ICS 13.310
A 91

中华人民共和国国家标准

GB/T 25724—2010

安全防范监控数字视音频编解码技术要求

Technical specification of surveillance video and audio coding

2010-12-23 发布　　2011-05-01 实施

中华人民共和国国家质量监督检验检疫总局
中国国家标准化管理委员会　发布

目　次

前　言

请注意，本标准的某些内容有可能涉及专利，本标准的发布机构不承担识别这些专利的责任。

本标准的附录 A～附录 H 为规范性附录，附录 I 和附录 J 为资料性附录。

本标准由中华人民共和国公安部提出。

本标准由全国安全防范报警系统标准化技术委员会(SAC/TC 100)归口。

本标准起草单位：公安部第一研究所、北京中星微电子有限公司、北京中盾安全技术开发公司、中星电子股份有限公司、清华大学、香港大学、大连理工大学、江苏东奇信息科技有限公司、中国传媒大学信息工程学院、国家多媒体软件工程技术研究中心、宁波艾利特科技发展有限公司、杭州恒生数字设备科技有限公司、公安部第三研究所、浙江大华技术股份有限公司、北京声迅电子有限公司、天津市亚安科技电子有限公司、深圳市艾立克电子有限公司、浙江大立科技股份有限公司、北京国通创安信息技术有限公司、天津天地伟业数码科技有限公司、金鹏电子信息机器有限公司、北京蛙视通信技术有限责任公司、杭州海康威视数字技术股份有限公司、中国科学院软件研究所、深圳中兴力维技术有限公司、北京汉邦高科数字技术有限公司、宁波舜宇光电信息有限公司、数维科技(北京)有限公司、新太科技股份有限公司、星际控股集团有限公司、浙江警官职业学院、北京富盛星电子有限公司、杭州华三通信技术有限公司、广东志成冠军集团有限公司。

本标准主要起草人：陈朝武、邓中翰、李晓峰、杨晓东、张跃、邱嵩、冯宇红、卢京辉、余子龙、袁丽蓉、费宝顶、高嵩、林冬、陈喆、钟兴业、王生进、杨磊、房子河、杨国胜、范京京、邹章彪、郅晨、王耀辉、王浩、李鹏飞、王建勇、高磊、王晛、魏一、孙大瑞、闫建新、余和初、戴林、陈瑞军、于烨、黄麒麟、季鹏飞、韩大炜、刘蕾蕾、陈玉、周志文、向稳新、吴参毅。

引　言

目前国内、国际没有专门针对安全防范监控应用的视音频编解码标准，现有的视音频编解码标准，都是针对广播电视和大众娱乐方面的应用，在安全防范领域直接采用具有很大的不适应性。本标准专门针对安防监控领域应用的特殊性，如：视频图像的实时传输性、全天候 24 h 监控环境的适应性、场景视音频信息的忠实还原性等要求制定。本标准主要技术特点有：

a) 支持高精度视频数据编码，适应宽动态范围，保留更多的图像细节，满足忠实于场景的要求。视频支持 8 bit～10 bit 数据，并保留未来扩充到 12 bit～16 bit 的可能；

b) 支持帧内 4×4 预测与变换量化、自适应帧—场编码(AFF)和上下文自适应二进制算术编码(CABAC)等技术，获得更好的图像质量和更高的编码效率；

c) 支持感兴趣区域(ROI)变质量编码，在传输网络带宽或数据存储空间有限的情况下，优先保证 ROI 图像质量，节省非 ROI 的开销，提供更符合监控需要的高质量视频编码，提高监控系统整体性能；

d) 支持可伸缩性视频编码(SVC)，对视频数据分层次编码，满足不同传输网络带宽和数据存储环境的需求；

e) 支持代数码书激励线性预测(ACELP)和变换音频编码(TAC)切换的双核音频编码，既保证对语音信号具有较好的编码效果，也保证环境(背景)声音的编码效果；

f) 支持声音识别特征参数的编码，避免编码失真对语音识别和声纹识别的影响；

g) 支持绝对时间参考信息、特殊监控事件等监控专用信息。监控专用信息通过专门语法与视音频压缩编码数据一起传输和存储，便于快速检索、分类查询、视音频同步和监控数据的综合应用；

h) 支持数据安全保护，规定加密和认证接口及数据格式，保证数据的安全性、完整性和非否认性。既保证格式的统一，便于互联互通，也保留足够的扩展灵活性，支持更高性能的加密和认证方式的增加和扩充。

相关专利情况说明

本文件的发布机构提请注意，声明符合本文件时，可能涉及与 5.2.3.1、5.2.3.2、5.2.3.8、5.2.4.2、5.2.4.4、5.2.4.10、5.3.6.7、6.1.2、6.1.4、6.2.6.1.3、6.2.6.1.4.10 中有关内容相关的专利的使用。

本文件的发布机构对于该专利的真实性、有效性和范围无任何立场。

该专利持有人已向本文件的发布机构表示，他愿意同任何申请人在合理且无歧视的条款和条件下，就专利授权许可进行谈判。该专利持有人的声明已在本文件的发布机构备案。相关信息可以通过以下联系方式获得：

专利持有人名称	联　系　地　址
北京中星微电子有限公司	北京海淀学院路 35 号世宁大厦(100191)
北京中盾安全技术开发公司	北京海淀区首体南路 1 号(100048)
中星电子股份有限公司	天津经济技术开发区第四大街 80 号天大科技园 A1 座 2 层(300457)
清华大学	北京海淀区清华园(100084)
数维科技(北京)有限公司	北京海淀区中关村南大街 2 号(100086)
武汉大学	湖北武汉市武汉大学(430079)

联系人：曾娟鹃
通讯地址：北京海淀区学院路35号世宁大厦16层
邮政编码：100191
电子邮件：zengjuanjuan@vimicro.com
电话：010-68948888-8950
传真：010-68944075

联系人：马志江
通讯地址：北京海淀区首体南路1号
邮政编码：100048
电子邮件：mzj76@yahoo.com
电话：010-88513553-828
传真：010-68454099

请注意除上述专利外，本文件的某些内容仍可能涉及专利。本文件的发布机构不承担识别这些专利的责任。

安全防范监控数字视音频编解码技术要求

1 范围

本标准规定了安全防范领域监控应用的数字视音频编码、解码过程的技术要求。

本标准适用于安全防范领域的视音频实时压缩、传输、播放和存储等业务，对于其他需要视音频编解码的领域也可参考采用。

2 规范性引用文件

下列文件中的条款通过本标准的引用而成为本标准的条款。凡是注日期的引用文件，其随后所有的修改单(不包括勘误的内容)或修订版均不适用于本标准，然而，鼓励根据本标准达成协议的各方研究是否可使用这些文件的最新版本。凡是不注日期的引用文件，其最新版本适用于本标准。

GB/T 20090.2—2006 信息技术 先进音视频编码 第2部分：视频

3 术语、定义和缩略语

下列术语、定义和缩略语适用于本标准。

3.1 术语和定义

3.1.1

“Z”字形扫描 zig-zag scan

变换系数从较低的空域频率到较高空域频率(近似)的一个明确排列顺序。“Z”字形扫描用于帧宏块中的变换系数。

3.1.2

B条带 B slice

根据同一条带内的已解码样点利用帧内预测进行解码，或者根据先前解码的参考图像利用双向预测进行解码的条带，对每个块进行帧间预测时最多使用两个运动矢量和参考索引。

3.1.3

I条带 I slice

根据同一条带内的已解码样点利用帧内预测进行解码的条带。

3.1.4

P条带 P slice

根据同一条带内的已解码样点利用帧内预测进行解码，或者根据先前解码的参考图像利用前向预测进行解码的条带，对每个块进行帧间预测时最多使用一个运动矢量和参考索引。

3.1.5

NAL单元 NAL unit

一个语法结构，包含后续数据的类型指示和所包含的字节数，数据以RBSP形式出现，必要时其中还包括认证数据及散布的防伪字节。

3.1.6

NAL单元流 NAL unit stream

由NAL单元组成的序列。

3.1.7

保留 reserved

某些语法元素的特定取值，供中国安全防范监控数字视音频编解码技术标准工作组将来使用。符合本标准的比特流不应使用这些值，但是这些值将来可能在本标准的扩展版本中用到。

3.1.8

闭环基音搜索 closed-loop pitch search

即自适应码书搜索，从加权输入信号和长时预测滤波器状态估计基音延迟的过程。

3.1.9

比特流 bitstream

编码视音频及其相关数据，构成一个或多个编码视音频序列的比特序列。比特流既可用来表示NAL单元流，也可表示字节流。

3.1.10

变换系数 transform coefficient

频率域的标量，与解码过程的反变换部分中一个特定的一维或二维频率索引相关联的系数。

3.1.11

变换系数幅值 transform coefficient level

一个与特定二维频率索引相关联的整数量值，解码过程中用于计算变换系数的值。

3.1.12

编码场 coded field

一个场的编码表示。

3.1.13

编码过程 encoding process

产生符合本标准的比特流的过程，本标准对视频编码过程不做规定。

3.1.14

编码器 encoder

实现编码过程的实体，包括软件及硬件。

3.1.15

编码视频序列 coded video sequence

按照解码顺序排列的IDR图像和紧随其后的零个或多个非IDR图像组成的图像序列。

3.1.16

编码条带NAL单元 coded slice NAL unit

包含编码图像的一个条带的NAL单元。

3.1.17

编码图像 coded picture

一幅图像的编码表示。一个编码图像可以是一个编码场，也可以是一个编码帧。

3.1.18

编码图像缓存区 coded picture buffer

一个先入先出缓存区，其存储方式按解码顺序排列。

3.1.19

编码帧 coded frame

一个帧的编码表示。

3.1.20

残差 residual

样点或数据元素预测值与解码值之间的差值。

3.1.21

参考场 reference field

一个标记为参考图像的场，用于编码场中P条带和B条带的解码过程中的帧间预测。

3.1.22

参考索引 reference index

参考图像的索引。

3.1.23

参考图像 reference picture

对解码顺序上后续图像的解码过程进行帧间预测的样点图像。

3.1.24

参考帧 reference frame

一个标记为参考图像的帧，用于编码帧中的P条带和B条带的解码过程中的帧间预测。

3.1.25

参数 parameter

序列参数集、图像参数集或安全参数集中的一个语法元素。参数也用于量化参数一词中。

3.1.26

层 layer

没有分支等级关系中的一组句法结构。高层包含低层。编码层指编码图像序列层、图像层、条带层和宏块层。对于可伸缩性视频编码图像，不同层的图像具有不同的可伸缩性(如不同的空间分辨率)。

3.1.27

场 field

一帧中的相隔行的集合。一帧由两场组成，包括一个顶场和一个底场。

3.1.28

场宏块 field macroblock

所包含的样点仅来自一个编码场的宏块。一个编码场的所有宏块均为场宏块。

3.1.29

场扫描 field scan

变换系数的排列顺序。与"Z"字形扫描顺序不同的是，它对列的扫描快于对行的扫描。场扫描用于场宏块中的变换系数。

3.1.30

代数码书 algebraic codebook

脉冲幅度和位置组成的一个集合。通过码字索引k按照一定的规则得到第k个激励码矢量的脉冲幅度和位置。

3.1.31

档次 profile

本标准中的一个特定语法子集。

3.1.32

底场 bottom field

组成帧的两个场中的一个。底场的每一行在空间位置上均位于其对应的顶场行的下面。

3.1.33

电导频谱对 immittance spectral pair

线性预测系数的变换称为电导频谱对。将逆滤波器传输函数$A(z)$分解为一个偶对称和一个奇对称多项式函数，该函数在单位圆上的根，即电导频谱对。

3.1.34

顶场 top field

组成帧的两个场中的一个。顶场的每一行在空间位置上均位于其对应的底场行的上面。

3.1.35

短时合成滤波器 short term synthesis filter

对声道脉冲响应进行建模的滤波器。激励信号通过该滤波器可得到合成信号。

3.1.36

二进制位 bin

二进制位串中的 1 bit。

3.1.37

二进制位串 bin string

一串二进制位。二进制位串为二值化的语法元素值的二进制表示。

3.1.38

二值化 binarization

语法元素所有可能值与一组二进制位串之间的唯一映射。

3.1.39

反变换 inverse transform

解码过程的一部分，将变换系数矩阵转换为空域样点矩阵的过程。

3.1.40

防伪字节 emulation prevention byte

一个字节，其值等于 0x03，可能在 NAL 单元中出现。防伪字节的出现可以保证在 NAL 单元的后续字节对齐的字节流中不会含有起始码前缀。

3.1.41

非参考图像 non-reference picture

不用于对任何其他图像进行帧间编码的图像。

3.1.42

分量 component

图像的三个样点矩阵(一个亮度矩阵，两个色度矩阵)中的一个矩阵或矩阵中的单个样点。

在音频部分，也指矢量中的元素或信号中的某些频率成分。

3.1.43

感知加权滤波 perceptual weighting filter

利用共振峰处的噪声掩蔽特性，在共振峰区域分配比较大的失真，来减少峰谷主观感觉噪声的滤波。

3.1.44

功率谱 power spectrum

信号通过傅立叶变换后得到幅度谱的平方。

3.1.45

光栅扫描 raster scan

矩形二维图像到一维图像的映射过程，一维图像的第一组值来自于二维图像最上边一行的从左到右扫描，然后依次是第二行、第三行等。对于图像每行(由上到下)都是从左到右扫描的。

3.1.46

宏块 macroblock

一个 16×16 的亮度样点块和相应的两个色度样点块。

3.1.47

宏块索引 macroblock index

编码帧中，宏块索引为帧图像的宏块光栅扫描顺序的序号，起始序号为0。编码场中，宏块索引为场图像的宏块光栅扫描顺序的序号，起始序号为0。

3.1.48

后向预测 backward prediction

使用显示顺序上在后的解码图像中的样点对当前图像中的样点进行预测。

3.1.49

划分 partitioning

将一个集合分为子集的过程。集合中的每个元素属于且只属于某一个子集。

3.1.50

基本层图像 base layer picture

不需要参考其他图像层信息即可以解码的图像。

3.1.51

级别 level

本标准中的一个特定档次中的参数取值的限定集合。一个档次可以包含一个或多个级别。对所有档次定义了一组相同的级别，不同档次的每个级别大部分特性都是通用的。对于一个独立的实现，在一定的约束条件下，可以支持多个级别。

3.1.52

即时解码刷新(IDR)图像 instantaneous decoding refresh (IDR) picture

一幅编码图像，其中所有条带为I条带。IDR图像解码之后，解码顺序上所有后续的编码图像都可以不用根据任何在IDR图像之前解码的图像来进行帧间预测解码。每个编码视频序列的第一幅图像为IDR图像。

3.1.53

假设参考解码器 hypothetical reference decoder

一个假设的解码器模型，规定了对于符合本标准的NAL单元流或字节流的可变性的约束。

3.1.54

解码过程 decoding process

读入编码的比特流后产生解码图像或者音频数据的过程。

3.1.55

解码器 decoder

实现解码过程的实体，包括软件及硬件。

3.1.56

解码顺序 decoding order

解码过程中处理语法元素的顺序。

3.1.57

解码图像 decoded picture

通过解码一幅编码图像得到的图像。一幅解码图像既可以是一个解码帧，也可以是一个解码场。一个解码场可以是顶场，也可以是底场。

3.1.58

解码图像缓存区 decoded picture buffer

保存解码图像的缓存区，用于附录A中规定的预测参考、输出重排序或输出延时等。

3.1.59

开环基音搜索 open-loop pitch search

直接从加权输入信号中估计最优基音延迟的过程。开环基音搜索简化了基音分析，并且将闭环基音搜索限定在开环基音搜索的延迟值附近。

3.1.60

可变长度编码 variable length coding

可逆的熵编码过程，为出现概率大的符号分配较短的码字，为出现概率小的符号分配较长的码字。

3.1.61

可伸缩性视频编码 scalable video coding

编码序列中的图像具有一定的可伸缩性。具有可伸缩性的图像通常包含基本层图像和增强层图像。

3.1.62

块 block

在视频信号空间中，指一个 $M\times N$(M 列 N 行)的样点矩阵，或者一个 $M\times N$ 的变换系数矩阵。

在音频信号空间中，指一个一维矢量。

3.1.63

亮度 luma

一个样点矩阵或单个样点，用于描述信号的单色表示。亮度所用符号为 Y。

3.1.64

量化参数 quantization parameter

解码过程中对变换系数幅值进行反量化时使用的参数。

3.1.65

零输入响应 zero input response

滤波器当前输入为零时，由过去输入而产生的输出。

3.1.66

美尔 Mel

一种非线性的频率刻度，根据主观音高进行划分。

3.1.67

美尔频率倒谱系数 Mel-frequency cepstral coefficients

用 FFT 将时域信号转化到频域，对其对数能量谱依照 Mel 刻度分布的三角滤波器组进行卷积，对各个滤波器的输出构成的向量进行 DCT 得到的系数，即美尔频率倒谱系数。

3.1.68

内部采样频率 internal sampling frequency

音频编码器的采样频率，范围为 12 800 Hz～38 400 Hz，采用 F_s 表示。

3.1.69

逆滤波器 inverse filter

去除信号短时相关性的滤波器。

3.1.70

频率索引 frequency index

与解码过程中反变换之前的变换系数相关的一维或二维索引。

3.1.71

起始码前缀 start code prefix

字节流中唯一等于 0x000001 的 3 个字节的序列，作为每个 NAL 单元的前缀。解码器可以利用起

始码前缀的位置来确定一个新的 NAL 单元的开始和前一个 NAL 单元的结束。NAL 单元中通过加入防伪字节来防止假冒的起始码前缀出现。

3.1.72

前向预测 forward prediction

使用显示顺序上在前的解码图像中的样点对当前图像中的样点进行预测。

3.1.73

前向帧间解码图像 forward inter decoded picture

P 图像

帧间预测中只使用前向预测解码的图像。

3.1.74

色度 chroma

一个样点矩阵或单个样点，用于描述代表两个相对于基色的色差信号中的一个。色度所用符号为 Cb 和 Cr。

3.1.75

上下文自适应二进制算术编码 context adaptive binary arithmetic coding

一种熵编码方法，根据上下文内容对二进制位进行编码，产生比特流。

3.1.76

声纹识别 voiceprint recognition

根据语音的声学特征识别该段语音所对应的说话人的过程。

3.1.77

数据比特串 string of data bits

语法元素的若干比特位的序列，出现在原始字节序列负载中原始字节序列负载截止位之前。在 SODB 中，最左边的比特位表示第一位即最高位，最右边的比特位表示最后一位即最低位。

3.1.78

双向预测 bidirectional prediction

使用显示顺序上在前及在后的解码图像中的样点对当前图像中的样点进行预测。

3.1.79

双向帧间解码图像 bidirectional inter decoded picture

B 图像

帧间预测中使用双向预测解码的图像。

3.1.80

条带 slice

特定区域内部按照光栅扫描顺序排列的整数个宏块。虽然一个条带包含区域内部按照光栅扫描顺序排列的整数个宏块，但这些宏块在图像内部并不一定是按照光栅扫描顺序连续排列的。宏块索引可以通过条带的第一个宏块的索引以及宏块到条带的映射得到。

3.1.81

条带头 slice header

编码条带的一部分，包含与该条带中第一个或者全部宏块有关的数据元素。

3.1.82

跳过宏块 skipped macroblock

没有编码数据的宏块。

3.1.83

图像 picture

源、编码或重构的图像数据，场或帧的通称。对于逐行扫描视频，一幅图像指一帧；而对于隔行扫描

视频，一幅图像指一帧，或组成该帧的顶场或底场。

3.1.84

图像参数集 picture parameter set

一个语法结构，包含应用于零个或多个编码图像的语法元素，由每个条带头中的语法元素 pic_parameter_set_id 确定。

3.1.85

维纳滤波器 wiener filter

根据最小均方误差准则，即滤波器的输出信号与期望信号之差的均方值最小，计算得到的最佳线性滤波器，称为维纳滤波器。

3.1.86

线性预测系数 LP coefficients

短时预测滤波器系数，也称为 LPC 系数。

3.1.87

序列参数集 sequence parameter set

一个语法结构，包含应用于零个或多个完整编码视频序列的语法元素，由条带头中的语法元素 pic_parameter_set_id 确定所引用的图像参数集，由图像参数集中的语法元素 seq_parameter_set_id 确定所引用的序列参数集。

3.1.88

音频超帧 audio superframe

由若干音频帧组成，目前本标准规定音频超帧中只包含一个音频帧。

3.1.89

音频子帧 audio subframe

音频帧的一部分，在 $F_s/2$ 采样频率下，由 64 个样本构成的数据块。

3.1.90

游程 run

解码过程中连续出现的数据元素的数目。在某些上下文环境中，游程指“Z”字形扫描或场扫描后产生的变换系数数组中非 0 系数之前的 0 值变换系数的数目。

3.1.91

预测 prediction

使用预测值来提供当前解码的样点值或数据元素的估计。

3.1.92

预测值 predictor

以前解码的样点值或数据元素的线性组合。

3.1.93

语法结构 syntax structure

零个或多个语法元素按照规定顺序一起出现在比特流中。

3.1.94

语法元素 syntax element

比特流中表示数据的元素。

3.1.95

语音识别 speech recognition

根据语音的声学特征和语言模型，将该段语音翻译为文本的过程。

3.1.96

源 source

编码前视音频素材或者素材的某些属性。

3.1.97

原始字节序列负载 raw byte sequence payload

一个语法结构，包含整数个封装于 NAL 单元中的字节。RBSP 或者为空，或者包含具有数据比特串形式的语法元素，其后跟随 RBSP 截止位和零个或多个连续的 0 值比特。

3.1.98

原始字节序列负载(RBSP)截止位 raw byte sequence payload (RBSP) stop bit

值为 1 的一个比特，出现在原始字节序列负载(RBSP)中的数据比特串之后。RBSP 中数据比特串的结束位置可以通过搜索 RBSP 中的 RBSP 截止位得到。

3.1.99

运动矢量 motion vector

二维矢量，用于帧间预测，表示匹配对象在解码图像和参考图像中的位置偏移。

3.1.100

增强层图像 enhance layer picture

需要参考其他图像层信息进行解码的图像。本标准中的一个增强层图像在解码时可以参考位于其下的相邻的图像层信息，同时该增强层图像的空间分辨率在水平和垂直方向上均为位于其下的相邻的(用来参考的)图像层的二倍。

3.1.101

帧 frame

在视频信号空间中由一个亮度样点矩阵(Y)和两个可能存在的色度样点矩阵(Cb 和 Cr)构成。

在音频信号空间中，作为音频处理的基本数据块。在 F_s 采样频率下，512 个样本构成一帧，在 $F_s/2$ 采样频率下，256 个样本构成一帧。

3.1.102

帧宏块 frame macroblock

一个编码帧中的所有宏块均为帧宏块。

3.1.103

帧间编码 inter coding

使用帧间预测对块、宏块、条带或图像进行编码。

3.1.104

帧间预测 inter prediction

利用已解码的参考图像得到当前样点的预测值的过程。

3.1.105

帧内编码 intra coding

使用帧内预测对块、宏块、条带或图像进行编码。

3.1.106

帧内解码图像 intra decoded picture

I 图像

只使用帧内预测解码的图像。

3.1.107

帧内预测 intra prediction

利用同一图像中已解码的样点得到当前样点的预测值的过程。

3.1.108

字节 byte

连续的 8 bit，读写时左边第一位为最高位，右边第一位为最低位。表示为比特序列时，字节的最高有效位为第一位。

3.1.109

字节对齐 byte-aligned

从比特流的第一个比特开始的 8 的倍数的位置为字节对齐的位置。比特或字节或语法元素为字节对齐的，指它出现在比特流中字节对齐的位置上。

3.1.110

字节流 byte stream

NAL 单元流的封装，包含起始码前缀和附录 B 定义的 NAL 单元。

3.1.111

自适应码书 adaptive codebook

通过长时预测滤波器状态得到的码书，由每个子帧自适应的激励矢量构成。

3.1.112

直流偏置 DC-offset

音频信号的直流分量。

3.2 缩略语

ACELP Algebraic Code Excited Linear Prediction 代数码书激励线性预测

BWE Bandwidth Extension 带宽扩展

CABAC Context Adaptive Binary Arithmetic Coding 上下文自适应二进制算术编码

CBR Constant Bit Rate 恒定比特率

CPB Coded Picture Buffer 编码图像缓存区

CRC Cyclic Redundancy Code 循环冗余校验码

DCT Discrete Cosine Transform 离散余弦变换

DFT Discrete Fourier Transform 离散傅立叶变换

DPB Decoded Picture Buffer 解码图像缓存区

FFT Fast Fourier Transform 快速傅立叶变换

FIR Finite Impulse Response 有限冲击响应

HRD Hypothetical Reference Decoder 假设参考解码器

IDCT Inverse Discrete Cosine Transform 离散余弦逆变换

IDFT Inverse Discrete Fourier Transform 离散傅立叶逆变换

IDR Instantaneous Decoding Refresh 即时解码刷新

IFFT Inverse Fast Fourier Transform 快速傅立叶逆变换

ISF Immittance Spectral Frequency 电导谱频率

ISP Immittance Spectral Pair 电导谱对

LP Linear Prediction 线性预测

LPC Linear Predictive Coding 线性预测编码

LSB Least Significant Bit 最低有效位

LTP Long Term Predictor 长时预测

MA Moving Average 滑动平均

MB Macroblock 宏块

MFCC Mel-Frequency Cepstral Coefficients 美尔频率倒谱系数

MSB Most Significant Bit 最高有效位
MSVQ Multi-Stage Vector Quantization 多级矢量量化
NAL Network Abstraction Layer 网络抽象层
PCM Pulse Code Modulation 脉冲编码调制
RBSP Raw Byte Sequence Payload 原始字节序列负载
ROI Region Of Interest 感兴趣区域
SEI Supplement Enhancement Information 补充增强信息
SNR Signal Noise Ratio 信噪比
SODB String Of Data Bits 数据比特串
SVC Scalable Video Coding 可伸缩性视频编码
TAC Transform Audio Coding 变换域音频编码
TVC Transform Vector Coding 变换域矢量编码
VAD Voice Activity Detection 语音活动检测
VBR Variable Bit Rate 可变比特率
VCL Video Coding Layer 视频编码层
VLC Variable Length Coding 可变长度编码
VQ Vector Quantization 矢量量化
VUI Video Usability Information 视频可用性信息

4 约定

4.1 算术运算符

算术运算符定义见表1。

表1 算术运算符定义

编 号	符 号	说 明
1	+	加法运算
2	−	减法运算(二元运算符)或取反(一元前缀运算符)
3	×	乘法运算
4	$\otimes$	卷积运算
5	x^y	指数运算,表示 x 的 y 次幂。在不是表示指数的情况下也可表示上标
6	/	除法运算,不做截断或四舍五入
7	÷	除法运算,不做截断或四舍五入
8	$\frac{x}{y}$	除法运算,不做截断或四舍五入
9	$\sum_{i=x}^{y} f(i)$	自变量 i 取由 x 到 y(含 y)的所有整数值时,函数 $f(i)$ 的累加和
10	$x\%y$	模运算,x 除以 y 的余数,其中 x 与 y 都是正整数

在没有以插入括号来明确指定运算优先次序的情况下,遵守如下规则:

——乘法和除法运算先于加法和减法运算;

——乘法和除法运算从左到右进行;

——加法和减法运算从左到右进行。

4.2 逻辑运算符

逻辑运算符定义见表2。

表 2　逻辑运算符定义

编　号	符　号	说　　明
1	&&	逻辑“与”运算
2	||	逻辑“或”运算
3	!	逻辑“非”运算
4	x? y:z	如果 x 为真或非 0 值，则取值为 y；否则取值为 z

4.3　关系运算符

关系运算符定义见表 3。

表 3　关系运算符定义

编　号	符　号	说　　明
1	>	大于
2	>=	大于或等于
3	<	小于
4	<=	小于或等于
5	==	等于
6	!=	不等于

4.4　位运算符

位运算符定义见表 4。

表 4　位运算符定义

编　号	符　号	说　　明
1	&	按位“与”运算。对整数进行运算时，以整数的二进制补码形式进行操作。如果两个二进制运算数中一个位数小于另外一个，则较短的运算数高位加 0 补齐
2	|	按位“或”运算。对整数进行运算时，以整数的二进制补码形式进行操作。如果两个二进制运算数中一个位数小于另外一个，则较短的运算数高位加 0 补齐
3	$x \gg y$	将 x 以 2 的补码整数表示的形式向右移 y 位。仅当 y 取非负数时定义此运算。右移运算移入 MSB 的位应该等于移位运算前 x 的 MSB 的值
4	$x \ll y$	将 y 以 2 的补码整数表示的形式向左移 y 位。仅当 y 取非负数时定义此运算。左移运算移入 LSB 的位值为 0

4.5　赋值运算符

赋值运算定义见表 5。

表 5　赋值运算定义

编　号	符　号	说　　明
1	=	赋值运算符
2	++	递增，例如 x++ 相当于 $x=x+1$；当用于数组下标时，在自加运算前先求变量值
3	--	递减，例如 x-- 相当于 $x=x-1$；当用于数组下标时，在自减运算前先求变量值
4	+=	自加指定值，例如 x+=3 相当于 $x=x+3$，x+=(−3)相当于 $x=x+(-3)$
5	-=	自减指定值，例如 x-=3 相当于 $x=x-3$，x-=(−3)相当于 $x=x-(-3)$

4.6　数学函数

数学函数计算公式如下：

$$\mathrm{Abs}(x)=\begin{cases}x, x>=0\\ -x, x<0\end{cases}$$

$\mathrm{Ceil}(x)$ 取不小于 x 的最小整数

$\mathrm{Clip1Y}(x)=\mathrm{Clip3}(0,(1\ll \mathrm{BitDepthY})-1,x)$

$\mathrm{Clip1C}(x)=\mathrm{Clip3}(0,(1\ll \mathrm{BitDepthC})-1,x)$

$$\mathrm{Clip3}(x,y,z)=\begin{cases}x, & z<x\\ y, & z<y\\ z, & \text{其他}\end{cases}$$

$\mathrm{Cos}(x)$ 表示 x 的余弦函数

$C_M^N=\dfrac{M!}{N!\ (M-N)!}$ 表示从 M 个数中取出 N 个数的组合数

$\mathrm{Exp}(x)$ 表示 e 的 x 次幂

$\mathrm{Floor}(x)$ 取不大于 x 的最大整数

$\mathrm{Ln}(x)$ 取以 e 为底的 x 的对数

$\mathrm{Log}_{10}(x)$ 取以 10 为底的 x 的对数

$\mathrm{Median}(x,y,z)=x+y+z-\mathrm{Min}(x,\mathrm{Min}(y,z))-\mathrm{Max}(x,\mathrm{Max}(y,z))$

$$\mathrm{Min}(x,y)=\begin{cases}x, & x<=y\\ y, & x>y\end{cases}$$

$$\mathrm{Max}(x,y)=\begin{cases}x, & x>=y\\ y, & x<y\end{cases}$$

$\mathrm{Round}(x)=\mathrm{Sign}(x)\times\mathrm{Floor}(\mathrm{Abs}(x)+0.5)$

$$\mathrm{Sign}(x)=\begin{cases}1, & x>=0\\ -1, & x<0\end{cases}$$

$\mathrm{Sin}(x)$ 表示 x 的正弦函数

4.7 语法元素、变量和表

语法元素在比特流中以粗体字出现。当表格或正文中用到某个语法元素的值时，则以常规字体出现。每个语法元素均表示为名称(所有字母小写，以下划线连接)，和一到两个代表其编码表示方式的描述符。解码过程根据语法元素以及之前已解码的语法元素的取值进行解码。

某些情况下语法表可能使用根据语法元素值导出的其他变量的值。这些变量出现在语法表或正文中，以小写和大写混合的形式命名，并且名称中不含下划线。以大写字母开头的变量是根据当前语法结构和所有相关语法结构的解码导出的。

在某些情况下，语法元素值或变量值的识记名称与其数值等同。有时，识记名称与其值无关。二者的关联在正文中做出规定。识记名称由一组或多组字母由下划线连接而成。每组字母均以大写字母开头，可包括多个大写字母。

函数用名称来描述，函数名由语法元素名称和左右圆括号中的零个或多个以逗号(若有多个变量时)分隔的变量名称(用于定义)或值(用于使用)构成。

一维的阵列称为数组，二维的阵列称为矩阵。阵列可以是语法元素，也可以是变量。下标或方括号可用来表示一个阵列的索引。对于一个矩阵，第一个下标为行(垂直)索引，第二个下标为列(水平)索引。使用方括号表示时，索引的顺序则正好相反。比如，一个矩阵 S 中的水平位置 x 和垂直位置 y 上的元素可表示为 $S[x,y]$ 或 S_{yx}。

单引号之间的一串比特值为二进制符号。例如，'10000100'表示一个第一位和倒数第三位等于 1 的 8 比特串。

十六进制符号，以前缀"0x"表示，当所表示的比特位数为 4 的整数倍时可替代二进制符号使用。例

如,“0x84”表示一个第一位和倒数第三位等于 1 的 8 比特串。

不使用单引号括起来的或不带前缀“0x”的数值为十进制值。

条件语句中等于 0 的值代表假(FALSE)的情况。用其他非零值代表真(TRUE)。

4.8 逻辑运算符的文字描述

在正文中,含有逻辑运算符的下列伪码语句:

```
if(条件 0)
  语句 0
else if(条件 1)
  语句 1
……
else / * 解释其他情况的注释 * /
  语句 n
```

可描述如下:

——如果条件 0,则语句 0

——否则,如果条件 1,语句 1

——……

——否则(说明性文字,表示其他情况),语句 n

正文中的每个“如果……否则,如果……否则……”语句都是由“……如下……”或“……应用下列规则”引导的,后面紧跟“如果……”。最后一个“如果……否则,如果……否则……”语句的条件一般是“否则,……”。交替出现的“如果……否则,如果……否则……”语句可以通过将“……如下……”或“……应用下列规则”和最后的“否则,……”配对加以识别。

正文中,一个以下列伪码描述的逻辑运算语句:

```
if(条件 0a && 条件 0b)
  语句 0
else if(条件 1a || 条件 1b)
  语句 1
……
else
  语句 n
```

可描述如下:

——如果下列所有条件为真,声明 0

——条件 0a

——条件 0b

——否则,如果下列任何一个条件为真,声明 1

——条件 1a

——条件 1b

——……

——否则,声明 n

正文中,一个以下列伪码描述的逻辑运算语句:

```
if(条件 0)
  声明 0
if(条件 1)
  声明 1
```

可描述如下：

——当条件 0 时，声明 0

——当条件 1 时，声明 1

4.9 过程

过程用于描述语法元素的解码。所有属于当前语法结构的语法元素和大写的变量，以及相关的语法结构，在过程的规范和调用中都是可用的。过程的规范中可能还含有明确指定为输入的小写的变量。每个规范均明确地规定了输出。输出可以是大写的变量，也可以是小写的变量。

在过程的规范中，一个特定宏块可用一个值与其宏块索引相等的变量名指代。

5 视频部分

5.1 编码比特流和输出数据的格式

5.1.1 比特流格式

本条规定 NAL 单元流和字节流之间的关系，二者均称为比特流。

NAL 单元流格式由一系列称为 NAL 单元的语法结构组成，按照解码顺序排序。NAL 单元流中 NAL 单元的解码顺序和内容是受约束的。

字节流可以用 NAL 单元流构造，通过将 NAL 单元按照解码顺序排列，并且为每个 NAL 单元添加一个起始码前缀和若干零值字节形成一个字节流。NAL 单元流格式可以通过在字节流中搜索唯一的起始码前缀，从字节流格式中提取出来。除字节流格式以外，构造 NAL 单元的其他方法，本标准不做规定。字节流格式在附录 B 中规定。

5.1.2 图像格式

本条规定由比特流确定的源与已解码帧和场之间的关系。

比特流所表示的视频源是一系列按解码顺序排列的帧或场或帧场的组合。

每个源或已解码图像(帧或场)都是由一个或多个视频样点阵列组成的：

——仅亮度(Y)(单色)的阵列；

——亮度和两个色度(YCbCr)的阵列；

——绿、蓝和红(GBR，也称为 RGB)的阵列；

——表示其他未定义的单色或三基色样点(例如 YZX，也称为 XYZ)的阵列。

为了便于标记和命名，本标准不考虑实际使用的颜色表示方法，与这些阵列相关的变量和词语均指亮度和色度，亮度阵列用 Y 表示，两个色度阵列分别用 Cb 和 Cr 表示。

本标准支持的色彩格式有 4∶0∶0(单色)，4∶2∶0 和 4∶2∶2，见表 6。

变量 SubWidthC 和 SubHeightC 在表 6 中规定，它们取决于通过 chroma_format_idc 表示的色度采样结构。

表 6 由 chroma_format_idc 决定的 SubWidthC 和 SubHeightC 的值

chroma_format_idc	色彩格式	SubWidthC	SubHeightC
0	4∶0∶0(单色)	—	—
1	4∶2∶0	2	2
2	4∶2∶2	2	1
注："—"表示 SubWidthC 或 SubHeightC 的值未定义。			

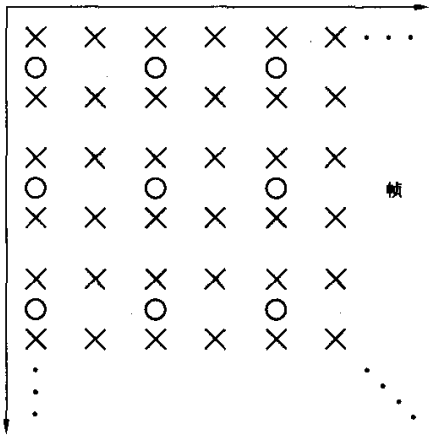
在单色采样中只有一个样点阵列，名义上当作亮度阵列。亮度阵列的高度和宽度为 16 的整数倍。在 4∶2∶0 格式下，两个色度阵列的高度和宽度均为亮度阵列的一半；色度阵列的高度和宽度为 8 的整数倍。在 4∶2∶2 格式下，两个色度阵列的高度等于亮度阵列的高度，宽度为亮度阵列的一半；色度阵列的宽度为 8 的整数倍，高度为 16 的整数倍。如果作为两场独立编码，整帧图像的亮度阵列的高度为 32 的整数倍，

4∶2∶0格式下色度矩阵的高度为16的整数倍，而4∶2∶2格式下色度矩阵的高度为32的整数倍。

除非特别说明，亮度和色度（当出现时）阵列的语法顺序为：当三个分量的数据都出现时，首先是亮度阵列的数据，然后是Cb阵列数据，最后是Cr阵列数据。

对于使用同一个序列参数集编码的场和帧，它们宽度相同，场的高度是帧的一半。

视频序列中用来表示每个亮度或色度样点的比特位数至少为8，表示亮度阵列样点的比特位数和表示色度阵列样点的比特位数可能不相同。在4∶2∶0格式下，一帧中亮度和色度样点的垂直和水平相对位置如图1所示。

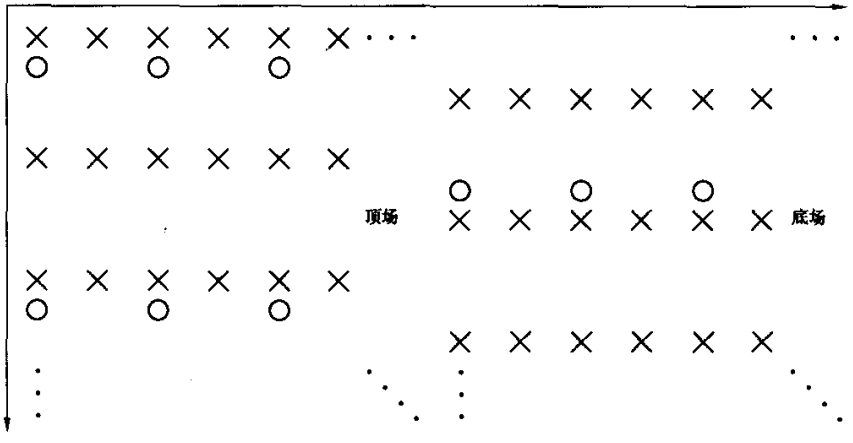


×——亮度样点的位置；○——色度样点的位置。

图1　帧图像中4∶2∶0亮度和色度样点垂直和水平位置

一幅编码图像可以是一个编码帧，也可以是一个编码场。符合本标准的编码视频序列可能含有编码帧和编码场的组合。场编码图像分为两类：顶场和底场。帧中序号为0（最上行）、2、4……的行为顶场行，帧中序号为1、3、5……的行为底场行。顶场由帧中所有的顶场行组成，底场由帧中所有底场行组成。也就是说，一帧图像也可以由一个顶场和一个底场组成。如果以场的形式显示，应先显示顶场，后显示底场。当两场同时显示，或者联合起来被用做参考帧时，它们是以帧的形式交织在一起的。如果一个解码帧的顶场或底场被用做参考场，这时仅使用该解码帧中的顶场行或底场行。

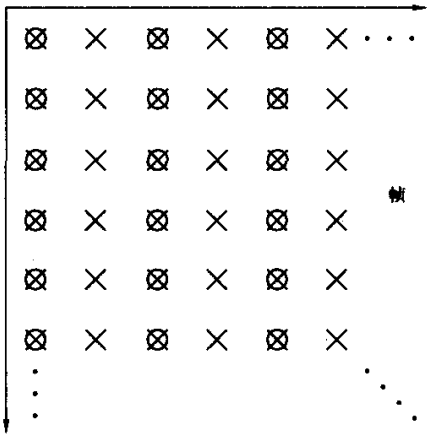
在4∶2∶0格式下，顶场和底场中亮度和色度样点的垂直和水平相对位置如图2所示。顶场中色度样点的垂直样点位置相对于场的采样格点上移四分之一个亮度样点的高度。底场中色度样点的垂直位置相对于场的采样格点下移四分之一个亮度样点的高度。



×——亮度样点的位置；○——色度样点的位置。

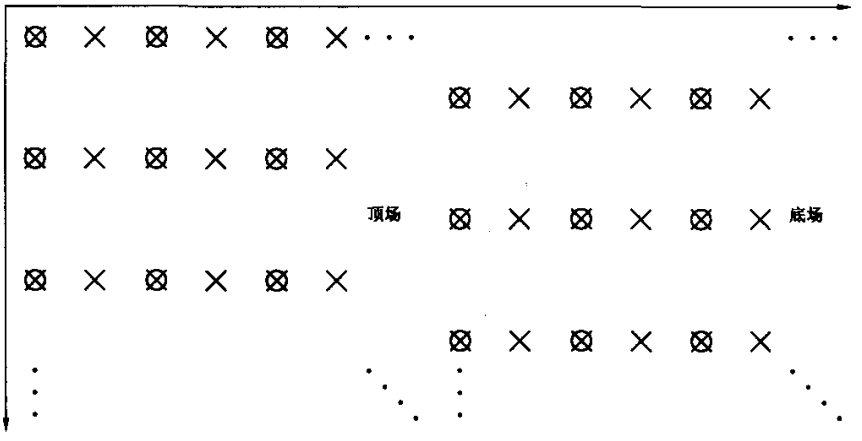
图2　顶场和底场中4∶2∶0亮度和色度样点的垂直和水平位置

在4∶2∶2格式下，色度样点和对应的亮度样点处于同一位置上，帧和场中的样点位置分别如图3和图4所示。



×——亮度样点的位置；○——色度样点的位置。

图3 帧图像中4∶2∶2亮度和色度样点的垂直和水平位置



×——亮度样点的位置；○——色度样点的位置。

图4 顶场和底场中4∶2∶2亮度和色度样点的垂直和水平位置

一帧图像中左上角亮度样点的位置坐标(x,y)为(0,0)，样点每右移一列，x的取值增加1，样点每下移一行，y的取值增加1。

样点是以宏块为单元进行处理的。每个宏块中的样点阵列的高和宽均为16个样点。变量MbWidthC和MbHeightC分别规定了每个宏块中色度阵列的宽度和高度，其推导过程如下：

——如果chroma_format_idc等于0，则MbWidthC和MbHeightC均为0(单色视频没有色度阵列)；

——否则，MbWidthC和MbHeightC按下式得到：

MbWidthC=16/SubWidthC

MbHeightC=16/SubHeightC

5.1.3 图像和条带的空间分割

5.1.3.1 条带的划分

本条规定一幅图像如何分割为条带和宏块。图像被划分为条带，条带由一系列的宏块组成。

每个宏块均包含一个 16×16 的亮度阵列，当色彩格式不是单色时，还包含两个相应的色度阵列。每个宏块代表图像中的一个空间矩形区域。如图 5 所示，一幅图像被分为两个条带。

