchical B Structure 层次B帧结构
HD High Definition 高清 (视频)
HDCIF High Definition CIF 高清通用图像格式
HDMI High Definition Multimedia Interface 高清多媒体接口
HDR High Dynamic Range 高动态范围
HDTV High Definition TeleVision 高清晰度电视
HEVC High Efficiency Video Coding 高效视频编码
HM HEVC test Model HEVC测试模式

HP High Profile 高档次
H10P High 10 Profile 高10档次
H4:2:2P High 4:2:2 Profile 高4:2:2档次
H4:4:4PP High 4:4:4 Predictive Profile 高4:4:4预测档次
HRD Hypothetical Reference Decoder 理想参考解码器
HSS Hypothetical Stream Scheduler 理想流调度
HTM 3D-HEVC Test Model 3D-HEVC测试模型
HVS Human Visual System 人类视觉系统
I Intra 帧内
IBDI Internal Bit Depth Increase 内部比特深度增加
IBR Image Based Rendering 基于图像的绘制
IC Illumination Compensation 亮度补偿
IDCT Inverse Discrete Cosine Transform 反离散余弦变换
IDE Integrated Development Environment 集成开发环境
IDR Instantaneous Decoding Refresh 即时解码刷新
IDW Image Domain Warping 图像域变形
IEC International Electrotechnical Commission 国际电工委员会
IQ Inverse Quantization 反量化
IRAP Intra Random Access Point 帧内随机接入点
IS International Standard 国际标准
ISA Instruction Set Architecture 指令集架构
ISO International Standardization Organization 国际标准化组织
ITU International Telecommunication Union 国际电信联盟（电联）
ITU-T ITU for Telecommunication standardization sector 国际电联通信标准化组织
ITU-R ITU-Radio communications sector 国际电联无线电通信部门
JCT-VC Joint Collaborative Team on Video Coding 视频编码联合协作组
JCT-3V JCT on 3D Video coding extension development 3D视频编码联合

协作组

JM Joint Model 联合模型 (AVC测试模型)

JPEG Joint Photographic Experts Group 静止图像专家组

JTC Joint Technical Committee 联合技术委员会

JVT Joint Video Team 联合视频组

KLT Karhunen-Loeve Tranform KL变换

KTA Key Technical Areas 关键技术领域

LCD Liquid Crystal Display 液晶显示

LD Low Delay 低时延

LDTV Low Definition TV 低清晰度电视

LED Light Emitting Diode 发光二极管

LF Loop Filtering 环路滤波

LP Leading Picture 前置图像

LPS Least Probable Symbol 小概率符号

LSB Least Significant Bit 最不重要比特

MB Macro Block 宏块

MBR Model Based Rendering 三维模型的绘制

MC Motion Compensation 运动补偿

MDCS Mode-Dependent Coefficient Scan 依赖模式的系数扫描

ME Motion Estimation 运动估计

MGS Medium Grain Scalability 中粒度可分级

MMX Multi Media eXtension 多媒体扩展

MOS Mean Opinion Score 平均评价分

MP Main Profile 主档次

MPEG Moving Picture Experts Group 活动图像专家组

MPM Most Probable Mode 最可能模式

MPS Most Probable Symbol 大概率符号

MSB Most Significant Bit 最重要比特

MSE Mean Squared Error 均方误差

MTU Maximum Transmission Unit 最大网络传输单元

MV Motion Vector 运动矢量

MVC MV Competition 运动矢量竞争

MVC Multiview Video Coding 多视点视频编码

MVD MV Difference 运动矢量差

MVD Multiview Video plus Depth 多视点视频+深度

MVP MV Prediction 运动矢量预测

MV-HEVC MultiView HEVC 多视点高效视频编码

MVV Multi View Video 多视点视频

NAL Network Abstraction Layer 网络提取层

NALU Network Abstraction Layer Unit NAL单元

NBDV Neighboring Block based Disparity Vector 基于相邻块的视差矢量

NTSC National Television Systems Committee 国家电视系统委员会

PAL Phase Alternating Line 逐行倒相

PB Prediction Block 预测块

PCM Pulse Code Modulation 脉冲编码调制

PDAM Proposed Draft Amendment 提议草案

POC Picture Order Count 图像顺序计数

PPS Picture Parameter Set 图像参数集

PSNR Peak Signal to Noise Ratio 峰值信噪比

PU Prediction Unit 预测单元

QCIF Quarter CIF 1/4CIF

QFHD Quad Full HD 4倍全高清

QHD Quarter High Definition 1/4高清

QP Quantization Parameter 量化参数

RA Random Access 随机接入

RADL Random Access Decodable Leading 可解码随机接入前置（图

像)