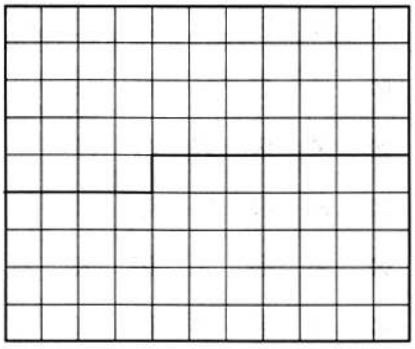


图 5　分割为两个条带的 11×9 个宏块的图像

当 roi_flag 等于 1 时，一帧图像被分为一个背景区域和 num_roi 个感兴趣区域(ROI)。每个 ROI 是一个由 top_left 和 bottom_right 所规定的矩形区域，不同 ROI 不应重叠。每个条带由同一区域内的一系列宏块组成。例如，如图 6 所示，一幅图像有 2 个 ROI，其中一个由 B1、B2、B3 和 B4 条带组成，另一个由 B5 条带组成，图像的其他区域(背景区域)被分为 A、C、D、E、F、G 及 H 条带。ROI 的大小必须为 16×32 样点阵列的整倍数，同时 ROI 中左上角亮度样点的位置坐标(x_0, y_0)和右下角的位置坐标(x_1, y_1)应满足下述条件：

$$\begin{cases} x_0 = 16 \times j \\ y_0 = 32 \times k \end{cases}, \quad \begin{cases} x_1 = 16 \times m - 1 \\ y_1 = 32 \times n - 1 \end{cases}$$

其中 j、k、m 及 n 为整数。

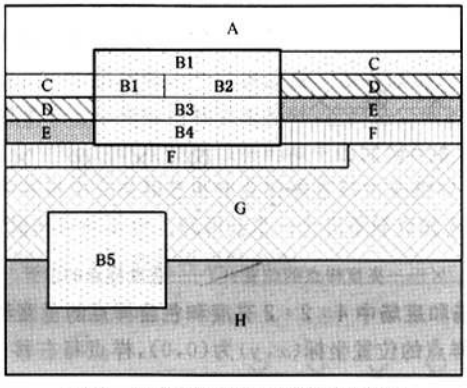


图 6　包含 ROI 的图像条带分割

当 roi_flag 等于 0 时，一个条带由图像内按光栅扫描顺序排列的连续宏块组成。当 roi_flag 等于 1 时，一个条带由所在区域内按光栅扫描顺序排列的连续宏块组成。

一个视频序列中的所有图像条带的解码顺序应与其编码顺序及在码流中的传送顺序相同。

5.1.3.2　宏块的划分

宏块左上角的点不应超出图像边界。在比特流中，当图像以编码场的形式出现时，任一宏块的样点应来自同一场。

宏块的划分如图 7 所示，这种划分用于运动补偿。图 7 中矩形里的数字表示宏块划分后运动矢量和参考索引在编码视频序列中的顺序。

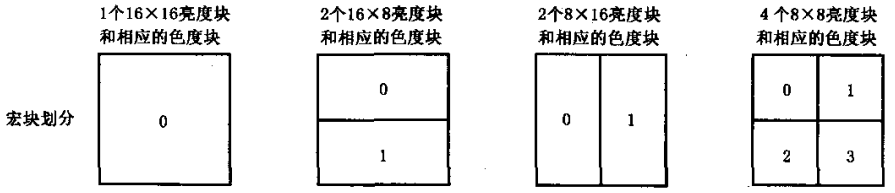


图 7　宏块的划分

在 4∶2∶0 格式下，一个宏块包括 4 个 8×8 亮度块(Y)和 2 个 8×8 色度块(1 个 Cb，1 个 Cr)。如图 8 所示，图中数字为宏块中 8×8 块在编码视频序列中的顺序。

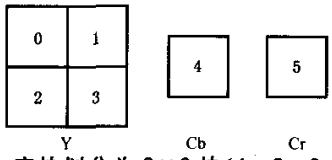


图 8　宏块划分为 8×8 块(4∶2∶0 格式)

在 4∶2∶2 格式下，一个宏块包括 4 个 8×8 亮度块(Y)和 4 个 8×8 色度块(2 个 Cb，2 个 Cr)。如图 9 所示，图中数字为宏块中 8×8 块在编码视频序列中的顺序。

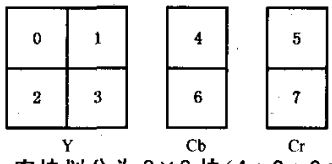


图 9　宏块划分为 8×8 块(4∶2∶2 格式)

一个 8×8 亮度块可能会被划分为 4 个 4×4 子块。如图 10 所示，图中数字为 4×4 块在编码视频序列中的顺序。

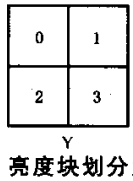


图 10　8×8 亮度块划分为 4×4 子块

5.1.3.3　相邻块可用性的推导过程

一个块 E 和它的相邻块 A、B、C 和 D 之间的空间位置如图 11 所示。如果 E 属于一个场宏块，A、B、C 和 D 为同一场图像中与 E 相邻的块。E 的大小可以是 16×16、16×8、8×16、8×8 或者 4×4。块 A 是块 E 的左边块，块 B 是块 E 的上边块。设块 E 左上角样点在图像中的坐标是(x_0, y_0)，右上角样点在图像中的坐标是(x_1, y_1)，块 X(X 为 A、B、C 或 D)为表 7 中列出的样点所属的块。表 7 中坐标均为样点在帧图像中的位置坐标。

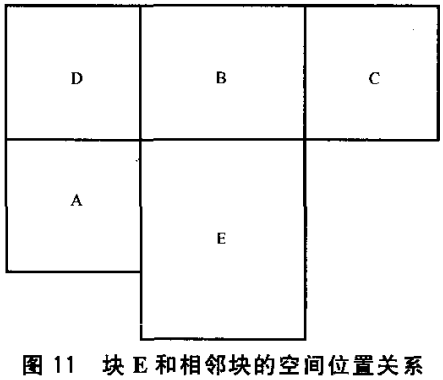


图 11　块 E 和相邻块的空间位置关系

表 7　块 A、B、C 和 D 的位置

field_pic_flag	块 A 右上角样点位置	块 B 左下角样点位置	块 C 左下角样点位置	块 D 右下角样点位置
1	(x_0-1, y_0)	(x_0, y_0-2)	(x_1+1, y_1-2)	(x_0-1, y_0-2)
0	(x_0-1, y_0)	(x_0, y_0-1)	(x_1+1, y_1-1)	(x_0-1, y_0-1)

如果一相邻块 X(X 为 A、B、C 或 D)在图像内并且该块与当前块 E 属于同一条带，则该相邻块标记为存在；否则该相邻块标记为不存在。

如果一相邻块标记为不存在或者尚未解码，则该块标记为不可用；否则该块标记为可用。如果某样点所在的块标记为不存在或者该样点尚未解码，则该样点标记为不可用；否则该样点标记为可用。

5.2　语法和语义

5.2.1　以表格形式描述语法的方法

语法表格规定了所有允许的比特流的超集。附加的语法限定可能在其他条中直接或间接规定。

注：实际的解码器宜有识别比特流入口点的方法，并且可以分辨和处理不一致的比特流。分辨和处理错误以及类似情形的方法不在本标准中描述。

表 8 给出了描述语法的伪代码例子。规定了当 syntax_element 出现时，从比特流中解析语法元素，并将指针移向比特流中下一个语法元素位置上的过程。

表 8　伪代码例程表

伪代码描述语言	描述符
/* 语句可以是一个关联某一语法类别的语法元素和描述符，或者用于说明语法元素的存在、类型和数值的表达式，下面给出两个例子。 */	
syntax_element	ue(v)
条件语句	
/* 花括号括起来的语句组是复合语句，在功能上视作单个语句。 */	
{	
语句	
语句	
…	
}	
/* "while" 语句测试条件是否为 TRUE，如果为 TRUE，则重复执行循环体，直到条件不为 TRUE。 */	
while(条件)	
语句	
/* "do…while"语句先执行循环体一次，然后测试条件是否为 TRUE，如果为 TRUE，则重复执行循环体，直到条件不为 TRUE。 */	
do	
语句	
while(条件)	

表 8（续）

伪代码描述语言	描述符
/*“if…else”语句首先测试条件，如果为 TRUE，则执行主要语句，否则执行另选语句。如果另选语句不需要执行，结构的“else”部分和相关的另选语句可忽略。*/	
if(条件)	
主要语句	
else	
另选语句	
/*“for”语句首先执行最初语句，然后测试条件，如果条件为 TRUE，则重复执行主要语句和随后语句直到条件不为 TRUE。*/	
for(最初语句；条件；随后语句)	
主要语句	

5.2.2 语法函数和描述符的规范

5.2.2.1 语法函数的规范

以下函数用于语法描述。这些函数假定解码器中存在一个比特流指针，这个指针指向比特流中解码过程要读取的下一个比特的位置。

byte_aligned( )的规定如下：

——如果比特流的当前位置是在字节的边界，即比特流中的下一个比特是字节的第一个比特，那么 byte_aligned( )的返回值为 TRUE；

——否则，byte_aligned( )的返回值为 FALSE。

more_data_in_byte_stream( )，在附录 B 规定的字节流 NAL 单元语法结构中使用，规定如下：

——如果字节流中后续还有更多数据，more_data_in_byte_stream( )的返回值为 TRUE；

——否则，more_data_in_byte_stream( )的返回值为 FALSE。

more_rbsp_data( )的规定如下：

——如果在 rbsp_trailing_bits( )之前的 RBSP 中有更多数据，more_rbsp_data( )的返回值为 TRUE；

——否则，more_rbsp_data( )的返回值为 FALSE。

判断 RBSP 中是否有更多数据的方法由应用规定。

next_bits(n)提供比特流中接下来的 n 个比特，不改变比特流指针。该函数使比特流中的下 n 个比特可见，n 在这里是函数的参数。当用在附录 B 规定的字节流中时，如果剩余的字节流已不足 n 个比特，next_bits(n)返回值为 0。

read_bits(n)从比特流中读取下面的 n 个比特，并且将比特流指针向前移动 n 个比特。当 n 等于 0 时，read_bits(n)的返回值为 0 并且不移动比特流指针。

5.2.2.2 描述符的规范

下述描述符规定了每个语法元素的解析过程。对于某些语法元素，使用通过竖线分开的两个描述符。在这些情况下，左边的描述符在 entropy_coding_mode_flag 等于 0 的时候有效，右边的描述符在 entropy_coding_mode_flag 等于 1 的时候有效：

——ae(v)：上下文自适应二进制算术编码语法元素。该描述符的解析过程在 5.4.3 中规定；

——b(8)：任意形式的 8 比特字节。该描述符的解析过程通过函数 read_bits(8)的返回值来规定；

——ce(v)：可变长度熵编码语法元素。该描述符的解析过程在 5.4.2.3 中规定；

——f(n):n 位比特串(由左至右),左位在先,该描述符的解析过程通过函数 read_bits(n)的返回值来规定;

——i(n):n 位有符号整数。在语法表中,如果 n 是'v',其比特数由其他语法元素值确定。解析过程由函数 read_bits(n)的返回值规定,该返回值用最高有效位在前的 2 的补码表示;

——me(v):映射的指数哥伦布码编码的语法元素,左位在先。解析过程在 5.4.2.2 中定义;

——se(v):有符号整数指数哥伦布码编码的语法元素,左位在先。解析过程在 5.4.2.2 中定义;

——u(n):n 位无符号整数。在语法表中,如果 n 是'v',其比特数由其他语法元素值确定。解析过程由函数 read_bits(n)的返回值规定,该返回值用最高有效位在前的二进制表示;

——ue(v):无符号整数指数哥伦布码编码的语法元素,左位在先。解析过程在 5.4.2.2 中定义。

5.2.3 以表格形式表示的语法

5.2.3.1 NAL 单元语法

NAL 单元语法见表 9。

表 9 NAL 单元语法表

nal_unit(NumBytesInNALunit) {	描述符
forbidden_zero_bit	f(1)
nal_ref_idc	u(1)
nal_unit_type	u(4)
encryption_idc	u(1)
authentication_idc	u(1)
NumBytesInHeader=1	
if(authentication_idc) {	
authentication_data_length_minus2	u(8)
NumBytesInHeader+=1	
}	
NumBytesInPayload=0	
for(i=NumBytesInHeader;i<NumBytesInNALunit;i++) {	
if(i+2<NumBytesInNALunit && next_bits(24)==0x000003) {	
payload_byte[NumBytesInPayload++]	b(8)
payload_byte[NumBytesInPayload++]	b(8)
i+=2	
emulation_prevention_three_byte /* 应等于 0x03 */	f(8)
}	
else	
payload_byte[NumBytesInPayload++]	b(8)
}	
}	

5.2.3.2 RBSP 语法

5.2.3.2.1 序列参数集 RBSP 语法

序列参数集 RBSP 语法见表 10。

表 10　序列参数集 RBSP 语法表

seq_parameter_set_rbsp( ) {	描述符
profile_id	u(8)
level_id	u(8)
seq_parameter_set_id	ue(v)
chroma_format_idc	u(2)
bit_depth_luma_minus8	ue(v)
bit_depth_chroma_minus8	ue(v)
pic_width_in_mbs_minus1	ue(v)
pic_height_in_mbs_minus1	ue(v)
progressive_seq_flag	u(1)
roi_flag	u(1)
svc_flag	u(1)
vui_parameters_present_flag	u(1)
if(vui_parameters_present_flag)	
vui_parameters( )	
rbsp_trailing_bits( )	
}	

5.2.3.2.2　图像参数集 RBSP 语法

图像参数集 RBSP 语法见表 11。

表 11　图像参数集 RBSP 语法表

pic_parameter_set_rbsp( ) {	描述符
pic_parameter_set_id	ue(v)
seq_parameter_set_id	ue(v)
entropy_coding_mode_flag	u(1)
pic_init_qp	u(v)
if(roi_flag) {	
num_roi	ue(v)
if(num_roi>0) {	
non_roi_skip_flag	u(1)
if(! non_roi_skip_flag && svc_flag)	
scalable_non_roi_skip_flag	u(1)
pic_init_qp_for_roi	u(v)
for(i=0;i<num_roi;i++) {	
top_left[i]	ue(v)
bottom_right[i]	ue(v)
}	

表 11（续）

pic_parameter_set_rbsp( ) {	描述符
}	
}	
fixed_pic_qp	u(1)
weighting_pred_flag	u(1)
loop_filter_disable_flag	u(1)
rbsp_trailing_bits( )	
}	

5.2.3.2.3 安全参数集 RBSP 语法

安全参数集 RBSP 语法见表 12。

表 12 安全参数集 RBSP 语法表

sec_parameter_set_rbsp( ) {	描述符
encryption_flag	u(1)
authentication_flag	u(1)
if(encryption_flag)	
encryption_type	u(4)
if(authentication_flag) {	
hash_type	u(4)
hash_hierarchy_flag	u(1)
hash_discard_p_pictures	u(1)
hash_discard_b_pictures	u(1)
hash_discard_extension_data	u(1)
signature_type	u(4)
if(signature_type>0) {	
successive_hash_pictures_minus1	u(8)
signature_data_length_minus1	u(8)
}	
}	
rbsp_trailing_bits( )	
}	

5.2.3.2.4 补充增强信息 RBSP 语法

补充增强信息 RBSP 语法见表 13。

表 13 补充增强信息 RBSP 语法表

sei_rbsp( ) {	描述符
do	
sei_message( )	

表 13（续）

sei_rbsp( ) {	描述符
while(more_rbsp_data( ))	
rbsp_trailing_bits( )	
}	

补充增强信息消息语法见表 14。

表 14 补充增强信息消息语法表

sei_message( ) {	描述符
PayloadType=0	
while(next_bits(8)==0xFF) {	
ff_byte /* 应等于 0xFF */	f(8)
PayloadType+=255	
}	
last_payload_type_byte	u(8)
PayloadType+=last_payload_type_byte	
PayloadSize=0	
while(next_bits(8)==0xFF) {	
ff_byte /* 应等于 0xFF */	f(8)
PayloadSize+=255	
}	
last_payload_size_byte	u(8)
PayloadSize+=last_payload_size_byte	
sei_payload(PayloadType, PayloadSize)	
}	

5.2.3.2.5 序列结尾 RBSP 语法

序列结尾 RBSP 语法见表 15。

表 15 序列结尾 RBSP 语法表

end_of_seq_rbsp( ) {	描述符
}	

5.2.3.2.6 流结尾 RBSP 语法

流结尾 RBSP 语法见表 16。

表 16 流结尾 RBSP 语法表

end_of_stream_rbsp( ) {	描述符
}	

5.2.3.2.7 编码条带 RBSP 语法

编码条带 RBSP 语法见表 17。

表 17　编码条带 RBSP 语法表

slice_layer_rbsp( ) {	描述符
slice_header( )	
slice_data( )	
rbsp_trailing_bits( )	
}	

5.2.3.2.8　RBSP 尾比特语法

RBSP 尾比特语法见表 18。

表 18　RBSP 尾比特语法表

rbsp_trailing_bits( ) {	描述符
rbsp_stop_one_bit /* 应等于 1 */	f(1)
while(! byte_aligned( ))	
rbsp_alignment_zero_bit /* 应等于 0 */	f(1)
}	

5.2.3.2.9　填充数据 RBSP 语法

填充数据 RBSP 语法见表 19。

表 19　填充数据 RBSP 语法表

filler_data_rbsp( ) {	描述符
while(next_bits(8)==0xFF)	
ff_byte /* 应等于 0xFF */	f(8)
rbsp_trailing_bits( )	
}	

5.2.3.3　条带头语法

条带头语法见表 20。

表 20　条带头语法表

slice_header( ) {	描述符
pic_parameter_set_id	ue(v)
frame_num	u(8)
if(nal_unit_type==2 || nal_unit_type==4)	
idr_pic_id	ue(v)
if(! progressive_seq_flag) {	
field_pic_flag	u(1)
if(field_pic_flag)	
bottom_field_flag	u(1)
}	
first_mb_in_slice	ue(v)
slice_type	ue(v)

表 20（续）

slice_header( ) {	描述符
if(! (slice_type==B && ! field_pic_flag) && ! (slice_type==I))	
picture_reference_flag	u(1)
if(! fixed_pic_qp) {	
fixed_slice_qp	u(1)
slice_qp_delta	se(v)
}	
if(slice_type ! =I && weighting_pred_flag) {	
slice_weighting_flag	u(1)
if(slice_weighting_flag) {	
num_of_references_minus1	u(2)
for (i=0;i<num_of_references;i++) {	
luma_scale	u(8)
luma_shift	i(8)
marker_bit	f(1)
chroma_scale	u(8)
chroma_shift	i(8)
marker_bit	f(1)
}	
mb_weighting_flag	u(1)
}	
}	
if(! loop_filter_disable) {	
loop_filter_parameter_flag	u(1)
if(loop_filter_parameter_flag) {	
slice_alpha_c0_offset	se(v)
slice_beta_offset	se(v)
}	
}	
}	

5.2.3.4 条带数据语法

条带数据语法见表 21。

表 21 条带数据语法表

slice_data( ) {	描述符
if(entropy_coding_mode_flag)	
while(! byte_aligned( ))	

表 21（续）

slice_data( ) {	描述符
cabac_alignment_one_bit	f(1)
moreDataFlag=1	
do{	
if(slice_type ! =I) {	
if(! entropy_coding_mode_flag) {	
mb_skip_run	ue(v)
moreDataFlag=more_rbsp_data( )	
}	
else {	
mb_skip_flag	ae(v)
moreDataFlag=! mb_skip_flag	
}	
}	
if(moreDataFlag) {	
if(svc_flag && (nal_unit_type==4 || nal_unit_type==3))	
macroblock_scalable( )	
else	
macroblock( )	
}	
if(! entropy_coding_mode_flag) {	
moreDataFlag=more_rbsp_data( )	
}	
else {	
end_of_slice_flag	ae(v)
moreDataFlag=! end_of_slice_flag	
}	
} while(moreDataFlag)	
}	

5.2.3.5 宏块语法

宏块语法见表 22。

表 22 宏块语法表

macroblock( ) {	描述符
mb_type	ue(v)|ae(v)
if(MbType! =P_Skip && MbType! =B_Skip) {	
if(MbType==B_8×8) {	

表 22（续）

macroblock( ) {	描述符
for(i=0;i<4;i++)	
mb_part_type	u(2)|ae(v)
}	
if(MbType==I_8×8) {	
for(i=0;i<4;i++) {	
pred_mode_flag	u(1)|ae(v)
if(! pred_mode_flag)	
intra_luma_pred_mode	u(2)|ae(v)
}	
if(chroma_format_idc>0)	
intra_chroma_pred_mode	ue(v)|ae(v)
if(chroma_format_idc==2)	
intra_chroma_pred_mode_422	ue(v)|ae(v)
}	
if(MbType==I_4×4) {	
for(i=0;i<16;i++) {	
pred_mode_flag	u(1)|ae(v)
if(! pred_mode_flag)	
intra_luma_pred_mode	u(2)|ae(v)
}	
if(chroma_format_idc>0)	
intra_chroma_pred_mode	ue(v)|ae(v)
if(chroma_format_idc==2)	
intra_chroma_pred_mode_422	ue(v)|ae(v)
}	
if(slice_type==P || (slice_type==B && field_pic_flag) && ! picture_reference_flag) {	
for(i=0;i<MvNum;i++)	
mb_reference_index	u(1)|u(2)|ae(v)
}	
for(i=0;i<MvNum;i++) {	
mv_diff_x	se(v)|ae(v)
mv_diff_y	se(v)|ae(v)
}	
if(mb_weighting_flag)	

表 22（续）

macroblock( ) {	描述符
weighting_prediction	u(1)|ae(v)
if(MbType ! =I_8×8 || entropy_coding_mode_flag)	
coded_block_pattern	me(v)|ae(v)
if(chroma_format_idc==2)	
coded_block_pattern_422	me(v)|ae(v)
if(MbType==I_4×4) {	
for(i=0;i<4;i++)	
if(MbCBP&1≪i)	
coded_block_pattern_4×4	me(v)|ae(v)
}	
if((MbCBP>0 || MbCBP422>0) && ! fixed_slice_qp)	
mb_qp_delta	se(v)|ae(v)
for(i=0;i<CbNum;i++)	
block(i)	
}	
}	

5.2.3.6 可伸缩性视频编码(SVC)增强层宏块语法

SVC 增强层宏块语法见表 23。

表 23 SVC 增强层宏块语法表

macroblock_scalable( ) {	描述符
mb_type	ue(v)|ae(v)
if(mb_type==I_4×4 || mb_type==I_8×8)	
svc_mode_flag	u(1)|ae(v)
if(! svc_mode_flag && MbType ! =P_Skip && MbType! =B_Skip) {	
if(MbType==B_8×8) {	
for(i=0;i<4;i++)	
mb_part_type	u(2)|ae(v)
}	
if(MbType==I_8×8) {	
for(i=0;i<4;i++) {	
pred_mode_flag	u(1)|ae(v)
if(! pred_mode_flag)	
intra_luma_pred_mode	u(2)|ae(v)
}	
if(chroma_format_idc>0)	

表 23（续）

macroblock_scalable( ) {	描述符
intra_chroma_pred_mode	ue(v)|ae(v)
if(chroma_format_idc==2)	
intra_chroma_pred_mode_422	ue(v)|ae(v)
}	
if(MbType==I_4×4) {	
for(i=0;i<16;i++) {	
pred_mode_flag	u(1)|ae(v)
if(! pred_mode_flag)	
intra_luma_pred_mode	u(2)|ae(v)
}	
if(chroma_format_idc>0)	
intra_chroma_pred_mode	ue(v)|ae(v)
if(chroma_format_idc==2)	
intra_chroma_pred_mode_422	ue(v)|ae(v)
}	
if(slice_type==P || (slice_type==B && field_pic_flag) && ! picture_reference_flag) {	
for(i=0;i<MvNum;i++)	
mb_reference_index	u(1)|u(2)|ae(v)
}	
for(i=0;i<MvNum;i++) {	
mv_diff_x	se(v)|ae(v)
mv_diff_y	se(v)|ae(v)
}	
if(mb_weighting_flag)	
weighting_prediction	u(1)|ae(v)
}	
if(MbType ! =P_Skip && MbType ! =B_Skip) {	
if(MbType ! =I_8×8 || entropy_coding_mode_flag)	
coded_block_pattern	me(v)|ae(v)
if(chroma_format_idc==2)	
coded_block_pattern_422	me(v)|ae(v)
if(MbType==I_4×4) {	
for(i=0;i<4;i++)	
if(MbCBP&1≪i)	

表 23（续）

macroblock_scalable( ) {	描述符
coded_block_pattern_4×4	me(v)|ae(v)
}	
if((MbCBP>0 || MbCBP422>0) && ! fixed_slice_qp)	
mb_qp_delta	se(v)|ae(v)
for(i=0;i<CbNum;i++)	
block(i)	
}	
}	

5.2.3.7 块语法

块语法见表 24。

表 24 块语法表

block(i) {	描述符
if(cbp_8×8 & 1≪i) {	
if(Transform8×8Flag) {	
do {	
trans_coefficient	ce(v)|ae(v)
if(trans_coefficient>=59 && ! entropy_coding_mode_flag)	
escape_level_diff	ce(v)
} while (trans_coefficient! ='EOB')	
}	
else {	
for(j=0;j<4;j++) {	
if(cbp_4×4[i] & 1≪j) {	
do {	
trans_coefficient	ce(v)|ae(v)
if(trans_coefficient>=59 && ! entropy_coding_mode_flag)	
escape_level_diff	ce(v)
} while (trans_coefficient ! ='EOB')	
}	
}	
}	
}	
}	

5.2.3.8 监控扩展数据单元语法

5.2.3.8.1 监控扩展数据单元语法通则

监控扩展数据单元语法见表 25。

表 25　监控扩展数据单元语法表

surveillance_extension_rbsp( ) {	描述符
while(next_bits(8) ! =0x80) {	
if(next_bits(8)==0x01)	
roi_extension( )	
else if(next_bits(8)==0x02)	
event_extension( )	
else if(next_bits(8)==0x03)	
alert_extension( )	
else if(next_bits(8)==0x04)	
time_extension( )	
else	
reserved_extension( )	
}	
surveillance_extension_stop_byte	f(8)
}	

5.2.3.8.2　感兴趣区域扩展语法

感兴趣区域扩展语法见表 26。

表 26　感兴趣区域扩展语法表

roi_extension( ) {	描述符
extension_id	u(8)
extension_length	u(8)
position_idc	u(8)
camera_idc	u(16)
region_num	u(4)
reserved_bits	f(4)
for(i=0;i<region_num;i++) {	
region_top_left_mbx[i]	u(8)
region_top_left_mby[i]	u(8)
region_width_in_mbs_minus1[i]	u(8)
region_height_in_mbs_minus1[i]	u(8)
}	
}	

5.2.3.8.3　监控事件扩展语法

监控事件扩展语法见表 27。

表 27　监控事件扩展语法表

event_extension( ) {	描述符
extension_id	u(8)
extension_length	u(8)
position_idc	u(8)
camera_idc	u(16)
region_num	u(4)
reserved_bits	f(4)
for(i=0;i<=region_num;i++) {	
event_num [i]	u(8)
for(j=0;j<event_num[i];j++) {	
region_event_id [i, j]	u(8)
}	
}	
}	

5.2.3.8.4　监控报警扩展语法

监控报警扩展语法见表 28。

表 28　监控报警扩展语法表

alert_extension( ) {	描述符
extension_id	u(8)
extension_length	u(8)
position_idc	u(8)
camera_idc	u(16)
alert_num	u(6)
reserved_bits	f(2)
for(i=0;i<alert_num;i++) {	
alert_region_id [i]	u(4)
reserved_bits	f(4)
alert_event_id [i]	u(8)
frame_num [i]	u(8)
}	
}	

5.2.3.8.5　绝对时间信息扩展语法

绝对时间信息扩展语法见表 29。

表 29　绝对时间信息扩展语法表

time_extension( ) {	描述符
extension_id	u(8)
extension_length	u(8)
position_idc	u(8)
camera_idc	u(16)
hour_bits	u(5)
minute_bits	u(6)
second_bits	u(6)
second_fraction_bits	u(14)
ref_date_flag	u(1)
if(ref_date_flag) {	
year_minus2000_bits	u(7)
month_bits	u(4)
day_bits	u(5)
}	
frame_num	u(8)
}	

5.2.4　语义

5.2.4.1　概述

本条规定了与语法结构和语法结构中的语法元素相关的语义。当一个语法元素的语义用一个或一组表格表示时，表格中未指定的任何值都不应出现在比特流中，除非在本标准中另外规定。

5.2.4.2　NAL 单元语义

NumBytesInNALunit 指示 NAL 单元的长度，单位为字节。一个 NAL 单元由单元头部分和单元负载部分组成，其中单元负载部分包含一个 RBSP 语法结构及可能存在的认证数据负载，同时还可能包含一些 emulation_prevention_three_byte。在 NAL 单元解码时需要用到 NumBytesInNALunit。为了能够推导出 NumBytesInNALunit，需要对 NAL 单元的边界进行划分。附录 B 规定了一种用于字节流格式的划分方法。其他划分方法可能会在本标准之外给出。

forbidden_zero_bit 在正常的情况下，forbidden_zero_bit 等于 0。

注 1：如果网络传输单元或接收单元发现 NAL 单元包含错误比特，可以将 forbidden_zero_bit 置为 1。

nal_ref_idc 不等于 0 时，表示 NAL 单元的内容包含一个序列参数集，或一个图像参数集，或一个安全参数集，或一个参考图像的条带。当一个编码图像的一个条带 NAL 单元的 nal_ref_idc 等于 0，该编码图像的所有条带 NAL 单元的 nal_ref_idc 都应等于 0。

nal_unit_type 指示 NAL 单元中的 RBSP 数据结构的类型，见表 30。VCL NAL 单元是指那些 nal_unit_type 值等于 1,2,3 或者 4 的 NAL 单元。所有其他的 NAL 单元都称为非 VCL NAL 单元。

注 2：VCL 的规定是为了有效的表示视频数据的内容。NAL 的规定则是为了数据的格式化，并提供头信息以便于存储或在多种通信信道上传输。每个 NAL 单元都包含整数字节。NAL 单元规定了一种既适用于面向分组系统又适用于比特流系统的通用格式。

在不影响 nal_unit_type 不等于 5 或 12 的 NAL 单元的解码过程和不影响本标准一致性的前提下，nal_unit_type 等于 5 和 12 的 NAL 单元可以被解码器丢弃。

注 3：本标准不规定 nal_unit_type 为保留值 NAL 单元的解码过程。解码器可以忽略(从比特流中去除并丢弃)所有 nal_unit_type 为保留值的 NAL 单元的内容。

当一个编码图像条带 NAL 单元的 nal_unit_type 值等于 2 时，编码同一图像的其他所有条带 NAL 单元的 nal_unit_type 值都应相同，与之对应的 SVC 增强层编码图像的所有条带 NAL 单元的 nal_unit_type 值都应等于 4。这样的图像称作 IDR 图像。

NAL 单元类型见表 30。

表 30 NAL 单元类型表

nal_unit_type	NAL 单元中 RBSP 语法结构的内容
0	保留
1	非 IDR 图像的编码条带 slice_layer_rbsp( )
2	IDR 图像的编码条带 slice_layer_rbsp( )
3	非 IDR 图像的 SVC 增强层编码条带 slice_layer_rbsp( )
4	IDR 图像的 SVC 增强层编码条带 slice_layer_rbsp( )
5	监控扩展数据单元 surveillance_extension_rbsp( )
6	补充增强信息 sei_rbsp( )
7	序列参数集 seq_parameter_set_rbsp( )
8	图像参数集 pic_parameter_set_rbsp( )
9	安全参数集 sec_parameter_set_rbsp( )
10	序列结尾 end_of_seq_rbsp( )
11	流结尾 end_of_stream_rbsp( )
12	填充数据 filler_data_rbsp( )
13	本标准第 6 章使用
14	保留
15	保留

encryption_idc 指示 NAL 单元中的 RBSP 是否加密。encryption_idc 等于 0 表示该 NAL 单元中的 RBSP 没有加密，encryption_idc 等于 1 表示该 NAL 单元中的 RBSP 以安全参数集中指定的加密方法加密。

authentication_idc 指示 NAL 单元是否包含认证数据。authentication_idc 等于 0 表示该 NAL 单元负载不包含认证数据，authentication_idc 等于 1 表示该 NAL 单元负载中的 RBSP 之后包含认证数据负载。authentication_idc 等于 1 的 NAL 单元应为一个图像的最后一个条带 NAL 单元。

authentication_data_length_minus2 加 2 表示认证数据的长度，以字节为单位。authentication_data_length_minus2 的取值应为 0～255(包括 0 和 255)。如果 signature_type 等于 0，认证数据的长度等于 hash_type 对应的摘要数据长度加 1；如果 signature_type 大于 0，认证数据的长度等于签名数据的长度加 1。

如果authentication_idc 等于 1，Authentication_DataLength 由下式得到：

AuthenticationDataLength = authentication_data_length_minus2+2

如果 authentication_idc 等于 0，AuthenticationDataLength 也等于 0。

payload_byte[i]为一个 NAL 单元负载的第 i 个字节，可能等于 rbsp_byte[j]或者 authentication_data[k]。一个 NAL 单元负载定义为一个字节的有序序列，包括一个 RBSP(如果 encryption_idc 等于 1，则为 RBSP 加密后生成的字节序列)，如果 authentication_idc 等于 1，RBSP 之后紧随着一个认证数据负载。

authentication_data[k] 为一个认证数据负载的第 k 个字节。一个认证数据负载定义为一个有序的字节序列，包括有效的认证数据和一个 authentication_stop_byte。

对 VCL NAL 单元，

——如果 signature_type 等于 0 并且 hash_hierarchy_flag 等于 0，认证数据为 5.2.4.4.3 中所述的初次摘要数据；

——如果 signature_type 等于 0 并且 hash_hierarchy_flag 等于 1，认证数据为 5.2.4.4.3 中所述的二次摘要数据；

——如果 signature_type 等于 1 并且 hash_hierarchy_flag 等于 0，认证数据为对 5.2.4.4.3 中所述的初次摘要数据进行签名产生的签名数据；

——如果 signature_type 等于 1 并且 hash_hierarchy_flag 等于 1，认证数据为对 5.2.4.4.3 中所述的二次摘要数据进行签名产生的签名数据。

对非 VCL NAL 单元，如序列参数集、图像参数集和监控扩展数据单元，

——如果 signature_type 等于 0，认证数据为 5.2.4.4.3 中所述的摘要数据；

——如果 signature_type 等于 1，认证数据为 5.2.4.4.3 中所述的签名数据。

authentication_stop_byte 为认证数据负载的最后一个字节(k 等于 Authentication_DataLength 减 1)，应等于 0x80。

rbsp_byte[j] 为一个 RBSP 的第 j 个字节。如果 encryption_idc 等于 1，rbsp_byte[j]为 RBSP 加密后的字节序列的第 j 个字节，RBSP 需要经过解密过程得到，解密过程不在本标准中规定。

一个 RBSP 定义为一个字节的有序序列，包含一个 SODB，如下所述：

——如果 SODB 为空(长度为 0 比特)，RBSP 也为空；

——否则 RBSP 包括如下 SODB：

- RBSP 的第一个字节包括(最高位在前)8 比特的 SODB；RBSP 的下一个字节应包括接下来的 8 比特的 SODB，依此类推，直到剩下的 SODB 少于 8 比特；
- rbsp_trailing_bits( )用于 SODB 之后，如下：
  最后 RBSP 字节中前面的(从最高位算起)比特包括 SODB 的剩下的比特(如果有的话)；下一个比特为单个 rbsp_stop_one_bit，其值为 1，并且当 rbsp_stop_one_bit 不是一个字节对齐的字节的最后一个比特时，一个或更多的 rbsp_alignment_zero_bit 就会出现以形成一个字节对齐。

具有这些 RBSP 属性的语法结构在语法表中用“_rbsp”后缀表示。这些结构在 NAL 单元中作为 rbsp_byte[j]数据字节的内容携带。NAL 单元到 RBSP 语法结构的关联见表 30。

注 4：当 RBSP 的边界已知时，解码器能够通过将 RBSP 字节连接成比特串并丢弃最后(最右边的)一个等于 1 的比特 rbsp_stop_one_bit 以及随后的任何等于 0 的比特从 RBSP 中解析出 SODB。解码过程所必须的数据包含

在 RBSP 的 SODB 部分。

注 5：监控扩展数据单元也满足 RBSP 的语法结构，其中的 surveillance_extension_stop_byte 相当于 rbsp_trailing_bits( )。

emulation_prevention_three_byte 等于 0x03。当一个 emulation_prevention_three_byte 出现在 NAL 单元中时，应被解码过程丢弃。NAL 单元的最后一个字节不能等于 0x00。在 NAL 单元中，下面的三字节序列不应在任何字节对齐的位置出现：

0x000000

0x000001

0x000002

在一个 NAL 单元中，除了下列序列，任何以 0x000003 开头的四字节的序列都不能出现在任何字节对齐的位置：

0x00000300

0x00000301

0x00000302

0x00000303

注 6：当 nal_unit_type 等于 0 时，在设计编码器时要避免上面列出的三字节和四字节形式出现在 NAL 单元语法结构的开头，以免 emulation_prevention_three_byte 语法元素成为 NAL 单元的第三字节。

5.2.4.3 NAL 单元的封装及约束

5.2.4.3.1 将 RBSP 和认证数据封装为 NAL 单元

使用 emulation_prevention_three_byte 将一个 RBSP 和认证数据(如果存在)封装到一个 NAL 单元中的目的：

——允许 NAL 单元中出现任意的 SODB，但要防止在 NAL 单元中出现伪起始码；

——通过在 NAL 单元的结尾查找 *authentication_stop_byte*，以便能够识别 NAL 单元中的认证数据；

——通过在 RBSP 的结尾查找其比特 rbsp_stop_one_bit，以便能够识别 NAL 单元中 SODB 的结尾。

编码器通过下列步骤能够从一个 RBSP 中产生一个 NAL 单元：

从 RBSP 及其后的认证数据负载(如果存在)中查找字节对齐的下面二进制比特图案：

'00000000 00000000 000000xx'(其中 xx 代表任意 2 比特图案：'00'、'01'、'10'或'11')，并且使用其中插入一个等于 0x03 的字节：

'00000000 00000000 00000011 000000xx'

得到的字节序列加上包含标识 RBSP 数据结构类型的 NAL 单元的单元头部分就形成了整个 NAL 单元。

该过程允许任何 SODB 在一个 NAL 单元中出现，同时可以确保：

——该 NAL 单元中没有字节对齐的伪起始码；

——无论是否字节对齐，在 NAL 单元中没有 8 个值为 0 的比特后跟随起始码的序列。

5.2.4.3.2 NAL 单元的顺序及其与编码图像和视频序列的关系

5.2.4.3.2.1 序列、图像参数集 RBSP 的顺序及其激活

图像参数集 RBSP 包括的参数可以被一个或多个编码图像的编码条带 NAL 单元使用。在解码过程的开始，所有图像参数集 RBSP 都处于未激活的状态。至多一个图像参数集 RBSP 在解码过程中的指定时刻为激活的，并且任何特定图像参数集 RBSP 的激活会导致先前激活的图像参数集 RBSP(如果有的话)去激活。

当一个图像参数集 RBSP(具有特定的 pic_parameter_set_id 值)为未激活的，并且被编码条带

NAL 单元使用时(使用该 pic_parameter_set_id),则该图像参数集 RBSP 激活。此图像参数集 RBSP 称作活动图像参数集 RBSP,直到由于另一个图像参数集 RBSP 的激活而去激活。一个具有特定 pic_parameter_set_id 值的图像参数集 RBSP 在激活之前对解码过程来说就应为可用的。

任何图像参数集 RBSP 与活动图像参数集 RBSP 具有相同 pic_parameter_set_id 值,应与该活动图像参数集 RBSP 具有相同的内容,除非其位于一个编码图像的最后一个 VCL NAL 单元之后,另一个编码图像的第一个 VCL NAL 单元之前。

一个序列参数集 RBSP 包括的参数可以被一个或多个图像参数集 RBSP 或者包含缓存周期 SEI 消息的 SEI NAL 单元引用。所有序列参数集 RBSP 在解码过程的开始都处于未激活的状态。至多一个序列参数集 RBSP 在解码过程中的指定时刻为激活的,并且任何特定的序列参数集 RBSP 的激活会导致先前激活的序列参数集 RBSP(如果有的话)去激活。

当一个序列参数集 RBSP(具有一个 seq_parameter_set_id 的特定值)为未激活的,并且被一个激活的图像参数集 RBSP 或者一个包含缓存周期 SEI 消息的 SEI NAL 单元引用时(使用该 seq_parameter_set_id 值),该序列参数集 RBSP 激活。该序列参数集 RBSP 称作活动序列参数集 RBSP,直到由于另一个序列参数集 RBSP 的激活而去激活。一个具有特定 seq_parameter_set_id 值的序列参数集 RBSP 在激活之前对解码过程来说应为可用的。一个活动的序列参数集 RBSP 在整个编码视频序列中保持激活。

任何序列参数集 RBSP 与活动序列参数集 RBSP 具有相同的 seq_parameter_set_id 值,应与该活动序列参数集 RBSP 具有相同的内容,除非其位于一个编码视频序列之后,另一个编码视频序列的第一个 VCL NAL 单元和第一个包含缓存周期 SEI 消息的 SEI NAL 单元之前。

注:如果图像参数集 RBSP 或序列参数集 RBSP 在比特流中传送,这些规定分别对包含图像参数集 RBSP 或序列参数集 RBSP 的 NAL 单元强加了一个顺序约束,以保证它们在解码过程中适时的可用。在某些应用中,序列参数集和图像参数集也可以通过其他的可靠机制传送到解码器端。

对序列参数集和图像参数集中的语法元素值与其他语法元素之间的关系所作出的规定,仅针对活动序列参数集和活动图像参数集。

在解码过程中,活动图像参数集和活动序列参数集的参数值应保持有效。

5.2.4.3.2.2 安全参数集 RBSP 的激活

安全参数集 RBSP 包括一些参数,这些参数可以被一个或多个其他类型的 NAL 单元使用。在解码过程的开始,每个安全参数集 RBSP 在被解码器收到的同时激活,并且会导致先前激活的安全参数集 RBSP(如果有的话)去激活。至多一个安全参数集 RBSP 在解码过程中的指定时刻为激活的。

注:在某些应用中,安全参数集也可以通过其他的可靠机制传送到解码器端。

5.2.4.3.2.3 编码图像的第一个 VCL NAL 单元的检测

本条规定了 VCL NAL 单元的语法,使其能够检测到每个编码图像的第一个 VCL NAL 单元。

当前编码图像的任何编码条带 NAL 单元应与前一个编码图像的任何编码条带 NAL 单元以下列方式中的一种或多种进行区分:

——frame_num 值不同;

——pic_parameter_set_id 值不同;

——field_pic_flag 值不同;

——bottom_field_flag 在两个条带中都出现而且值不同;

——nal_ref_idc 值不同;

——nal_unit_type 值不同,而且其中一个 nal_unit_type 的值等于 2 或者 4;

——nal_unit_type 值相同,idr_pic_id 值不同。

5.2.4.3.2.4 VCL NAL 单元的顺序及其与编码图像的关系

每个 VCL NAL 单元都是一个编码图像的一部分。

一个编码图像中的 VCL NAL 单元的顺序规定如下：

——如果 roi_flag 等于 0，一个图像的编码条带顺序应按编码条带的第一个宏块索引递增的顺序。如果 roi_flag 等于 1，位于同一区域的编码条带顺序应按编码条带的第一个宏块索引递增的顺序；

——具有 nal_unit_type 等于 6 的 NAL 单元应位于编码图像的第一个 VCL NAL 单元之前；

——具有 nal_unit_type 等于 0，12，13，14 及 15 的 NAL 单元不能位于编码图像的第一个 VCL NAL 单元之前。

5.2.4.4 原始字节序列负载及 RBSP 尾比特语义

5.2.4.4.1 序列参数集 RBSP 语义

profile_id 和 level_id 指示比特流所遵守的档次和级别，见附录 C。

seq_parameter_set_id 指示序列参数集的编号。seq_parameter_set_id 的值应在 0～31 的范围内（包括 0 和 31）。

注：如果序列参数集语法元素的值改变，编码器宜使用不同的 seq_parameter_set_id 值。

chroma_format_idc 指示 5.1.2 所述的图像格式。chroma_format_idc 等于 0 表示图像格式为 4∶0∶0，每个宏块的编码块个数 CbNum 为 4；chroma_format_idc 等于 1 表示图像格式为 4∶2∶0，CbNum 为 6；chroma_format_idc 等于 2 表示图像格式为 4∶2∶2，CbNum 为 8。

bit_depth_luma_minus8 加 8 等于亮度阵列样点的比特位宽，如下所示：

$$BitDepthY = 8 + bit\_depth\_luma\_minus8$$

bit_depth_luma_minus8 取值范围应在 0～2 之间（包括 0 和 2）。

bit_depth_chroma_minus8 加 8 等于色度阵列样点的比特位宽，如下所示：

$$BitDepthC = 8 + bit\_depth\_chroma\_minus8$$

bit_depth_chroma_minus8 取值范围应在 0～2 之间（包括 0 和 2）。

pic_width_in_mbs_minus1 加 1 等于以宏块为单位的一个解码帧的宽度。

以宏块为单位的帧图像宽度由下列公式得出：

$$PicWidthInMbs = pic\_width\_in\_mbs\_minus1 + 1$$

帧图像的亮度阵列的宽度由下列公式得出：

$$PicWidthInSamplesL = PicWidthInMbs \times 16$$

帧图像的色度阵列的宽度由下列公式得出：

$$PicWidthInSamplesC = PicWidthInMbs \times MbWidthC$$

对增强层图像，

以宏块为单位的增强层帧图像宽度由下列公式可得：

$$PicWidthInMbsEL = PicWidthInMbs \times 2$$

增强层帧图像的亮度阵列的宽度由下列公式得出：

$$PicWidthInSamplesLEL = PicWidthInSamplesL \times 2$$

增强层帧图像的色度阵列的宽度由下列公式得出：

$$PicWidthInSamplesCEL = PicWidthInSamplesC \times 2$$

pic_height_in_mbs_minus1 加 1 等于以宏块为单位的一个解码帧的高度。

以宏块为单位的帧图像高度由下列公式得出：

$$PicHeightInMbs = pic\_height\_in\_mbs\_minus1 + 1$$

帧图像的亮度阵列的高度由下列公式得出：

$$PicHeightInSamplesL = PicHeightInMbs \times 16$$

帧图像的色度阵列的高度由下列公式得出：

$$PicHeightInSamplesC = PicHeightInMbs \times MbHeightC$$

帧图像的宏块数由下列公式得出：

PicSizeInMbs=PicWidthInMbs × PicHeightInMbs

对增强层图像，

以宏块为单位的增强层帧图像高度由下列公式可得：

PicHeightInMbsEL=PicHeightInMbs × 2

增强层帧图像的亮度阵列的高度由下列公式得出：

PicHeightInSamplesLEL=PicHeightInSamplesL × 2

增强层帧图像的色度阵列的高度由下列公式得出：

PicHeightInSamplesCEL=PicHeightInSamplesC × 2

增强层帧图像的宏块数由下列公式得出：

PicSizeInMbsEL=PicWidthInMbsEL × PicHeightInMbsEL

progressive_seq_flag 等于 0 表示编码视频序列中的编码图像可为编码场或编码帧。progressive_seq_flag 等于 1 表示编码视频序列中的编码图像均为编码帧。

roi_flag 等于 1 表示编码视频序列中的图像允许按区域划分后进行条带编码，roi_flag 等于 0 表示整个图像没有区域划分直接编码。

svc_flag 等于 1 表示编码视频序列中的图像可包含 nal_unit_type 等于 3 和 4 的 NAL 单元，svc_flag 等于 0 表示编码视频序列中的图像不包含 nal_unit_type 等于 3 和 4 的 NAL 单元。

vui_parameters_present_flag 等于 1 表示后续码流中存在 vui_parameters( )语法结构，见附录 D。vui_parameters_present_flag 等于 0 表示后续码流中不存在 vui_parameters( )语法结构。

5.2.4.4.2 图像参数集 RBSP 语义

pic_parameter_set_id 指示图像参数集的编号。pic_parameter_set_id 的值应在 0～255 的范围内(包括 0 和 255)。

seq_parameter_set_id 指示图像参数集引用的序列参数集的编号。seq_parameter_set_id 的值应在 0～31 的范围内(包括 0 和 31)。

entropy_coding_mode_flag 指示语法表中某些包含两个描述符的语法元素的熵编码方式，具体如下：

——如果 entropy_coding_mode_flag 等于 0，那么采用语法表中左边描述符所指定的 VLC 方法；

——否则(entropy_coding_mode_flag 等于 1)，就采用语法表中右边描述符所指定的 CABAC 方法。

pic_init_qp 表示图像中量化因子的初始值。如果 roi_flag 等于 1，pic_init_qp 表示背景区域量化因子的初始值。如果对应序列参数集中的 bit_depth_luma_minus8 等于 0，pic_init_qp 为 6 位无符号整数，否则 pic_init_qp 为 7 位无符号整数。pic_init_qp 的取值范围为 0～(55+(BitDepthY − 8) × 7)(包括 0 和 55+(BitDepthY − 8) × 7)。

num_roi 表示一个图像中的 ROI 的个数。如果 num_roi 不在码流中，默认其值等于 0。

non_roi_skip_flag 指示编码视频序列中的非 IDR 图像中是否包含背景区域部分的图像信息。如果为非 IDR 图像且 non_roi_skip_flag 等于 1，表示该编码图像中不包含背景区域部分的图像信息。如果为非 IDR 图像且 non_roi_skip_flag 等于 0，表示该编码图像中包含背景区域部分的图像信息。编码视频序列中的 IDR 图像中应包含背景区域部分的图像信息，对应的 non_roi_skip_flag 默认等于 0。如果 non_roi_skip_flag 不在码流中，默认其值等于 0。

scalable_non_roi_skip_flag 指示编码视频序列中的非 IDR 图像的增强层条带是否包含背景区域部分的图像信息。如果为非 IDR 图像且 scalable_non_roi_skip_flag 等于 1，表示该编码图像的增强层条带中不包含背景区域部分的图像信息。如果为非 IDR 图像且 scalable_non_roi_skip_flag 等于 0，表示该编码图像的增强层条带中包含背景区域部分的图像信息。编码视频序列中的 IDR 图像的增强层条

带中应包含背景区域部分的图像信息，对应的 scalable_non_roi_skip_flag 默认等于 0。如果 scalable_non_roi_skip_flag 不在码流中，默认其值等于 non_roi_skip_flag。

pic_init_qp_for_roi 表示图像中 ROI 内量化因子的初始值。如果对应序列参数集中的 bit_depth_luma_minus8 等于 0，pic_init_qp_for_roi 为 6 位无符号整数，否则 pic_init_qp_for_roi 为 7 位无符号整数。pic_init_qp_for_roi 的取值范围为 0～(55＋(BitDepthY － 8) × 7)(包括 0 和 55＋(BitDepthY － 8) × 7)，如果 pic_init_qp_for_roi 不在码流中，默认其值等于 pic_init_qp。

top_left[i]和 bottom_right[i] 表示该 ROI 矩形区域内的左上角和右下角的宏块所对应的位置，其值等于该宏块在编码帧图像中按光栅扫描顺序的宏块索引，宏块索引与宏块位置的对应关系见 5.3.3.3.1。对第 i 个 ROI，top_left[i]和 bottom_right[i]的取值应符合下述约束：

——top_left[i]小于或等于 bottom_right[i]，且 bottom_right[i]小于 PicSizeInMbs；

——(top_left[i] % PicWidthInMbs)小于或等于(bottom_right[i] % PicWidthInMbs)。

增强层图像中对应 ROI 内的左上角和右下角的宏块在基本层图像中所对应的位置与 top_left[i]及 bottom_right[i]相同。

fixed_pic_qp 等于 1 表示同一编码图像中的量化因子取值不变，fixed_pic_qp 等于 0 表示同一编码图像中的量化因子取值可变。

weighting_pred_flag 等于 1 表示图像解码过程中使用加权预测，weighting_pred_flag 等于 0 表示图像解码过程中不使用加权预测。

loop_filter_disable_flag 等于 1 表示图像解码过程中不调用去块效应滤波过程，loop_filter_disable_flag 等于 0 表示图像解码过程中调用去块效应滤波过程。

5.2.4.4.3 安全参数集 RBSP 语义

encryption_flag 等于 1 表示支持对图像编码条带，或序列参数集，或图像参数集，或扩展数据单元进行加密，即 NAL 单元中的 RBSP 可能经过加密。encryption_flag 等于 0 表示不支持对 NAL 单元中的 RBSP 进行加密。

authentication_flag 等于 1 表示支持对图像编码条带，或序列参数集，或图像参数集，或扩展数据单元进行认证，即 NAL 单元的负载中可能存在上述数据内容的认证数据。authentication_flag 等于 0 表示不支持对上述数据内容进行认证，即 NAL 单元的负载中不包含认证数据。

如果 svc_flag 等于 1，同一图像的基本层编码条带和增强层编码条带分别进行加密和认证。

encryption_type 指示加密所采用的算法，具体对应关系见表 31。

表 31 encryption_type 与具体加密算法的对应关系

encryption_type	加 密 算 法
0	SM1
1	DES
2	AES
3…15	保留

hash_type 表示进行认证所采用的算法，见表 32。

表 32 hash_type 与具体算法的对应关系

hash_type	认 证 算 法	摘要数据长度(byte)
0	MD5	16
1	SHA-1	20
2	SHA-224	28

表 32（续）

hash_type	认证算法	摘要数据长度(byte)
3	SHA-256	32
4	SHA-384	48
5	SHA-512	64
6	HMAC-MD5	16
7…15	保留	保留

hash_hierarchy_flag 等于 1 表示对一个图像的每个编码条带 NAL 单元中的 RBSP 分别进行认证产生初次摘要数据，并对整个图像中各条带的摘要数据按条带解码顺序排列后再次认证产生该图像的二次摘要数据。如果在码流中传送，该二次摘要数据应在对应图像的最后一个编码条带 NAL 单元中。hash_hierarchy_flag 等于 0 表示只对一个图像中各个编码条带 NAL 单元中的 RBSP 按条带解码顺序排列后进行认证产生该图像的初次摘要数据。如果在码流中传送，该初次摘要数据应在对应图像的最后一个编码条带 NAL 单元中。如果 NAL 单元负载为序列参数集，或图像参数集，或扩展数据单元，只对该 NAL 单元负载单独认证产生摘要数据，与 hash_hierarchy_flag 的取值无关。如果 hash_hierarchy_flag 不在码流中，默认其值等于 0。

hash_discard_p_pictures 等于 1 表示对 P 图像不进行认证，hash_discard_p_pictures 等于 0 表示对 P 图像进行认证。如果 hash_discard_p_pictures 不在码流中，默认其值等于 1。

hash_discard_b_pictures 等于 1 表示对 B 图像不进行认证，hash_discard_b_pictures 等于 0 表示对 B 图像进行认证。如果 hash_discard_b_pictures 不在码流中，默认其值等于 1。

hash_discard_extension_data 等于 1 表示对序列参数集 NAL 单元，图像参数集 NAL 单元和扩展数据 NAL 单元中的 RBSP 不进行认证，hash_discard_extension_data 等于 0 表示对序列参数集 NAL 单元，图像参数集 NAL 单元和扩展数据 NAL 单元中的 RBSP 进行认证。如果 hash_discard_extension_data 不在码流中，默认其值等于 1。

signature_type 表示是否对图像的摘要数据进行数字签名，以及签名的算法，见表 33。

表 33　signature_type 与具体签名算法的对应关系

signature_type	签名算法
0	不做签名
1	RSA
2	DSA
3…15	保留

如果 NAL 单元负载为序列参数集，或图像参数集，或扩展数据，只对该 NAL 单元负载产生的摘要数据进行签名；否则

——如果 hash_hierarchy_flag 等于 1，进行签名的摘要数据为一个图像的二次摘要数据；

——如果 hash_hierarchy_flag 等于 0，进行签名的摘要数据为一个图像的初次摘要数据。

successive_hash_pictures_minus1 加 1 表示按解码顺序进行数字签名的连续图像个数。successive_hash_pictures_minus1 的取值应为 0～255。

如果 successive_hash_pictures_minus1 大于 0，首先对按解码顺序连续的 SuccessiveHashPictures 个图像的摘要数据产生树型摘要数据，再对树顶摘要数据进行数字签名。如图 12 所示，n 个图像的树顶摘要数据是对前 n－1 个图像的树顶摘要数据和第 n 个图像的摘要数据排列后按 hash_type 所示的方法产生的摘要数据。如果在码流中传送，该签名数据应在对应的第 SuccessiveHashPictures 个图像的最后一个条带 NAL 单元中。

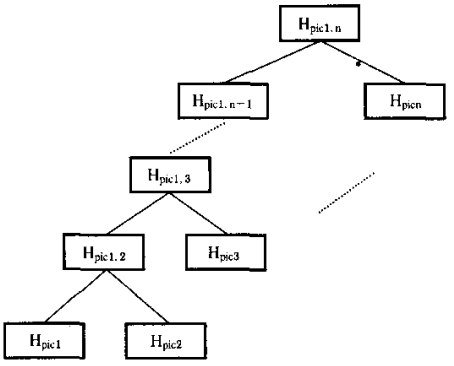


图 12　树型摘要示意图

SuccessiveHashPictures=successive_hash_pictures_minus1+1

注：安全参数集激活后的第一个包含摘要数据的图像应为连续 SuccessiveHashPictures 个图像的第一个。包含签名数据的图像的下一个图像应为连续 SuccessiveHashPictures 个图像的第一个。一个 IDR 图像应为连续 SuccessiveHashPictures 个图像的第一个。如果之前包含摘要数据的连续图像不足 SuccessiveHashPictures，签名数据应在 IDR 图像之前最后一个包含摘要数据的编码图像的最后一个条带 NAL 单元中。

如果 successive_hash_pictures_minus1 等于 0，对一个图像摘要数据进行数字签名。如果在码流中传送，该签名数据应在对应图像的最后一个条带 NAL 单元中。

signature_data_length_minus1 加 1 表示签名数据的长度，以字节为单位。signature_data_length_minus1 的取值应为 0～255。

5.2.4.4.4　补充增强信息 RBSP 语义

补充增强信息(SEI)消息与图像的输出及显示有关，在 VCL NAL 单元的解码过程中不是必要的。

一个 SEI NAL 单元可以包括多个 SEI 消息。每个 SEI 消息中规定了 SEI 负载类型 PayloadType 和 SEI 负载长度 PayloadSize。

ff_byte 为一个等于 0xFF 的字节。

last_payload_type_byte 表示负载类型的最后一个字节。

last_payload_size_byte 表示负载长度的最后一个字节。

sei_payload 的规定见附录 E。

5.2.4.4.5　序列结尾 RBSP 语义

视频序列结尾 RBSP 表示按解码顺序在比特流中下一个(如果存在)紧跟的编码图像将是一个 IDR 编码图像。视频序列结尾 RBSP 的内容为空。

5.2.4.4.6　流结尾 RBSP 语义

流结尾 RBSP 表示按解码顺序，在流结尾 RBSP 之后没有任何其他的 NAL 单元。流结尾 RBSP 的内容为空。

5.2.4.4.7　编码条带 RBSP 语义

编码条带 RBSP 由条带头和条带数据组成。

5.2.4.4.8　RBSP 尾比特语义

rbsp_stop_one_bit 应等于 1。

rbsp_alignment_zero_bit 应等于 0。

5.2.4.4.9　填充数据 RBSP 语义

填充数据 RBSP 包括的字节取值均等于 0xFF。填充数据 RBSP 不用于图像的解码过程。

ff_byte 应等于 0xFF。

5.2.4.5　条带头语义

如果存在，条带头中的语法元素 pic_parameter_set_id，frame_num，idr_pic_id，field_pic_flag，bottom_field_flag 的值在同一编码图像的所有条带头中应一致。

pic_parameter_set_id 指示使用的图像参数集。pic_parameter_set_id 的值应在 0～255 范围内(包括 0 和 255)。

frame_num 表示编码视频序列中每帧图像的显示顺序的低 8 比特。一个编码视频序列中第一帧图像的 frame_num 应等于 0。如果一帧图像由两个编码场组成,这两个场的所有编码条带头中的 frame_num 的取值应相同。

idr_pic_id 指示 IDR 图像的编号。编码视频序列中的连续的 IDR 图像的 idr_pic_id 应不同。idr_pic_id 的取值范围应在 0～255 范围内(包括 0 和 255)。

field_pic_flag 等于 1 表示该条带为一个编码场的条带。field_pic_flag 等于 0 表示该条带为一个编码帧的条带。如果 field_pic_flag 不在码流中,默认其值等于 0。

bottom_field_flag 等于 1 表示该条带为一个编码底场的一部分。bottom_field_flag 等于 0 表示该条带为一个编码顶场的一部分。如果 bottom_field_flag 不在码流中,默认其值等于 0。

first_mb_in_slice 表示在条带中第一个宏块在图像中的宏块索引 MbIndex。当 roi_flag 等于 0 时,一个条带的 first_mb_in_slice 的值应不小于同一图像的任何在该条带之前(按解码顺序)的其他条带的 first_mb_in_slice 的值。如果 field_pic_flag 等于 0,first_mb_in_slice 的取值范围应为 0～(PicSizeInMbs-1),如果 field_pic_flag 等于 1,first_mb_in_slice 的取值范围应为 0～(Floor(PicSizeInMbs / 2)-1)。

宏块索引的取值 MbIndex 的计算方式见 3.1.47。

slice_type 表示条带的编码类型,见表 34。

表 34　slice_type 的名称关联

slice_type	slice_type 的名称
0	P(P 条带)
1	B(B 条带)
2	I(I 条带)
3	P(P 条带)
4	B(B 条带)
5	I(I 条带)

slice_type 的值在 3～5 范围内表示,除了当前条带的编码类型,所有当前编码图像的其他条带的 slice_type 值应与当前条带的 slice_type 值一样,或者等于当前条带的 slice_type 值减 3。

当 nal_unit_type 等于 2 或 4(IDR 图像)时,slice_type 应等于 2 或 5。

picture_reference_flag 等于 1 表示所有宏块都使用缺省参考图像;picture_reference_flag 等于 0 表示由每个宏块在宏块层确定参考图像。确定参考图像的方法见 5.3.3.4 。

fixed_slice_qp 等于 1 说明在该条带内量化因子不变,否则(fixed_slice_qp 等于 0)说明在该条带内量化因子可变。如果 fixed_slice_qp 不在码流中,默认其值等于 1。

slice_qp_delta 表示条带的 SliceQPY 与 pic_init_qp 或 pic_init_qp_for_roi 的差值。slice_qp_delta 的取值范围为－32～31(包括－32 和 31)。如果 slice_qp_delta 不在码流中,默认其值等于 0。

SliceQPY 按下面的公式计算:

——如果 roi_flag 等于 1 且该条带在任一 ROI 内,

SliceQPY＝pic_init_qp_for_roi＋(fixed_pic_qp＝＝1? 0:slice_qp_delta);

——否则,

SliceQPY＝pic_init_qp＋(fixed_pic_qp＝＝1? 0:slice_qp_delta)。

SliceQPY 的取值范围应在 0～(55 ＋ (BitDepthY－ 8)×7)(包括 0 和 55＋(BitDepthY－8)×7)。

slice_weighting_flag 等于 1 表示宏块的运动补偿使用加权预测，slice_weighting_flag 等于 0 表示宏块的运动补偿不使用加权预测。如果 slice_weighting_flag 不在码流中，默认其值等于 0。

num_of_references_minus1 加 1 表示条带中宏块解码中用到的参考图像数目 number_of_references。number_of_references 的取值范围见 5.3.3.4 。

luma_scale 表示预测亮度时的缩放参数，取值范围为 0～255(包括 0 和 255)。

luma_shift 表示预测亮度时的平移参数，取值范围为－128～127(包括－128 和 127)。

marker_bit 应等于 1。

注：marker_bit 用于防止码流中出现长 0 序列。

chroma_scale 表示预测色度时的缩放参数，取值范围为 0～255(包括 0 和 255)。

chroma_shift 表示预测色度时的平移参数，取值范围为－128～127(包括－128 和 127)。

如果使用加权预测，同一个方向的参考图像的加权预测参数 luma_scale，luma_shift，chroma_scale 和 chroma_shift 在码流中的顺序按参考图像索引由小到大排列。对 B 条带，前向参考图像的加权预测参数在前，后向参考图像的加权预测参数在后。

mb_weighting_flag 等于 0 表示所有帧间预测宏块都采用加权运动补偿，mb_weighting_flag 等于 1 表示每个帧间预测宏块由 weighting_prediction 决定是否采用加权预测。

loop_filter_parameter_flag 等于 1 表示条带头中包含语法元素 slice_alpha_c0_offset 和 slice_beta_offset，loop_filter_parameter_flag 等于 0 表示条带头中不包含语法元素 slice_alpha_c0_offset 和 slice_beta_offset。

slice_alpha_c0_offset 表示去块效应滤波过程中 α 和 C 索引的偏移，取值范围为－8～8(包括－8 和 8)，去块效应滤波参数 AlphaCOffset 等于 slice_alpha_c0_offset。如果 slice_alpha_c0_offset 不在码流中，默认其值等于 0。

slice_beta_offset 表示去块效应滤波过程中 β 索引的偏移，取值范围为－8～8(包括－8 和 8)，去块效应滤波参数 BetaOffset 等于 slice_beta_offset。如果 slice_beta_offset 不在码流中，默认其值等于 0。

5.2.4.6 条带数据语义

cabac_alignment_one_bit 应等于 1。

mb_skip_run 表示连续的跳过宏块数。当解码一个 P 条带时，这些跳过宏块的宏块类型为 P_Skip；当解码一个 B 条带时，这些跳过宏块的宏块类型为 B_Skip。mb_skip_run 的值应在 0～(PicSizeInMbsMbIndex)范围内(包括边界值)。

mb_skip_flag 等于 1 表示当前宏块的类型为跳过宏块，mb_skip_flag 等于 0 表示当前宏块的类型不是跳过宏块。

end_of_slice 等于 1 表示当前条带中没有未解码宏块，end_of_slice 等于 0 表示当前条带中还有其他未解码宏块。

5.2.4.7 宏块语义

mb_type 表示宏块类型 MbType，其语义依赖于 slice_type。

如果 slice_type 为 I，条带内的宏块都为 I 宏块，MbTypeIdx 等于 mb_type，对应宏块类型见表 35。

如果 slice_type 为 P，条带内的宏块可为 I 宏块或 P 宏块，如果 mb_type 值大于等于 4，宏块为 I 宏块，MbTypeIdx 等于 mb_type 减 4，对应宏块类型见表 35；否则宏块为 P 宏块，MbTypeIdx 等于 mb_type，对应宏块类型见表 36。

如果 slice_type 为 B，条带内的宏块可为 I 宏块或 B 宏块，如果 mb_type 值大于等于 23，宏块为 I 宏块，MbTypeIdx 等于 mb_type 减 23，对应宏块类型参见表 35；否则宏块为 B 宏块，MbTypeIdx 等于 mb_type，对应宏块类型见表 37。

表 35 I 宏块类型

MbTypeIdx	MbType	MvNum	MbPredMode
0	I_4×4	0	
1	I_8×8	0	

表 36 P 宏块类型

MbTypeIdx	MbType	MvNum	MbPredMode
0	P_16×16	1	前向
1	P_16×8	2	前向
2	P_8×16	2	前向
3	P_8×8	4	前向
Inferred	P_Skip	0	无

表 37 B 宏块类型

MbTypeIdx	MbType	MvNum	MbPredMode
0	B_Direct_16×16	0	双向
1	B_Fwd_16×16	1	前向
2	B_Bck_16×16	1	后向
3	B_Sym_16×16	1	双向
4	B_Fwd_Fwd_16×8	2	上前、下前
5	B_Fwd_Fwd_8×16	2	左前、右前
6	B_Bck_Bck_16×8	2	上后、下后
7	B_Bck_Bck_8×16	2	左后、右后
8	B_Fwd_Bck_16×8	2	上前、下后
9	B_Fwd_Bck_8×16	2	左前、右后
10	B_Bck_Fwd_16×8	2	上后、下前
11	B_Bck_Fwd_8×16	2	左后、右前
12	B_Fwd_Sym_16×8	2	上前、下双
13	B_Fwd_Sym_8×16	2	左前、右双
14	B_Bck_Sym_16×8	2	上后、下双
15	B_Bck_Sym_8×16	2	左后、右双
16	B_Sym_Fwd_16×8	2	上双、下前
17	B_Sym_Fwd_8×16	2	左双、右前
18	B_Sym_Bck_16×8	2	上双、下后
19	B_Sym_Bck_8×16	2	左双、右后
20	B_Sym_Sym_16×8	2	上双、下双
21	B_Sym_Sym_8×16	2	左双、右双
22	B_8×8	0…4	前向、后向、双向
Inferred	B_Skip	0	无

I 宏块的 MbType 和 MvNum 见表 35；

——如果 MbTypeIdx 等于 0，宏块类型为 I_4×4；

——如果 entropy_coding_mode_flag 等于 1 并且 MbTypeIdx 等于 1，宏块类型为 I_8×8；

——如果 entropy_coding_mode_flag 等于 0 并且 MbTypeIdx 大于等于 1，宏块类型为 I_8×8，MbCBP的 CodeNum 等于 MbTypeIdx 减 1，MbCBP 的解析过程见 5.4.2.2。

P 宏块的 MbType 和 MvNum 见表 36，其中 P_Skip 类型(跳过宏块)可由条带数据语法推导得到。

B 宏块的 MbType 和 MvNum 见表 37，其中 B_Skip 类型(跳过宏块)可由条带数据语法推导得到。

Transform8×8Flag 的规定如下：

——如果 MbType 为 I_4×4，Transform8×8Flag 等于 0；

——否则，Transform8×8Flag 等于 1。

mb_part_type 如果 MbType 等于 B_8×8，MbPartType 和 MbPartMvNum 的值用 mb_part_type 查表 38 得到，MvNum 为该宏块所有块的 MbPartMvNum 的和。

表 38　B_8×8 子模式

mb_part_type	MbPartType	MbPartMvNum	MbPartPredMode
0	SB_Direct_8×8	0	双向
1	SB_Fwd_8×8	1	前向
2	SB_Bck_8×8	1	后向
3	SB_Sym_8×8	1	双向

在表 35～表 38 中定义的 MbType 和 MbPartType 中，有 Skip 字样的称为跳过模式，有 Direct 字样的称为直接模式，有 Sym 字样的称为对称模式。MvNum 表示宏块的运动矢量数。对称模式为一种双向预测模式，此时比特流中只包含前向参考索引和前向运动矢量，对称模式的后向参考索引和后向运动矢量的推导见 5.3.5.2。

pred_mode_flag 等于 1 表示对应亮度块的帧内预测模式等于帧内预测模式的预测值，pred_mode_flag 等于 0 表示对应亮度块的帧内预测模式根据 intra_luma_pred_mode 确定。

intra_luma_pred_mode 用于确定宏块中对应亮度块的帧内预测模式，解码过程见 5.3.4 。

intra_chroma_pred_mode 用于确定宏块中顺序号为 4 和 5 的色度块的帧内预测模式。解码过程见 5.3.4 。

intra_chroma_pred_mode_422 用于确定 4∶2∶2 格式下宏块中顺序号为 6 和 7 的色度块的帧内预测模式。解码过程见 5.3.4。

mb_reference_index 表示运动矢量对应的参考图像索引。如果 field_pic_flag 等于 0 或者 slice_type 为 B，mb_reference_index 为 1 位无符号整数。如果 field_pic_flag 等于 1 并且 slice_type 的值为 P，mb_reference_index 为 2 位无符号整数。参考图像索引的规定见 5.3.3.4。如果存在两个方向的运动矢量，前向参考索引在前，后向参考索引在后。

mv_diff_x 和 mv_diff_y 表示运动矢量差值，单位为四分之一样点，取值范围为－4096～4095(按亮度样点为－1024～1023.75)，如果存在的话，前向运动矢量差值在前，后向运动矢量差值在后。

weighting_prediction 等于 1 表示宏块使用加权预测，weighting_prediction 等于 0 表示宏块不使用加权预测。如果 weighting_prediction 不在码流中，默认其值等于 slice_weighting_flag。

coded_block_pattern 指示宏块中的 8×8 亮度块和 8×8 色度块是否包含非零残差数据。如果 chroma_format_idc 等于 1，由 coded_block_pattern 解析得到一个 6 位无符号整数 MbCBP，指示宏块中的 4 个 8×8 亮度块和 2 个 8×8 色度块是否包含非零残差数据；如果 chroma_format_idc 等于 0 或 2，由 coded_block_pattern 解析得到一个 4 位无符号整数 MbCBP，指示宏块中的 4 个 8×8 亮度块是否包含非零残差数据。解析过程见 5.4.2.2。

coded_block_pattern_422 指示 chroma_format_idc 等于 2 时 4 个 8×8 色度块是否包含非零残差数据。由 coded_block_pattern_422 解析得到一个 4 位无符号整数 MbCBP422，解析过程见 5.4.2.2。

宏块的残差编码模式 cbp_8×8 可以统一表示为如下：

——如果 chroma_format_idc 等于 0 或 1，则 cbp_8×8 的值等于 MbCBP；

——如果 chroma_format_idc 等于 2，则 cbp_8×8 的值等于(MbCBP422≪4) + MbCBP。

coded_block_pattern_4×4 表示 MbType 为 I_4×4 时，一个 8×8 亮度块中的 4 个 4×4 亮度块是否包含非零残差数据。解析过程见 5.4.2.2，解析得到第 i 个 8×8 亮度块的 cbp_4×4[i]。

mb_qp_delta 表示宏块的 QPY 与 SliceQPY 之间的差值。mb_qp_delta 的取值范围为－32～31(包括－32 和 31)。宏块的量化参数 QPY 按下面的公式计算：

QPY＝SliceQPY＋(fixed_slice_qp＝＝1? 0:mb_qp_delta)

QPY 的取值范围应在 0～(55＋(BitDepthY－8)×7)(包括 0 和 55＋(BitDepthY－8)×7)。

5.2.4.8 可伸缩性视频编码(SVC)增强层宏块语义

svc_mode_flag 等于 1 表示只使用跨层预测，svc_mode_flag 等于 0 表示不使用跨层预测。其他语义定义同 5.2.4.7 相关语义定义。

5.2.4.9 块语义

trans_coefficient 定义如下：

——如果 entropy_coding_mode_flag 等于 0，trans_coefficient 用于确定游程长度和量化非零系数值的联合索引，或用于确定转逸游程长度和转逸系数值的符号。trans_coefficient 的解析过程见 5.4.2.3；

——如果 entropy_coding_mode_flag 等于 1，trans_coefficient 用于确定游程长度和量化非零系数值。trans_coefficient 的解析过程见 5.4.3。

escape_level_diff 当 trans_coefficient 不能确定游程长度和量化非零系数值的联合索引时，escape_level_diff用于确定转逸系数的绝对值，escape_level_diff 的解析过程见 5.4.2.3。

5.2.4.10 监控扩展数据单元语义

5.2.4.10.1 监控扩展数据单元语义通则

监控扩展数据单元用于传递监控相关信息，由扩展单元标识(extension_id)区分。有效的 extension_id 不应等于 0x80。

注：extension_id 等于 0x05 的监控扩展数据单元用于第 6 章。

监控扩展数据单元中的信息在图像解码过程中不是必要的。

surveillance_extension_stop_byte 应等于 0x80。

5.2.4.10.2 感兴趣区域扩展语义

extension_id 8 位无符号整数，感兴趣区域扩展的标号 extension_id 应等于 1。

extension_length 8 位无符号整数，指示 extension_length 之后的本扩展语法元素长度，以字节为单位。

position_idc 8 位无符号整数，用于指示图像来源的位置。

camera_idc 16 位无符号整数，用于指示图像来源的摄像机标识。

region_num 指示感兴趣区域的个数。

reserved_bits 保留比特位，每个相应的比特位取值应为 1。

region_top_left_mbx[i]和 region_top_left_mby[i]分别表示第 i 个 ROI 内左上角宏块的横坐标值、纵坐标值。坐标值以宏块为单位。

以样点为单位计算的第 i 个 ROI 左上角位置的横坐标值、纵坐标值为：

roiTopLeftSamplePositionX[i]＝region_top_left_mbx[i]×16

roiTopLeftSamplePositionY[i]＝region_top_left_mby[i]×16

region_width_in_mbs_minus1[i]加 1 和region_height_in_mbs_minus1[i]加 1 分别等于第 i 个 ROI 的宽度和高度，以宏块为单位。

以样点为单位计算的第 i 个 ROI 的宽度、高度为：

roiWidthInSample[i]=(region_width_in_mbs_minus1[i]+1)×16

roiHeightInSample[i]=(region_height_in_mbs_minus1[i]+1)×16

5.2.4.10.3 监控事件扩展语义

extension_id 8 位无符号整数，监控事件扩展的标号 extension_id 应等于 2。

extension_length 8 位无符号整数，指示 extension_length 之后的本扩展语法元素长度，以字节为单位。

position_idc 8 位无符号整数，用于指示图像来源的位置。

camera_idc 16 位无符号整数，用于指示图像来源的摄像机标识。

region_num 指示图像监控区域的个数。

reserved_bits 保留比特位，每个相应的比特位取值应为 1。

event_num[i]表示第 i 个区域内所监控事件的个数，每个区域内监控事件的个数不应超过 16。

region_event_id 区域事件标识符，用于指示区域内可能的事件特征。区域事件特征由应用决定，不在本标准中定义。

5.2.4.10.4 监控报警扩展语义

extension_id 8 位无符号整数，监控报警扩展的标号 extension_id 应等于 3。

extension_length 8 位无符号整数，指示 extension_length 之后的本扩展语法元素长度，以字节为单位。

position_idc 8 位无符号整数，用于指示图像来源的位置。

camera_idc 16 位无符号整数，用于指示图像来源的摄像机标识。

alert_num 表示报警信息的个数。

reserved_bits 保留比特位，每个相应的比特位取值应为 1。

alert_region_id 指示报警的区域，其取值应与感兴趣区域的索引值对应，如果没有定义感兴趣区域，alert_region_id 应等于 0。

alert_event_id 报警事件标识符，用于指示报警事件特征。报警事件特征由应用决定，不在本标准中定义，但应与监控事件扩展中的区域事件标识符 region_event_id 相对应。

frame_num 取值应为发生所指报警事件所对应图像的 frame_num，该编码图像的编码条带 NAL 单元中的 frame_num 的取值与监控报警扩展中的 frame_num 相等，位于监控报警扩展所处监控扩展数据单元之前，且最为临近，满足上述条件的编码图像对应该监控报警信息。

5.2.4.10.5 绝对时间信息扩展语义

extension_id 8 位无符号整数，绝对时间信息扩展的标号 extension_id 应等于 4。

extension_length 8 位无符号整数，指示 extension_length 之后的本扩展语法元素长度，以字节为单位。

position_idc 8 位无符号整数，用于指示图像来源的位置。

camera_idc 16 位无符号整数，用于指示图像来源的摄像机标识。

hour_bits 表示小时信息。hour_bits 取值范围应在 0～23 之间(包括 0 和 23)。

minute_bits 表示分钟信息。minute_bits 取值范围应在 0～59 之间(包括 0 和 59)。

second_bits 表示秒信息。second_bits 取值范围应在 0～59 之间(包括 0 和 59)。

second_fraction_bits 表示秒的分数信息，以 1/16 384 秒为单位。second_fraction_bits 取值范围应在 0～16 383 之间(包括 0 和 16 383)。

ref_date_flag 等于 1 表示绝对时间信息扩展中包含绝对日期参考信息，ref_date_flag 等于 0 表示绝对时间信息扩展中不包含绝对日期参考信息。

year_minus2000_bits 加 2000 表示年份信息，如下所示：

Year＝year_minus2000_bits＋2000

year_minus2000_bits 取值范围应在 0～127 之间(包括 0 和 127)。

month_bits 表示月份信息，month_bits 取值范围应在 1～12 之间(包括 1 和 12)。

day_bits 表示日期信息，date_bits 取值范围应在 1～31 之间(包括 1 和 31)。

frame_num 表示绝对时间信息所对应的编码图像的 frame_num，该编码图像的编码条带 NAL 单元中的 frame_num 的取值与绝对时间信息扩展中的 frame_num 相等，位于绝对时间信息扩展所处监控扩展数据单元之后，且最为临近，满足上述条件的编码图像的起始时间对应该绝对时间信息。

5.3 解码过程

5.3.1 视频解码器

视频解码流程示例见图 13。视频解码器接收编码比特流，对条带中的宏块，经熵解码、逆扫描、反量化及反变换产生一组残差数据 D'，并根据码流中的信息通过帧内预测或帧间预测得到预测数据 PRED，预测数据与残差数据通过计算生成重建图像 F'。重建图像经去块滤波产生最终的解码图像。

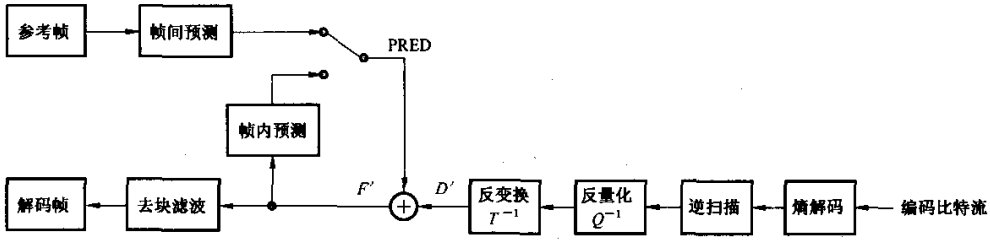


图 13 解码流程示例

5.3.2 NAL 单元解码过程

NAL 单元的解码过程的输入为 NAL 单元，输出为封装在 NAL 单元中的 RBSP 语法结构。该过程为从 NAL 单元中提取出 RBSP 语法结构。如果 encryption_idc 等于 1，从 NAL 单元中提取 RBSP 语法结构时需要对加密的 RBSP 进行解密处理，得到未加密的 RBSP 语法结构。该解密过程不在本标准中规定。

对 NAL 单元中的 RBSP 语法结构按照如下的方式进行解码：

——nal_unit_type 的值为 1、2、3 和 4 时 NAL 单元的解码过程，见 5.3.3；

——nal_unit_type 的值为 1 和 2 时 NAL 单元中的 I 宏块的帧内预测过程，见 5.3.4；

——nal_unit_type 的值为 1 时 NAL 单元中的 P 和 B 宏块的帧间预测过程，见 5.3.5；

——nal_unit_type 的值为 1 和 2 时 NAL 单元中的宏块在去块效应滤波前变换系数的解码过程和图像重建过程，见 5.3.6；

——nal_unit_type 的值为 1、2、3 和 4 时 NAL 单元的重建图像的去块效应滤波过程，见 5.3.7；

——nal_unit_type 的值为 3 和 4 时 NAL 单元中的宏块在去块效应滤波之前的解码过程，见 5.3.8；

——nal_unit_type 的值为 7、8 和 9 时 NAL 单元中的 RBSP 分别为序列参数集，图像参数集及安全参数集。活动的序列参数集，图像参数集及安全参数集用于其他 NAL 单元的解码过程；

——nal_unit_type 的值为 13 的 NAL 单元的解码过程，见第 6 章；

——nal_unit_type 的值为 0、14 和 15 的 NAL 单元的解码过程不在本标准规定。

5.3.3 条带解码过程

5.3.3.1 图像，条带及宏块的分类和对应关系

编码视频序列中可能解码出以下四种图像：

——帧内解码图像(I 图像)；

——即时解码刷新图像(IDR 图像);

——前向帧间解码图像(P 图像);

——双向帧间解码图像(B 图像)。

所有图像可为帧图像,也可为场图像。

I 图像中所有的条带均为 I 条带。

IDR 图像中所有的条带均为 I 条带。也就是说,IDR 图像同时也是 I 图像。与 I 图像不同的是,IDR 图像解码之后,解码顺序上所有后续的编码图像都可以不用根据任何在该 IDR 图像之前解码的图像来进行帧间预测解码。每个编码视频序列的第一幅图像为 IDR 图像。可以通过 nal_unit_type 来识别一幅图像是否为 IDR 图像。

P 图像中可以包含 I 条带和 P 条带。

B 图像中可以包含 I 条带,P 条带和 B 条带。

I 图像和 P 图像可以作为参考图像,B 图像不能作为参考图像。

注:I 条带中只包含 I 宏块,P 条带中可包含 P 宏块和 I 宏块,而 B 条带可包含 B 宏块和 I 宏块。

5.3.3.2 图像顺序号的解码过程

本过程的输入为 frame_num,field_pic_flag,bottom_field_flag,输出为图像顺序号 CurrPicCnt。对于参考图像,如果为参考帧,field_pic_flag 等于 0;如果为参考场,field_pic_flag 等于 1,顶场的 bottom_field_flag 等于 0,底场的 bottom_field_flag 等于 1。

图像顺序号指示当前图像及参考图像的图像顺序,用于解码 P 和 B 条带时的运动矢量导出过程中计算块之间的距离。

图像顺序号 CurrPicCnt 由下列公式得出:

——如果 field_pic_flag 等于 0,则 CurrPicCnt 等于 2×frame_num;

——否则(field_pic_flag 等于 1),如果 bottom_field_flag 等于 0,则 CurrPicCnt 等于 2×frame_num;

——否则,CurrPicCnt 等于 2×frame_num+1。

5.3.3.3 条带中宏块的映射

5.3.3.3.1 宏块索引 MbIndex 与宏块位置(X, Y)的映射关系

宏块索引 MbIndex 对应图像中的宏块位置(X_0, Y_0)的计算方法如下:

——如果 field_pic_flag 等于 0,

X_0 = MbIndex % PicWidthInMbs,

Y_0 = Floor(MbIndex / PicWidthInMbs);

——否则(field_pic_flag 等于 1),

a) 如果 bottom_field_flag 等于 0,

X_0 = MbIndex % PicWidthInMbs,

Y_0 = Floor(MbIndex/PicWidthInMbs)×2;

b) 否则(bottom_field_flag 等于 1),

X_0 = MbIndex % PicWidthInMbs,

Y_0 = Floor(MbIndex/PicWidthInMbs)×2+1。

5.3.3.3.2 宏块位置(X, Y)所属区域的计算

如果 num_roi 大于 0,码流中当前宏块的宏块位置(X, Y)所属区域的计算方法如下:

```
RegionNum=num_roi
for(i=0;i<num_roi;i++) {
  if(X>=Xtl[i]&&X<=Xbr[i]&&Y>=Ytl[i]&&Y<=Ybr[i]) {
      RegionNum=i
      break
```

}

}

如果 RegionNum 等于 num_roi，宏块位置(X，Y)所属区域为非 ROI，否则宏块位置(X，Y)属于由 top_left[i]和 bottom_right[i]确定的那个 ROI。

5.3.3.3.3 下一个宏块的宏块索引的计算

5.3.3.3.3.1 num_roi 等于 0 时下一宏块的宏块索引的计算

如果 num_roi 等于 0，码流中当前宏块的宏块为索引 MbIndex[n]，计算下一个宏块的宏块索引 MbIndex[n+1]的方法如下：

MbIndex[n+1]=MbIndex[n]+1

5.3.3.3.3.2 num_roi 大于 0 时下一宏块的宏块索引的计算

如果 num_roi 大于 0，计算方法如下：

1) 计算数组 top_left[i]和 bottom_right[i]对应的宏块位置(Xtl[i]，Ytl[i])和(Xbr[i]，Ybr[i])，以及 first_mb_in_slice 对应的宏块位置(X，Y)，方法见 5.3.3.3.1；
2) 根据(X，Y)判断当前条带所属区域，方法见 5.3.3.3.2；
3) 如 5.3.3.3.3.1 所示的计算下一宏块的宏块索引，并计算对应的位置(X′，Y′)及所属区域；
4) 如果(X′，Y′)和(X，Y)属于同一区域，步骤 3)结果即为下一宏块对应的宏块位置，否则回到步骤 3)继续计算。

5.3.3.3.4 宏块解码顺序

条带中的宏块解码顺序按下述步骤确定：

1) 条带解码开始时当前宏块的 MbIndex 等于 first_mb_in_slice；
2) 如果 entropy_coding_mode_flag 等于 0，则如果 mb_skip_run 存在，由 mb_skip_run 解析得到 SkipMbCount；否则 SkipMbCount 等于 0。如果 SkipMbCount 不等于 0，则从当前宏块开始的 SkipMbCount 个宏块的 MbType 根据条带类型设为 P_Skip 或 B_Skip，这些宏块按照跳过宏块类型进行处理。如果 SkipMbCount 等于 0，则解码当前宏块。如果 entropy_coding_mode_flag 等于 1，则解析 mb_skip_flag，如果当前宏块的 MbType 为 P_Skip 或 B_Skip，则当前宏块按照跳过宏块类型进行处理。如果宏块的 MbType 不是 P_Skip 或 B_Skip，则解码当前宏块；
3) 如果条带中还有未解码宏块，则计算下一宏块的索引及位置，回到步骤 2)继续解码，否则条带解码结束。

5.3.3.4 参考图像选择

P 条带或 B 条带最多可有两个参考帧或四个参考场，它们应为最临近当前解码图像的参考帧或参考场。P 条带最多可有前向的两个参考帧或者四个参考场，一帧图像中后解码的场(底场)中的条带还可参考当前帧的前一场(顶场)。B 条带的参考图像为前向及后向的各一个参考帧或者各两个参考场。

运动矢量所指的参考样点可能越过参考图像的边界。越过参考图像边界外的整数参考样点应使用图像内距离该整数参考样点所指位置最近的整数样点进行边界扩展获得。对亮度样点矩阵，参考块的样点在水平和垂直方向均不应越过参考图像边界外 16 个样点。对色度样点矩阵：

——如果图像格式为 4∶2∶0，参考块的样点在水平和垂直方向均不应越过参考图像边界外 8 个样点；

——如果图像格式为 4∶2∶2，参考块的样点在水平方向不应越过参考图像边界外 8 个样点，在垂直方向不应越过参考图像边界外 16 个样点。

参考索引值用于指示条带解码过程中所用到的参考图像，不同的参考图像有不同的参考索引值，参考索引的取值范围为 0～3。参考索引值随着参考图像与当前条带所在图像距离(显示顺序)的增加而增加，索引值为 0 的参考图像与当前条带所在图像的距离最近，索引值为 1 的图像距离其次，索引值为

3 的图像距离最远。两个方向单独排序。

参考图像的参考索引值的标记方法及可能的参考图像数 number_of_reference 的定义如下(见图 14):

a) 如果当前条带为 P 条带,field_pic_flag 等于 0,如图 14a)所示,number_of_reference 可能等于 1 或 2;