- RAP Random Access Point 随机访问点
- RASL Random Access Skipped Leading 跳过随机接入前置 (图像)
- RBC Regin Boundary Chain 区域边界链码
- RBSP Raw Byte Sequence Payload 原始字节序列载荷
- RD Rate-Distortion 率失真
- RDO RD Optimization 率失真优化
- RDOQ RDO Quantization 率失真优化的量化
- RExt Range Extensions 范围扩展
- RGB Red Green Blue 红绿蓝
- ROI Region Of Interesting 感兴趣区间
- RPL Reference Picture List 参考图像列表
- RPS Reference Picture Set 参考图像集
- RQT Residual Quad-Tree 残差四叉树
- RTP Real-time Transport Protocol 实时传输协议
- SAD Sum of Absolute Differences 绝对差值之和
- SAE Sum of Absolute Error 绝对误差之和
- SAO Sample Adaptive Offset 样点自适应补偿
- SAR Sample Aspect Ration 样点宽高比
- SB Sub-Block 子块
- SCC Screen Content Coding 屏幕内容编码
- SD Standard Definition 标准清晰度 (视频)
- SDTV Standard Definition TV 标准清晰度电视
- SEI Supplemental Enhancement Information 附加增强信息
- SG Slice Group 条组
- SG Study Group 研究组
- SHM SHVC Test Model SHVC的测试模式
- SHV Super Hi-Vision 超高分辨率

SHVC Scalable High efficiency Video Coding 可分级高效视频编码

SI Side Information 边信息

SIMD Single Instruction Multiple Data 单指令多数据流

SODB String Of Data Bits 数据比特串

SOP Structure Of Pictures 图像结构

SP Switching Prediction 切换预测

SPS Sequence Parameter Set 序列参数集

SQICF Sub-QCIF 准QCIF

SS Slice Segment 条分割

SSE Streaming SIMD Extensions 单指令多数据流扩展

SSCQE Single Stimulus Continuous Quality Evaluation 单刺激连续质量

评分

SSD Sum of Squared Differences 差值平方和

SSE Sum of Squared Error 误差平方和

SSIM Structure Similarity Image Measurement 构相似度图像评价

STSA Step-wise Temporal Sub-layer Access 逐步时域子层接入

SVC Scalable Video Coding 可分级视频编码

TB Transform Block 变换块

TMVP Temporal Motion Vector Predictor 时域运动矢量预测

TP Trailing Pictures 后置图像

TR Truncated Rice 截断Rice (编码)

TS Temporal Sub-layer 时域子层

TSA Temporal Sub-layer Access 时域子层接入

TSS Three Step Search 三步搜索

TU Transform Unit 变换单元

TUB Truncated Unary Binarization 二进制一元截断 (编码)

UDP User Datagram Protocol 用户数据报协议

UHDTV Ultra High Definition TV 超高清晰度电视

URQ Uniform Reconstruction Quantizers 均匀重建量化器

VBR Variable Bit Rate 可变比特率

VCEG Visual Coding Experts Group 视频编码专家组

VCD Video Compact Disc 视频光盘

VCL Video Coding Layer 视频编码层

VGA Video Graphics Array 视频图形阵列

VLC Variable Length Coding 变字长编码

VPS Video Parameter Set 视频参数集

VPS Video Parameter Set 视频参数集

VQ Vector Quantization 矢量量化

VQEG Video Quality Experts Group 视频质量专家组

VSP View Synthesis Prediction 视点合成预测

VUI Video Usability Information 视频可用信息 WD Working Draft 工作

草案

WG Working Group 工作组

WPP Wavefront Parallel Processing 波前并行处理

反侵权盗版声明

电子工业出版社依法对本作品享有专有出版权。任何未经权利人书面许可，复制、销售或通过信息网络传播本作品的行为；歪曲、篡改、剽窃本作品的行为，均违反《中华人民共和国著作权法》，其行为人应承担相应的民事责任和行政责任，构成犯罪的，将被依法追究刑事责任。

为了维护市场秩序，保护权利人的合法权益，我社将依法查处和打击侵权盗版的单位和个人。欢迎社会各界人士积极举报侵权盗版行为，本社将奖励举报有功人员，并保证举报人的信息不被泄露。

举报电话：（010）88254396; （010）88258888

传真：（010）88254397

E-mail: dbqq@hei.com.cn

通信地址：北京市万寿路173信箱 电子工业出版社总编办公室

邮编：100036