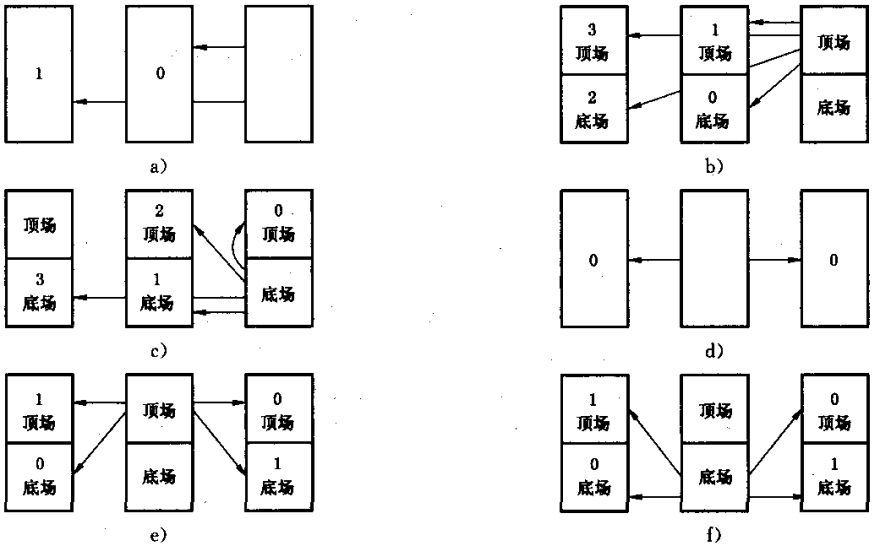
b) 如果当前条带为 P 条带,field_pic_flag 等于 1 且 bottom_field_flag 等于 0,如图 14b)所示,number_of_reference 可能等于 2 或 4;

c) 如果当前条带为 P 条带,field_pic_flag 等于 1 且 bottom_field_flag 等于 1,如图 14c)所示,number_of_reference 可能等于 1,3 或 4;

d) 如果当前条带为 B 条带,field_pic_flag 等于 0,如图 14d)所示,number_of_reference 等于 2;

e) 如果当前条带为 B 条带,field_pic_flag 等于 1 且 bottom_field_flag 等于 0,如图 14e)所示,number_of_reference 等于 4;

f) 如果当前条带为 B 条带,field_pic_flag 等于 1 且 bottom_field_flag 等于 1,如图 14f)所示,number_of_reference 等于 4。



注:图中的数字表示参考索引值,箭头所指的为参考图像。如果其中一个参考图像为 IDR 图像,则该 IDR 图像之前的图像不能作为参考图像。当一个帧编码的 IDR 图像的顶场或底场作为参考场时,顶场被视为 IDR 图像,底场视为非 IDR 图像。

图 14 参考图像的选择及索引标记方法

5.3.4 帧内预测过程

5.3.4.1 8×8 亮度块帧内预测模式

当前宏块的每一个 8×8 亮度块采用以下方法确定其预测模式:

a) 计算当前块预测模式的预测值:

1) 如果左边块可用并且为帧内预测块,则将左边块的帧内预测模式赋值给 intraPredModeA;如果左边块可用但不是帧内预测块,则 intraPredModeA 等于 2;否则 intraPredModeA 等于−1;

2) 如果上边块可用并且为帧内预测块,则将上边块的帧内预测模式赋值给 intraPredModeB;如果上边块可用但不是帧内预测块,则 intraPredModeB 等于 2;否则 intraPredModeB 等于−1;

3) 如果 intraPredModeA 或 intraPredModeB 等于－1，则 predIntraPredMode 等于 2；否则 predIntraPredMode 等于 Min(intraPredModeA，intraPredModeB)。

b) 如果 pred_mode_flag 的值为 1，则预测模式 IntraLumaPredMode 等于 predIntraPredMode；

c) 如果 pred_mode_flag 的值为 0，并且 intra_luma_pred_mode 小于 predIntraPredMode，则 IntraLumaPredMode 等于 intra_luma_pred_mode；否则 IntraLumaPredMode 等于 intra_luma_pred_mode 加 1。

确定左边块或上边块是否存在或可用的方法见 5.1.3.3。

左边块或上边块的帧内预测模式的确定：

a) 如果左边宏块或上边宏块为 I_4×4，则以左边或上边 8×8 块中序号为 0 的 4×4 块的预测模式为左边块或上边块的预测模式；

b) 如果左边宏块或上边宏块为 I_8×8，则以左边或上边 8×8 块的预测模式为左边块或上边块的预测模式。

IntraLumaPredMode 的值与亮度块预测模式的关系见表 39。

表 39 8×8 亮度块帧内预测模式

IntraLumaPredMode	名　　称
0	Intra_8×8_Vertical
1	Intra_8×8_Horizontal
2	Intra_8×8_DC
3	Intra_8×8_Down_Left
4	Intra_8×8_Down_Right
注：所示的 8×8 亮度块帧内预测模式示意见图 15。	

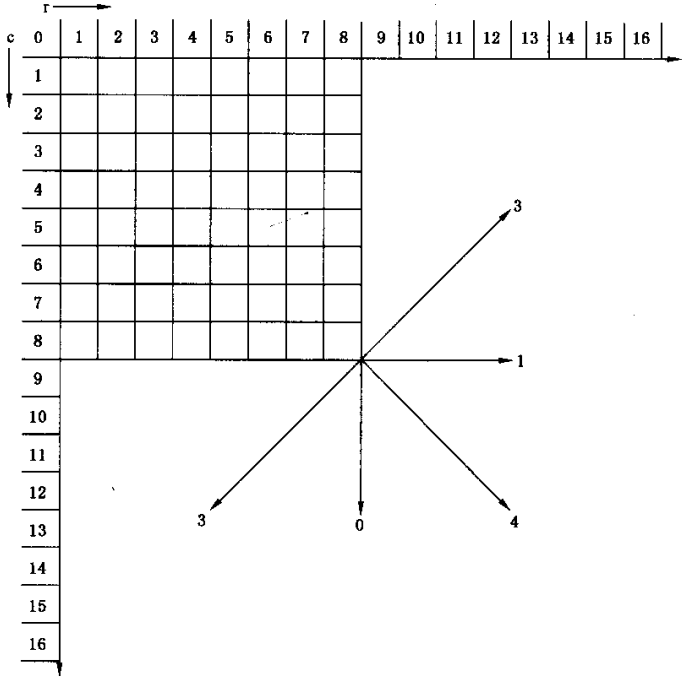


图 15 8×8 亮度块帧内预测模式

5.3.4.2 8×8 亮度块帧内预测

5.3.4.2.1 参考样点的获得

8×8 帧内预测块由其上边和左边的参考样点 r[i](i=0...16)和 c[i](i=0...16)来预测(r、c 可表示亮度或色度参考样点),见图 15,其中 r[0]等于 c[0]。如果帧内预测需要用到 i 大于 16 的上边和左边的参考样点,则 r[i]=r[16],c[i]=c[16](i>16)。参考样点的值为去块滤波之前的重建样点值。本条定义 8×8 帧内预测参考样点获取的具体方法。

设当前块所属图像的亮度样点矩阵为 I。

设当前块左上角样点在当前图像中的坐标为(x_0, y_0),其参考样点按以下规则获得:

——如果坐标为(x_0+i−1, y_0−1)(i=1...8)的样点可用,则 r[i] (i=1...8)等于 I[x_0+i−1, y_0−1],r[i]可用;否则 r[i]不可用;

——如果坐标为(x_0+i−1,y_0−1)(i=9...16)的样点可用,则 r[i] (i=9...16)等于 I[x_0+i−1, y_0−1],r[i]可用;否则 r[i] (i=9... 16)等于 r[8],r[i]是否可用由 r[8]是否可用决定;

——如果坐标为(x_0−1,y_0+i−1)(i=1...8)的样点可用,则 c[i] (i=1...8)等于 I[x_0−1,y_0+i−1],c[i]可用;否则 c[i]不可用;

——如果坐标为(x_0−1,y_0+i−1)(i=9...16)的样点可用,则 c[i] (i=9...16)等于 I[x_0−1,y_0+i−1],c[i]可用;否则 c[i] (i=9... 16)等于 c[8] ,c[i]是否可用由 c[8]是否可用决定;

—— 如果坐标为(x_0− 1, y_0− 1)的样点可用,则 r[0]等于 I[x_0−1,y_0−1],r[0]可用;否则

1) 如果 r[1]和 c[1]都可用,则 r[0]等于(r[1]+c[1]+1)≫1,r[0]可用;否则
2) 如果 r[1]可用并且 c[1]不可用,则 r[0]等于 r[1],r[0]可用;否则
3) 如果 c[1]可用并且 r[1]不可用,则 r[0]等于 c[1],r[0]可用;否则 r[0]不可用。

5.3.4.2.2 帧内预测过程

根据 IntraLumaPredMode 决定 8×8 亮度块帧内预测模式。各模式的 8×8 帧内预测过程如下:

a) Intra_8×8_Vertical 预测(IntraLumaPredMode 等于 0)

当 r[i](i=1...8)可用时,该模式可使用,此时:

predMatrix[x,y]=r[x+1](x,y=0...7)

b) Intra_8×8_Horizontal 预测(IntraLumaPredMode 等于 1)

当 c[i](i=1...8)可用时,该模式可使用,此时:

predMatrix[x,y]=c[y+1] (x,y=0...7)

c) Intra_8×8_DC 预测(IntraLumaPredMode 等于 2)

1) 如果 r[i]、c[i](i=0...9)都可用,则:

predMatrix[x,y]=(((r[x]+2×r[x+1]+r[x+2]+2)≫2)+((c[y]+2×c[y+1]+c[y+2]+2)≫2))≫1(x,y=0...7);否则:

2) 如果 r[i](i=0...9)可用,则:

predMatrix[x,y]=(r[x]+2×r[x+1]+r[x+2]+2)≫2(x,y=0...7);否则

3) 如果 c[i](i=0...9)可用,则

predMatrix[x,y]=(c[y]+2×c[y+1]+c[y+2]+2)≫2(x,y=0...7);否则

4) predMatrix[x,y]=1≪(BitDepthY−1)(x,y=0...7)。

d) Intra_8×8_Down_Left 预测(IntraLumaPredMode 等于 3)

当 r[i]、c[i](i=1...16)均可用时,该模式可使用,此时:

predMatrix[x,y]=(((r[x+y+1]+2×r[x+y+2]+r[x+y+3]+2)≫2)+((c[x+y+1]+2×c[x+y+2]+c[x+y+3]+2)≫2))≫1(x,y=0...7)。

e) Intra_8×8_Down_Right 预测(IntraLumaPredMode 等于 4)

当 r[i]、c[i](i=0...8)均可用时,该模式可使用,此时:

1) 如果 x 等于 y,predMatrix[x,y]=(c[1]+2×r[0]+r[1]+2)≫2(x,y=0...7);否则

2) 如果 x 大于 y,predMatrix[x,y]=(r[x−y+1]+2×r[x−y]+r[x−y−1]+2)≫2(x,y=0...7);否则

3) 如果 y 大于 x,predMatrix[x,y]=(c[y−x+1]+2×c[y−x]+c[y−x−1]+2)≫2(x,y=0...7)。

本过程输出为一个 8×8 预测亮度样点矩阵 predMatrix。

5.3.4.3 4×4 亮度块帧内预测模式

当前宏块的每一个 4×4 亮度块采用以下方法确定其预测模式:

a) 计算当前块预测模式的预测值:

1) 如果左边块可用并且为帧内预测块,则将左边块的帧内预测模式赋值给 intraPredModeA;如果左边块可用但不是帧内预测块,则 intraPredModeA 等于 2;否则 intraPredModeA 等于−1;

2) 如果上边块可用并且为帧内预测块,则将上边块的帧内预测模式赋值给 intraPredModeB;如果上边块可用但不是帧内预测块,则 intraPredModeB 等于 2;否则 intraPredModeB 等于−1;

3) 如果 intraPredModeA 或 intraPredModeB 等于−1,则 predIntraPredMode 等于 2;否则 predIntraPredMode 等于 Min(intraPredModeA, intraPredModeB)。

b) 如果 pred_mode_flag 的值为 1,则预测模式 IntraLumaPredMode 等于 predIntraPredMode;

c) 如果 pred_mode_flag 的值为 0,并且 intra_luma_pred_mode 小于 predIntraPredMode,则 IntraLumaPredMode 等于 intra_luma_pred_mode;否则 IntraLumaPredMode 等于 intra_luma_pred_mode 加 1。

确定左边块或上边块是否存在或可用的方法见 5.1.3.3。

左边块或上边块的帧内预测模式的确定:

——如果左边宏块或上边宏块为 I_4×4,则以左边或上边 4×4 块的预测模式为左边块或上边块的预测模式;

——如果左边宏块或上边宏块为 I_8×8,则以左边或上边 8×8 块的预测模式为左边块或上边块的预测模式。

IntraLumaPredMode 的值与亮度块预测模式的关系见表 40。

表 40 4×4 亮度块帧内预测模式

IntraLumaPredMode	名 称
0	Intra_4×4_Vertical
1	Intra_4×4_Horizontal
2	Intra_4×4_DC
3	Intra_4×4_Down_Left
4	Intra_4×4_Down_Right
注:所示的 4×4 亮度块帧内预测模式示意见图 16。	

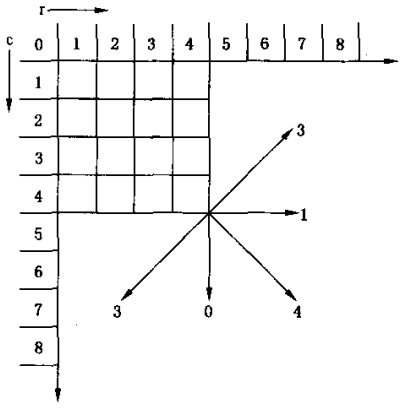


图 16　4×4 亮度块帧内预测模式

5.3.4.4　4×4 亮度块帧内预测

5.3.4.4.1　参考样点的获得

4×4 帧内预测块由其上边和左边的参考样点 r[i](i=0...8)和 c[i](i=0...8)来预测(r、c 表示亮度参考样点),参见图 16,其中 r[0]等于 c[0]。如果帧内预测需要用到 i 大于 8 的上边和左边的参考样点,则 r[i]=r[8],c[i]=c[8](i>8)。参考样点的值为去块滤波之前的重建样点值。本条定义 4×4 帧内预测参考样点获取的具体方法。

设当前块所属的图像的亮度样点矩阵为 I。

设当前块左上角样点在当前图像中的坐标为(x_0, y_0),其参考样点按以下规则获得:

——如果坐标为(x_0+i-1, y_0-1)(i=1...4)的样点可用,则 r[i] (i=1...4)等于 I[x_0+i-1, y_0-1],r[i]可用;否则 r[i]不可用;

——如果坐标为(x_0+i-1,y_0-1)(i=5...8)的样点可用,则 r[i] (i=5...8)等于 I[x_0+i-1,y_0-1],r[i]可用;否则 r[i] (i=5...8)等于 r[4],r[i]是否可用由 r[4]是否可用决定;

——如果坐标为(x_0-1,y_0+i-1)(i=1...4)的样点可用,则 c[i](i=1...4)等于 I[x_0-1,y_0+i-1],c[i]可用;否则 c[i]不可用;

——如果坐标为(x_0-1,y_0+i-1)(i=5...8)的样点可用,则 c[i](i=5...8)等于 I[x_0-1,y_0+i-1],c[i]可用;否则 c[i](i=5...8)等于 c[4],c[i]是否可用由 c[4]是否可用决定;

——如果坐标为(x_0-1,y_0-1)的样点可用,则 r[0]等于 I[x_0-1,y_0-1],r[0]可用;否则

1)　如果 r[1]和 c[1]都可用,则 r[0]等于(r[1]+c[1]+1)≫1;r[0]可用;否则

2)　如果 r[1]可用并且 c[1]不可用,则 r[0]等于 r[1];r[0]可用;否则

3)　如果 c[1]可用并且 r[1]不可用,则 r[0]等于 c[1];r[0]可用;否则 r[0]不可用。

5.3.4.4.2　帧内预测过程

根据 IntraLumaPredMode 决定 4×4 亮度块帧内预测模式。各模式的 4×4 帧内预测过程如下:

a)　Intra_4×4_Vertical 预测(IntraLumaPredMode 等于 0)

当 r[i](i=1...4)可用时,该模式可使用,此时:

predMatrix[x,y]=r[x+1](x,y=0...3)

b)　Intra_4×4_Horizontal 预测(IntraLumaPredMode 等于 1)

当 c[i](i=1...4)可用时,该模式可使用,此时:

predMatrix[x,y]=c[y+1](x,y=0...3)

c)　Intra_4×4_DC 预测(IntraLumaPredMode 等于 2)

1)　如果 r[i]、c[i](i=0...5)都可用,则:

predMatrix[x,y]=(((r[x]+2×r[x+1]+r[x+2]+2)≫2)+((c[y]+2×c[y+1]+

c[y+2]+2))≫2))≫1(x,y=0...3)；否则

2) 如果 r[i](i=0...5)可用，则：
predMatrix[x,y]=(r[x]+2×r[x+1]+r[x+2]+2)≫2(x,y=0...3)；
否则

3) 如果 c[i](i=0...5)可用，则：
predMatrix[x,y]=(c[y]+2×c[y+1]+c[y+2]+2)≫2(x,y=0...3)；
否则

4) predMatrix[x,y]=1≪(BitDepthY−1)(x,y=0...3)。

d) Intra_4×4_Down_Left 预测(IntraLumaPredMode 等于 3)
当 r[i]、c[i](i=1...8)均可用时，该模式可使用，此时：
predMatrix[x,y]=(((r[x+y+1]+2×r[x+y+2]+r[x+y+3]+2)≫2)+((c[x+y+1]+2×c[x+y+2]+c[x+y+3]+2)≫2))≫1(x,y=0...3)。

e) Intra_4×4_Down_Right 预测(IntraLumaPredMode 等于 4)
当 r[i]、c[i](i=0...4)均可用时，该模式可使用，此时：

1) 如果 x 等于 y，predMatrix[x,y]=(c[1]+2×r[0]+r[1]+2)≫2(x,y=0...3)；否则

2) 如果 x 大于 y，predMatrix[x,y]=(r[x−y+1]+2×r[x−y]+r[x−y−1]+2)≫2(x,y=0...3)；否则

3) 如果 y 大于 x，predMatrix[x,y]=(c[y−x+1]+2×c[y−x]+c[y−x−1]+2)≫2(x,y=0...3)。

本过程输出为一个 4×4 预测亮度样点矩阵 predMatrix。

5.3.4.5 色度块帧内预测模式

如果 chroma_format_idc 不等于 0，宏块中顺序号为 4 和 5 的色度块的帧内预测模式 IntraChromaPredMode 等于 intra_chroma_pred_mode。如果 chroma_format_idc 等于 2，宏块中顺序号为 6 和 7 的色度块的帧内预测模式 IntraChromaPredMode 等于 intra_chroma_pred_mode_422。

IntraChromaPredMode 的值与色度块预测模式的关系见表 41。

表 41 8×8 色度块帧内预测模式

IntraChromaPredMode	名　　称
0	Intra_Chroma_DC
1	Intra_Chroma_Horizontal
2	Intra_Chroma_Vertical
3	Intra_Chroma_Plane

5.3.4.6 色度块帧内预测

色度块的帧内预测块由其上边和左边的参考样点 r[i](i=0...16)和 c[i](i=0...16)来预测，参见图 15，其中 r[0]等于 c[0]，参考样点的值为去块滤波前的重建样点值。色度块参考样点的获取方法与亮度块相同(见 5.3.4.2.1)，与亮度块帧内预测不同之处在于 r、c 表示色度参考样点。根据 IntraChromaPredMode 决定 8×8 色度块帧内预测模式。各模式的 8×8 帧内预测过程如下：

a) Intra_Chroma_DC 预测(IntraChromaPredMode 等于 0)

1) 如果 r[i]、c[i](i=0...9)都可用，则：
predMatrix[x,y]=(((r[x]+2×r[x+1]+r[x+2]+2)≫2)+((c[y]+2×c[y+1]+c[y+2]+2)≫2))≫1(x,y=0...7)；否则

2) 如果 r[i](i=0...9)可用，则：
predMatrix[x,y]=(r[x]+2×r[x+1]+r[x+2]+2)≫2(x,y=0...7)；

否则

3) 如果 c[i](i=0...9)可用，则：

predMatrix[x,y]=(c[y]+2×c[y+1]+c[y+2]+2)≫2(x,y=0...7)；

否则

4) predMatrix[x,y]=1≪(BitDepthC−1)(x,y=0...7)。

b) Intra_Chroma_Horizontal 预测(IntraChromaPredMode 等于 1)

当 c[i](i=1...8)可用时，该模式可使用，此时：

predMatrix[x,y]=c[y+1](x,y=0...7)。

c) Intra_Chroma_Vertical 预测(IntraChromaPredMode 等于 2)

当 r[i](i=1...8)可用时，该模式可使用，此时：

predMatrix[x,y]=r[x+1](x,y=0...7)。

d) Intra_Chroma_Plane 预测(IntraChromaPredMode 等于 3)

当 r[i]、c[i](i=0...8)均可用时，该模式可使用，此时：

predMatrix[x,y]=Clip1((ia+(x−3)×ib+(y−3)×ic+16)≫5)(x,y=0...7)。

其中，ia=(r[8]+c[8])≪4，ib=(17×ih+16)≫5，ic=(17×iv+16)≫5，

$ih=\sum_{i=0}^{3}(i+1)\times(r[5+i]-r[3-i])$，$iv=\sum_{i=0}^{3}(i+1)\times(c[5+i]-c[3-i])$。

本过程输出为一个 8×8 预测色度样点矩阵 predMatrix。

5.3.5 帧间预测过程

5.3.5.1 概述

当解码 P、B 类型的宏块时，调用本过程。

本过程的输出为当前宏块的帧间预测样点矩阵 predMatrix，包含一个 16×16 亮度矩阵 predL，如果 chroma_format_idc 不等于 0，还包含相应的色度样点矩阵 predCb 及 predCr，分别对应色度分量 Cb 和 Cr。

5.3.5.2 运动矢量

5.3.5.2.1 BlockDistance 的计算方法

块的距离索引 DistanceIndex 定义如下：DistanceIndex 等于所属图像的 CurrPicCnt 值。

当前块(属于当前图像)和它的运动矢量所指向的参考块(属于参考图像)之间的距离 BlockDistance 计算如下：

——如果参考块在当前块之前(显示顺序)，BlockDistance 等于当前块的 DistanceIndex 减去参考块的 DistanceIndex 的差加上 512 的和模 512；

——如果参考块在当前块之后(显示顺序)，BlockDistance 等于参考块的 DistanceIndex 减去当前块的 DistanceIndex 的差加上 512 的和模 512。

5.3.5.2.2 亮度运动矢量预测

设图 11 中块 A、B、C、D 的原始运动矢量或转换后的运动矢量为 mvA、mvB、mvC、mvD，对应的 BlockDistance 为 BlockDistanceA、BlockDistanceB、BlockDistanceC、BlockDistanceD，对应的参考索引为 referenceIndexA、referenceIndexB、referenceIndexC、referenceIndexD。

——如果 A 不可用或者采用帧内预测模式，或者与当前块 E 没有同一预测方向的运动矢量，mvA 为零矢量，BlockDistanceA 等于 1，A 的参考索引值为−1；

——如果 B 不可用或者采用帧内预测模式，或者与当前块 E 没有同一预测方向的运动矢量，mvB 为零矢量，BlockDistanceB 等于 1，B 的参考索引值为−1；

——如果 D 不可用或者采用帧内预测模式，或者与当前块 E 没有同一预测方向的运动矢量，mvD 为零矢量，BlockDistanceD 等于 1，D 的参考索引值为−1；

——如果 C 不可用，那么 mvC 等于 mvD，BlockDistanceC 等于 BlockDistanceD，C 的参考索引值等于 D 的参考索引值；

——如果 C 采用帧内预测模式，或者与当前块 E 没有同一预测方向的运动矢量，mvC 为零矢量，BlockDistanceC 等于 1，C 的参考索引值为－1。

当前块 E 的运动矢量预测值 MVEPred 计算过程如下：

第一步，如果 A、B 和 C 三者中只有一个块 X 的参考索引值不为－1，那么 MVEPred 等于 mvX(X 为 A、B 或 C)；否则进行第二步。

第二步，如果 E 所在宏块按 16×8 或 8×16 模式编码，计算过程如下(见图 17)：

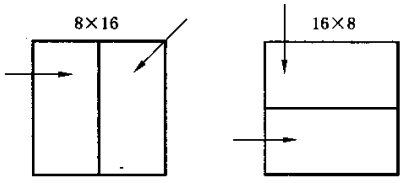


图 17　8×16 或 16×8 模式预测

a)　8×16 模式：

1)　E 为左块：如果 A 和 E 的参考索引值相同，MVEPred 等于 mvA；否则进行第三步；

2)　E 为右块：如果 C 和 E 的参考索引值相同，MVEPred 等于 mvC ；否则进行第三步。

b)　16×8 模式：

1)　E 为上块：如果 B 和 E 的参考索引值相同，MVEPred 等于 mvB；否则进行第三步；

2)　E 为下块：如果 A 和 E 的参考索引值相同，MVEPred 等于 mvA；否则进行第三步。

第三步，首先根据 BlockDistanceA，BlockDistanceB，BlockDistanceC，BlockDistanceE 对 mvA(mvA_x，mvA_y)，mvB(mvB_x，mvB_y)，mvC(mvC_x，mvC_y)进行缩放，得到 MVA (MVA_x，MVA_y)，MVB (MVB_x，MVB_y)，MVC(MVC_x，MVC_y)，然后计算 MVEPred。

MVA_x＝Sign(mvA_x)×((Abs(mvA_x)×BlockDistanceE×Floor(512/BlockDistanceA)＋256)≫9)

MVA_y＝Sign(mvA_y＋delt1)×((Abs(mvA_y＋delt1)×BlockDistanceE×Floor(512/BlockDistanceA)＋256)≫9)－delt2

MVB_x＝Sign(mvB_x)×((Abs(mvB_x)×BlockDistanceE×Floor(512/BlockDistanceB)＋256)≫9)

MVB_y＝Sign(mvB_y＋delt1)×((Abs(mvB_y＋delt1)×BlockDistanceE×Floor(512/BlockDistanceB)＋256)≫9)－delt2

MVC_x＝Sign(mvC_x)×((Abs(mvC_x)×BlockDistanceE×Floor(512/BlockDistanceC)＋256)≫9)

MVC_y＝Sign(mvC_y＋delt1)×((Abs(mvC_y＋delt1)×BlockDistanceE×Floor(512/BlockDistanceC)＋256)≫9)－delt2

其中 BlockDistanceE 为当前块 E 对应的 BlockDistance。

delt1 和 delt2 按照以下方式决定：

a)　如果 field_pic_flag 的值为 1，则，

1)　如果当前块所在的场为顶场，并且 mvX(X 为 A、B 或 C)指向的场为底场，则 delt1＝2；

2)　如果当前块所在的场为底场，并且 mvX(X 为 A、B 或 C)指向的场为顶场，则 delt1＝－2；

3)　如果当前块所在的场为顶场，并且 mvX(X 为 A、B 或 C)指向的场为顶场，则 delt1＝0；

4)　如果当前块所在的场为底场，并且 mvX(X 为 A、B 或 C)指向的场为底场，则 delt1＝0；

5)　如果当前块所在的场为顶场，并且 MVX(X 为 A、B 或 C)指向的场为底场，则 delt2＝2；

6) 如果当前块所在的场为底场，并且 MVX(X 为 A、B 或 C)指向的场为顶场，则 delt2＝－2；

7) 如果当前块所在的场为顶场，并且 MVX(X 为 A、B 或 C)指向的场为顶场，则 delt2＝0；

8) 如果当前块所在的场为底场，并且 MVX(X 为 A、B 或 C)指向的场为底场，则 delt2＝0。

b) 否则，delt1＝0，delt2＝0。

定义距离 Dist(MV1，MV2)＝Abs(x1－x2)＋Abs(y1－y2)，其中运动矢量 MV1＝[x1，y1]，MV2＝[x2，y2]。定义 VAB 等于 Dist(MVA，MVB)，VBC 等于 Dist(MVB，MVC)，VCA 等于 Dist(MVC，MVA)。

MVEPred 的计算如下：

——计算 FMV 等于 Median(VAB，VBC，VCA)；

——如果 FMV 和 VAB 相等，则 MVEPred 等于 MVC；

——否则，如果 FMV 和 VBC 相等，则 MVEPred 等于 MVA；

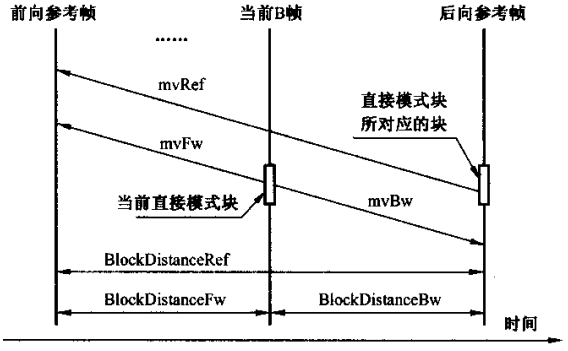
——否则，MVEPred 等于 MVB。

5.3.5.2.3 亮度运动矢量解码

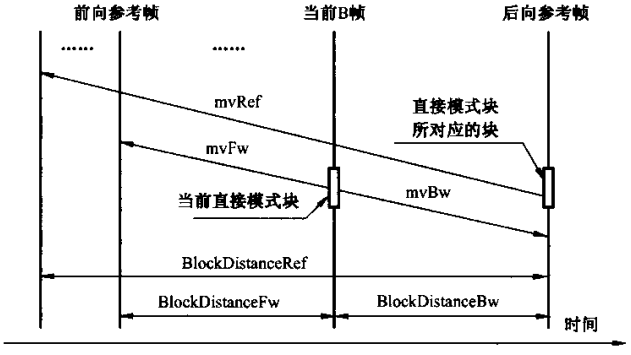
当前块 E 的运动矢量 mvE 等于 MVEPred 加上由 mv_diff_x 和 mv_diff_y 解码得到的运动矢量增量的和，其基本单位为四分之一样点。当前块 E 的运动矢量范围应符合 5.3.3.4 的规定。

5.3.5.2.4 亮度运动矢量导出

如果当前块类型为跳过模式、直接模式或对称模式，其运动矢量按照下面定义的方法导出(见图 18，图 19)，否则将 5.3.5.2.3 解码得到的运动矢量，按照宏块划分顺序分配给相应块。

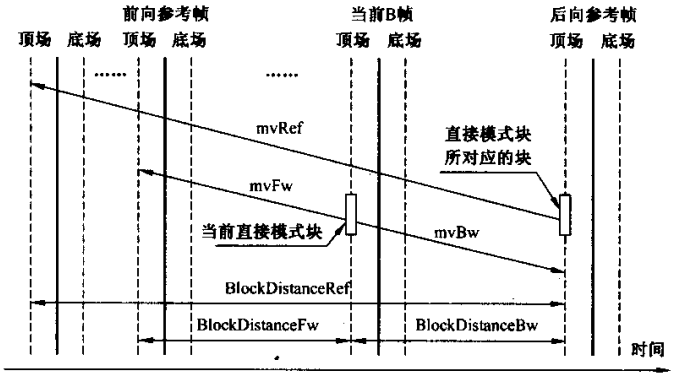


a) field_pic_flag 等于 0 时，直接或者跳过模式所对应的块指向标记为 0 的参考图像

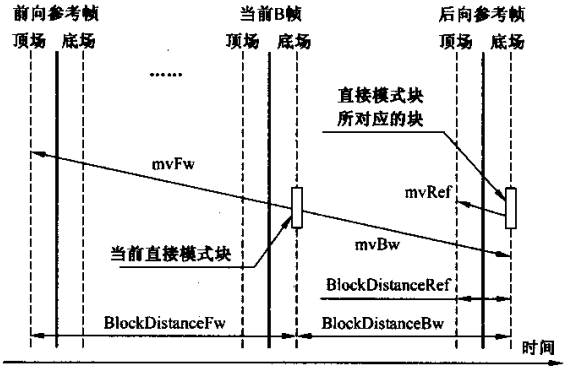


b) field_pic_flag 等于 0 时，直接或者跳过模式所对应的块指向标记为 1 的参考图像

图 18 直接模式的运动矢量推导过程

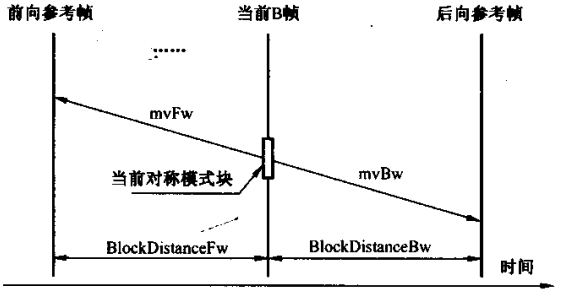


c) field_pic_flag 等于 1 时，直接或者跳过模式所对应的块指向标记为 3 的参考图像

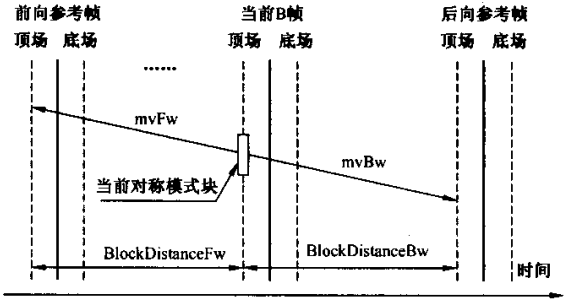


d) field_pic_flag 等于 1 时，直接或者跳过模式所对应的块指向标记为 0 的参考图像

图 18（续）



a) field_pic_flag 等于 0 时的对称模式



b) field_pic_flag 等于 1 时的对称模式

图 19　对称模式的运动矢量推导过程

a)　如果当前宏块类型为 P_Skip，当前块的前向参考图像为缺省参考图像，即图 14 中标记为 0 的

图像，且 mvE=MVEPred；

b) 如果当前宏块类型为 B_Skip 或 B_Direct_16×16，或当前块类型为 SB_Direct_8×8，对每个 8×8块分别进行以下步骤：

第一步：

1) 如果后向参考图像中与当前 8×8 块的左上角样点位置对应的样点所在的编码宏块类型为 I_8×8 或 I_4×4，则当前块的前后向参考图像均为缺省参考图像，即图 14 中标记为 0 的图像，该块的前后向运动矢量为根据 5.3.5.2.2 得到的当前块所在宏块(5.3.5.2.2 中块 E 为当前块所在宏块)的前后向运动矢量预测值。

2) 否则，如果当前块所在条带的 field_pic_flag 等于 0，

——当前块的前后向参考图像均为缺省参考图像，即图 14d)中标记为 0 的图像，其前后向距离索引分别为 DistanceIndexFw 和 DistanceIndexBw；当前块的前后向 BlockDistance 分别为 BlockDistanceFw 和 BlockDistanceBw；

——在后向参考图像中与当前块的左上角样点位置对应的样点所在的编码块的运动矢量为 mvRef(mvRef_x，mvRef_y)，该编码块的距离索引为 DistanceIndexCol，该运动矢量指向的参考块的距离索引为 DistanceIndexRef。

3) 否则，如果当前块所在图像的 field_pic_flag 等于 1，

——在后向参考图像中与当前块的左上角样点位置对应的样点所在的编码块的运动矢量为 mvRef(mvRef_x，mvRef_y)，该编码块的距离索引为 DistanceIndexCol，该运动矢量指向的参考块的距离索引为 DistanceIndexRef；

——当前块的前向参考索引为 0 和 1 的参考场中的参考块的距离索引分别为 DistanceIndexFw0 和 DistanceIndexFw1。如果 DistanceIndexRef 等于 DistanceIndexFw0，则当前块的前向参考索引为 0，DistanceIndexFw 等于 DistanceIndexFw0；否则当前块的前向参考索引为 1，DistanceIndexFw 等于 DistanceIndexFw1；

——当前块的后向参考索引为 0 和 1 的参考场中的参考块的距离索引分别为 DistanceIndexBw0 和 DistanceIndexBw1。如果当前块所在条带的 bottom_field_flag 等于 0，则当前块的后向参考场为参考索引标记为 0 的场，DistanceIndexBw 等于 DistanceIndexBw0；否则当前块的后向参考场为参考索引标记为 1 的场，DistanceIndexBw 等于 DistanceIndexBw1。

第二步：

1) BlockDistanceRef = (DistanceIndexCol − DistanceIndexRef + 512)% 512

当前块的距离索引为 DistanceIndexCur，则：

BlockDistanceFw = (DistanceIndexCur − DistanceIndexFw + 512)% 512

BlockDistanceBw = (DistanceIndexBw − DistanceIndexCur + 512)% 512

2) 如果当前块所在图像的 field_pic_flag 等于 0，并且后向参考图像的 field_pic_flag 等于 1，则 mvRef_y=mvRef_y×2；

3) 如果当前块所在图像的 field_pic_flag 等于 1，并且后向参考图像的 field_pic_flag 等于 0，则 mvRef_y=Sign(mvRef_y)×Floor(Abs(mvRef_y)/2)。

第三步：

1) 当前块的前向运动矢量 mvFw(mvFw_x，mvFw_y)为：

——如果 mvRef_x 小于 0，则 mvFw_x=−((Floor(16384 / BlockDistanceRef)×(1−mvRef_x×BlockDistanceFw)−1)≫14)；否则 mvFw_x=(Floor(16384/BlockDistanceRef)×(1+mvRef_x×BlockDistanceFw)−1) ≫14；

——如果(mvRef_y+delt1)小于 0，则 mvFw_y=−((Floor(16384/BlockDistanceRef)×

(1－(mvRef_y＋delt1)×BlockDistanceFw)－1)≫14)－delt2；否则 mvFw_y＝((Floor(16384/BlockDistanceRef)×(1＋(mvRef_y＋delt1)×BlockDistanceFw)－1)≫14)－delt2。delt1、delt2 的值见 5.3.5.2.2，mvRef 相当于 5.3.5.2.2 中的 mvX，mvFw 相当于 5.3.5.2.2 中的 MVX。

2) 当前块的后向运动矢量 mvBw(mvBw_x，mvBw_y)为：

——如果 mvRef_x 小于 0，则 mvBw_x＝(Floor(16384/BlockDistanceRef)×(1－mvRef_x×BlockDistanceBw)－1)≫14；否则 mvBw_x＝－((Floor(16384/BlockDistanceRef)×(1＋mvRef_x×BlockDistanceBw)－1)≫14)；

——如果(mvRef_y＋delt1)小于 0，则 mvBw_y＝((Floor(16384/BlockDistanceRef)×(1－(mvRef_y＋delt1)×BlockDistanceBw)－1)≫14)－delt2；否则 mvBw_y＝－((Floor(16384/BlockDistanceRef)×(1＋(mvRef_y＋delt1)×BlockDistanceBw)－1)≫14)－delt2。delt1、delt2 的值见 5.3.5.2.2，mvRef 相当于 5.3.5.2.2 中的 mvX，mvBw 相当于 5.3.5.2.2 中的 MVX。

c) 如果当前块类型为对称模式，运动矢量按照下面定义的方法导出：

前向参考索引由比特流中的 mb_reference_index 解码得到。如果 field_pic_flag 等于 0，则后向参考索引与前向参考索引相等；否则，后向参考索引等于 1 减去前向参考索引。对称模式块的前向运动矢量 mvFw 按照 5.3.5.2.3 的方法得到，当前块的后向运动矢量 mvBw(mvBw_x，mvBw_y)为：

mvBw_x＝－((mvFw_x×BlockDistanceBw×Floor(512/BlockDistanceFw)＋256)≫9)

mvBw_y＝－(((mvFw_y＋delt1)×BlockDistanceBw×Floor(512/BlockDistanceFw)＋256)≫9)－delt2

其中 delt1、delt2 的值见 5.3.5.2.2，mvFw 相当于 5.3.5.2.2 中的 mvX，mvBw 相当于 5.3.5.2.2中的 MVX。

5.3.5.3 参考样点的导出过程

5.3.5.3.1 亮度样点插值过程

首先对亮度进行二分之一样点和四分之一样点的插值，然后根据运动矢量得到相应的参考样点。如果插值过程中所参考的整数样点在参考图像外，应通过参考图像的边界扩展获得。

图 20 给出了参考图像中整数样点、二分之一样点和四分之一样点的位置，其中用大写字母标记的为整数样点位置，用小写字母标记的为二分之一和四分之一样点位置。

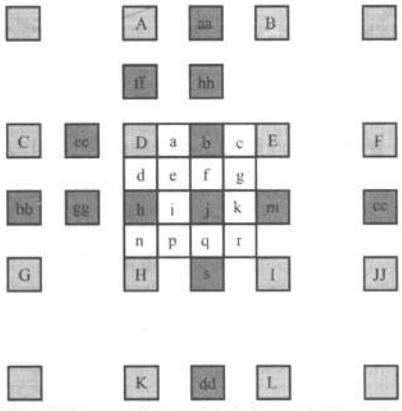


图 20 整数样点、二分之一样点和四分之一样点的位置

二分之一样点位置的预测值通过 4 抽头滤波器 $F_1(-1,5,5,-1)$ 计算得到。四分之一样点位置的预测值通过 4 抽头滤波器 $F_2(1,7,7,1)$ 计算得到。

二分之一样点的计算过程如下：

a) 二分之一样点 b：首先用 F_1 对水平方向上最近的 4 个整数样点滤波，得到中间值 $b'=(-C+5D+5E-F)$；最终的预测值 $b=Clip1((b'+4)\gg3)$；

b) 二分之一样点 h：首先用 F_1 对垂直方向上最近的 4 个整数样点滤波，得到中间值 $h'=(-A+5D+5H-K)$；最终的预测值 $h=Clip1((h'+4)\gg3)$；

c) 二分之一样点 j：首先用 F_1 在水平或垂直方向上对最近的 4 个二分之一样点中间值滤波，得到中间值 $j'=(-bb'+5h'+5m'-cc')$，或者 $j'=(-aa'+5b'+5s'-dd')$。其中 aa′，dd′和 s′为相应位置二分之一样点中间值(用 F_1 在水平方向滤波得到)，bb′，cc′和 m′为相应位置二分之一样点中间值(用 F_1 在垂直方向滤波得到)。最终的预测值 $j=Clip1((j'+32)\gg6)$，采用水平方向或垂直方向滤波得到的值相同。

四分之一样点的计算过程如下：

a) 四分之一样点 a：首先用 F_2 在水平方向上对 ee′，D′，b′和 E′四个值滤波，得到中间值 $a'=(ee'+7D'+7b'+E')$；最终的预测值 $a=Clip1((a'+64)\gg7)$。其中 ee′和 b′为相应位置二分之一样点中间值，D′和 E′为相应位置整数样点放大 8 倍的值。四分之一样点 c 的插值过程与 a 的插值过程相同；

b) 四分之一样点 d：首先用 F_2 在垂直方向上对 ff′，D′，h′和 H′四个值滤波，得到中间值 $d'=(ff'+7D'+7h'+H')$；最终的预测值 $d=Clip1((d'+64)\gg7)$。其中 ff′和 h′为相应位置二分之一样点中间值，D′和 H′为相应位置整数样点放大 8 倍的值。四分之一样点 n 的插值过程与 d 的插值过程相同；

c) 四分之一样点 i：首先用 F_2 在水平方向上对 gg′，h″，j′和 m″四个值滤波，得到中间值 $i'=(gg'+7h''+7j'+m'')$；最终的预测值 $i=Clip1((i'+512)\gg10)$。其中 gg′和 j′为相应位置二分之一中间值，h″和 m″为相应位置二分之一样点中间值放大 8 倍的值。四分之一样点 k 的插值过程与 i 的插值过程相同；

d) 四分之一样点 f：首先用 F_2 在垂直方向上对 hh′，b″，j′和 s″四个值滤波，得到中间值 $f'=(hh'+7b''+7j'+s'')$；最终的预测值 $f=Clip1((f'+512)\gg10)$。其中 hh′和 j′为相应位置二分之一中间值，b″和 s″为相应位置二分之一样点中间值放大 8 倍的值。四分之一样点 q 的插值过程与 f 的插值过程相同；

e) 四分之一样点 e，g，p 和 r：

1) $e=Clip1((D''+j'+64)\gg7)$；

2) $g=Clip1((E''+j'+64)\gg7)$；

3) $p=Clip1((H''+j'+64)\gg7)$；

4) $r=Clip1((I''+j'+64)\gg7)$。

其中 D″，E″，H″和 I″为相应位置整数样点放大 64 倍的值，j′为相应位置二分之一样点中间值。

图 20 中当前亮度块 E 的左上角整样点位置为(x，y)，预测样点矩阵的元素 predMatrix[x，y]根据表 42 赋值，表 42 中 xFracL 等于 mvE 的水平分量 mvE_x & 3，yFracL 等于 mvE 的垂直分量 mvE_y & 3。

表 42 预测样点矩阵元素

xFracL	0	0	0	0	1	1	1	1	2	2	2	2	3	3	3	3	0
yFracL	0	1	2	3	0	1	2	3	0	1	2	3	0	1	2	3	0
predMatrix[x,y]	D	d	h	n	a	e	i	p	b	f	j	q	c	g	k	r	D

5.3.5.3.2 色度样点插值过程

色度样点插值使用与对应亮度块的运动矢量 mvE(mvE 的水平分量为 mvE_x，垂直分量为 mvE_y)对

应的运动矢量 mvC。mvC 的水平分量为 mvC_x，垂直分量为 mvC_y，单位为八分之一样点，mvC 的范围应符合 5.3.3.4 的规定。如果 chroma_format_idc 等于 1，则 mvC_x 的值等于 mvE_x，mvC_y 的值等于 mvE_y；如果 chroma_format_idc 等于 2，则 mvC_x 的值等于 mvE_x，mvC_y 的值等于 mvE _y×2。色度样点插值如图 21 所示，其中 A，B，C，D 为距离被插值样点最近的整数样点值，dx 与 dy 分别为预测样点与 A 的水平和垂直距离，dx 等于 mvC_x & 7，dy 等于 mvC_y & 7。

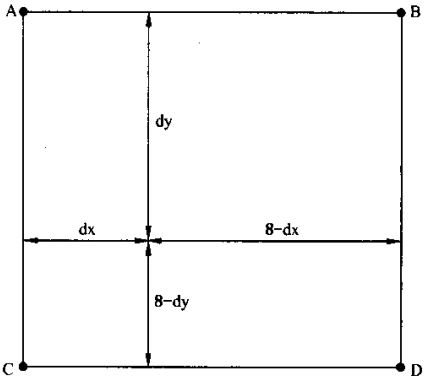


图 21　八分之一色度插值

预测样点矩阵的元素 predMatrix[x,y]根据下式计算：

predMatrix[x,y]=Clip1(((8−dx)×(8−dy)×A+dx×(8−dy)×B+(8−dx)×(dy×C+dx×dy×D+32)≫6)

5.3.5.3.3　加权预测

如果条带层或宏块层语法指示宏块使用了加权预测，在如 5.3.5.3.1 和 5.3.5.3.2 得到预测样点矩阵后调用本过程。

亮度样点加权预测如下：

predMatrix[x,y]=Clip1(((predMatrix[x,y]×luma_scale+16)≫5)+luma_shift)

色度样点加权预测如下：

predMatrix[x,y]=Clip1(((predMatrix[x,y]×chroma_scale+16)≫5)+chroma_shift)

5.3.6　变换系数解码过程以及图像重建过程

5.3.6.1　概述

本条所述过程在去块滤波过程之前进行，包括块解码，逆扫描，反量化，反变换和重建。

块解码的输入为块中连续的语法元素 trans_coefficient，输出为数组 QuantCoeffArray。

逆扫描的输入为由块解码生成的数组 QuantCoeffArray，输出为量化系数矩阵 QuantCoeffMatrix。

反量化的输入为量化系数矩阵 QuantCoeffMatrix，当前宏块的量化参数 QPY，输出为反量化后的变换系数矩阵 CoeffMatrix。

反变换的输入为 4×4 或 8×8 变换系数矩阵 CoeffMatrix，输出为 4×4 或 8×4 残差样点矩阵 ResidueMatrix。

重建过程的输入为残差样点矩阵 ResidueMatrix 和预测样点矩阵 predMatrix，输出为重建样点矩阵 RecMatrix。

5.3.6.2　块解码

利用 trans_coefficient 生成 Run 数组和 Level 数组的过程如下：

——如果 entropy_coding_mode_flag 等于 0，

a)　Run 数组和 Level 数组的地址索引等于 0；

b)　解析 trans_coefficient，方法参见 5.4.2.3；

c)　如果 trans_coefficient 小于 59：

1) 如果 CurrentVLCTable 中包含此 trans_coefficient 值的索引项，则以 trans_coefficient 为索引查 CurrentVLCTable，得到 Level 值和 Run 值，把它们分别放入 Level 数组和 Run 数组；
2) 如果 CurrentVLCTable 中不包含此 trans_coefficient 值的索引项，则以(trans_coefficient-1)为索引查 CurrentVLCTable，得到 Level 值和 Run 值，并将 Level 值符号取反后放入 Level 数组，将 Run 值放入 Run 数组；
3) Run 数组和 Level 数组的地址加 1，更新 CurrentVLCTable，解析下一个 trans_coefficient 及 Level 值和 Run 值；

d) 如果 trans_coefficient 大于或等于 59：
1) run 等于 Floor((trans_coefficient-59)/2)，将 run 放入 Run 数组；
2) 由 CurrentVLCTable 查表 F.20 得到 MaxRun，如果 run 大于 MaxRun，RefAbsLevel 等于 1；否则以 run 为索引查 CurrentVLCTable 得到 RefAbsLevel；
3) 如果 trans_coefficient 是奇数，将-(RefAbsLevel+escape_level_diff)放入 Level 数组；否则，将(RefAbsLevel+escape_level_diff)放入 Level 数组；
4) Run 数组和 Level 数组的地址加 1，更新 CurrentVLCTable，解析下一个 trans_coefficient 及 Level 值和 Run 值；

e) 如果 trans_coefficient 等于 EOB，Run 数组和 Level 数组的解码结束。

——如果 *entropy_coding_mode_flag* 等于 1，

a) Run 数组和 Level 数组的地址索引等于 0；
b) 通过解析 trans_coefficient 得到 AbsLevel，如果为该块的第一个 AbsLevel，则 AbsLevel=AbsLevel+1；
c) 如果 AbsLevel 不等于 0：
1) level 等于 AbsLevel；
2) 通过解析 trans_coefficient 得到 coeffSign，如果 coeffSign 为 0，将 level 放入 Level 数组；如果 coeffSign 为 1，将-level 放入 Level 数组；
3) 通过解析 trans_coefficient 得到 RunVal，将 RunVal 放入 Run 数组；
4) Level 数组和 Run 数组的地址加 1，解析下一个 trans_coefficient；
d) 如果 AbsLevel 等于 0，Run 数组和 Level 数组的解码结束(即 trans_coefficient 等于 EOB)。

由 Level 数组和 Run 数组生成数组 QuantCoeffArray 的步骤如下：

——将 QuantCoeffArray 数组的所有元素初始化为 0；

——将非 0 量化系数的值赋给 QuantCoeffArray 数组中相应的元素。定义 j 和 coeffNum，令 j 等于 Run 数组或 Level 数组中最后一个赋值元素的地址索引，*coeffNum* 等于−1，并有，

```
while(j>=0)
{
    coeffNum+=(Run[j]+1)
    QuantCoeffArray[coeffNum]=Level[j]
j—
}
```

5.3.6.3 逆扫描

设 QuantCoeffArray 数组中某元素的地址为 k，通过逆扫描找到值为 k 的单元对应的列号 i 和行号 j，然后将 QuantCoeffArray[k]赋给 QuantCoeffMatrix[i,j]。根据帧场的不同，块大小的不同，扫描方式定义如图 22。

j \ i	0	1	2	3	4	5	6	7
0	0	1	5	6	14	15	27	28
1	2	4	7	13	16	26	29	42
2	3	8	12	17	25	30	41	43
3	9	11	18	24	31	40	44	53
4	10	19	23	32	39	45	52	54
5	20	22	33	38	46	51	55	60
6	21	34	37	47	50	56	59	61
7	35	36	48	49	57	58	62	63

a) 8×8 帧扫描

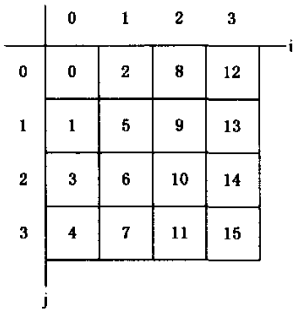
j \ i	0	1	2	3	4	5	6	7
0	0	3	11	16	22	32	38	55
1	1	6	12	20	25	33	42	57
2	2	7	15	21	28	37	43	58
3	4	10	19	27	31	39	47	59
4	5	14	24	30	36	44	50	60
5	8	17	26	35	41	48	52	61
6	9	18	29	40	46	51	54	62
7	13	23	34	45	49	53	56	63

b) 8×8 场扫描

j \ i	0	1	2	3
0	0	1	5	6
1	2	4	7	12
2	3	8	11	13
3	9	10	14	15

c) 4×4 帧扫描

图 22　8×8 块和 4×4 块的逆扫描



d) 4×4 场扫描

图 22（续）

5.3.6.4 系数反量化

5.3.6.4.1 量化参数

亮度量化参数 QPY 的取值范围应在 0～(55＋(BitDepthY－8)×7)。

亮度块的量化参数 QPY 由码流中计算得到(见 5.2.4.7)。色度块的量化参数 QPC 以 QPY 为索引查表 43 得到。在表 43 中，QP1 ＝ CLIP3(－(BitDepthY－8)×7,55,QPY－(BitDepthY－8)×7)，QPC＝QP2＋(BitDepthC－8)×7。

表 43 色度块的 QPC 与 QPY 的映射关系

QP1	<31	31	32	33	34	35	36	37	38	39	40	41	42
QP2	QP1	31	32	33	33	34	35	35	36	36	37	37	38
QP1	43	44	45	46	47	48	49	50	51	52	53	54	55
QP2	38	39	39	39	40	40	40	41	41	41	42	42	42

5.3.6.4.2 4×4 反量化

本条定义根据量化参数 QP(QPY 或 QPC)将 4×4 量化系数矩阵 QuantCoeffMatrix 转换为 4×4 变换系数矩阵 CoeffMatrix 的过程。其中，量化系数的取值应在$-2^{2+\mathrm{BitDepth}}$～$(2^{2+\mathrm{BitDepth}}-1)$范围内。

变换系数矩阵 CoeffMatrix 的元素 w_{ji} 由下式得到：

$$w_{ji}=\mathrm{QuantCoeffMatrix}[i,j]\times \mathrm{iqcoeff4}[\mathrm{QP}\%7,i,j]\ll \mathrm{Floor}(\mathrm{QP}/7)$$

其中，i,j ＝0...3。iqcoeff4 见表 44。

表 44 iqcoeff4

QP%7	i	j			
		0	1	2	3
0	0	16	16	16	16
	1	16	17	16	17
	2	16	16	16	16
	3	16	17	16	17
1	0	18	18	18	18
	1	18	19	18	19
	2	18	18	18	18
	3	18	19	18	19

表 44（续）

QP%7	i	j			
		0	1	2	3
2	0	20	21	20	21
	1	21	22	21	22
	2	20	21	20	21
	3	21	22	21	22
3	0	22	23	22	23
	1	23	24	23	24
	2	22	23	22	23
	3	23	24	23	24
4	0	26	27	26	27
	1	27	28	27	28
	2	26	27	26	27
	3	27	28	27	28
5	0	28	29	28	29
	1	29	30	29	30
	2	28	29	28	29
	3	29	30	29	30
6	0	30	31	30	31
	1	31	33	31	33
	2	30	31	30	31
	3	31	33	31	33

从符合本标准的比特流中解码得到的变换系数的取值应在$-2^{6+\mathrm{BitDepth}}$～$(2^{6+\mathrm{BitDepth}}-1)$范围内。

5.3.6.4.3 8×8 反量化

本条定义根据量化参数 QP(QPY 或 QPC)将 8×8 量化系数矩阵 QuantCoeffMatrix 转换为 8×8 变换系数矩阵 CoeffMatrix 的过程。其中，量化系数的取值应在$-2^{3+\mathrm{BitDepth}}$～$(2^{3+\mathrm{BitDepth}}-1)$范围内。

变换系数矩阵 CoeffMatrix 的元素 w_{ji} 由下式得到：

$$w_{ji}=\mathrm{QuantCoeffMatrix}[i,j]\times \mathrm{iqcoeff8}[\mathrm{QP}\%7,i\&3,j\&3]\ll \mathrm{Floor}(\mathrm{QP}/7)$$

其中，i,j=0...7。iqcoeff8 见表 45。

表 45 iqcoeff8

QP%7	i&3	j&3			
		0	1	2	3
0	0	32	30	33	30
	1	30	28	31	28
	2	33	31	35	31
	3	30	28	31	28

表 45（续）

QP%7	i&3	j&3			
		0	1	2	3
1	0	36	33	37	33
	1	33	31	35	31
	2	37	35	39	35
	3	33	31	35	31
2	0	40	37	42	37
	1	37	35	39	35
	2	42	39	44	39
	3	37	35	39	35
3	0	44	41	46	41
	1	41	39	43	39
	2	46	43	48	43
	3	41	39	43	39
4	0	52	49	54	49
	1	49	46	51	46
	2	54	51	57	51
	3	49	46	51	46
5	0	56	52	58	52
	1	52	49	55	49
	2	58	55	61	55
	3	52	49	55	49
6	0	60	56	63	56
	1	56	53	59	53
	2	63	59	66	59
	3	56	53	59	53

从符合本标准的比特流中解码得到的变换系数的取值应在$-2^{8+\mathrm{BitDepth}}$～$(2^{8+\mathrm{BitDepth}}-1)$范围内。

5.3.6.5 4×4 反变换

4×4 反变换的步骤如下：

a) 对变换系数矩阵进行如下水平反变换：

$H'=CoeffMatrix\times T_4^T$

其中，T_4 为 4×4 反变换矩阵，T_4^T 为 T_4 的转置矩阵，H'表示水平反变换后的中间结果。

$$T_4=\begin{bmatrix}4 & 5 & 4 & 2\\ 4 & 2 & -4 & -5\\ 4 & -2 & -4 & 5\\ 4 & -5 & 4 & -2\end{bmatrix}$$

b) 对矩阵 H'进行如下垂直反变换：

$H=T_4\times H'$

其中，H 表示反变换后的 4×4 矩阵。从符合本标准的比特流中解码得到的 H 矩阵元素取值范围应为$-2^{10+BitDepth}\sim(2^{10+BitDepth}-1)$。

c) 残差样点矩阵 ResidueMatrix 的元素 r_{ji}计算如下：

$r_{ji}=(h_{ji}+2^9)\gg10(i,j=0...3)$

其中 h_{ji}为 H 矩阵的元素。

5.3.6.6 8×8 反变换

8×8 反变换的步骤如下：

a) 对变换系数矩阵进行如下水平反变换：

$H'=CoeffMatrix\times T_8^T\gg3$

其中，T_8 为 8×8 反变换矩阵，T_8^T 为 T_8 的转置矩阵，H'表示水平反变换后的中间结果。

$$T_8=\begin{bmatrix}8 & 12 & 10 & 10 & 8 & 6 & 4 & 3\\ 8 & 10 & 4 & -3 & -8 & -12 & -10 & -6\\ 8 & 6 & -4 & -12 & -8 & 3 & 10 & 10\\ 8 & 3 & -10 & -6 & 8 & 10 & -4 & -12\\ 8 & -3 & -10 & 6 & 8 & -10 & -4 & 12\\ 8 & -6 & -4 & 12 & -8 & -3 & 10 & -10\\ 8 & -10 & 4 & 3 & -8 & 12 & -10 & 6\\ 8 & -12 & 10 & -10 & 8 & -6 & 4 & -3\end{bmatrix}$$

b) 对矩阵 H'进行如下垂直反变换：

$H=T_8\times H'\gg3$

其中，H 表示反变换后的 8×8 矩阵。从符合本标准的比特流中解码得到的 H 矩阵元素取值范围应为$-2^{8+BitDepth}\sim(2^{8+BitDepth}-1)$。

c) 残差样点矩阵 ResidueMatrix 的元素 r_{ji}计算如下：

$r_{ji}=(h_{ji}+2^7)\gg8\quad(i,j=0...7)$

其中 h_{ji}为 H 矩阵的元素。

5.3.6.7 重建

本过程的输入为残差样值矩阵 ResidueMatrix 和预测样点矩阵 predMatrix，输出为重建样点矩阵 RecMatrix。

a) 如果当前块为 I 或 P 宏块类型，重建样点矩阵 RecMatrix 计算如下：

RecMatrix[x,y]=Clip1(predMatrix[x,y]+ResidueMatrix[x,y])

b) 如果当前块为 B 宏块类型，并且有两个运动向量，重建样点矩阵 RecMatrix 计算如下：

RecMatrix[x, y] = Clip1 (((predMatrixFw[x, y] + predMatrixBw[x, y] + 1) ≫ 1) + ResidueMatrix[x,y])

c) 如果当前块为 B 宏块类型，并且只有前向运动向量，重建样点矩阵 RecMatrix 计算如下：

RecMatrix[x,y]=Clip1(predMatrixFw[x,y]+ResidueMatrix[x,y])

d) 如果当前块为B宏块类型，并且只有后向运动向量，重建样点矩阵RecMatrix计算如下：RecMatrix[x,y]=Clip1(predMatrixBw[x,y]+ResidueMatrix[x,y])，其中predMatrixFw为前向参考图像的预测样点矩阵，predMatrixBw为后向参考图像的预测样点矩阵。

e) 如果解码图像为非IDR图像，同时roi_flag等于1并且non_roi_skip_flag等于1，在图像的所有条带都解码后，背景区域的宏块的重建样点矩阵的取值等于显示顺序上位于当前解码图像之前且距当前解码图像最近的一个已解码的non_roi_skip_flag等于0的参考图像中相同位置上宏块的样点矩阵。

5.3.7 去块效应滤波过程

5.3.7.1 进行滤波的边界及滤波的顺序

如果loop_filter_disable_flag等于0，在位于去块效应滤波过程之前的图像重建过程完成以后，对整幅重建图像调用去块效应滤波过程。去块效应滤波过程以宏块为单位对图像中的所有的宏块按照光栅扫描的顺序进行。宏块的所有边界都应进行滤波，除非满足以下情况之一：

a) 该边界为图像边界；

b) 该边界两边的块属于不同的条带；

c) 解码图像为非IDR图像并且roi_flag等于1，同时该边界所属宏块不属于任一ROI并且non_roi_skip_flag等于1；

d) 解码图像为增强层非IDR图像并且roi_flag等于1，同时该边界所属宏块不属于任一ROI并且scalable_non_roi_skip_flag等于1。

此处宏块的所有边界定义为宏块内各个8×8块和(或者)4×4子块之间的边界，以及当前宏块与上边及左边相邻宏块之间的边界。

注：因为在图像重建过程完成以后才开始对整幅重建图像进行去块效应滤波过程，所以宏块以及位于当前宏块左边、上边的宏块的解码样点通常都可用。

亮度和色度分量分别进行滤波。对于每个宏块以及每个分量，先滤波纵向的边界，从宏块的左侧的边界开始，按照从左到右的顺序进行处理，然后滤波横向的边界，从宏块的上部边界开始，按照从上到下的顺序进行处理。

图23显示了4∶2∶0格式下一个宏块需要滤波的边界。如果宏块类型为I_4×4，那么实线和虚线边界都需要滤波。如果为其他类型宏块，只需要对实线边界进行滤波。先滤波纵向边界，再滤波横向边界。

如果图像格式为4∶0∶0格式，只需要对亮度边界进行滤波，没有色度边界需要滤波。

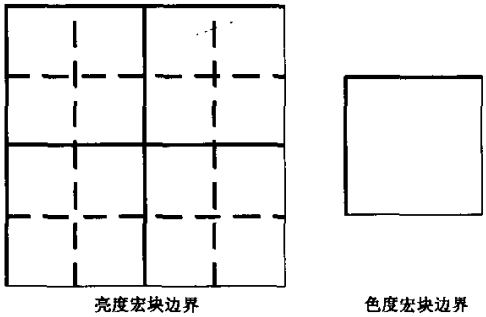


图23 4∶2∶0格式下需要滤波的宏块边界

帧内预测使用去块效应滤波前的重建图像样点值。

注：当前宏块的上边或者左边的宏块的样点值可能在前面的去块滤波过程中被修改，当前宏块的去块滤波的输入为这些修改后的样点值，并且当前宏块去块滤波可能进一步修改这些样点值。当前宏块垂直边界滤波过程中修改的样点值作为水平边界滤波过程的输入。

每个8×8亮度块之间的边界有一个“边界强度”Bs，色度块的边界强度用对应位置亮度块边界的

Bs 代替，如图 23 所示。如果 Bs 等于 0，则不要对边界滤波，否则根据局部样点值的特性和 Bs 值对边界滤波。

5.3.7.2 边界滤波强度的推导过程

根据宏块类型、宏块中 8×8 块的运动矢量等信息，按如下方法计算 Bs 的值：

a) 如果边界两边的两个块中至少有一个属于 I 宏块并且边界为当前宏块与相邻宏块之间的边界，则 Bs 等于 3；

b) 否则，如果当前宏块为 I 宏块并且边界为宏块内 8×8 块或 4×4 子块之间的边界，则 Bs 等于 2；

c) 否则，边界两边的两个块中至少有一个存在非零残差数据，则 Bs 等于 2；

d) 否则，如果为 P 条带并且满足以下两个条件中的任一个，则 Bs 等于 1；如果为 P 条带并且以下两个条件都不满足，则 Bs 等于 0；

1) 两个块的参考图像不同；

2) 两个块的参考图像相同，且两个块的运动矢量分量中任一分量的差值大于或等于一个整样点。

e) 否则，按以下方法计算 Bs 的值：

1) 如果两个块 p 和 q 的前向索引参考值和后向索引参考值分别相等：
以下两个条件中任一个成立，则 Bs 等于 1：
——两个块的前向运动矢量分量中任一个分量的差值大于或等于一个整样点；
——两个块的后向运动矢量分量中任一个分量的差值大于或等于一个整样点。
否则，Bs 等于 0；

2) 否则，Bs 等于 1。

5.3.7.3 块边界阈值的推导过程

图 24 表示块 p 和块 q 在水平或垂直边界两侧的 8 个样点(边界用黑色粗线表示)。用 P0、P1、Q0 和 Q1 分别表示 p0、p1、q0 和 q1 滤波后的样点值并有如下定义：

$ap=Abs(p2-p0)$，$aq=Abs(q2-q0)$

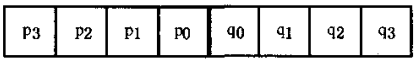


图 24 块水平或垂直边界两侧的样点

如果下式为真，对边界样点滤波：

$Bs!=0\ \&\&\ Abs(p0-q0)<\alpha\ \&\&\ Abs(p1-p0)<\beta\ \&\&\ Abs(q1-q0)<\beta$

其中 α 和 β 为块边界阈值。

对于亮度块，可以根据两个块的 QPY 平均值 QPav，以及 AlphaCOffset 和 BetaOffset 计算查表索引 IndexA 和 IndexB。

两个块的 QPY 平均值 QPav 为：

$QPav=(QPYp+QPYq+1)\gg 1$

索引 IndexA 和 IndexB 为：

$IndexA=Clip3(0,55,QPav+AlphaCOffset-(BitDepthY-8)\times 7)$

$IndexB=Clip3(0,55,QPav+BetaOffset-(BitDepthY-8)\times 7)$

根据索引 IndexA 和 IndexB 与门限变量 α'和 β'间的对应关系，由表 46 得到 α'、β'的取值，根据 IndexA查表得到 α'，根据 IndexB 查表得到 β'。通过下述方式可以得到相应的块边界阈值 α 和 β。

$\alpha=\alpha'\times(1\ll(BitDepthY-8))$

$\beta=\beta'\times(1\ll(BitDepthY-8))$

对于色度块，计算方法和亮度块相同，但是 QPY 和 BitDepthY 应替换为 QPC 以及 BitDepthC。

表 46　门限变量 α'和 β'与 IndexA 和 IndexB 的关系

索引	α'	β'	索引	α'	β'	索引	α'	β'	索引	α'	β'
0	0	0	14	0	0	28	20	7	42	101	14
1	0	0	15	0	0	29	22	7	43	113	14
2	0	0	16	4	2	30	25	8	44	127	15
3	0	0	17	4	2	31	28	8	45	144	15
4	0	0	18	5	2	32	32	9	46	162	16
5	0	0	19	6	3	33	36	9	47	182	16
6	0	0	20	7	3	34	40	10	48	203	17
7	0	0	21	8	3	35	45	10	49	226	17
8	0	0	22	9	3	36	50	11	50	255	18
9	0	0	23	10	4	37	56	11	51	255	18
10	0	0	24	12	4	38	63	12	52	255	19
11	0	0	25	13	4	39	71	12	53	255	19
12	0	0	26	15	6	40	80	13	54	255	20
13	0	0	27	17	6	41	90	13	55	255	20

5.3.7.4　Bs 等于 3 时的边界滤波过程

对亮度块边界两边的样点 p0、p1、p2、q0、q1 和 q2 的滤波过程如下：

```
if(ap<β && Abs(p0−q0)<((α≫2)+2)){
  P0=(p2+p1+3×p0+2×q0+q1+4)≫3
  P1=(p2+p1+p0+q0+2)≫2
  P2=(2×p3+2×p2+2×p1+p0+q0+4)≫3
}
else
  P0=(2×p1+p0+q1+2)≫2
if(aq<β && Abs(p0−q0)<((α≫2)+2)){
  Q0=(q2+q1+3×q0+2×p0+p1+4)≫3
  Q1=(q2+q1+q0+p0+2)≫2
  Q2=(2×q3+2×q2+2×q1+q0+p0+4)≫3
}
else
  Q0=(2×q1+q0+p1+2)≫2
```

其中 P0 和 Q0 分别为 p0 和 q0 滤波后的值。如果在上面的滤波过程中 P2(Q2)、P1(Q1)不被赋值，则不对 p2(q2)、p1(q1)滤波；否则，P2(Q2)和 P1(Q1)为 p2(q2)和 p1(q1)滤波后的值。

对色度块边界两边的样点 p0 和 q0 采用同样的方法滤波，不对 p2、p1 和 q2、q1 滤波。

上述滤波过程中，ap 和 aq 的定义见 5.3.7.3。

4.3.7.5　Bs 等于 2 或 1 时的边界滤波过程

边界滤波强度 Bs 的值为 2 或 1 时，对 p0 和 q0 滤波的计算过程如下(P0 和 Q0 分别为 p0 和 q0 滤波后的值)：

a)　如果为亮度边界：

Tc=C+((ap<β)? 1 : 0)+((aq<β)? 1 : 0)

delta=Clip3(−Tc,Tc,(((q0−p0)×4+(p1−q1)+4)≫3))

P0=Clip1Y(p0+delta)

Q0=Clip1Y(q0−delta)

b) 如果为色度边界：

Tc=C+1

delta=Clip3(−Tc,Tc,(((q0−p0)×4+(p1−q1)+4)≫3))

P0=Clip1C(p0+delta)

Q0=Clip1C(q0−delta)

对 p1 和 q1 滤波的计算过程如下：

a) 如果为色度边界，不对 p1 和 q1 滤波。

b) 对亮度边界，如果 ap 小于 β，则对 p1 滤波，滤波后的值为

P1=Clip1Y(p1+Clip3(−C,C,((p3+p2+p0+q0−4×p1+2)≫2)

c) 对亮度边界，如果 aq 小于 β，则对 q1 滤波，滤波后的值为

Q1=Clip1Y(q1+Clip3(−C,C,((q3+q2+q0+p0−4×q1+2)≫2)

上述滤波过程中，ap 和 aq 的定义见 5.3.7.3，C 称为滤波裁减参数。

对于亮度块，可以根据两个块的 QPY 平均值 QPav，以及 AlphaCOffset 计算查表索引 IndexC。

两个块的 QPY 平均值 QPav 为：

QPav=(QPYp+QPYq+1)≫1

索引 IndexC 为：

IndexC=Clip3(0,55,QPav+(bs−1)+AlphaCOffset−(BitDepthY−8)×7)

滤波裁减参数变量 C'与 IndexC 之间的关系见表 47。滤波裁减参数 C 的值由下式计算出：

C=C'×(1≪(BitDepthY−8))

对于色度块，计算方法和亮度块相同，但是 QPY 和 BitDepthY 应替换为 QPC 以及 BitDepthC。

表 47 滤波裁减参数变量 C'与 IndexC 的关系

索引	C'	索引	C'	索引	C'	索引	C'
0	0	14	0	28	1	42	3
1	0	15	0	29	1	43	3
2	0	16	1	30	2	44	3
3	0	17	1	31	2	45	4
4	0	18	1	32	2	46	4
5	0	19	1	33	2	47	4
6	0	20	1	34	2	48	5
7	0	21	1	35	2	49	5
8	0	22	1	36	2	50	5
9	0	23	1	37	2	51	6
10	0	24	1	38	3	52	6
11	0	25	1	39	3	53	6
12	0	26	1	40	3	54	7
13	0	27	1	41	3	55	7

5.3.8 可伸缩性视频编码(SVC)增强层条带中宏块的解码过程

5.3.8.1 概述

解码 nal_unit_type 的值为 3 和 4 的增强层条带 NAL 单元中的宏块时调用本过程。增强层解码图像帧的宽度和高度均为基本层解码图像帧的 2 倍。

增强层条带的解码过程同 5.3.3。

注：不特殊说明的情况下，解码增强层图像时应用增强层图像宽度和高度的参数替代基本层的对应参数。

5.3.8.2 层间对应的空间位置的计算

一个高分辨率的增强层图像中的亮度及色度样点的空间位置与位于其下的相邻图像层的低分辨率图像中亮度及色度样点的空间位置的对应关系如下所示：

$$Bx' = Ex \times 2$$
$$By' = Ey \times 2$$
$$Bx = Floor((Bx' + 2)/4)$$
$$By = Floor((By' + 2)/4)$$
$$Ex = Bx \times 2$$
$$Ey = By \times 2$$

其中 Ex,Ey 分别为高分辨率图像中亮度或色度样点的空间位置，Bx,By 分别为 Ex,Ey 在低分辨率图像中对应的空间位置，Bx′,By′分别为 Ex,Ey 在低分辨率图像中对应的四分之一样点精度的空间位置。

一个高分辨率的增强层图像中的亮度及色度块(2N×2N)在位于其下的相邻图像层的低分辨率图像中的对应块(N×N)由块左上角的坐标对应确定。

5.3.8.3 低分辨率层图像样点到对应位置的高分辨率层图像样点的插值计算

从低分辨率图像样点到对应位置的高分辨率层图像样点的插值计算方法如下：

首先进行水平方向插值，水平方向二分之一样点位置的样点值 $P(Bx_0 + \frac{1}{2})$ 通过 4 抽头滤波器(−1,9,9,−1)计算得到：

$$P(Bx_0 + \frac{1}{2}) = (-P(Bx_0 - 1) + 9 \times P(Bx_0) + 9 \times P(Bx_0 + 1) - P(Bx_0 + 2) + 8) \gg 4$$

然后进行垂直方向插值，垂直方向二分之一样点位置的样点值 $P(By_0 + \frac{1}{2})$ 通过 4 抽头滤波器(−1,9,9,−1)计算得到：

$$P(By_0 + \frac{1}{2}) = (-P(By_0 - 1) + 9 \times P(By_0) + 9 \times P(By_0 + 1) - P(By_0 + 2) + 8) \gg 4$$

如果插值所用的整数样点位置在图像边界外，则使用与之距离最近的整数样点位置的值替代。

5.3.8.4 增强层宏块的预测及图像重建过程

5.3.8.4.1 svc_mode_flag 等于 0 时的预测及图像重建过程

如果 svc_mode_flag 等于 0，增强层宏块的解码只使用增强层内的帧间或帧内预测。其帧内预测过程同 5.3.4，帧间预测过程同 5.3.5，重建过程同 5.3.6。

如果解码图像为非 IDR 图像，同时 roi_flag 等于 1 并且 scalable_non_roi_skip_flag 等于 1，在图像的所有条带都解码后，增强层图像中背景区域的宏块的重建样点矩阵的取值等于相邻低分辨率层的对应位置的块的样点矩阵经 5.3.8.3 所示方法插值后生成的样点矩阵。

5.3.8.4.2 *svc_mode_flag* 等于 1 时的预测及图像重建过程

如果 svc_mode_flag 等于 1，增强层宏块的解码过程中只使用跨层预测。本过程的输入为增强层码流解析生成的残差样点矩阵 EResMatrix，以及低分辨率解码图像的样点矩阵 LRecMatrix(经过去块滤波过程后)，输出为增强层重建样点矩阵 ERecMatrix。

增强层残差样点矩阵 EResMatrix 由码流解析生成的 QuantCoeffArray 经逆扫描，反量化及反变

换过程得到，具体过程同 5.3.6.3～5.3.6.6。

层间重建预测矩阵 InterRecMatrix 的计算方式如下：

a) 根据当前块左上角样点的位置(Ex0,Ey0)计算对应低分辨率层图像中的对应位置(Bx0,By0)，方法同 5.3.8.2；

b) 在 LRecMatrix 中得到左上角样点位置为(Bx0,By0)的 8×8 块的样点矩阵 LRec8×8；

c) 由 LRec8×8 插值得到 InterRecMatrix，方法同 5.3.8.3。

增强层 I,P 或 B 图像中相应宏块的重建样点矩阵 ERecMatrix 计算如下：

ERecMatrix[x,y]=Clip1(EResMatrix[x,y]+InterRecMatrix[x,y])

如果解码图像为非 IDR 图像，同时 roi_flag 等于 1 并且 scalable_non_roi_skip_flag 等于 1，在图像的所有条带都解码后，增强层图像中背景区域的宏块的重建样点矩阵的取值等于相邻低分辨率层的对应位置的块的样点矩阵经 5.3.8.3 所示方法插值后生成的样点矩阵。

5.4 解析过程

5.4.1 概述

本过程的输入为 RBSP 的比特流。本过程的输出为语法元素值。

对于 5.2.3.1～5.2.3.3 的语法表格中描述符等于 ue(v),me(v),se(v),ce(v)的语法元素，解析过程见 5.4.2。

对于 5.2.3.4～5.2.3.7 中的语法元素，如果 entropy_coding_mode_flag 等于 0，解析过程见 5.4.2；如果 entropy_coding_mode_flag 等于 1，解析过程见 5.4.3。

5.4.2 指数哥伦布编码的解析过程

5.4.2.1 k 阶指数哥伦布码

解析 k 阶指数哥伦布码时，首先从比特流的当前位置开始寻找第一个非零比特，并将找到的零比特个数记为 leadingZeroBits，然后根据 leadingZeroBits 计算 CodeNum。用伪代码描述如下：

```
leadingZeroBits=－1
for (b=0;! b;leadingZeroBits++)
    b= read_bits(1)
CodeNum=2^(leadingZeroBits+k)－2^k+read_bits(leadingZeroBits+k)
```

表 48 给出了 0 阶、1 阶、2 阶和 3 阶指数哥伦布码的结构。指数哥伦布码的比特串分为“前缀”和“后缀”两部分。前缀由 leadingZeroBits 个连续的‘0’和一个‘1’构成。后缀由 leadingZeroBits + k 个比特构成，即表中的 xi 串，i 的范围为从 0～(leadingZeroBits + k － 1)，每个 xi 的值为 0 或 1。

表 48 k 阶指数哥伦布码表

阶数	码字结构	CodeNum 取值范围
k=0	1	0
	01 x0	1…2
	001 x1 x0	3…6
	0001 x2 x1 x0	7…14
	……	……
k=1	1 x0	0…1
	01 x1 x0	2…5
	001 x2 x1 x0	6…13
	0001 x3 x2 x1 x0	14…29
	……	……

表 48（续）

阶数	码字结构	CodeNum 取值范围
k=2	1 x1 x0	0…3
	01 x2 x1 x0	4…11
	001 x3 x2 x1 x0	12…27
	0001 x4 x3 x2 x1 x0	28…59
	……	……
k=3	1 x2 x1 x0	0…7
	01 x3 x2 x1 x0	8…23
	001 x4 x3 x2 x1 x0	24…55
	0001 x5 x4 x3 x2 x1 x0	56…119
	……	……

5.4.2.2 **ue(v)，se(v)和 me(v)**

ue(v)，se(v)和 me(v)描述的语法元素使用 0 阶指数哥伦布码，其解析过程为：

ue(v)：语法元素的取值等于 CodeNum；

se(v)：语法元素的取值与 CodeNum 的映射关系见表 49；

me(v)：MbCBP、MbCBP422 及 cbp_4×4 的取值与 CodeNum 的映射关系分别见表 50 和表 51。

表 49 se(v)中 CodeNum 与语法元素取值的映射关系

CodeNum	语法元素值
0	0
1	1
2	−1
3	2
4	−2
5	3
6	−3
k	$(-1)^{k+1}\times \mathrm{Ceil}(k\div 2)$

表 50 4：2：0 格式下 MbCBP 与 CodeNum 的映射关系

CodeNum	MbCBP	
	帧内编码宏块	帧间编码宏块
0	63	0
1	15	15
2	31	63
3	47	31
4	0	16
5	14	32
6	13	47

表 50（续）

CodeNum	MbCBP	
	帧内编码宏块	帧间编码宏块
7	11	13
8	7	14
9	5	11
10	10	12
11	8	5
12	12	10
13	61	7
14	4	48
15	55	3
16	1	2
17	2	8
18	59	4
19	3	1
20	62	61
21	9	55
22	6	59
23	29	62
24	45	29
25	51	27
26	23	23
27	39	19
28	27	30
29	46	28
30	53	9
31	30	6
32	43	60
33	37	21
34	60	44
35	16	26
36	21	51
37	28	35
38	19	18
39	35	20
40	42	24

表 50（续）

CodeNum	MbCBP	
	帧内编码宏块	帧间编码宏块
41	26	53
42	44	17
43	32	37
44	58	39
45	24	45
46	20	58
47	17	43
48	18	42
49	48	46
50	22	36
51	33	33
52	25	34
53	49	40
54	40	52
55	36	49
56	34	50
57	50	56
58	52	25
59	54	22
60	41	54
61	56	57
62	38	41
63	57	38

表 51　4：0：0 MbCBP 和 4：2：2 MbCBP/MbCBP422 以及 cbp_4×4 与 CodeNum 的映射关系

CodeNum	MbCBP/MbCBP422/cbp_4×4	
	帧内编码宏块	帧间编码宏块
0	15	0
1	0	1
2	7	2
3	11	4
4	13	8
5	14	3
6	3	5
7	5	10

表 51（续）

CodeNum	MbCBP/MbCBP422/cbp_4×4	
	帧内编码宏块	帧间编码宏块
8	10	12
9	12	15
10	1	7
11	2	11
12	4	13
13	8	14
14	6	6
15	9	9

5.4.2.3 ce(v)

ce(v)描述的语法元素采用 0 阶、1 阶、2 阶或 3 阶指数哥伦布码进行解析，其阶数由下面的规则决定：

a) 帧内编码块亮度系数的 escape_level_diff 采用 1 阶指数哥伦布码；

b) 帧间编码块亮度系数的 escape_level_diff 采用 0 阶指数哥伦布码；

c) 色度系数的 escape_level_diff 采用 0 阶指数哥伦布码。

本标准定义了 19 个与 ce(v)有关的变长码表，即 VLC0_Intra，VLC1_Intra，VLC2_Intra，VLC3_Intra，VLC4_Intra，VLC5_Intra，VLC6_Intra，VLC0_Inter，VLC1_Inter，VLC2_Inter，VLC3_Inter，VLC4_Inter，VLC5_Inter，VLC6_Inter，VLC0_Chroma，VLC1_Chroma，VLC2_Chroma，VLC3_Chroma，VLC4_Chroma，见附录 F。不同的码表决定了 ce(v)所用的指数哥伦布码的阶数。其中 VLC0_Inter 采用 3 阶指数哥伦布码，VLC2_Chroma 及 VLC3_Chroma 采用 1 阶指数哥伦布码，VLC1_Chroma 及 VLC4_Chroma 采用 0 阶指数哥伦布码，其他码表均采用 2 阶指数哥伦布码。

本条中上述 19 个码表的选择按照以下规则进行：

编码块第一个解码量化系数的码表选择：

a) 帧内预测编码块的亮度系数，CurrentVLCTable = VLC0_Intra，见表 F.1；

b) 帧间预测编码块的亮度系数，CurrentVLCTable = VLC0_Inter，见表 F.8；

c) 色度系数，CurrentVLCTable = VLC0_Chroma，见表 F.15。

maxAbsLevel 等于 0。

absLevel 等于第一个解码量化系数值的绝对值。

其他解码量化系数的码表选择：

a) 对于帧内预测编码块的亮度系数，如果 absLevel 大于 maxAbsLevel，按如下方式进行码表切换：

1) 如果 absLevel 等于 1，CurrentVLCTable = VLC1_Intra，见表 F.2；

2) 如果 absLevel 等于 2，CurrentVLCTable = VLC2_Intra，见表 F.3；

3) 如果 absLevel 等于 3 或 4，CurrentVLCTable = VLC3_Intra，见表 F.4；

4) 如果 absLevel 等于 5、6 或 7，CurrentVLCTable = VLC4_Intra，见表 F.5；

5) 如果 absLevel 等于 8、9 或 10，CurrentVLCTable = VLC5_Intra，见表 F.6；

6) 如果 absLevel 大于 10，CurrentVLCTable = VLC6_Intra，见表 F.7。

b) 对于帧间预测编码块的亮度系数，如果 absLevel 大于 maxAbsLevel，按如下方式进行码表切换：

1) 如果 absLevel 等于 1,CurrentVLCTable = VLC1_Inter,见表 F.9;
2) 如果 absLevel 等于 2,CurrentVLCTable = VLC2_Inter,见表 F.10;
3) 如果 absLevel 等于 3,CurrentVLCTable = VLC3_Inter,见表 F.11;
4) 如果 absLevel 等于 4、5 或 6,CurrentVLCTable = VLC4_Inter,见表 F.12;
5) 如果 absLevel 等于 7、8 或 9,CurrentVLCTable = VLC5_Inter,见表 F.13;
6) 如果 absLevel 大于 9,CurrentVLCTable = VLC6_Inter,见表 F.14。

c) 对于色度块的系数,如果 absLevel 大于 maxAbsLevel,按如下方式进行码表切换:
1) 如果 absLevel 等于 1,CurrentVLCTable = VLC1_Chroma,见表 F.16;
2) 如果 absLevel 等于 2,CurrentVLCTable = VLC2_Chroma,见表 F.17;
3) 如果 absLevel 等于 3 或 4,CurrentVLCTable = VLC3_Chroma,见表 F.18;
4) 如果 absLevel 大于 4,CurrentVLCTable = VLC4_Chroma,见表 F.19。

d) 如果 absLevel 大于 maxAbsLevel,则 maxAbsLevel 等于 absLevel。

e) absLevel 等于当前解码量化系数值的绝对值。

ce(v)描述的语法元素解析过程如下:

语法元素 trans_coefficient 等于 CodeNum。如果 trans_coefficient 大于或等于 59,解析下一个 ce(v)语法元素,得到一个新的 CodeNum,escape_level_diff 等于 CodeNum。

5.4.3 条带数据的 CABAC 解析过程

5.4.3.1 概述

本过程的输入为先前已解析的语法元素值和解析语法元素值的请求。本过程的输出为语法元素值。

ae(v)描述的语法元素解析过程如下:

a) 对条带进行解析前,首先进行初始化,见 5.4.3.2;

b) 对语法元素做二值化,见 5.4.3.3;

c) 对二值化得到的二进制位串进行解析,见 5.4.3.4;
1) 二进制位串中每个二进制位的索引号为 binIdx,对应唯一的 ctxIdx,见 5.4.3.4.2;
2) 根据 ctxIdx 解析二进制位,见 5.4.3.4.3;
3) 完成一个二进制位的解析后,将得到的二进制位串与二值化过程得到的二进制位串集合进行比较。如果得到的二进制位串与集合中某个二进制位串相匹配,则输出相应语法元素值;否则继续解析二进制位。

ae(v)描述的语法元素解析过程用伪代码描述如下:

```
if(当前语法元素为条带的第一个语法元素){
    初始化所有上下文模型
    初始化算术码解码器
}
语法元素二值化
binIdx=-1
do{
    binIdx++
    得到与 binIdx 对应的 ctxIdx
    得到与 ctxIdx 对应的上下文模型 ctx
    根据 ctx 解析二进制位
}while((b0,... ,bbinIdx)不是语法元素的二进制位串)
输出语法元素值
```

5.4.3.2 初始化

5.4.3.2.1 初始化上下文模型

每个上下文模型 ctx 需要初始化三个变量 mps、cycno 和 lgPmps。mps 和 cycno 的值应初始化为 0，lgPmps 的值应初始化为 1023。

5.4.3.2.2 初始化算术码解码器

rS1、rT1、valueS 和 valueT 为用于算术码解码器的变量。rS1 的值应初始化为 0，rT1 的值应初始化为 0xFF。

valueS 和 valueT 的初始化过程用伪代码描述如下：

```
valueS = 0
    valueT = read_bits(9)
    while( ! ((valueT≫8)& 0x01) ){
        valueT=(valueT≪1)|read_bits(1)
        valueS++
}
valueT = valueT & 0xFF
```

5.4.3.3 二值化

各个语法元素的二值化过程如下：

a) mb_skip_flag、svc_mode_flag、P 条带的 mb_reference_index、I 条带和 P 条带的 mb_type 的二值化过程如下：

语法元素的值为 synElVal，synElVal 与二进制位串的关系见表 52。

表 52 synElVal 与二进制位串的关系

synElVal	二进制位串					
0	1					
1	0	1				
2	0	0	1			
3	0	0	0	1		
4	0	0	0	0	1	
5	0	0	0	0	0	1
…						
binIdx	0	1	2	3	4	5

b) B 条带的 mb_type 的二值化过程如下：

语法元素的值为 synElVal，synElVal 与二进制位串的关系见表 53。

表 53 synElVal 与二进制位串的关系

synElVal	二进制位串					
0	0					
1	1	1				
2	1	0	1			
3	1	0	0	1		
4	1	0	0	0	1	

表 53（续）

synElVal	二进制位串					
5	1	0	0	0	0	1
…						
binIdx	0	1	2	3	4	5

c） mb_part_type 的二值化过程如下：

语法元素的值为 synElVal，synElVal 与二进制位串的关系见表 54。

表 54 synElVal 与二进制位串的关系

synElVal	二进制位串	
0	0	0
1	0	1
2	1	0
3	1	1
binIdx	0	1

d） pred_mode_flag 的二值化过程如下：

语法元素的值为 synElVal。如果二进制位串为‘0’，synElVal 的值为 0。如果二进制位串为‘1’，synElVal 的值为 1。

e） intra_luma_pred_mode 的二值化过程如下：

语法元素的值为 synElVal，synElVal 与二进制位串的关系见表 55。

表 55 synElVal 与二进制位串的关系

synElVal	二进制位串		
0	1		
1	0	1	
2	0	0	1
3	0	0	0
binIdx	0	1	2

f） intra_chroma_pred_mode 的二值化过程如下：

语法元素的值为 synElVal，synElVal 与二进制位串的关系见表 56。

表 56 synElVal 与二进制位串的关系

synElVal	二进制位串		
0	0		
1	1	0	
2	1	1	0
3	1	1	1
binIdx	0	1	2

g) mv_diff_x 和 mv_diff_y 的二值化过程如下：

语法元素分为绝对值 mvdAbs 和符号位 mvdSign 两部分，先解析 mvdAbs，再解析 mvdSign。mvdAbs 的值为 synElVal，mvdSign 的解析过程见 5.4.3.4。synElVal 与二进制位串的关系见表 57 所示。如果 synElVal 的值大于或等于 9，二进制位串的前两位为'11'，后续位为 mvdAbs-9 对应的 3 阶指数哥伦布码(见表 57)。如果 mvdAbs 为 0，此时不应解析 mvdSign。mvdSign 的值为 0 表示 mv_diff_x 或 mv_diff_y 的值为 synElVal；mvdSign 的值为 1 表示其值为－synElVal。

表 57 synElVal 与二进制位串的关系

synElVal	二进制位串										
0	0										
1	1	0	1								
2	1	0	0	1							
3	1	0	0	0	1						
4	1	0	0	0	0	1					
5	1	0	0	0	0	0	1				
6	1	0	0	0	0	0	0	1			
7	1	0	0	0	0	0	0	0	1		
8	1	0	0	0	0	0	0	0	0		
9	1	1	0	0	0	0					
10	1	1	0	0	0	1					
11	1	1	0	0	1	0					
…											
16	1	1	0	1	1	1					
17	1	1	1	0	0	0	0	0			
18	1	1	1	0	0	0	0	1			
…											
binIdx	0	1	2	3	4	5	6	7	8	9	10

h) mb_qp_delta 的二值化过程如下：

语法元素先映射为无符号整数 CodeNum，映射关系同表 49，CodeNum 与二进制位串的关系见表 58。

表 58 CodeNum 与二进制位串的关系

CodeNum	二进制位串					
0	0					
1	1	0				
2	1	1	0			
3	1	1	1	0		
4	1	1	1	1	0	
5	1	1	1	1	1	0
…						
binIdx	0	1	2	3	4	5

i) coded_block_pattern 的二值化过程如下：

语法元素的值为 synElVal，如果 synElVal 小于 16，二进制位串为 synElVal 的四位前缀，后缀为‘0’；如果 synElVal 大于 16 并且小于或等于 31，二进制位串为 synElVal 的四位前缀，后续位为‘100’；如果 synElVal 大于 31 并且小于或等于 47，二进制位串为 synElVal 的四位前缀，后续位为‘101’；如果 synElVal 大于 47，二进制位串为 synElVal 的四位前缀，后续位为‘11’。synElVal 与二进制位串的关系见表 59。

表 59 synElVal 与二进制位串的关系

synElVal	二进制位串						
	四位前缀				后缀		
0	0	0	0	0	0		
1	1	0	0	0	0		
2	0	1	0	0	0		
3	1	1	0	0	0		
…							
15	1	1	1	1	0		
16	0	0	0	0	1	0	0
17	1	0	0	0	1	0	0
…							
31	1	1	1	1	1	0	0
32	0	0	0	0	1	0	1
33	1	0	0	0	1	0	1
…							
47	1	1	1	1	1	0	1
48	0	0	0	0	1	1	
49	1	0	0	0	1	1	
…							
63	1	1	1	1	1	1	
binIdx	0	1	2	3	4	5	6

j) trans_coefficient 的二值化过程如下：

语法元素分为 coeffLevel、coeffSign 和 coeffRun 三部分，先解析 coeffLevel，再解析 coeffSign，最后解析 coeffRun。AbsLevel 为解析 coeffLevel 得到的 synElVal。coeffSign 为一个二进制位‘0’或‘1’。RunVal 为解析 coeffRun 得到的 synElVal。synElVal 与二进制位串的关系见表 52。

k) weighting_prediction 的二值化过程如下：

语法元素的值为 synElVal。如果二进制位串为‘0’，synElVal 的值为 0。如果二进制位串为‘1’，synElVal 的值为 1。

l) B 条带的 mb_reference_index 的二值化过程如下：

语法元素的值为 synElVal。如果二进制位串为‘0’，synElVal 的值为 1。如果二进制位串为‘1’，synElVal 的值为 0。

5.4.3.4 二进制位串解析

5.4.3.4.1 概述

对每个二进制位，置 contextWeighting 的值为 0。如果二进制位为 mvdSign 或 coeffSign 或 weighting_prediction，或该二进制位属于 mv_diff_x 和 mv_diff_y 的 3 阶指数哥伦布码部分，则 BypassFlag 的值为 1；否则 BypassFlag 的值为 0。解析二进制位时，如果 BypassFlag 的值为 0，则先由 binIdx 确定 ctxIdx(见 5.4.3.4.2)，然后再解析二进制位(见 5.4.3.4.3)；否则，直接解析二进制位(见 5.4.3.4.3)。

在解析二进制位串的过程中，binIdx 从 0 开始，每解析一个二进制位，binIdx 加 1，并且将得到的二进制位串与从表 47～表 54 中相应的二进制位串比对，得到 synElVal。

5.4.3.4.2 确定 ctxIdx

ctxIdx 由起始索引号和 ctxIdxInc 决定，如下：

ctxIdx = 起始索引号+ctxIdxInc

语法元素的起始索引号见表 60。

表 60 语法元素起始索引号

语法元素	类 型	起始索引号
mb_skip_flag		0
mb_type		2
svc_mode_flag		17
mb_part_type		19
pred_mode_flag		22
intra_luma_pred_mode		23
intra_chroma_pred_mode		26
mb_reference_index		30
mv_diff_x		36
mv_diff_y		42
code_block_pattern		48
code_block_pattern_422		48
code_block_pattern_4×4		48
mb_qp_delta		60
rans_coefficient	帧模式亮度	64
	帧模式色度	131
	场模式亮度	198
	场模式色度	265

确定 mb_skip_flag 的 ctxIdxInc 的步骤如下：

ctxIdxInc=Min(binIdx, 1)

确定 mb_type 的 ctxIdxInc 的步骤如下：

1) 如果当前为 I 条带或 P 条带，则：

ctxIdxInc=Min(binIdx, 4)

2) 否则，如果当前为 B 条带，并且 binIdx 等于 0，则：

ctxIdxInc=5+a+b

3) 否则，如果当前为 B 条带，并且 binIdx 小于或等于 7，则：

ctxIdxInc=7+binIdx

4) 否则，如果当前为 B 条带，并且 binIdx 大于 7，则：

ctxIdxInc=14

如果当前块 E 的左边块 A(或上边块 B)可用，并且块 A(或块 B)的宏块类型不等于 0，则 a(或 b)的值为 1；否则 a(或 b)的值为 0。块 E 和块 A、块 B 的关系见 5.1.3.3。

确定 svc_mode_flag 的 ctxIdxInc 的步骤如下：

ctxIdxInc=binIdx

确定 mb_part_type 的 ctxIdxInc 的步骤如下：

1) 如果 binIdx 等于 0，则：

ctxIdxInc=0

2) 否则，如果 binIdx 为 0 的二进制位为'0'，则：

ctxIdxInc=1

3) 否则：

ctxIdxInc=2

确定 pred_mode_flag 的 ctxIdxInc 的步骤如下：

ctxIdxInc=0

确定 intra_luma_pred_mode 的 ctxIdxInc 的步骤如下：

ctxIdxInc=Min(binIdx, 2)

确定 intra_chroma_pred_mode 的 ctxIdxInc 的步骤如下：

1) 如果 binIdx 等于 0，则：

ctxIdxInc=a+b

2) 否则：

ctxIdxInc=3

如果当前块 E 的左边块 A(或上边块 B)可用，并且块 A(或块 B)的预测模式不等于 0，则 a(或 b)的值为 1；否则 a(或 b)的值为 0。如果块 A(或块 B)所在宏块的 svc_mode_flag 等于 1，则 a(或 b)的值为 0。块 E 和块 A、块 B 的关系见 5.1.3.3。

确定 mb_reference_index 的 ctxIdxInc 的步骤如下：

1) 如果 binIdx 等于 0，则：

ctxIdxInc=a+2×b

2) 否则，如果 binIdx 等于 1，则：

ctxIdxInc=4

3) 否则：

ctxIdxInc=5

如果当前块 E 的左边块 A(或上边块 B)可用，并且块 A(或块 B)的参考索引值大于 0，则 a(或 b)的值为 1；否则 a(或 b)的值为 0。块 E 和块 A、块 B 的关系见 5.1.3.3。

确定 mv_diff_x、mv_diff_y 的 ctxIdxInc 的步骤如下：

1) 如果 binIdx 等于 0，则：

```
if(mvda<2)
  ctxIdxInc=0
else if(mvda<16)
```

ctxIdxInc=1
else
ctxIdxInc=2

2) 否则，如果 binIdx 等于 1，则：
ctxIdxInc=3

3) 否则，如果 binIdx 等于 2，并且 binIdx 为 1 的二进制位为'0'，则：
ctxIdxInc=4

4) 否则，如果 binIdx 为 1 的二进制位为'0'，则：
ctxIdxInc=5

如果当前块 E 的左边块 A 可用，并且块 A 的 mv_diff_x 或 mv_diff_y 在比特流存在，则 mvda 的值为块 A 的 mv_diff_x 或 mv_diff_y 的值；否则 mvda 的值为 0。块 E 和块 A 的关系见 5.1.3.3。

确定 code_block_pattern，code_block_pattern_422 及 code_block_pattern_4×4 的 ctxIdxInc 的步骤如下：

1) 如果 binIdx 的值小于或等于 3，则：
ctxIdxInc=a+2×b

2) 否则，如果 binIdx 的值等于 4，则：
if(chroma_format_idc==1)
ctxIdxInc=4+c+2×d
else
ctxIdxInc=4

3) 否则，如果 binIdx 的值等于 5，则：
ctxIdxInc=4+e+2×f

4) 否则：
ctxIdxInc=8+e+2×f

如果当前块 E 的左边块 A(或上边块 B)可用，并且块 A(或块 B)中不包含非零系数，则 a(或 b)的值为 1；否则 a(或 b)的值为 0。块 E 和块 A、块 B 的关系见 5.1.3.3。

对 code_block_pattern，如果当前宏块 E 的左边宏块 A(或上边宏块 B)可用并且宏块 A(或宏块 B)中顺序号等于 4 或 5 的块包含非零系数，则 c(或 d)的值为 1；否则 c(或 d)的值为 0。

对 code_block_pattern，如果当前宏块 E 的左边宏块 A(或上边宏块 B)可用并且宏块 A(或宏块 B)中顺序号等于 5 的块包含非零系数，则 e(或 f)的值为 1；否则 e(或 f)的值为 0。

确定 mb_qp_delta 的 ctxIdxInc 的步骤如下：

1) 如果 binIdx 等于 0，则：
if(PreviousDeltaQP ! =0)
ctxIdxInc=1
else
ctxIdxInc=0

2) 否则，如果 binIdx 等于 1，则：
ctxIdxInc=2

3) 否则：
ctxIdxInc=3

确定 trans_coefficient 的 coeffLevel 的 ctxIdxInc 的步骤如下：

1) 如果 lMax 等于 0 或者 binIdx 不等于 0，则：
contextWeighting=0

ctxIdxInc=priIdx×7+secIdx

2) 否则：

contextWeighting=1

ctxIdxInc=priIdx×7+secIdx

ctxIdxIncW=35+(pos ≫5)×16+(pos≫1)&0x0F

如果 contextWeighting=1，则 ctxIdxW=起始索引号+ctxIdxIncW

其中 priIdx 和 secIdx 分别见表 61 和表 62；pos 记录了待解析的量化系数幅值在逆扫描中的位置。对每个块进行解析前，pos 应初始化为 0。

表 61 priIdx

lMax	0	1	2	3	4	≥5
priIdx	0	1	2	3	3	4

lMax 为当前块已解析系数的最大幅值。对每个块进行解析前，lMax 应初始化为 0。

表 62 secIdx

secIdx	含义		
	coeffLevel		coeffRun
	lMax==0	lMax ! = 0	
0		binIdx 等于 0	AbsLevel 的值等于 1，并且 binIdx 等于 0
1	binIdx 等于 0	binIdx 等于 1	AbsLevel 的值等于 1，并且 binIdx 大于等于 1
2	binIdx 大于等于 1	binIdx 大于等于 2	AbsLevel 的值大于 1，并且 binIdx 等于 0
3			AbsLevel 的值大于 1，并且 binIdx 大于等于 1

确定 trans_coefficient 的 coeffRun 的 ctxIdxInc 的步骤如下：

ctxIdxInc=priIdx×7+3+secIdx

其中 priIdx 和 secIdx 分别见表 61 和表 62。

5.4.3.4.3 二进制位解析

5.4.3.4.3.1 概述

二进制位的解析过程如下：

a) 首先，解析二进制位值 binVal

1) 如果 BypassFlag 的值为 1，执行 decode_bypass 过程(见 5.4.3.4.3.3)；

2) 否则，如果解析的语法元素为 end_of_slice，则执行 decode_final 过程(见 5.4.3.4.3.4)；

3) 否则，执行 decode_decision 过程(见 5.4.3.4.3.2)。

b) 第二步，如果 binVal 的值为 0，则二进制位为'0'；如果 binVal 的值为 1，则二进制位为'1'。

5.4.3.4.3.2 decode_decision

如果 contextWeighting 的值为 1，decode_decision 过程的输入为 rS1、rT1、valueS、valueT 以及上下文模型 ctx1、ctx2；否则 decode_decision 过程的输入为 rS1、rT1、valutS、valueT 以及上下文模型 ctx。此处 ctx 及 ctx1 对应 5.4.3.4.2 得到的 ctxIdx，ctx2 对应 5.4.3.4.2 得到的 ctxIdxW。decode_decision 过程的输出为二进制位值 binVal。decode_decision 过程用伪代码描述如下：

```
decode_decision( )
{
    if(contextWeighting==1) {
        if(ctx1->mps==ctx2->mps) {
          predMps=ctx1->mps
```

```
    lgPmps=Floor((ctx1->lgPmps+ctx2->lgPmps)/2)
}
else {
    if(ctx1->lgPmps1<ctx2->lgPmps) {
      predMps=ctx1->mps
      lgPmps=1023-((ctx2->lgPmps-ctx1->lpPmps)>>1)
    }
    else {
      predMps=ctx2->mps
      lgPmps=1023-((ctx1->lgPmps-ctx2->lgPmps)>>1)
    }
  }
}
else {
  predMps=ctx->mps
  lgPmps=ctx->lpPmps
}
if(rT1 >=(lgPmps>>2)) {
  rS2=rS1
  rT2=rT1-(lgPmps>>2)
  sFlag=0
}
else {
  rS2=rS1+1
  rT2=256+rT1-(lgPmps>>2)
  sFlag=1
}
if(rS2 > valueS||(rS2==valueS && valueT >= rT2)) {
  binVal=! predMps
  if(sFlag==0)
    tRlps=lgPmps>>2
  else
    tRlps=rT1+(lgPmps>>2)
  if(rS2==valueS)
    valueT=valueT-rT2
  else
    valueT=(valueT<<1)|read_bits(1)-rT2
  while(tRlps<0x100) {
  tRlps=tRlps<<1
  valueT=(valueT<<1)|read_bits(1)
}
rS1=0
rT1=tRlps & 0xFF
```

```
        valueS=0
        while(valueT<0x100) {
          valueS++
          valueT=(valueT≪1) | read_bits(1)
        }
        valueT=valueT & 0xFF
      }
      else {
        binVal=predMps
        rS1=rS2
        rT1=rT2
      }
      if(contextWeighting==1) {
        ctx1=update_ctx(binVal, ctx1)
        ctx2=update_ctx(binVal, ctx2)
      }
      else
        ctx=updata_ctx(binVal, ctx)
      return(binVal)
  }
```

5.4.3.4.3.3 **decode_bypass**

decode_bypass 过程的输入为 rS1、rS2、valueS 和 valueT。decode_bypass 过程的输出为二进制位值 binVal。decode_bypass 过程用伪代码描述如下：

```
  decode_bypass()
  {
    predMps=0
    lgPmps=1023
    if(rT1 )=(lgPmps≫2)) {
      rS2=rS1
      rT2=rT1-(lgPmps≫2)
      sFlag=0
  }
  else {
      rS2=rS1+1
      rT2=256+rT1-(lgPmps≫2)
      sFlag=1
  }
  if(rS2>valueS||(rS2==valueS && valueT )= rT2)) {
      binVal=! predMps
      if(sFlag==0)
        tRlps=lgPmps≫2
      else
        tRlps=rT1+(lgPmps≫2)
```

```
        if(rS2==valueS)
          valueT=valueT-rT2
        else
          valueT=(valueT≪1) | read_bits(1)-rT2
        while(tRlps<0x100) {
          tRlps=tRlps≪1
          valueT=(valueT≪1) | read_bits(1)
        }
        rS1=0
        rT1=tRlps & 0xFF
        valueS=0
        while(valueT<0x100) {
          valueS++
          valueT=(valueT≪1) | read_bits(1)
        }
        valueT=valueT & 0xFF
      }
      else {
        binVal=predMps
        rS1=rS2
        rT1=rT2
      }
      return(binVal)
    }
```

5.4.3.4.3.4 decode_final

decode_final 过程的输入为 rS1、rS2、valueS 和 valueT。decode_final 过程的输出为二进制位值 binVal。decode_final 过程用伪代码描述如下：

```
    decode_final()
    {
      predMps=0
      lgPmps= 4
      if(rT1 >=(lgPmps≫2)) {
        rS2=rS1
        rT2=rT1-(lgPmps≫2)
        sFlag=0
      }
      else {
        rS2=rS1+1
        rT2=256+rT1-(lgPmps≫2)
        sFlag=1
      }
      if(rS2>valueS ||(rS2==valueS && valueT >= rT2)) {
        binVal=! predMps
```

```
        if(sFlag==0)
          tRlps=lgPmps≫2
        else
          tRlps=rT1+(lgPmps≫2)
        if(rS2==valueS)
          valueT=valueT-rT2
        else
          valueT=(valueT≪1) | read_bits(1)-rT2
        while(tRlps<0x100) {
          tRlps=tRlps≪1
          valueT=(valueT≪1) | read_bits(1)
        }
        rS1=0
        rT1=tRlps & 0xFF
        valueS=0
        while(valueT<0x100) {
          valueS++
          valueT=(valueT≪1) | read_bits(1)
        }
        valueT=valueT & 0xFF
      }
      else {
        binVal=predMps
        rS1=rS2
        rT1=rT2
      }
      return(binVal)
    }
```

5.4.3.4.3.5 update_ctx

update_ctx 过程的输入为 binVal 和 ctx。update_ctx 过程的输出为更新后的 ctx。update_ctx 过程用伪代码描述如下：

```
update_ctx()
{
  if(ctx->cycno<=1)
    cwr=3
  else if(ctx->cycno==2)
    cwr=4
  else
    cwr=5
  if(binVal != ctx->mps){
    if(ctx->cycno<=2)
      ctx=ctx->cycno+1
    else
```

```
        ctx=3
    }
      else if(ctx->cycno==0)
        ctx->cycno=1
      if(binVal==ctx->mps)
        ctx->lgPmps=ctx->lgPmps-(ctx->lgPmps≫cwr)-(ctx->lgPmps ≫(cwr+2))
      else {
        switch(cwr) {
          case 3:
            ctx->lgPmps=ctx->lgPmps+197
            break
          case 4:
            ctx->lgPmps=ctx->lgPmps+95
            break
          default:
            ctx->lgPmps=ctx->lgPmps+46
        }
        if(ctx->lgPmps>1023) {
          ctx->lgPmps=2047-ctx->lgPmps
          ctx->mps=! (ctx->mps)
        }
      }
      return(ctx)
  }
```

6 音频部分

6.1 总体描述

6.1.1 模拟信号和数字信号之间转换

模拟信号和数字信号之间的转换描述如下：

a) 模拟信号到数字线性 PCM 信号转换：

——麦克风；

——输入电平调节设备；

——抗混叠滤波器；

——采样保持设备，采样频率为 16 kHz、24 kHz、32 kHz 和 48 kHz；

——模拟信号转换为 16 比特数字线性 PCM，采用二进制补码表示。

b) 数字线性 PCM 信号到模拟信号转换：

——16 比特数字线性 PCM 信号转换为模拟信号；

——转换保持设备；

——补偿重建滤波器；

——输出电平调节设备；

——耳机或喇叭。

6.1.2 编解码框架的描述

图 25 和图 26 给出了音频编码器和解码器框图。

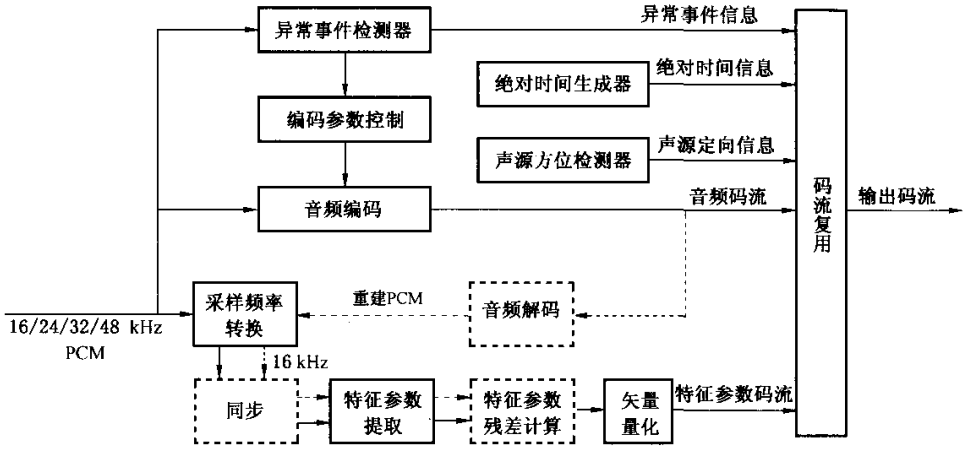


图 25　音频编码器框图

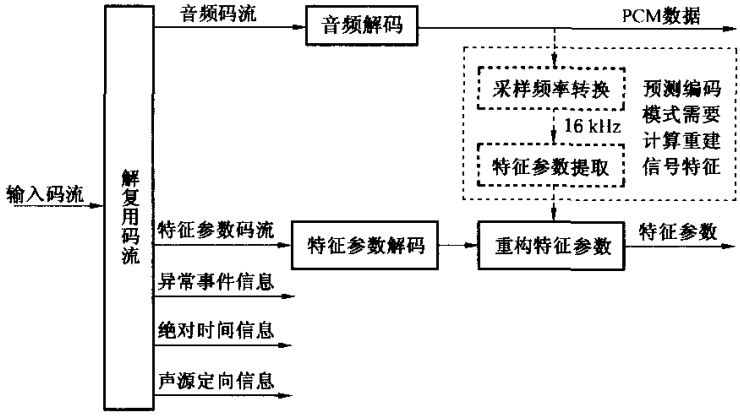


图 26　音频解码器框图

在编码器端，输入声音信号首先经过异常事件检测器，如果检测到异常（比如尖叫声、枪声、爆炸声等），将检测结果传递给编码参数控制模块，编码参数控制模块将根据检测到的事件的重要性设置音频编码器的编码参数，以实现变质量音频编码。音频编码器低频和高频用不同的方法进行编码，低频使用基于 ACELP 和 TAC 切换的双核编码器进行编码，高频用相当少的比特进行 BWE 编码。识别特征参数编码提供两种编码模式：直接编码模式和预测编码模式。直接编码模式直接对提取识别特征参数进行矢量量化；而预测编码模式则需要对音频编码器的码流进行解码得到重建信号，重建信号和原始信号分别提取识别特征参数，使用重建信号识别特征参数作为原始信号识别特征参数的预测，最后对预测残差进行矢量量化。如果输入信号采样频率不是 16 kHz，则输入信号先经过采样频率转换模块，转换为 16 kHz 采样的信号，再进行识别特征参数提取。对于预测编码模式，由于重建信号和原始信号之间存在一定的延迟，为了保证所提取的识别特征参数在时间上对应关系，需要经过同步模块同步。最后音频码流、识别特征参数码流、异常事件信息、声源定向信息和绝对时间信息复用成一路码流。

在解码器端，先进行解复用以得到音频码流、识别特征参数码流、异常事件信息、声源定向信息和绝对时间信息。然后音频解码器直接解码输出重建音频信号；识别特征参数解码器从码流中解码出识别特征参数，如果当前编码模式是直接编码模式，则解码得到的就是最终的识别特征参数；如果当前解码模式是预测编码模式，则解码得到的是识别特征参数的残差，需对音频解码器输出的重建信号，先经过采样频率转换模块，转换为 16 kHz 采样的信号，再提取识别特征参数作为预测值，最后这两部分相加

就得到了最终的识别特征参数。

6.1.3 音频编解码描述

图 27 给出了双核音频编码器流程图。输入音频信号首先经过预处理，分成两个频带，分别是低频信号和高频信号，采样频率都是 $Fs/2$。然后低频信号使用基于 ACELP 和 TAC 切换的双核编码器，ACELP 是基于时域预测编码技术，适合语音信号和瞬态信号，TAC 是基于变换域编码技术，更适合音乐信号和稳态信号。高频信号使用 BWE 进行编码。最后将低频参数、高频参数和编码模式信息复用成一路码流。

图 28 给出了双核音频解码器流程图。首先通过码流解复用，得到低频参数、高频参数和编码模式信息。然后低频参数通过 ACELP 和 TAC 双核解码，高频参数通过 BWE 解码。解码后低频信号和高频信号通过后处理恢复成全带信号。

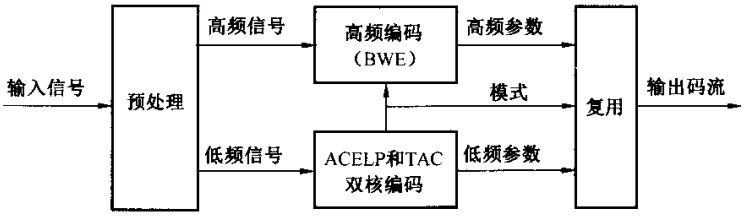


图 27 双核音频编码器流程图

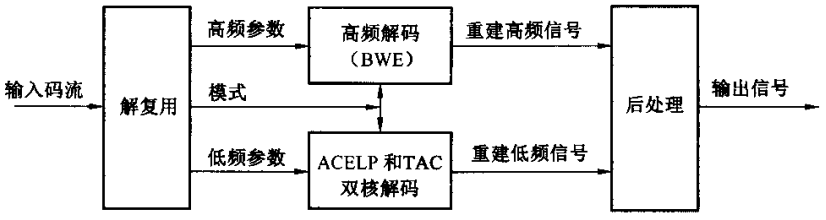


图 28 双核音频解码器流程图

6.1.4 识别特征参数编码描述

图 29 给出了识别特征参数编码框图，整个框架主要分为两个模块：特征提取和特征压缩。16 kHz 采样频率的宽带语音信号先进行去直流偏置，噪声消除，波形预处理，再进行倒谱计算，最后对得到的倒谱特征进行去信道干扰的均衡处理。VAD 模块检测语音帧和非语音帧，输出 1 比特的 VAD 标志位，同识别特征参数合并压缩，具体 VAD 算法描述参见附录 I。识别特征参数选取的是 13 维 MFCC 系数和一个对数能量系数，构成一个 14 维的特征矢量。特征提取时帧长为 25 ms(16 kHz 采样频率下 400 个样本)，帧移为 10 ms(16 kHz 采样频率下 160 个样本)。特征压缩模块使用矢量量化对特征矢量或残差矢量进行量化。

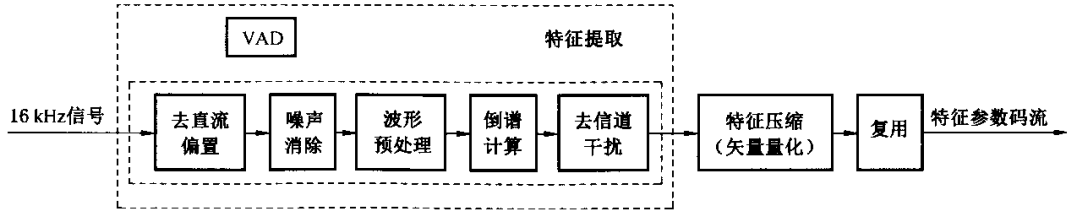


图 29 识别特征参数编码框图

识别特征参数编码提供两种编码模式：直接编码模式和预测编码模式。直接编码模式直接对提取特征进行矢量量化；而预测编码模式则需要对音频编码器的码流进行解码得到重建信号，重建信号和原

始信号分别提取识别特征参数，使用重建信号识别特征参数作为原始信号识别特征参数的预测，最后对预测残差进行矢量量化。

详细编码过程见 6.2.6。

6.2 编码器功能描述

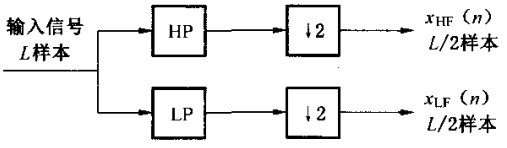
6.2.1 预处理

6.2.1.1 采样频率转换

音频信号的输入采样频率有 16 kHz、24 kHz、32 kHz 和 48 kHz，需要将各种不同采样频率的输入信号在编码之前的预处理中进行重采样，转换为内部采样频率 Fs。同理，在解码的后处理中同样需要采样频率转换。

6.2.1.2 低频信号和高频信号分解

采样频率为 Fs 的输入信号通过截止频率在 $Fs/4$ 的低通滤波器，再作 2 倍临界下采样，得到 $Fs/2$ 采样低频信号 $x_{LF}(n)$；同样，经过截止频率在 $Fs/4$ 的高通滤波器，再作 2 倍临界下采样，得到 $Fs/2$ 采样高频信号 $x_{HF}(n)$，见图 30。



L——音频超帧长度。

图 30 低频和高频信号分解框图

6.2.1.3 低频信号高通滤波

低频信号经过高通滤波器，目的是为了滤掉不需要的低频成分。高通滤波器的传递函数如下：

$$H_{h1}(z)=\frac{b_0-b_1z^{-1}+b_2z^{-2}}{1-a_1z^{-1}+a_2z^{-2}} \qquad \cdots\cdots(1)$$

式中的滤波器系数取决于采样频率。

6.2.2 低频信号编码概述

6.2.2.1 ACELP 和 TAC 双核编码

$Fs/2$ 频率采样的低频信号(0～$Fs/4$ 频带)使用基于 ACELP 和 TAC 切换的双核编码器。ACELP 属于时域预测的编码技术，适合语音信号和瞬态信号编码。TAC 属于变换域的编码技术，更适合典型的音乐信号和稳态信号编码，本标准采用了其中一种称为 TVC 的变换域编码技术。

6.2.2.2 ACELP 和 TVC 的时间图

ACELP 和 TVC 双核编码的输入是按 $Fs/2$ 频率采样的单声道信号，以连续 256 个采样点组成一帧进行处理。每帧可采用两种模式编码，采用哪一种取决于信号特征，见图 31 所示。在 ACELP 模式中，采用 ACELP 核编码。在 TVC 模式中，采用 TVC 核编码，由于 TVC 是变换编码技术，需要加上下一帧的前 32 个样本用于帧重叠。

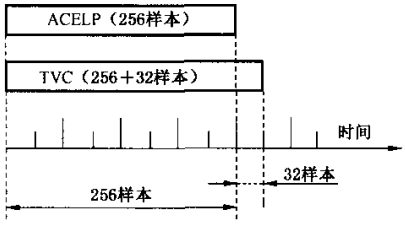


图 31 帧类型的时间图

6.2.2.3 ACELP 和 TVC 的闭环模式选择

音频帧首先使用多种模式分别编码，然后选择最好的模式。选择的标准是加权信号 $x_w(n)$ 和合成加权信号 $\hat{x}_w(n)$ 间的分段信噪比均值。子帧的分段信噪比公式如下：

$$segSNR_i = 20\log_{10}\left\{\frac{\sum_{n=0}^{N-1} x_w^2(n)}{\sum_{n=0}^{N-1}(x_w(n)-\hat{x}_w(n))^2}\right\} \quad \cdots\cdots(2)$$

式中：

N ——子帧长度(64 个样点)。

每帧分段信噪比均值计算公式如下：

$$\overline{segSNR} = \frac{1}{N_{SF}}\sum_{i=0}^{N_{SF}-1} segSNR_i \quad \cdots\cdots(3)$$

式中：

N_{SF} ——音频帧中子帧的数目，本标准规定 N_{SF} 的值是 4。

6.2.3 ACELP 编码

6.2.3.1 预加重

输入到 ACELP 核编码的信号，通过一阶的预加重滤波器 $H_{emph}(z)$，见 6.2.3.3。

6.2.3.2 LP 分析和量化

6.2.3.2.1 线性预测分析

线性预测分析是用 16 阶线性预测器作短时分析，采用莱文逊-杜宾(Levinson-Durbin)算法进行线性预测系数求解，对每帧分析一次得到一组线性预测系数。线性预测系数在编码前要先转化为 ISF 系数，然后再进行量化。LP 合成滤波器的传递函数如下：

$$H(z) = \frac{1}{\hat{A}(z)} = \frac{1}{1+\sum_{i=1}^{m}\hat{a}_i z^{-i}} \quad \cdots\cdots(4)$$

式中：

$\hat{a}_i$ ——量化后的线性预测系数，$m=16$ 是预测阶数。

LP 分析首先用 384 个样本的非对称窗加权预加重后的信号 $s(n)$，计算自相关系数，用莱文逊—杜宾算法求 LP 系数，然后转换为 ISP 系数并在 ISP 域插值，最后转到 ISF 域量化。

384 个样本的 LP 分析帧结构如图 32 所示，其中 256 个样本来自第 n 帧，64 个样本来自第 $n-1$ 帧，64 个样本来自第 $n+1$ 帧。第 n 帧分析窗与第 $n-1$ 帧分析窗有 128 个样本的重叠。因此 LP 分析需要前瞻 64 个样本。

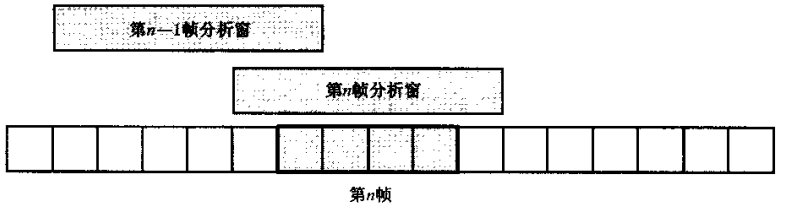


图 32 线性预测分析帧结构图

6.2.3.2.2 加窗和自相关函数的计算

分析窗采用重心在第四个子帧的非对称窗，该窗由两部分组成，第一部分是半个海明(Hamming)窗，第二部分是 1/4 余弦窗，公式为：

$$w(n)=\begin{cases}0.54-0.46\cos\left(\dfrac{2n\pi}{2L_1-1}\right), & n=0,\cdots,L_1-1\\ \cos\left(\dfrac{2\pi(n-L_1)}{4L_2-1}\right), & n=L_1,\cdots,L_1+L_2-1\end{cases} \quad \cdots\cdots(5)$$

式中：

$L_1=256$，$L_2=128$。

设加窗后信号为 $s'(n)$，则有

$$s'(n)=w(n)s(n) \qquad n=0,\cdots,383 \quad \cdots\cdots(6)$$

式中：

$w(n)$——加窗函数；

$s(n)$——预加重后的信号。

$s'(n)$对应的自相关函数为

$$r(k)=\sum_{n=k}^{383}s'(n)s'(n-k), \quad k=0,\cdots,16 \quad \cdots\cdots(7)$$

然后用滞后窗 $w_{lag}(i)$乘自相关函数使其具有 60 Hz 的带宽扩展，滞后窗 $w_{lag}(i)$的表达式为

$$w_{lag}(i)=\exp\left[-\frac{1}{2}\left(\frac{2\pi f_0 i}{f_s}\right)^2\right], \quad i=1,\cdots,16 \quad \cdots\cdots(8)$$

式中：

$f_0=60$ Hz——扩展的带宽；

$f_s=12.8$ kHz——采样频率。

另外 $r(0)$乘以白噪声校正因子 1.000 1。最后得到修正后的自相关函数 $r'(k)$：

$$r'(k)=\begin{cases}1.000\ 1r(0), & k=0\\ r(k)w_{lag}(k), & k=1,\cdots,16\end{cases} \quad \cdots\cdots(9)$$

6.2.3.2.3 用莱文逊—杜宾算法求解 LP 系数

用修正后的自相关函数 $r'(k)$求解线性预测系数 a_k，$k=1,\cdots,16$，即求解下述方程组：

$$\sum_{k=1}^{16}a_k r'(|i-k|)=-r'(i), i=1,\cdots,16 \quad \cdots\cdots(10)$$

该方程组可用莱文逊—杜宾算法求解，算法计算步骤如下：

$$\begin{aligned}&E(0)=r'(0);\\ &for(i=1;i<=16;++i)\\ &\{\\ &\quad k_i=-\left[r'(i)+\sum_{j=1}^{i-1}a_j^{i-1}r'(i-j)\right]/E(i-1);\\ &\quad a_i^{(i)}=k_i;\\ &\quad for(j=1;j<=i-1;++j)\\ &\quad\{\\ &\quad a_j^{(i)}=a_j^{(i-1)}+k_i a_{i-j}^{(i-1)};\\ &\quad\}\\ &\quad E(i)=(1-k_i^2)E(i-1);\\ &\}\end{aligned} \quad \cdots\cdots(11)$$

最后得到线性预测系数 $a_j=a_j^{(16)}$，$j=1,\cdots,16$。

线性预测系数转化成 ISP 系数，以便于量化和内插。

6.2.3.2.4 LP 系数转换为 ISP 系数

对于 16 阶线性预测器来说，ISP 系数就是下面这组多项式的根：

$$F'_1(z)=A(z)+z^{-16}A(z^{-1}) \quad \cdots\cdots(12)$$

$$F'_2(z)=A(z)-z^{-16}A(z^{-1}) \quad \cdots\cdots(13)$$

多项式 $F'_1(z)$和 $F'_2(z)$分别为对称和反对称多项式。它们的根在单位圆上，而且相互交替出现。其中 $F'_2(z)$有两个根分别为 $z=1(\omega=0)$和 $z=-1(\omega=\pi)$。为了消除这两个根，定义两个新的多项式记作 $F_1(z)$和 $F_2(z)$：

$$F_1(z)=F'_1(z) \quad \cdots\cdots(14)$$

$$F_2(z)=F'_2(z)/(1-z^{-2}) \quad \cdots\cdots(15)$$

多项式 $F_1(z)$和 $F_2(z)$分别有 8 个和 7 个共轭复根在单位圆上$(e^{\pm j\omega_i})$，由下面的式子表示：

$$F_1(z)=(1+a_{16})\prod_{i=0,2,\cdots,14}(1-2q_iz^{-1}+z^{-2}) \quad \cdots\cdots(16)$$

$$F_2(z)=(1-a_{16})\prod_{i=1,3,\cdots,13}(1-2q_iz^{-1}+z^{-2}) \quad \cdots\cdots(17)$$

式中：

q_i——ISP 系数，$q_i=\cos(\omega_i)$，$i=0,\cdots,15$；

ω_i——ISF 系数，ISF 系数满足顺序特性，即 $0<\omega_0<\omega_1<\cdots<\omega_{14}<\pi$；

a_{16}——最后一阶线性预测系数，$q_{15}=a_{16}$。

由下面递推关系得到 $F_1(z)$和 $F_2(z)$多项式的系数 $f_1(i)$和 $f_2(i)$：

```
for(i=0;i<=7;++i)
{
  f1(i)=ai+am-i;
  f2(i)=ai-am-i+f2(i-2);
}
f1(8)=2a8;
```

……(18)

式中：

$m=16$——预测器阶数；

$f_2(-2)=f_2(-1)=0$。

ISP 系数的求解过程描述如下：

首先将 0 到 π 初分成 100 点，找到符号变化的区间。然后使用切比雪夫多项式法在每个区间求根，符号变换的区间进一步细分为 4 个子区间。使用这个方法得到的根是 ISP 系数。

多项式 $F_1(z)$和 $F_2(z)$在 $z=e^{j\omega}$处可表示为

$$F_1(\omega)=2e^{-j8\omega}C_1(x) \text{ 和 } F_2(\omega)=2e^{-j7\omega}C_2(x) \quad \cdots\cdots(19)$$

$$C_1(x)=\sum_{i=0}^{7}(f_1(i)T_{8-i}(x)+f_1(8)/2) \text{ 和 } C_2(x)=\sum_{i=0}^{6}(f_2(i)T_{8-i}(x)+f_2(7)/2) \quad \cdots\cdots(20)$$

式中：

$T_m=\cos(m\omega)$——切比雪夫多项式第 m 个根。

多项式 $C(x)$在 $x=\cos(\omega)$处的递归计算如下：

```
for(k=nf-1;k<=1;--k)
{
  bk=2xbk+1-bk+2+f(nf-k);
}
C(x)=xb1-b2+f(nf)/2;
```

……(21)

对于多项式 $C_1(x)$，$n_f=8$；对于多项式 $C_2(x)$，$n_f=7$；迭代初始值 $b_{nf}=f(0)$和 $b_{nf+1}=0$。

6.2.3.2.5 ISP 系数转换为 LP 系数

一旦 ISP 系数被量化和内插，将被再次转回到 LP 系数。具体转换过程如下：

根据已量化和内插的 ISP 系数，用公式(16)、(17)求 $F_1(z)$和 $F_2(z)$的系数，用 $q_i=\cos(\omega_i)$，$i=0,\cdots,m-1$(其中 $m=16$)迭代计算系数 $f_1(i)$：

```
for(i=2;i<=m/2;++i)
{
  f1(i)=-2q2i-2f1(i-1)+2f1(i-2);
  for(j=i-1;j<=2;--j)
    f1(j)=f1(j)-2q2i-2f1(j-1)+f1(j-2);
  f1(1)=f1(1)-2q2i-2;
}
```

……(22)

式中：

初始值 $f_1(0)=1, f_1(1)=-2q_0$。

同理，计算 $f_2(i)$，只是需要用 q_{2i-1}代替公式(22)中的 q_{2i-2}，用 $m/2-1$ 代替 $m/2$，初始值变为 $f_2(0)=1, f_2(1)=-2q_1$。

求出 $f_1(i), i=0,\cdots,m/2$ 和 $f_2(i), i=0,\cdots,m/2-1$ 即得到 $F_1(z)$和 $F_2(z)$的系数，$F_2(z)$再乘以 $1-z^{-2}$就得到 $F_2'(z)$。那么 $F_1'(z)$和 $F_2'(z)$多项式的系数 $f_1'(i)$和 $f_2'(i)$为：

$$\begin{aligned} f'_2(i)&=f_2(i)-f_2(i-2), \quad i=2,\cdots,m/2-1\\ f'_1(i)&=f_1(i), \qquad\qquad i=0,\cdots,m/2 \end{aligned} \tag{23}$$

$F_1'(z)$和 $F_2'(z)$分别乘以 $1+q_{m-1}$和 $1-q_{m-1}$，多项式系数最终变为：

$$\begin{aligned} f'_2(i)&=(1-q_{m-1})f'_2(i), \quad i=0,\cdots,m/2-1\\ f'_1(i)&=(1+q_{m-1})f'_1(i), \quad i=0,\cdots,m/2 \end{aligned} \tag{24}$$

由于 $F_1'(z)$和 $F_2'(z)$分别是对称和反对称多项式，根据关系式 $A(z)=(F_1'(z)+F_2'(z))/2$，最后得到 LP 系数：

$$a_i=\begin{cases} 0.5f'_1(i)+0.5f'_2(i), & i=1,\cdots,m/2-1\\ 0.5f'_1(i)-0.5f'_2(i), & i=m/2+1,\cdots,m-1\\ 0.5f'_1(m/2), & i=m/2\\ q_{m-1}, & i=m \end{cases} \tag{25}$$

6.2.3.2.6 ISP 系数的量化

ISP 系数在量化之前先要转换为频域的 ISF 系数。ISF 系数表达式为：

$$f_i=\begin{cases} \dfrac{f_s}{2\pi}\arccos(q_i), & i=0,\cdots,14\\ \dfrac{f_s}{4\pi}\arccos(q_i), & i=15 \end{cases} \tag{26}$$

式中：

$f_i\in[0,6.4]$ kHz——ISF 系数；

$f_s=12.8$ kHz——采样频率。

ISF 矢量表示为 $f'=[f_0,f_1,\cdots,f_{15}]$，$t$ 表示矢量的转置。

用一阶 MA 预测法，先求出当前帧的 ISF 预测残差矢量，然后量化 ISF 预测残差矢量。

定义 $z(n)$为去均值后的当前帧 ISF 矢量。预测残差矢量为 $r(n)$，其表达式为：

$$z(n)=isf_n-mean\_isf \tag{27}$$

$$r(n)=z(n)-p(n) \tag{28}$$

式中：

isf_n——第 n 帧的 ISF 矢量；

$mean\_isf$——ISF 矢量均值；

$p(n)$——第 n 帧的预测 ISF 矢量。

由一阶 MA 预测法得到：

$$p(n)=\frac{1}{3}\hat{r}(n-1) \qquad \cdots\cdots(29)$$

式中：

$\hat{r}(n-1)$——前一帧量化后的 ISF 残差矢量。

为了利用 ISF 系数的帧内相关性，将 16 个 ISF 残差系数(矢量 $VQ1$)按照其索引号的奇偶顺序分为两组，如下所示：

分组 1：$res\_isf_0, res\_isf_2, res\_isf_4, res\_isf_6, res\_isf_8, res\_isf_{10}, res\_isf_{12}, res\_isf_{14}$

分组 2：$res\_isf_1, res\_isf_3, res\_isf_5, res\_isf_7, res\_isf_9, res\_isf_{11}, res\_isf_{13}, res\_isf_{15}$

将 $res\_isf_0, res\_isf_2, res\_isf_4$ 三个系数组成子矢量 $VQ2$，$res\_isf_6, res\_isf_8, res\_isf_{10}$ 三个系数组成子矢量 $VQ3$，分别对 $VQ2$ 和 $VQ3$ 进行矢量量化，$VQ2$ 矢量量化需要 10 比特，$VQ3$ 矢量量化需要 9 比特：

$$VQ2=\{res\_isf_0, res\_isf_2, res\_isf_4\} \text{ 和 } VQ3=\{res\_isf_6, res\_isf_8, res\_isf_{10}\} \qquad \cdots\cdots(30)$$

量化采用的误差准则为均方量化误差准则，见下式：

$$E=\sum_i (res\_isf_i - res\_isf_i^q)^2, i=\{0,2,\cdots,10\} \qquad \cdots\cdots(31)$$

量化得到对应的最佳量化矢量为 $VQ'2$ 和 $VQ'3$：

$$VQ'2=\{res\_isf_0^q, res\_isf_2^q, res\_isf_4^q\} \text{ 和 } VQ'3=\{res\_isf_6^q, res\_isf_8^q, res\_isf_{10}^q\} \qquad \cdots\cdots(32)$$

利用 $VQ'2$ 的 $res\_isf_0^q$ 和 $res\_isf_2^q$ 对 $VQ1$ 的 res_isf_1 系数进行预测，并计算相应的残差 $res\_isf'_1$。

$$\begin{aligned} res\_isf_1^{predict} &= \theta+\alpha_0\times res\_isf_0^q+\beta_2\times res\_isf_2^q \\ res\_isf'_1 &= res\_isf_1 - res\_isf_1^{predict} \end{aligned} \qquad \cdots\cdots(33)$$

式中：

θ——ISF 系数均值；

α_0, β_2——预测系数。

将矢量 $VQ1$ 中的 $res\_isf_{12}, res\_isf_{14}$ 和 $res\_isf'_1$ 组成子矢量 $VQ4$，并对矢量 $VQ4$ 进行矢量量化，得到对应的最佳量化矢量 $VQ'4$，$VQ4$ 矢量量化需要 9 比特：

$$\begin{aligned} VQ4 &= \{res\_isf_{12}, res\_isf_{14}, res\_isf'_1\} \\ VQ'4 &= \{res\_isf_{12}^q, res\_isf_{14}^q, res\_isf_1^q\} \end{aligned} \qquad \cdots\cdots(34)$$

至此 16 维矢量 $VQ1$ 中已经有下列 ISF 残差系数进行了矢量量化：

量化前：$res\_isf_0, res\_isf_2, res\_isf_4, res\_isf_6, res\_isf_8, res\_isf_{10}, res\_isf_{12}, res\_isf_{14}, res\_isf_1$

量化后：$res\_isf_0^q, res\_isf_2^q, res\_isf_4^q, res\_isf_6^q, res\_isf_8^q, res\_isf_{10}^q, res\_isf_{12}^q, res\_isf_{14}^q, res\_isf_1^q$

剩余的七个未量化 ISF 残差系数组成子矢量 $VQ5$，利用上述量化的 ISF 残差系数对 $VQ5$ 进行预测，并计算相应的残差子矢量 $VQ6$：

$$\begin{aligned} res\_isf_i^{predict} &= \theta_i+\alpha_{i-1}\times res\_isf_{i-1}^q+\beta_{i+1}\times res\_isf_{i+1}^q, i=3,5,7,9,11,13 \\ res\_isf'_i &= res\_isf_i - res\_isf_i^{predict}, \quad i=3,5,7,9,11,13 \end{aligned} \qquad \cdots\cdots(35)$$

式中：

$res\_isf_i^{predict}$——第 i 个 ISF 残差系数的预测值；

$res\_isf'_i$——第 i 个 ISF 残差系数与其预测值的残差；

θ_i——ISF 系数均值；

α_{i-1} 和 β_{i+1}——预测系数。

将 $VQ6$ 分为如下两个矢量：

$$VQ7=\{res\_isf'_3,res\_isf'_5,res\_isf'_7\}$$
$$VQ8=\{res\_isf'_9,res\_isf'_{11},res\_isf'_{13},res\_isf_{15}\} \quad \cdots\cdots(36)$$

对 $VQ7$ 矢量和 $VQ8$ 矢量分别进行矢量量化，$VQ7$ 需要 9 比特量化，$VQ8$ 需要 9 比特量化。

对 16 维 ISF 残差系数进行矢量量化总共需要的比特数：$VQ2$ 需要 10 比特，$VQ3$ 需要 9 比特，$VQ4$ 需要 9 比特，$VQ7$ 需要 9 比特，$VQ8$ 需要 9 比特。总共需要 46 比特。

6.2.3.2.7 ISP 系数的插值

定义 $q^{(n)}$ 是第 n 帧 LP 分析得到的 ISP 矢量，$q^{(n-1)}$ 是第 $n-1$ 帧 LP 分析得到的 ISP 矢量。每个子帧的 ISP 矢量 $q_i^{(n)}$ 插值如下：

$$\begin{aligned}q_1^{(n)}&=0.55q^{(n-1)}+0.45q^{(n)}\\q_2^{(n)}&=0.2q^{(n-1)}+0.8q^{(n)}\\q_3^{(n)}&=0.04q^{(n-1)}+0.96q^{(n)}\\q_4^{(n)}&=q^{(n)}\end{aligned} \quad \cdots\cdots(37)$$

得到每个子帧的 ISP 系数后，再将 ISP 系数转换为 LP 系数得到每个子帧的 LP 滤波器。

上面的插值公式既用于量化前的 ISP 系数，也用于量化后的 ISP 系数。

6.2.3.3 感知加权

对信号进行感知加权滤波处理，感知加权处理后的输出信号用于后续处理环节。

感知加权滤波器形式如下：

$$W(z)=\frac{A(z/\gamma_1)}{H_{emph}(z)} \quad \cdots\cdots(38)$$

式中：

$\gamma_1=0.92$；

$H_{emph}(z)$——预加重滤波器。

当信号的高频能量小于低频能量时，进行高频预加重滤波，用来提升信号的高频部分，预加重滤波器形式为：

$$H_{emph\_hi}(z)=1-\mu_1 z^{-1} \quad \cdots\cdots(39)$$

当信号的低频能量小于高频能量时，进行低频预加重滤波，用来提升信号的低频部分，预加重滤波器形式为：

$$H_{emph\_low}(z)=1+\mu_2 z^{-1} \quad \cdots\cdots(40)$$

预加重滤波器系数 μ_1 取 0.68，μ_2 取 0.18。在高频预加重和低频预加重模式切换时，为了避免出现切换噪声，需要进行过渡平滑。具体为将预加重滤波器的系数在一定范围内渐次平滑过渡到另一模式。此外，编码器端需要编码 1 比特预加重模式信息，以告诉解码器感知加权滤波器中预加重采用的模式。规定此标志位为 0 时，表示高频预加重，为 1 时，表示低频预加重。

解码时使用逆感知加权滤波器，逆感知加权滤波器形式如下：

$$\frac{1}{W(z)}=\frac{H_{emph}(z)}{A(z/\gamma_1)} \quad \cdots\cdots(41)$$

6.2.3.4 ACELP 激励编码

6.2.3.4.1 开环基音搜索

6.2.3.4.1.1 自相关函数序列的计算

开环基音搜索每两个子帧估计一次基音周期。进行开环基音搜索是为了估计出一个比较准确的基音周期，从而降低闭环基音周期搜索的复杂度。

开环基音周期搜索基于感知加权后的信号进行分析。加权信号 $s_w(n)$ 在进行基音周期搜索之前，先使用四阶 FIR 滤波器 $H_{decim2}(z)$ 进行滤波，然后再进行 2 倍下采样处理，得到信号 $s_{wd}(n)$ 进行开环基音周期搜索。

归一化的自相关函数，计算如下：

$$corr' = \frac{\sum_{n=0}^{63} s_{wd}(n)s_{wd}(n-delay)}{\sqrt{\sum_{n=0}^{63} s_{wd}^{2}(n)\sum_{n=0}^{63} s_{wd}^{2}(n-delay)}} \quad \cdots\cdots(42)$$

式中：

$s_{wd}(n)$——感知加权域中降采样信号；

$delay$——基音周期候选值，搜索范围同内部采样频率 F_s 相关，当 $F_s=25.6$ kHz 时，范围为 19～115；

$corr'$——对应于该候选值的自相关函数。

为了降低复杂度，计算 $corr=sign(corr')\times corr'\times corr'$ 代替公式(42)中自相关函数，其中 $sign(corr')$ 代表 $corr'$ 的符号。

对于每个基音周期候选值，计算自相关函数值，然后从中选取最多六个满足以下关系的基音周期候选值：

a) 该候选值对应的自相关函数值大于前一个候选值对应的自相关函数值，并且大于后一个候选值对应的自相关函数值；

b) 对于满足条件 a)的基音周期候选值，选取其对应自相关函数最大的最多六个值(若出现自相关函数值相同，则候选值较小的优先)按其对应的自相关函数值从大到小排列保存，并保存其对应的基音周期候选值序列。

由以上两个条件确定有序自相关函数序列 max$corr$[6]以及对应基音周期候选值序列 $peakpos$[6]。

6.2.3.4.1.2 基音周期全局参考确定

引入基音周期全局参考 global_pitch 进行辅助判断，使基音周期具有平滑性。基音周期全局参考确定方法如图 33 所示，具体描述如下：

利用 6.2.3.4.1.1 确定的基音周期候选值序列 $peakpos$[6]以及自相关函数序列 max$corr$[6]。

首先选择与前帧的基音周期全局参考接近的基音周期候选值，对其相应的自相关函数值乘以 1.2 进行加权。重新排列基音周期候选值序列 $peakpos$[6]以及自相关函数序列 max$corr$[6]。

然后对 $peakpos$[6]以及 max$corr$[6]消除基音周期加倍。倍周期的消除采用的是固定加权方法，其目的是找出一个最佳基音周期候选值，算法为：

a) 设定最佳的基音周期候选值为自相关函数最大值对应的基音周期候选值。考察基音周期候选序列，对每一个基音周期候选值，选择一个自相关函数值的缩放因子。根据基音周期候选值的大小来选择缩放因子，当基音周期候选值大于阈值 25 时，选择缩放因子为 1.2；否则，选择缩放因子为 1.11；

b) 比较该基音周期候选值对应的自相关函数值与自相关函数值序列中的最大值和缩放因子的比值，若同时满足：

1) 当前考虑的基音周期候选值小于当前最佳的基音周期；

2) 当前基音周期候选值对应的自相关函数值大于自相关函数序列中的最大值和缩放因子的比值。

则设定基音周期最佳候选值为当前的基音周期候选值。如此循环，直至基音周期候选值序列中的每一个基音周期候选值计算完成；

c) 判断自相关函数序列中的最大值对应的基音周期候选值是否为当前最佳基音周期候选值的加倍。若是，保持当前的基音周期最佳候选值；否则，设定自相关函数序列中最大值对应的基音周期候选值作为最佳基音周期候选值。

得到了最佳基音周期候选值后，要进行可靠的基音周期全局参考确定。确定基音周期全局参考的

算法为：

a) 满足以下四个条件之一，即可确定可靠的基音周期参考：

1) 基音周期候选值序列中，自相关函数最大值并不是最佳基音周期候选对应的自相关函数值的加倍，并且最佳的基音周期候选值同当前的基音周期全局参考(延续前一帧)的差值绝对值小于8；

2) 自相关函数序列中的最大值与其他值的比值均大于1.7；

3) 基音周期候选值序列中存在基音周期候选值为最佳基音周期候选值的加倍；

4) 当前的基音周期全局参考(延续前一帧)是当前最佳基音周期候选值的加倍，并且自相关函数的最大值要大于阈值0.36。

b) 如果当前帧能确定可靠的基音周期参考，则为新的基音周期全局参考；否则，当前帧要延续前一帧的基音周期全局参考。如果满足以下三个条件之一，则强制基音周期全局参考为0：1) 自相关函数最大值小于0.15；2) 保持基音周期全局参考的帧数超过2帧；3) 弱自相关函数的帧数超过1帧。

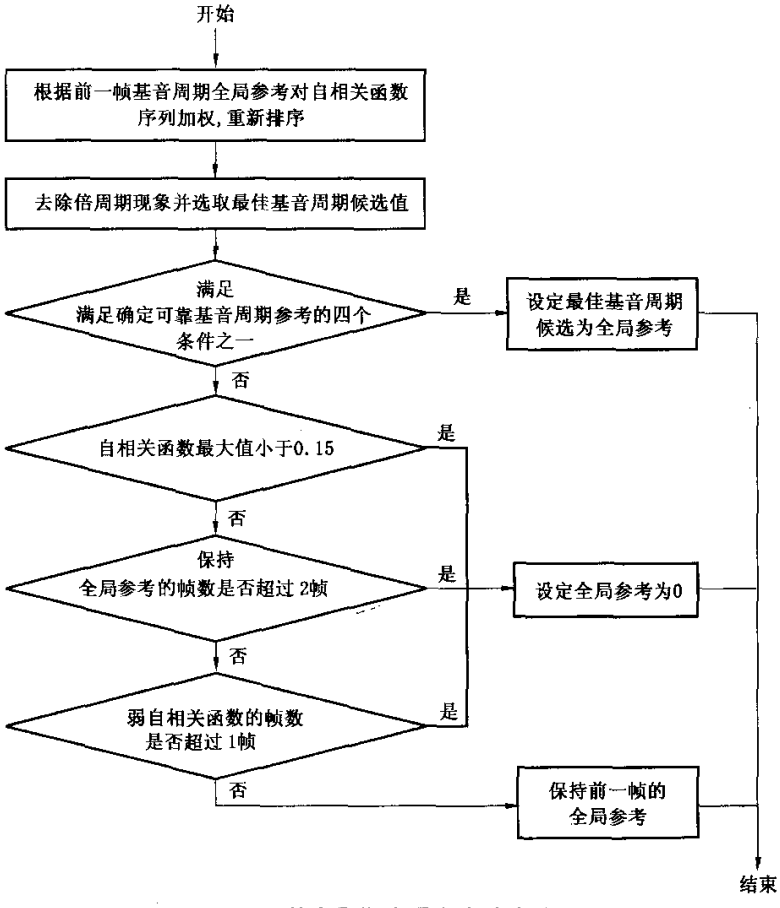


图33 基音周期全局参考确定流程

6.2.3.4.1.3 基音周期最终确定

在确定基音周期时候，利用自相关函数序列以及其对应的基音周期候选序列，将分三种情况确定最终的基音周期，具体方法见图34所示：

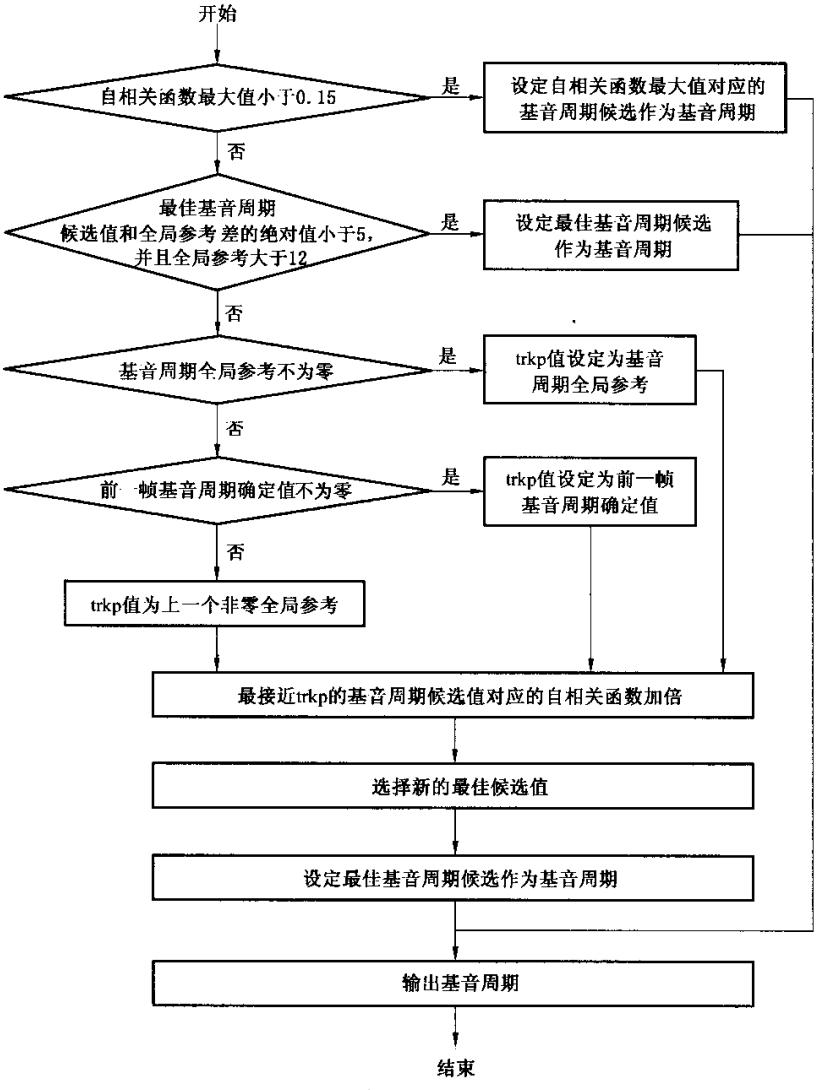


图 34 基音周期确定流程

a) 基音周期最佳候选值与基音周期全局参考的差值的绝对值小于 5，并且基音周期全局参考大于 12。

此时直接输出最佳基音周期值作为基音周期。

b) 最大自相关函数值小于阈值 0.15。

语音段信号的相关程度比较小，不易判断出明显的基音周期。当前的基音周期搜索没有实际意义，只是为闭环基音搜索提供一个最大程度去除长时相关性的参考。因而直接输出自相关函数最大值对应的基音周期搜索候选值作为基音周期。

c) 无法明显判断基音周期。

引入一个基音周期确定参考值(trkp)，该值用来对最后的基音周期的确定起到参考的作用，该值确定的步骤如下：

1) 若基音周期全局参考非零，trkp 值设定为基音周期的全局参考；

2) 否则，若前一帧基音周期确定值非零，则 trkp 值设定为前一帧基音周期确定值。

3) 否则，trkp 值设定为上一个不为 0 的基音周期全局参考。若上一个不为 0 的基音周期全

局参考保持超过 3 帧，则 trkp 值强制为 0。

利用上述的条件确定的 trkp 值，对整个基音周期候选序列进行搜索。找到基音周期候选值最接近 trkp 的一个值，将其对应的自相关函数加倍，并重新排序自相关函数值。最后将自相关函数最大的基音周期候选值作为基音周期输出。

6.2.3.4.2 脉冲响应计算

脉冲响应是指感知加权合成滤波器[见式(43)]的脉冲响应 $h(n)$。脉冲响应每个子帧计算一次，将滤波器 $A(z/\gamma_1)$的系数用零扩展后，通过滤波器 $1/\hat{A}(z)$和滤波器 $1/H_{emph}(z)$得到脉冲响应 $h(n)$。

$$H(z)W(z)=\frac{A(z/\gamma_1)}{\hat{A}(z)H_{emph}(z)} \qquad \cdots\cdots(43)$$

6.2.3.4.3 目标信号计算

目标信号 $x(n)$定义为加权信号 $s_w(n)$与感知加权合成滤波器 $H(z)W(z)$零输入响应 $\hat{s}_0$ 的差，即：

$$x(n)=s_w(n)-\hat{s}_0 \qquad \cdots\cdots(44)$$

6.2.3.4.4 自适应码书

6.2.3.4.4.1 闭环基音周期搜索

闭环基音搜索的准则是使原始信号和重建信号之间加权均方误差最小，即使 $R(k)$最大，见式(45)。

$$R(k)=\frac{\sum_{n=0}^{63}x(n)y_k(n)}{\sqrt{\sum_{n=0}^{63}y_k^2(n)}} \qquad \cdots\cdots(45)$$

式中：

$x(n)$——目标信号；

$y_k(n)$——延时 k 的滤波激励(即过去激励与 $h(n)$的卷积，$y_k(n)=h(n)\otimes exc(n-k)$)。

搜索范围限制在开环基音搜索值附近，对于第一和第三子帧，使用相应开环基音值 T_{op} 附近的值，而对于第二和第四子帧，使用前一子帧分数延时 T_1 的整数部分 $floor(T_1)$附近的值。

先计算搜索范围的第一个延时 t_{min}的卷积 $y_k(n)$，对其他整数延时 $k=t_{min}+1,\cdots,t_{max}$用下面式(46)递归计算：

$$y_k(n)=y_{k-1}(n-1)+exc(-k)h(n), \qquad n=0,\cdots,63 \qquad \cdots\cdots(46)$$

式中：

$exc(n)$，$n=-(231+17),\cdots,63$——激励缓冲器的值；

$y_{k-1}(-1)=0$。

在搜索阶段 $exc(n)$，$n=0,\cdots,63$ 是未知的，而且只有基音延迟小于 64 才需要，为使搜索简单化，将 LP 残差存入 $exc(n)$使公式(46)对所有延迟都有效。

确定最佳整数闭环延时后，则在最佳整数闭环延时附近按 1/4 样本分辨率搜索分数基音延时。内插归一化系数 $R(k)$，并搜索其最大值得到的分数基音周期。搜索使用的 FIR 插值滤波器为海明窗 sinc 函数，截断在±15 处，滤波器的截止频率(−3 dB)为 5.063 kHz。

确定分数基音延时后，在其对应的整数延时 k 和小数延时 t 处内插过去的激励 $exc(n)$来计算自适应码书激励 $v(n)$：

$$v(n)=\sum_{i=0}^{15}(exc(n-k-i)b_{64}(t+4i)+exc(n-k+1+i)b_{64}(4(i+1)-t)),$$
$$n=0,\cdots,63 \quad t=0,\cdots,3 \qquad \cdots\cdots(47)$$

式中：

内插滤波器 b_{64}——海明窗 sinc 函数，截断在±63 处，滤波器的截止频率(−3 dB)为 6.016 kHz。

6.2.3.4.4.2 自适应码书激励的滤波

由于宽带信号的周期性不一定会扩展到高频部分，所以为了改善宽带信号下自适应码书的性能，需

要对自适应码书激励信号进行滤波。

首先，对式(47)计算得到的自适应码书激励 $v(n)$做低通滤波得到其低频部分 $v\_low(n)$，其计算过程如下：

$$v\_low(n)=\sum_{k=-1}^{1}b(k)v(n-k),n=0,\cdots,63 \qquad \cdots\cdots(48)$$

式中：

$b(-1)=b(1)=0.26,b(0)=0.48$。

然后，计算自适应码书激励 $v(n)$的高频部分 $v\_high(n)$：

$$v\_high(n)=v(n)-v\_low(n),\quad n=0,\cdots,63 \qquad \cdots\cdots(49)$$

计算 LP 残差信号 $r(n)$：

$$r(n)=s(n)+\sum_{i=1}^{16}\hat{a}_i s(n-i),\quad n=0,\cdots,63 \qquad \cdots\cdots(50)$$

式中：

$s(n)$——经过预加重的信号；

$\hat{a}_i$——量化的线性预测系数。

计算 LP 残差信号 $r(n)$与自适应码书激励 $v(n)$的高频部分 $v_high(n)$的互相关 $corr$：

$$corr=\frac{\sum_{n=0}^{63}r(n)\times v\_high(n)}{\sqrt{\sum_{n=0}^{63}r^2(n)\times\sum_{n=0}^{63}v\_high^2(n)}} \qquad \cdots\cdots(51)$$

比较互相关 $corr$ 与给定阈值 $\alpha=0.19$ 的大小，并计算自适应码书增益：

a) 若 $corr>\alpha$，则最终的自适应码书激励信号为 $v(n)$；令加权合成信号为 $synth(n)$，则有 $synth(n)=h(n)\otimes v(n)$，此时增益 g_p 为(其中 $x(n)$为目标信号)：

$$g_p=\frac{\sum_{n=0}^{63}x(n)\times synth(n)}{\sum_{n=0}^{63}synth^2(n)} \qquad \cdots\cdots(52)$$

b) 若 $corr\leqslant\alpha$，则最终的自适应码书激励信号为 $v\_low(n)$；令加权合成信号为 $synth'(n)$，则有 $synth'(n)=h(n)\otimes v\_low(n)$。此时增益大小 g_p 为：

$$g_p=\frac{\sum_{n=0}^{63}x(n)\times synth'(n)}{\sum_{n=0}^{63}(synth'(n))^2} \qquad \cdots\cdots(53)$$

自适应码书增益 g_p 范围为[0,1.2]，对于一些特殊的相关性很强的信号(例如，正弦信号)，需要限制 g_p 范围在[0,0.95]，以保证 LP 滤波器的稳定。通过检测每帧量化前的 ISF 系数之间的最小距离小于 60 Hz，来保证 LP 滤波器的稳定。如果检测到 LP 滤波器可能处于不稳定状态，则限制 g_p 不超过 0.95，以防止滤波器发散。

在编码码流中使用 1 比特来标识自适应码书激励信号是否使用低通滤波。

6.2.3.4.5 代数码书

6.2.3.4.5.1 代数码书结构

代数码书结构采用的是正负交错脉冲设计。每个子帧的 64 个样本位置被分为 4 个轨道，每个轨道 16 个位置。每个轨道的脉冲个数由对应的码率所决定，具体码书结构见表 63 所示。

表 63　代数码书结构

轨　道	位　　置
1	0，4，8，12，16，20，24，28，32，36，40，44，48，52，56，60
2	1，5，9，13，17，21，25，29，33，37，41，45，49，53，57，61
3	2，6，10，14，18，22，26，30，34，38，42，46，50，54，58，62
4	3，7，11，15，19，23，27，31，35，39，43，47，51，55，59，63

6.2.3.4.5.2　10.8 kbps 模式

10.8 kbps 模式下，代数码书矢量有 4 个脉冲，每个脉冲的幅度为＋1 或－1。每个子帧的 64 个位置被分为 4 个轨道，每个轨道包含 1 个脉冲，见表 64 所示(脉冲 Pi 的序号 i 表示搜索顺序)。

表 64　10.8 kbps 代数码书脉冲结构

轨道	脉　冲	位　　置
1	P0	0，4，8，12，16，20，24，28，32，36，40，44，48，52，56，60
2	P1	1，5，9，13，17，21，25，29，33，37，41，45，49，53，57，61
3	P2	2，6，10，14，18，22，26，30，34，38，42，46，50，54，58，62
4	P3	3，7，11，15，19，23，27，31，35，39，43，47，51，55，59，63

每个轨道中 1 个脉冲的位置需要 4 比特编码，脉冲符号用 1 比特编码，4 个脉冲总共需要 4×(4＋1)＝20 比特。

6.2.3.4.5.3　12.4 kbps 模式

12.4 kbps 模式下，代数码书矢量有 6 个脉冲，每个脉冲的幅度为＋1 或－1。每个子帧的 64 个位置被分为 4 个轨道，其中轨道 1 和轨道 2 各包含 2 个脉冲，其余轨道各包含 1 个脉冲，见表 65 所示。

表 65　12.4 kbps 代数码书脉冲结构

轨道	脉　冲	位　　置
1	P0，P1	0，4，8，12，16，20，24，28，32，36，40，44，48，52，56，60
2	P2，P3	1，5，9，13，17，21，25，29，33，37，41，45，49，53，57，61
3	P4	2，6，10，14，18，22，26，30，34，38，42，46，50，54，58，62
4	P5	3，7，11，15，19，23，27，31，35，39，43，47，51，55，59，63

轨道 1 和轨道 2 中各有 2 个脉冲，各需要 2×4＋1＝9 比特。轨道 3 和轨道 4 中各有 1 个脉冲，各需要 4＋1＝5 比特，6 个脉冲总共需要 2×9＋2×5＝28 比特。

6.2.3.4.5.4　14.0 kbps 模式

14.0 kbps 模式下，代数码书矢量有 8 个脉冲，每个脉冲的幅度为＋1 或－1。每个子帧的 64 个位置被分为 4 个轨道，每个轨道各包含 2 个脉冲，见表 66 所示。

表 66　14.0 kbps 代数码书脉冲结构

轨道	脉　冲	位　　置
1	P0，P1	0，4，8，12，16，20，24，28，32，36，40，44，48，52，56，60
2	P2，P3	1，5，9，13，17，21，25，29，33，37，41，45，49，53，57，61
3	P4，P5	2，6，10，14，18，22，26，30，34，38，42，46，50，54，58，62
4	P6，P7	3，7，11，15，19，23，27，31，35，39，43，47，51，55，59，63

每个轨道中 2 个脉冲共需要 9 比特编码，8 个脉冲总共需要 4×9＝36 比特。

6.2.3.4.5.5 15.6 kbps 模式

15.6 kbps 模式下，代数码书矢量有 10 个脉冲，每个脉冲的幅度为+1 或−1。每个子帧的 64 个位置被分为 4 个轨道，其中轨道 1 和轨道 2 各包含 3 个脉冲，轨道 3 和轨道 4 各包含 2 个脉冲，见表 67 所示。

表 67 15.6 kbps 代数码书脉冲结构

轨道	脉 冲	位 置
1	P0,P1,P2	0, 4, 8, 12, 16, 20, 24, 28, 32, 36, 40, 44, 48, 52, 56, 60
2	P3,P4,P5	1, 5, 9, 13, 17, 21, 25, 29, 33, 37, 41, 45, 49, 53, 57, 61
3	P6,P7	2, 6, 10, 14, 18, 22, 26, 30, 34, 38, 42, 46, 50, 54, 58, 62
4	P8,P9	3, 7, 11, 15, 19, 23, 27, 31, 35, 39, 43, 47, 51, 55, 59, 63

轨道 1 和轨道 2 中各有 3 个脉冲，各需要 13 比特编码，轨道 3 和轨道 4 中各有 2 个脉冲，各需要 9 比特，10 个脉冲总共需要 2×13+2×9=44 比特。

6.2.3.4.5.6 17.2 kbps 模式

17.2 kbps 模式下，代数码书矢量有 12 个脉冲，每个脉冲的幅度为+1 或−1。每个子帧的 64 个位置被分为 4 个轨道，每个轨道各包含 3 个脉冲，见表 68 所示。

表 68 17.2 kbps 代数码书脉冲结构

轨道	脉 冲	位 置
1	P0,P1,P2	0, 4, 8, 12, 16, 20, 24, 28, 32, 36, 40, 44, 48, 52, 56, 60
2	P3,P4,P5	1, 5, 9, 13, 17, 21, 25, 29, 33, 37, 41, 45, 49, 53, 57, 61
3	P6,P7,P8	2, 6, 10, 14, 18, 22, 26, 30, 34, 38, 42, 46, 50, 54, 58, 62
4	P9,P10,P11	3, 7, 11, 15, 19, 23, 27, 31, 35, 39, 43, 47, 51, 55, 59, 63

每个轨道中 3 个脉冲各需要 13 比特编码，12 个脉冲总共需要 4×13=52 比特。

6.2.3.4.5.7 19.6 kbps 模式

19.6 kbps 模式下，代数码书矢量有 16 个脉冲，每个脉冲的幅度为+1 或−1。每个子帧的 64 个位置被分为 4 个轨道，每个轨道各包含 4 个脉冲，见表 69 所示。

表 69 19.6 kbps 代数码书脉冲结构

轨道	脉 冲	位 置
1	P0,P1,P2,P3	0, 4, 8, 12, 16, 20, 24, 28, 32, 36, 40, 44, 48, 52, 56, 60
2	P4,P5,P6,P7	1, 5, 9, 13, 17, 21, 25, 29, 33, 37, 41, 45, 49, 53, 57, 61
3	P8,P9,P10,P11	2, 6, 10, 14, 18, 22, 26, 30, 34, 38, 42, 46, 50, 54, 58, 62
4	P12,P13,P14,P15	3, 7, 11, 15, 19, 23, 27, 31, 35, 39, 43, 47, 51, 55, 59, 63

每个轨道各有 4 个脉冲，各需要 16 比特编码，16 个脉冲总共需要 4×16=64 比特。

6.2.3.4.5.8 21.2 kbps 模式

21.2 kbps 模式下，代数码书矢量有 18 个脉冲，每个脉冲的幅度为+1 或−1。每个子帧的 64 个位置被分为 4 个轨道，其中轨道 1 和轨道 2 各包含 5 个脉冲，轨道 3 和轨道 4 各包含 4 个脉冲，见表 70 所示。

表 70　21.2 kbps 代数码书脉冲结构

轨道	脉　冲	位　　置
1	P0,P1,P2,P3,P4	0, 4, 8, 12, 16, 20, 24, 28, 32, 36, 40, 44, 48, 52, 56, 60
2	P5,P6,P7,P8,P9	1, 5, 9, 13, 17, 21, 25, 29, 33, 37, 41, 45, 49, 53, 57, 61
3	P10,P11,P12,P13	2, 6, 10, 14, 18, 22, 26, 30, 34, 38, 42, 46, 50, 54, 58, 62
4	P14,P15,P16,P17	3, 7, 11, 15, 19, 23, 27, 31, 35, 39, 43, 47, 51, 55, 59, 63

轨道 1 和轨道 2 中各有 5 个脉冲，各需要 20 比特编码，轨道 3 和轨道 4 中各有 4 个脉冲，各需要 16 比特，18 个脉冲总共需要 2×20+2×16=72 比特。

6.2.3.4.5.9　24.4 kbps 模式

24.4 kbps 模式下，代数码书矢量有 24 个脉冲，每个脉冲的幅度为+1 或−1。每个子帧的 64 个位置被分为 4 个轨道，每个轨道各包含 6 个脉冲，见表 71 所示。

表 71　24.4 kbps 代数码书脉冲结构

轨道	脉　冲	位　　置
1	P0,P1,P2,P3,P4,P5	0, 4, 8, 12, 16, 20, 24, 28, 32, 36, 40, 44, 48, 52, 56, 60
2	P6,P7,P8,P9,P10,P11	1, 5, 9, 13, 17, 21, 25, 29, 33, 37, 41, 45, 49, 53, 57, 61
3	P12,P13,P14,P15,P16,P17	2, 6, 10, 14, 18, 22, 26, 30, 34, 38, 42, 46, 50, 54,58, 62
4	P18,P19,P20,P21,P22,P23	3, 7, 11, 15, 19, 23, 27, 31, 35, 39, 43, 47, 51, 55, 59, 63

每个轨道各有 6 个脉冲，各需要 22 比特编码，24 个脉冲总共需要 4×22=88 比特。

6.2.3.4.6　代数码书搜索前的预滤波

为提高合成信号质量，增强特定的频谱成分，在代数码书搜索前使用了预滤波器。该滤波器由两部分组成，一个是周期增强部分 $1/(1-0.85z^{-T})$（T 表示基音延时的整数部分），用来增强合成信号的谐波结构；另一个是频谱倾斜部分，同 6.2.3.1 中预加重滤波器有关。如果预加重滤波器为高频预加重，则频谱倾斜滤波器为 $(1-0.3z^{-1})$；如果预加重滤波器为低频预加重，则频谱倾斜滤波器为 $(1+0.3z^{-1})$。

6.2.3.4.7　代数码书搜索

代数码书搜索是按子帧进行，其搜索准则是使加权的输入信号和加权的合成信号之间的均方误差最小。搜索前需要对自适应码书闭环搜索中使用的目标矢量进行更新，具体方法是将原来的目标矢量减去自适应码书的贡献：

$$x'_0(n)=x_0(n)-g_p y_1(n) \qquad \cdots\cdots(54)$$

式中：

$x'_0(n)$——更新后的用于代数码书搜索的目标矢量；

$x_0(n)$——用于自适应码书闭环搜索的第一级目标矢量；

g_p——未量化的自适应码书增益；

$y_1(n)=v(n)\otimes h(n)$——自适应码矢量与感知加权合成滤波器脉冲响应的卷积。

令 g_c 表示未量化的代数码书增益，$y_2(n)=c(n)\otimes h(n)$ 表示代数码书矢量与感知加权合成滤波器脉冲响应的卷积，则使如下平方误差：

$$e=\sum_{n=0}^{N-1}[x_0(n)-g_c y_2(n)]^2 \qquad \cdots\cdots(55)$$

式中：

N——子帧长度。

最小的 g_c 为（通过 e 对 g_c 的偏导数为零得到）：

$$g_c=\frac{\sum_{n=0}^{N-1}x'_0(n)y_2(n)}{\sum_{n=0}^{N-1}y_2^2(n)} \quad \cdots\cdots(56)$$

将得到的 g_c 代入式(55)得最小平方误差为：

$$e_{min}=\sum_{n=0}^{N-1}x_0'^2(n)-\frac{[\sum_{n=0}^{N-1}x_0'(n)y_2(n)]^2}{\sum_{n=0}^{N-1}y_2^2(n)} \quad \cdots\cdots(57)$$

选择使 e_{min} 最小的代数码书激励矢量，即选择使公式(57)右边第二项最大的代数码书激励矢量作为合成信号的最佳激励。

若索引为 k 的代数码书激励矢量为 c_k，将式(57)右边第二项转换为矩阵形式：

$$Q_k=\frac{(x_0'^tHc_k)^2}{c_k^tH^tHc_k}=\frac{(d^tc_k)^2}{c_k^t\Phi c_k}=\frac{(R_k)^2}{E_k} \quad \cdots\cdots(58)$$

式中：

$d=H^tx'_0$——目标矢量 $x_0'(n)$ 和脉冲响应 $h(n)$ 的互相关；

$\Phi=H^tH$——$h(n)$ 的自相关矩阵。

矢量 d 和矩阵 Φ 在码书搜索前可预先计算为：

$$d(n)=\sum_{i=n}^{63}x_0(i)h(i-n),n=0,\cdots,63 \quad \cdots\cdots(59)$$

$$\Phi(i,j)=\sum_{n=j}^{63}h(n-i)h(n-j),i=0,\cdots,63,j=i,\cdots,63 \quad \cdots\cdots(60)$$

因为代数码矢量只有少量的非零脉冲，所以式(58)中的分子表示为：

$$C=\sum_{i=0}^{N_p-1}a_id(m_i) \quad \cdots\cdots(61)$$

式中：

m_i——第 i 个脉冲的位置；

a_i——第 i 个脉冲的幅度；

N_p——脉冲的个数。

式(58)的分母表示为：

$$E=\sum_{i=0}^{N_p-1}\phi(m_i,m_i)+2\sum_{i=0}^{N_p-2}\sum_{j=i+1}^{N_p-1}a_ia_j\phi(m_i,m_j) \quad \cdots\cdots(62)$$

由于 $d(n)$ 和 $\phi(i,j)$ 的值在码书搜索前已经计算好，a_i 的值又为 1 或 −1，所以码书搜索时仅进行简单的加减运算，这样大幅度降低码书搜索的运算量。

为了简化搜索过程，先通过参考信号 $b(n)$ 做脉冲幅度的预判决，即设置某位置上的脉冲幅度等于 $b(n)$ 在这个位置上的符号。参考信号 $b(n)$ 的计算公式如下：

$$b(n)=\frac{res'_{LTP}(n)}{\sqrt{\sum_{i=0}^{63}res'_{LTP}(i)res'_{LTP}(i)}}+\frac{d(n)}{\sqrt{\sum_{i=0}^{63}d(i)d(i)}},\quad n=0,\cdots,63 \quad \cdots\cdots(63)$$

$$res'_{LTP}(n)=res_{LTP}(n)\otimes w(n) \quad \cdots\cdots(64)$$

式中：

$w(n)$——谱倾斜滤波器 $(1-0.3z^{-1})$ 的脉冲响应；

$res_{LTP}(n)$——长时预测后的残差信号，即减去自适应码书贡献的 LP 残差信号。

参考信号 $b(n)$ 可以预测脉冲的位置，从而降低搜索复杂度。每个轨道对应 16 个不同的脉冲位置，

每个不同位置对应一个 $b(n)$值。在同一轨道内，依据 $b(n)$绝对值的大小，从小到大对 $b(n)$进行排序，记录下 8 个 $b(n)$绝对值最大值对应的脉冲位置。后续进行搜索时，可以直接对这 8 个脉冲位置进行搜索。

为简化搜索过程，在码书搜索前需要用符号信息修正 $d(n)$和 $\Phi(i,j)$。

第 1 步，计算出符号信息 $s_b(n)=\text{sign}[b(n)]$和信号 $d'(n)=d(n)s_b(n)$；

第 2 步，用符号信息修正 $\Phi(i,j)$，即 $\Phi'(i,j)=s_b(i)s_b(j)\Phi(i,j)$。

则式(61)和式(62)可分别转化为：

$$C=\sum_{i=0}^{N_p-1}d'(m_i) \qquad \cdots\cdots(65)$$

$$E=\sum_{i=0}^{N_p-1}\Phi'(m_i,m_j)+2\sum_{i=0}^{N_p-2}\sum_{j=i+1}^{N_p-1}\Phi'(m_i,m_j) \qquad \cdots\cdots(66)$$

按照从上往下顺序，依次搜索每一个轨道中的所有脉冲，在搜索完一个轨道的所有脉冲后，再开始搜索下一个轨道的脉冲。在搜索时，每次搜索确定同一轨道的两个脉冲，当一个轨道上剩余脉冲个数为一个时，可以与相邻下一轨道的第一个脉冲进行组合搜索。接着从下一轨道剩余的脉冲中确定两个脉冲继续搜索。

为了具体描述搜索方法，下面就以一个轨道两个脉冲的情形进行介绍，脉冲轨道划分见表 72 所示。

表 72　脉冲轨道示意

轨道(Tx)	脉冲	位　　置
1(T0)	P0,P1	0, 4, 8, 12, 16, 20, 24, 28, 32, 36, 40, 44, 48, 52, 56, 60
2(T1)	P2,P3	1, 5, 9, 13, 17, 21, 25, 29, 33, 37, 41, 45, 49, 53, 57, 61
3(T2)	P4,P5	2, 6, 10, 14, 18, 22, 26, 30, 34, 38, 42, 46, 50, 54, 58, 62
4(T3)	P6,P7	3, 7, 11, 15, 19, 23, 27, 31, 35, 39, 43, 47, 51, 55, 59, 63

假设脉冲轨道 1 到 4 分别用 T0、T1、T2 和 T3 表示，在轨道 T0 中，搜索脉冲 P0 和 P1。脉冲 P0 的位置由脉冲位置参考信号 $b(n)$在 T0 中的 8 个最大值所对应的位置中进行搜索确定。脉冲 P1 在轨道 T0 中的 16 个位置进行全搜索，总的搜索次数为：8×16=128(次)。在脉冲 P0 和 P1 搜索完成以后，判断轨道 T0 中是否还有未搜索脉冲，本实例中，轨道 T0 的所有脉冲已经搜索完毕，接下来开始搜索轨道 T1 的脉冲。最佳脉冲的判断准则是使公式(58)取值最大。

在轨道 T1、T2 和 T3 中的搜索方法与轨道 T0 中的方法相同。当四个轨道的所有脉冲搜索完毕之后，整个搜索过程结束，输出所有脉冲的最佳位置和符号。整个搜索过程中脉冲搜索顺序为：P0-P1、P2-P3、P4-P5 和 P6-P7，搜索次数为：8×16×4=512(次)。最后，再依次搜索脉冲起始位置(P0 的位置)在轨道 T1、T2 和 T3 的情形。总搜索次数为：512×4=2 048(次)。

6.2.3.4.8　代数码书矢量的编码

6.2.3.4.8.1　代数码书矢量的编码过程

对每个轨道搜索出的脉冲使用分类组合索引编码，编码过程如下：

a) 对轨道上需要编码的脉冲按照位置进行统计，获得有脉冲位置的数目 pos_num(设 pos_num 的值为 N)、有脉冲位置在轨道上的分布 P(N)和各个有脉冲位置上的脉冲数目 SU(N)；

b) 按照有脉冲位置的数目 pos_num 确定第一索引 I1。第一索引表示了在相同有脉冲位置的数目条件下，有脉冲位置在轨道上全部可能的分布情况；

c) 按照有脉冲位置在轨道上的分布 P(N)确定第二索引 I2；

d) 按照各个有脉冲位置上的脉冲数目 SU(N)确定第三索引 I3；

e) 最后生成编码索引 Index(N)，编码索引包括第一、二、三索引和脉冲符号索引信息。

分类组合索引编码步骤见 6.2.3.4.8.2～6.2.3.4.8.6。图 35 给出了分类组合索引编码的处理流程。

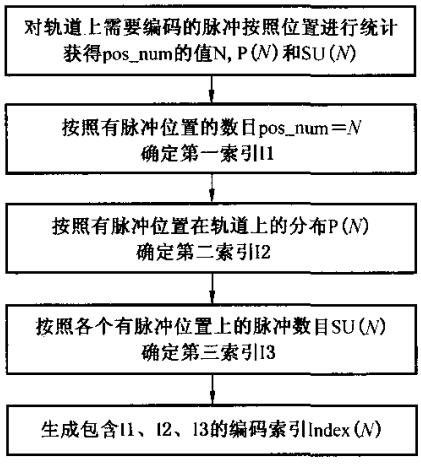


图 35　代数码书分类组合索引编码流程框图

6.2.3.4.8.2　预处理(脉冲统计)

对轨道上需要编码的脉冲按照位置进行统计,获得有脉冲位置的数目,有脉冲位置在轨道上的分布和各个有脉冲位置上的脉冲数目。

pulse_num 表示需要编码的脉冲总数,与码率模式相对应,设 pulse_num=ω。pos_num 表示有脉冲位置的数目,设 pos_num=N,由于 ω 个脉冲在轨道上的分布可能出现位置重叠,显然有 $N\in[1,\omega]$。有脉冲位置矢量 $P(N)=\{p(0),p(1),\cdots,p(N-1)\}$表示有脉冲位置在轨道上的分布。$p(n)$表示有脉冲位置在轨道上的位置序号,$n\in[0,N-1]$,$p(n)\in[0,M-1]$,$M=16$ 表示轨道上的位置总数,且 $0\leqslant p(0)<p(1)<\cdots<p(N-1)\leqslant 15$。脉冲数目矢量 $SU(N)=\{su(0),su(1),\cdots,su(N-1)\}$,表示各个有脉冲位置上的脉冲数目。$su(n)$表示 $p(n)$位置的脉冲数目,$su(0)+su(1)+\cdots+su(N-1)=\omega$。

对轨道上需要编码的脉冲按照位置进行统计时,还需要获得各个有脉冲位置的脉冲符号信息。脉冲符号矢量 $S(N)=\{s(0),s(1),\cdots,s(N-1)\}$表示各个有脉冲位置的脉冲符号信息,$s(n)$表示 $p(n)$位置的脉冲符号。采用 $s(n)=0$ 表示正脉冲,$s(n)=1$ 表示负脉冲的编码方法。

6.2.3.4.8.3　第一索引 I1(按位置数分类)

按照有脉冲位置的数目 pos_num=N 确定第一索引 I1。

每个轨道上共有 16 个位置。编码的脉冲数目为 6 时,有脉冲的位置数分别是 1、2、3、4、5、6,与其对应的第一索引分别为 0x1E0000、0x1D0000、0x1C0000、0x080000、0x100000、0x000000;编码的脉冲数目为 5 时,有脉冲位置的数目分别为 1、2、3、4、5,与其对应的第一索引分别为 0x00000、0x08000、0x10000、0x20000、0x40000;编码的脉冲数目为 4 时,有脉冲位置的数目分别为 1、2、3、4,与其对应的第一索引分别为 0x0000、0x2000、0x4000、0x8000;编码的脉冲数目为 3 时,有脉冲位置的数目分别为 1、2、3,与其对应的第一索引分别为 0x1C00、0x1800、0x0000;编码的脉冲数目为 2 时,有脉冲位置的数目分别为 1、2,与其对应的第一索引分别为 0x1E0、0x000;编码的脉冲数目为 1 时,有脉冲位置的数目为 1,与其对应的第一索引是 0x00。

6.2.3.4.8.4　第二索引 I2(位置编码)

第二索引 I2 指示当前有脉冲位置的分布情况。

将有脉冲位置在轨道上的分布对应为一个 N 维脉冲位置矢量 $P(N)$:

$$P(N)=\{p(0),p(1),\cdots,p(N-1)\} \quad \cdots\cdots(67)$$

式中:

$P(N)$的可能组合样本数为 C_M^N。这些可能的组合样本按 $p(0)$小的矢量排列在前,$p(0)$相同时,

$p(1)$小的矢量排列在前，后面依次类推顺序排列。这样所有可能的脉冲位置矢量就有了一个大小为C_M^N排序表。一个N维脉冲位置矢量$P(N)$在排序表的位置序号就是第二索引$I2$，计算公式如下：

$$I2 = C_M^N - C_{M-p(0)}^N + \sum_{n=1}^{N-1}[C_{M-p(n-1)-1}^{N-n} - C_{M-p(n)}^{N-n}] \quad \cdots\cdots(68)$$

式中：

C_M^N——$I2$全部可能的取值数，$I2$的值从0开始计数，且$I2 \in [0, C_M^N - 1]$。

6.2.3.4.8.5 第三索引 I3(按有脉冲位置上脉冲数量分布分类)

$SU(N)$与$P(N)$是同维度的矢量，但受限于$su(0)+su(1)+\cdots+su(N-1)=\omega$，且$\omega$的数值通常不大，一般为1～6，因此$SU(N)$的可能组合样本数$Class(N)$较小。$SU(N)$与第三索引I3关系在高维度情况下采用查询关系，在低维度情况采用计算关系。在某些极端情况下，例如$N=1$或$N=\omega$，此时$SU(N)$只有一种可能情况，无须由I3进行指示，可不编码I3。索引I3的值从0开始计数，且$I3 \in [0, Class(N)-1]$。

6.2.3.4.8.6 编码索引生成

由于$I2$、$I3$一般不能表示为2的整数幂，所以$I2$、$I3$合并成$I23$，合并公式如下：

$$I23 = I3 \times C_M^N + I2 \quad \cdots\cdots(69)$$

编码索引$Index(N)$还包含各个脉冲符号索引s(n)的信息，长度为N的脉冲符号矢量$S(N)$字段被添加到编码索引的最后N位。

编码索引Index(N)的表示如下：

$$Index(N) = I1 + I23 \times 2^N + s(0) \times 2^{N-1} + s(1) \times 2^{N-2} + \cdots + s(N-1) \quad \cdots\cdots(70)$$

6.2.3.4.8.7 各个脉冲数量下的索引编码的比特分配情况

各种情况下脉冲索引编码的比特详细分配情况见表73～表78。

表73 6脉冲比特分配表

脉冲位置数	编码起始值	比特														
		21	20	19	18	17	16	15	…	6	5	4	3	2	1	0
6	0x000000	0	0	0	$I2$						s(0)	s(1)	s(2)	s(3)	s(4)	s(5)
5	0x100000	0	1	$I3\times4368+I2$								s(0)	s(1)	s(2)	s(3)	s(4)
4	0x080000	0	0	1	$I3\times1820+I2$							s(0)	s(1)	s(2)	s(3)	
3	0x1C0000	0	1	1	1	0	0	$I3\times560+I2$					s(0)	s(1)	s(2)	
2	0x1D0000	0	1	1	1	0	1	$I3\times120+I2$						s(0)	s(1)	
1	0x1E0000	0	1	1	1	1	0	$I2$							s(0)	

表74 5脉冲比特分配表

脉冲位置数	编码起始值	比特														
		19	18	17	16	15	14	…	6	5	4	3	2	1	0	
5	0x40000	0	1	$I2$							s(0)	s(1)	s(2)	s(3)	s(4)	
4	0x20000	0	0	1	$I3\times1820+I2$							s(0)	s(1)	s(2)	s(3)	
3	0x10000	0	0	0	1	$I3\times560+I2$							s(0)	s(1)	s(2)	
2	0x08000	0	0	0	0	1	$I3\times120+I2$							s(0)	s(1)	
1	0x00000	0	0	0	0	0	I2								s(0)	

表 75　4 脉冲比特分配表

脉冲位置数	编码起始值	比特													
		15	14	13	12	11	10	…	6	5	4	3	2	1	0
4	0x8000	1	*I2*									s(0)	s(1)	s(2)	s(3)
3	0x4000	0	1	*I3*×560 + *I2*									s(0)	s(1)	s(2)
2	0x2000	0	0	1	*I3*×120 + *I2*									s(0)	s(1)
1	0x0000	0	0	0	0	*I2*									s(0)

表 76　3 脉冲比特分配表

脉冲位置数	编码起始值	比特												
		12	11	10	9	8	7	6	5	4	3	2	1	0
3	0x0000	0	*I2*									s(0)	s(1)	s(2)
2	0x1800	1	1	0	*I3*×120 + *I2*								s(0)	s(1)
1	0x1C00	1	1	1	*I2*									s(0)

表 77　2 脉冲比特分配表

脉冲位置数	编码起始值	比特								
		8	7	6	5	4	3	2	1	0
2	0x000	0	*I2*						s(0)	s(1)
1	0x1E0	1	1	1	1	*I2*				s(0)

表 78　单脉冲比特分配表

脉冲位置数	编码起始值	比特				
		4	3	2	1	0
1	0x00	*I2*				s(0)

6.2.3.4.9　自适应和代数码书增益量化

每个子帧的自适应码书和代数码书增益使用 7 比特进行联合矢量量化，其中自适应码书增益 g_p 直接量化，代数码书增益 g_c 通过对校正因子 γ 和每帧的代数码书平均能量 $\bar{E}_i$ 进行量化来实现。每帧的代数码书平均能量使用 2 比特量化。

设 $E_i(n)$ 是第 n 个子帧代数码书激励矢量的能量，计算公式如下：

$$E_i(n)=10\log_{10}\left(\frac{1}{N}g_c^2\sum_{i=0}^{N-1}c^2(i)\right)=20\log_{10}(g_c)+E_i \quad \cdots\cdots(71)$$

式中：

$N=64$——子帧的长度；

$c(i)$——代数码书激励矢量；

E_i 是按下式计算的能量值：

$$E_i=10\log_{10}\left(\frac{1}{N}\sum_{i=0}^{N-1}c^2(i)\right) \quad \cdots\cdots(72)$$

计算代数码书平均能量 $\bar{E}_i$ 并量化，然后用于计算代数码书预测增益 g'_c：

$$g'_c=10^{0.05(\bar{E}_i-E_i)} \quad \cdots\cdots(73)$$

上式是从公式 $\bar{E}_i=20\log_{10}(g'_c)+E_i$ 中推导出的。

代数码书增益 g_c 和预测增益 g'_c 之间的校正因子 γ 由下式得到：

$$\gamma = g_c / g'_c \qquad \cdots\cdots(74)$$

每个子帧的自适应码书增益 g_p 和代数码书增益的校正因子 γ 采用 7 比特联合矢量量化，码书的搜索准则使原始信号和重建信号之间加权均方误差最小，计算公式见下：

$$E = \frac{1}{N}\sum_{n=0}^{N-1}[x_0(n) - \hat{g}_p y_1(n) - \hat{g}_c y_2(n)]^2 \qquad \cdots\cdots(75)$$

式中：

$x_0(n)$——自适应码书搜索的目标信号；

$y_1(n)$——自适应码书矢量同感知加权合成滤波器脉冲响应的卷积；

$y_2(n)$——代数码书矢量同感知加权合成滤波器脉冲响应的卷积；

$\hat{g}_p$——量化的自适应码书增益；

$\hat{g}_c$——量化的代数码书增益。

代数码书平均能量 $\bar{E}_s$ 的计算和量化过程如下：

首先计算每个子帧 LP 预测残差的能量：

$$E_{res}(n) = 10\log_{10}\left(\frac{1}{N}\sum_{i=0}^{N-1} r^2(i)\right) \qquad \cdots\cdots(76)$$

然后计算每帧的平均残差能量：

$$\bar{E}_{res} = \frac{1}{4}\sum_{n=0}^{3} E_{res}(n) \qquad \cdots\cdots(77)$$

从残差能量中减掉自适应码书的贡献，得到每帧的代数码书平均能量 $\bar{E}_s$。这通过减掉每帧开环基音搜索得到的归一化自相关能量平均值实现，即：

$$\bar{E}_s = \bar{E}_{res} - 10\bar{R} \qquad \cdots\cdots(78)$$

式中：

$\bar{R}$——开环基音搜索得到的归一化自相关能量平均值。

平均能量 $\bar{E}_s$ 每帧用 2 比特量化，有 4 个量化级别：18，30，42，54。通过每次对 $\bar{E}_s$ 加上 12(量化索引加 1)迭代，来限制 $\bar{E}_s$ 量化范围 $(E_{max}-27)<\bar{E}_s\leqslant 54$。其中 E_{max} 是 4 个子帧中 $E_{res}(n)$ 最大值。

6.2.4 TVC 编码

6.2.4.1 TVC 编码过程

TVC 编码过程见图 36。

变换域矢量编码技术(TVC)编码步骤简述如下：

a) 音频输入信号通过感知加权滤波器得到目标信号；

b) 加自适应窗；

c) 通过 FFT 变换将时域信号变换到频域；

d) 在变换域中，进行峰值预整形和缩放因子调整，以减少低频噪声；

e) 对预整形后的信号进行基于变长分裂表矢量量化；

f) 增益平衡和峰值逆整形；

g) 逆时频变换，将频域信号变换到时域，得到量化后的时域信号；

h) 计算和量化全局增益；

i) 加自适应窗和重叠加，以减少因变换域量化而引起的块效应；

j) 为下一帧保存重叠信号；

k) 重建信号通过逆感知加权滤波器和 LP 分析滤波器得到激励信号，以更新 ACELP 的自适应码书，允许 TVC 和 ACELP 模式之间切换。

编码码流需要传输四个参数：噪声因子、缩放因子、频谱的量化值、全局增益。6.2.4.2～6.2.4.12 将详细阐述编码算法工作流程。

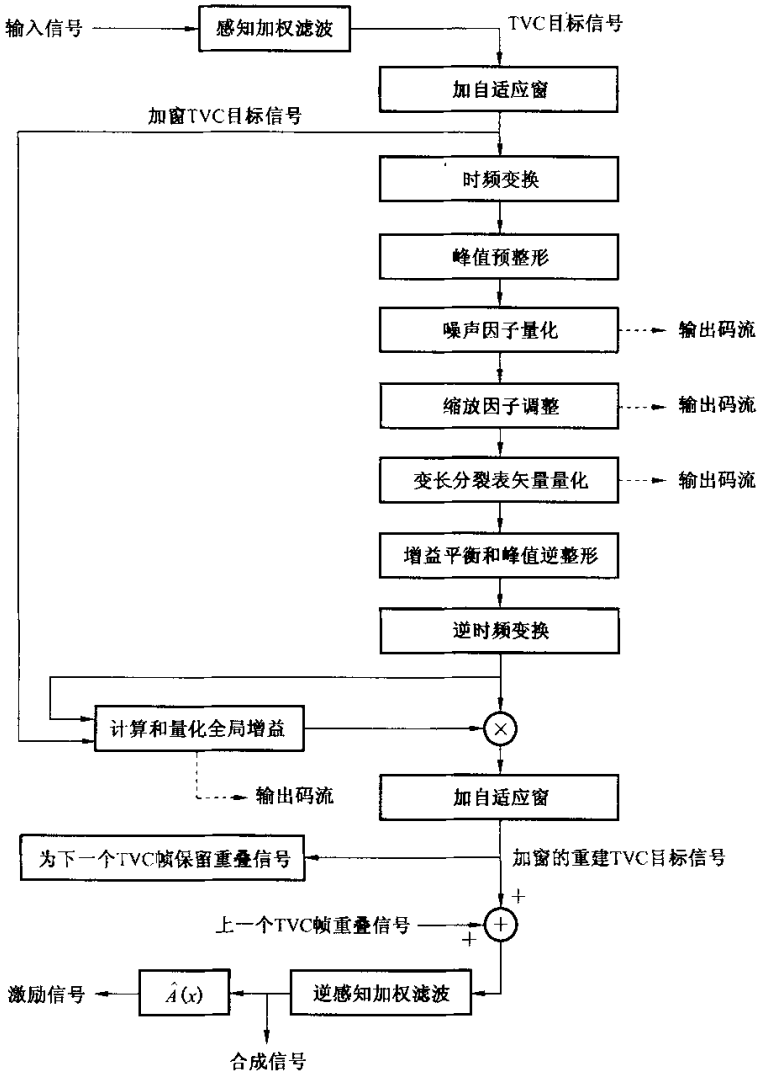


图 36 TVC 编码过程图

6.2.4.2 计算目标信号

对输入的信号进行感知加权滤波得到目标信号，后续计算都针对目标信号进行，感知加权滤波器传输函数如下：

$$W(z)=\frac{\hat{A}(z/\gamma_1)}{H_{emph}(z)} \qquad \cdots\cdots(79)$$

式中：

$\gamma_1=0.92$；

$\hat{A}(z)$——由量化系数构成的 LP 滤波器；

$H_{emph}(z)$——预加重滤波器。

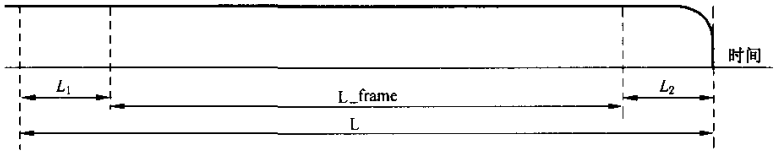
感知加权滤波器的详细描述见 6.2.3.3。

6.2.4.3 加自适应窗

自适应窗的具体窗型同上一帧编码模式以及编码模式切换有关。假设 L_frame 表示当前帧输入信号的长度；L 表示当前帧自适应窗的长度；L_1 表示与上一帧重叠的样本长度；L_2 表示与下一帧重叠

的样本长度。

上一帧使用 ACELP 编码时，自适应窗如图 37 所示：



$L_1=16$；

$L_2=16$；

L_frame=256；

L=288。

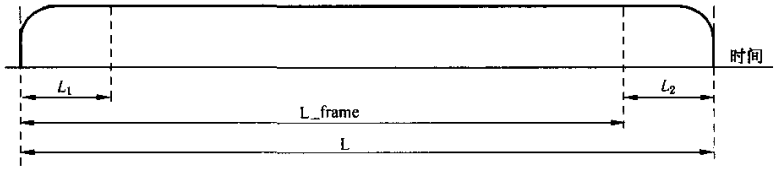
图 37　上一帧 ACELP 编码时自适应窗

图 37 中自适应窗的窗函数为：

$$\begin{aligned} w_1(n)&=1, n=0,\cdots,L-L_2-1 \\ w_2(n)&=\cos(2\pi n/(4L_2)), n=0,\cdots,L_2-1 \end{aligned} \quad \cdots\cdots(80)$$

因为当前帧与下一帧的重叠部分长度为 L_2，所以当下一帧还是 TVC 编码时，下一帧帧头所加的窗长要和 L_2 长度一致。

上一帧使用 TVC 编码时，自适应窗如图 38 所示：



$L_1=16$(上一帧由 ACELP 切换为 TVC 编码)或 32(上一帧没有发生模式切换)；

$L_2=32$；

L_frame=256；

L=288。

图 38　上一帧 TVC 编码时自适应窗

图 38 中对应自适应窗的窗函数为：

$$\begin{aligned} w_1(n)&=\sin(2\pi n/(4L_1)), && n=0,\cdots,L_1-1 \\ w_2(n)&=1, && n=0,\cdots,L-L_1-L_2-1 \\ w_3(n)&=\cos(2\pi n/(4L_2)), && n=0,\cdots,L_2-1 \end{aligned} \quad \cdots\cdots(81)$$

6.2.4.4　时频变换

加窗信号通过 DFT 变换到频域：

$$X(k)=\sum_{n=0}^{L_{DFT}-1} x(n)e^{\frac{-2\pi kn}{L_{DFT}}} \quad \cdots\cdots(82)$$

式中：

L_{DFT}——DFT 变换长度，取 288 个样本，具体实现可以使用基 9 的 FFT 快速算法。

6.2.4.5　峰值预整形

峰值预整形算法见图 39 所示：

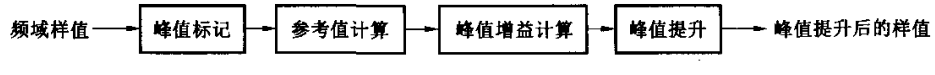


图 39　峰值预整形算法流程

处理步骤如下：

a) 计算频谱的幅值 $M(n)=\sqrt{\mathrm{Re}^2(n)+\mathrm{Im}^2(n)}$；

b) 标记前 1/4 频谱中的峰值集合$\{p_i\}$，p_i 定义为整形频谱段幅值的局部最大值。若 $M(n)>M(m)$，$\forall m\in[n-j,n+j]$，$m\neq n$，则 $M(n)\in\{p_i\}$表示一个 $2j+1$ 点局部的最大值，实际中 j 选择为 1；

c) 计算参考值 $ref_{\max}=\sqrt{E_{\max}/8}$，$E_{\max}$为前 1/4 频谱中划分八维频谱矢量块的最大能量；

d) 计算峰值 p_i 的放大因子 $R_i=(ref_{\max}/p_i)^{1/2}$。如果 $R_i>R_{i-1}$，则 $R_i=R_{i-1}$，以保证放大因子的递减性；

e) 在峰值集合$\{p_i\}$中除去 $ref_{\max}$相关的峰值点，保证 $ref_{\max}$的不变性。对剩余的峰值点 p_i 进行放大 $p_i=p_iR_i$。

该模块通过提升低频峰值，达到减小低频中较小峰值处量化噪声的目的。由于只对少量的频谱点进行放大，对全局增益影响很小。

6.2.4.6 噪声因子量化

预整形的频谱 X 划分成 $K=N/8$ 个八维矢量，定义 B_k 为第 k 个矢量，$k=0,1,\cdots,K-1$。计算 B_k 的能量 E_k 为：

$$E_k=\max\left(2,\sum_{m=0}^{7}B_k[m]B_k[m]\right) \qquad \cdots\cdots(83)$$

根据 E_k 得到消耗比特数的初始估计：

$$R_k(1)=5\log_2\left(\frac{E_k}{2}\right) \qquad \cdots\cdots(84)$$

比特消耗估计迭代：

初始条件：设 fac=128，offset=0 和 nbits_max=0.95×($b_{\max}$−K)，$b_{\max}$表示频谱量化可用比特数。

迭代执行 10 次：

a) offset=offset+fac；

b) $nbits=\sum_{k=1}^{K}\max(0,R_k(1)-offset)$；

c) if(nbits<=nbits_max)offset=offset−fac；

d) fac=fac/2；

迭代过程完成后，确定了参数 offset 值，再进行下面的迭代：

初始条件：nbits=0；n=1；

迭代过程：

```
for (k=K/2;k<=K-1;++k)
{
  tmp=Rk(1)-offset;
  if(tmp<5)
  {
    nbits=nbits+tmp;
    n=n+1;
  }
}
```

迭代过程完成后，nbits=nbits/n。

噪声因子计算：

$$\delta_{noise}=10^{\frac{\log_{10}(2)}{10}(nbits-5)} \qquad \cdots\cdots(85)$$

噪声因子范围在 0.1～0.8 之间，使用 3 比特量化，量化索引为 $idx=floor((8-10\times\delta_{noise})+0.5)$。

6.2.4.7 缩放因子控制

选择合适的缩放因子对整个频谱的不同部分进行调整，使各缩放因子控制的频带内量化噪声的分布更合理。TVC 采用两个缩放因子 g_1 和 g_2 分别对频谱样值进行调整，最终对调整后的频谱样值进行量化。缩放因子调整见图 40。

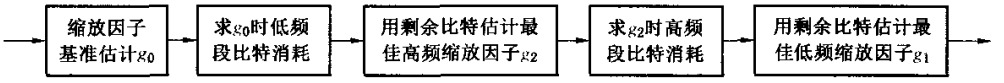


图 40 两个缩放因子的调整算法

假设频谱样值序列为 $X(0,1,\cdots,N)$，先将整个频带等分为低频带 $X(0,1,\cdots,N/2)$和高频带$X(N/2+1,\cdots,N)$，消耗比特数的估计函数为 $b=cons(X,N,g)$，b_{max}表示频谱量化可用比特数。按照下列步骤调整缩放因子：

a) 选择 $g=g_0$，使得全频带的比特消耗 $b_0=cons(X(0,1\cdots,N),N,g_0)$满足 $b_0<b_{max}$且最小化$(b_{max}-b_0)$，则 g_0 将作为两个缩放因子调整的基准值；

b) 计算低频带的比特消耗 $b_l=cons(X(0,1,\cdots,N/2),N/2,g_0)$；

c) 选择 $g=g_2$，使得高频带的比特消耗 $b_h=cons(X(N/2,\cdots,N),N/2,g_2)$满足 $b_n<b_{max}-b_l$ 且最小化$(b_{max}-b_l-b_h)$；

d) 选择 $g=g_1$，使得低频带的比特消耗 $b_l=cons(X(0,1,\cdots,N/2),N/2,g_1)$，满足 $b_l<b_{max}-b_h$且最小化$(b_{max}-b_l-b_h)$；

e) 用缩放因子 g_1 和 g_2 对序列 X 进行缩放。缩放后的频谱为$X'=[X(0,1,\cdots,N/2)/g_1,X(N/2+1,\cdots,N)/g_2]$，最后将 X'送入矢量量化器；

f) 对 g_1/g_2 进行 7 比特编码传输，为了保持编码比特数不变，应相应减少量化可用比特数。

6.2.4.8 变长分裂表矢量量化

6.2.4.8.1 变长分裂表矢量量化过程

TVC 使用格型量化器量化经过缩放的频谱 X'，频谱划分成八维矢量，使用由 Gosset 格子集(又称为 RE_8)构成的矢量码表量化。格的生成矩阵 G 产生一个格中的所有点，c=kG，k 是由整数值组成的行矢量，c 就是产生的格点。为了生成满足给定码率的矢量码表，只考虑位于给定半径的球面上的格点。使用多个不同的半径，就可以生成多码率码表。

基于分裂表的矢量量化方法框图见图 41，图中的变量 header 表示码头，split header 表示分裂量的码头，even_flag 则表示偶标志位。基于分裂表的矢量量化方法过程如下：

a) 在格中找到待编码数据 x 的最近邻点 y；

b) 判断 y 是否在基础码书 C 中，如果在，则直接计算索引 i，并编码输出到码流；

c) 若 y 不在基础码书 C 中，判断 y 是否属于 $2D_8$。如果 y 属于 $2D_8$，则将其减 1，置偶标志位(even_flag)为 1；如果 y 不属于 $2D_8$，则使用分裂表编码；

d) 再判断 y 是否在基础码书 C 中，如果在，则直接计算索引 i，并编码输出到码流；如果不在，则使用分裂表编码；

e) 使用分裂表编码时，将 y 中的每一个分量 $y(i)$分裂为 $c(i)$与分裂表中的某一个值 $y'(i)$的和。要选择合适的 $y'(i)$使得 $c(i)$的绝对值最小，并且使得生成的八个 $c(i)$组成的八维矢量为基础码书 C 中的某一矢量 c。分裂之后再检测八个 $y'(i)$值的大小，若均小于等于一级扩展阈值，则采用一级扩展编码方式，否则采用二级扩展编码方式。

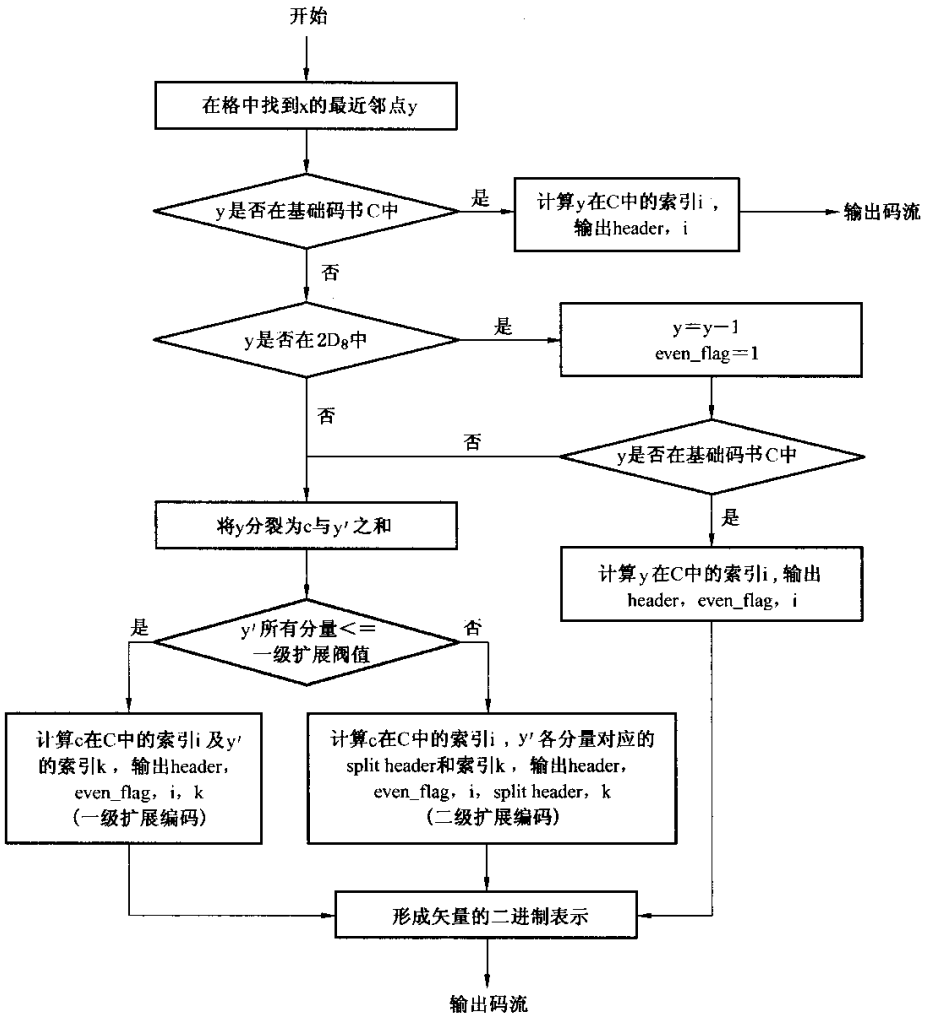


图 41　基于分裂表的矢量量化方法编码示意图

基于分裂表的矢量量化方法，首先判断待编码数据是否在基础码书中，如果在则直接利用基础码书编码；否则，尝试将其分裂为基础码书中的码字和分裂表中一个分裂量的和，再对基础码字和分裂量分别编码。详细过程见 6.2.4.8.2～6.2.4.8.5。

6.2.4.8.2　寻找最近邻点

将预整形后的频谱数据分组，每八个数为一组，组成一个八维的矢量 x，在格中寻找与该矢量最接近的点，即最近邻点 y。

6.2.4.8.3　基础码书的选取

如果编码矢量不在基础码书中，基于分裂表的矢量量化方法尝试进行分裂处理。为适应分裂处理，选取如下的 RE_8 基础码书：

$RE_8 = 2D_8 \cup \{2D_8 + (1,1,\cdots,1)\}$ 其中 $D_8 = \{(x_1,x_2,\cdots,x_8) \in Z^8 \mid x_1+\cdots+x_8 \text{ 为偶}\}$

RE_8 集合中所有数据之和是 4 的倍数，并且奇偶性相同。

基础码书 Q_0，Q_2，Q_3，Q_4 和 inv_Q_4 的特征码字定义见表 79。

表 79　基础码书的特征码字定义

特　征　码　字	Q_0	Q_2	Q_3	Q_4	inv_Q_4
(0,0,0,0,0,0,0,0)	√				
(2,0,0,0,0,0,0,0)		√	√		
(1,1,1,1,1,1,1,1)		√	√		
(2,2,0,0,0,0,0,0)		√	√		
(2,2,2,2,0,0,0,0)			√		
(3,1,1,1,1,1,1,1)			√		
(4,0,0,0,0,0,0,0)			√		
(3,3,1,1,1,1,1,1)				√	
(1,1,3,3,3,3,3,3)					√
(4,2,2,0,0,0,0,0)			√		
(3,3,3,1,1,1,1,1)				√	
(1,1,1,3,3,3,3,3)					√
(4,4,0,0,0,0,0,0)			√		
(5,1,1,1,1,1,1,1)				√	
(3,3,3,3,1,1,1,1)				√	
(5,3,1,1,1,1,1,1)				√	
(6,2,0,0,0,0,0,0)			√		
(5,3,3,1,1,1,1,1)				√	
(5,5,1,1,1,1,1,1)				√	
(7,1,1,1,1,1,1,1)				√	
(7,3,1,1,1,1,1,1)				√	
(3,3,3,3,3,3,3,1)				√	
(3,3,3,3,3,3,3,3)				√	
(9,1,1,1,1,1,1,1)				√	
(11,1,1,1,1,1,1,1)				√	
(13,1,1,1,1,1,1,1)				√	

不同的基础码书是用一个不同长度的二进制数来标识的，即基础码书的标识位 header，具体的表示方式如下：

header＝0　　基础码书为 Q_0

header＝10　　基础码书为 Q_2

header＝1 100　　基础码书为 Q_3

header＝1 110　　基础码书为 Q_4

header＝11 111 110　　基础码书为 inv_Q_4

6.2.4.8.4　基础码书编码

a)　直接编码

首先在基础码书中查找格中 x 的最近邻点 y。如果在基础码书 Q_0，Q_2，Q_3，Q_4 中，则直接计算码字 y 在基础码书中的索引 i。将索引 i 和基础码书的标识位 header 打包输出，输出格式见图 42。由于基础码书 Q_0 只包括一个特征码字(0,0,0,0,0,0,0,0)，因此只编码码书的

header，不编码码书索引。

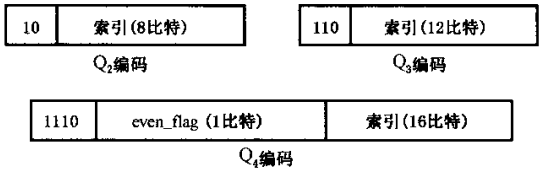


图 42　Q_2，Q_3，Q_4 输出编码格式

索引 i 的计算和编码过程如下：

——分解初始矢量，获得符号和初始矢量绝对值；

——对符号进行编码，获得符号编码；

——对初始绝对值矢量进行分层组合编码，获得绝对值矢量编码；

——组合符号编码及绝对值矢量编码，获得初始矢量的编码；

——初始矢量的编码加上矢量所属特征码字在整个基础码书中偏移量，得到基础码书的索引。

b)　特殊特征码字的编码

在对基础码书 inv_Q_4 进行搜索时，若检测到待编码数据符合特殊特征码字(1,1,1,3,3,3,3,3)或(1,1,3,3,3,3,3,3)，则将其进行位反转为(3,3,3,1,1,1,1,1)和(3,3,1,1,1,1,1,1)，并置反转标志位，即给数据加码头 header："11111110"。然后按直接编码方法计算(3,3,3,1,1,1,1,1)和(3,3,1,1,1,1,1,1)在码书 Q_4 中的索引，此时的输出格式见图 43。

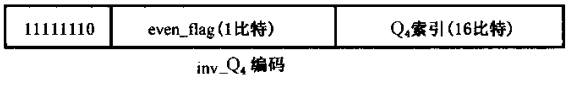


图 43　特殊特征码字的输出编码格式

c)　$2D_8$ 数据的奇化编码

在对基础码书进行搜索时，若码字 y 不在基础码书中，则还需判断它是否为 $2D_8$ 数据。若为 $2D_8$ 数据，则置偶标志位为 1，即 $even\_flag=1$。

置偶标志位后，将 y 的每一个分量都减 1，得到一个各分量均为奇数的矢量。此时再判断码字 y 是否在基础码书 Q_4 中。若在，则按直接编码方法计算其在 Q_4 中的索引作为输出；若为特殊特征码字，则按特殊特征码字的编码方法编码；若仍不在基础码书中，则使用分裂表编码。

d)　基于缓存的常见特征码字快速码书搜索

基于缓存的快速码书搜索方法，是借鉴缓存使用机理的一种 VQ 快速搜索方法。该方法基于 RE_8 格矢量量化中八位一组有许多是重复的数据，如果能够有效击中这些重复的数据，那么将大大提高搜索速度，并在可能的范围内节省比特。编码流程如图 44 所示。主要步骤如下所示：

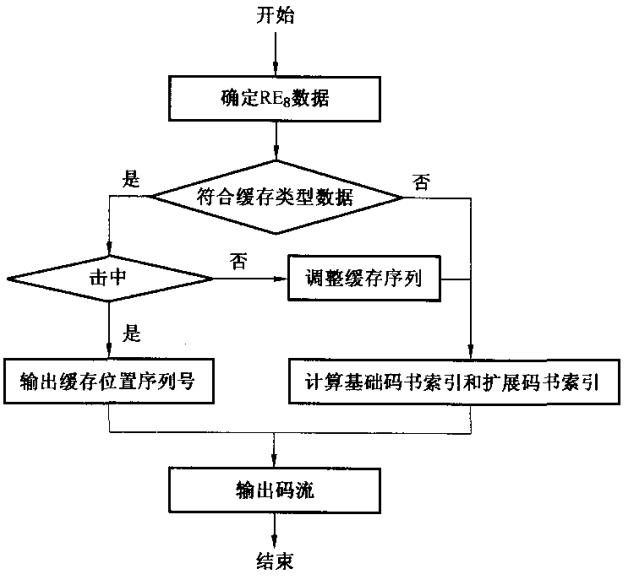


图 44　基于缓存的快速码书搜索编码流程图

第 1 步：确定 RE_8 数据

将预整形后的频谱数据分组，每八个数为一组，根据就近原则将这八个数量化为 RE_8 集合上的点。

第 2 步：确定缓存类型数据和缓存空间

将码书中出现机率最高的特征码字选出来，遇到这类数据时便对其进行与缓存相关的操作，这些数据称之为缓存类型数据。选取特征码字为(1,1,1,1,1,1,1,1)和(2,2,0,0,0,0,0,0)的数据作为缓存类型数据。选取了 16 个空间作为缓存空间，即如果用二进制表示，需要 4 比特。

编码时遇到这类数据，在缓存中查找是否有与其相同的数据。如果待编数据不是缓存类型数据，则直接计算其码书索引并输出到码流。如果是缓存类型数据，则将其缓存序列号输出到码流。

第 3 步：编码时缓存中的数据操作

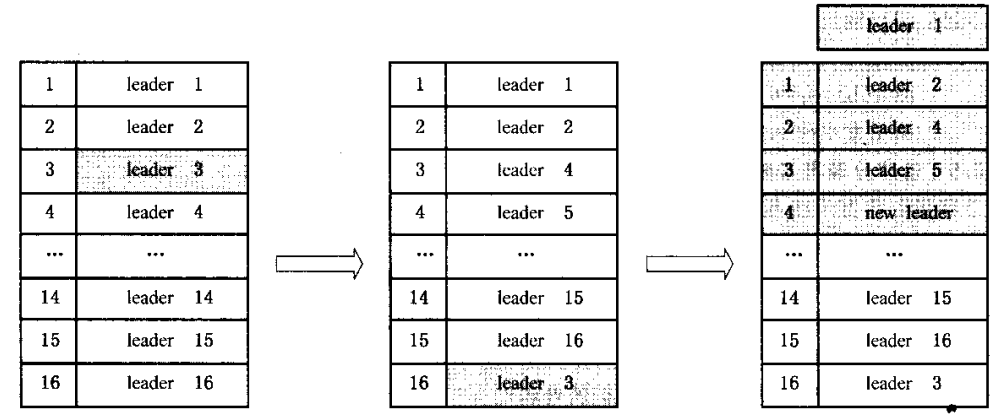
当判断得知待编码数据为缓存类型后，在缓存中查找是否有与其相同的数据。如果有，称之为“击中”，没有则称之为“未击中”。对于击中了的数据，输出其相应的缓存序列号；对于未击中的数据则将其放入缓存中。此时如果缓存已占满，则需要有一个数据被替换出来。采用的替换原则是：如果数据刚入缓存，则编号为 4；如果缓存中的数据被击中，则沉入最底部，编号为 16；编号为 1 的数据，最先被替换，每次数据被击中或者刚进来，编号都会有所改变。

具体操作过程见图 45。leader1 到 leader16 及 New leader 均为特征码字的标号。

图 45 a)为某状态下缓存中的数据存储情况。图中 16 个缓存空间均已填满。第 1 列为缓存空间编号，第 2 列为所存的特征码字。此时，若下一个数据与缓存中的某个数据相同，即击中该特征码字。比如击中第 3 个特征码字 leader3，则进行被击中数据沉入底部的操作，见图 45 b)所示。leader3 沉入第 16 个缓存空间，而 leader3 之后的数据均往上浮 1 个位置，leader4 上浮到了第 3 个空间，leader16 上浮到了第 15 个空间。对于未击中的情况，则见图 45 c)所示。图 45 c)表示新进来 1 个缓存数据后未被击中时缓存数据的更新过程。新数据进入第 4 个缓存空间，原来第 4 个空间里的数据上浮到第 3 个空间，第 3 个空

间里的数据上浮到第 2 个空间，第 2 个空间里的数据上浮到第 1 个空间，而原来第 1 个空间里的数据则被替换出来了，不再存放在缓存中。第 5 个空间到第 16 个空间里的数据则不作变动。

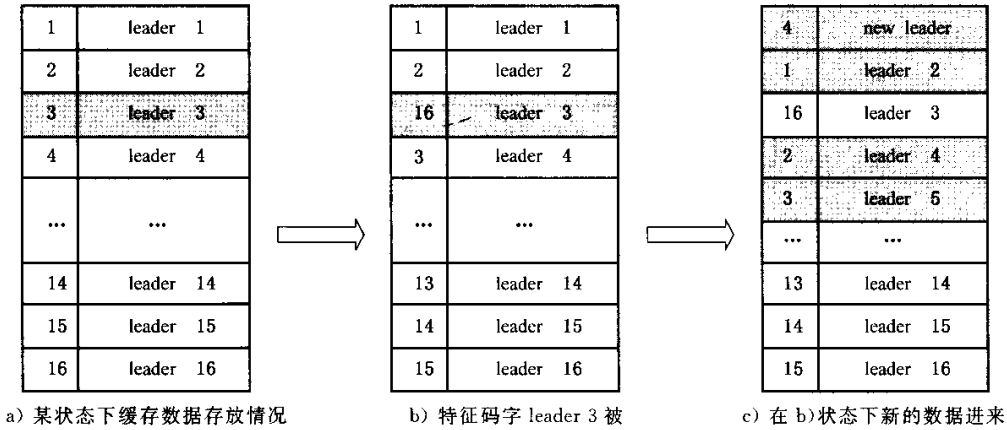
而在实际的编解码过程中，只是改变缓存空间的编号，并不变换数据在缓存中的存放位置。具体过程见图 46 所示。图 46 与图 45 表示的是同样的过程。



a) 某状态下缓存数据存放情况　　b) 特征码字 leader 3 被击中后数据更新情况　　c) 在 b)状态下新的数据进来后未被击中时缓存数据替换情况

图 45　缓存中的数据替换过程

图 46 a)为某状态下缓存里的数据存放情况，图 46 b)表示击中特征码字 leader 3 后缓存中的数据处理过程，图 46 c)表示未击中情况下新存入 1 个数据的过程。与图 45 不同的是，这里并不改变数据在缓存中的存放位置，而只改变空间的编号。具体过程为：击中编号为 3 的特征码字时，与它相应的缓存编号变为 16，而原来编号从 4～16 的特征码字的编号都相应减 1；未击中时，将新特征码字存入原来编号为 1 的特征码字的位置，而原来编号为 1 的特征码字则被替换，不再存放在缓存中，原来编号为 2、3 和 4 的特征码字编号都减 1，其余特征码字对应的编号不变。



a) 某状态下缓存数据存放情况　　b) 特征码字 leader 3 被击中后数据更新情况　　c) 在 b)状态下新的数据进来后未被击中时缓存数据替换情况

图 46　实际编码中缓存数据替换过程

第 4 步：缓存标识位的编码

为了表示特征码字是否被击中，需要在码流中增加缓存标识位，可通过修改基础码书的标识位 header 实现。修改后的 header 表示如下：

header=0　基础码书为 Q_0

header=10　基础码书为 Q_2

header=1100　基础码书为 Q_3

header=1101　数据为击中了的缓存数据

header=1110　基础码书为 Q_4

第 5 步：编码输出

对缓存类型数据的编码输出，如果击中了，则输出其在缓存中的地址，见图 46a)～图 46b)，击中了 leader 3，则输出为 leader 3 在缓存中的地址编号“3”；若未击中，则计算其码书索引作为输出。

6.2.4.8.5　使用分裂表的编码

对于待编码数据未在基础码书中（也包括奇化后仍不在基础码书中的 $2D_8$ 数据和非特殊特征码字），使用分裂表进行编码。

一个码字由于其分量的绝对值太大，不在基础码书中。对于这样的矢量编码方法是：将绝对值大的分量减去一个分裂量，得到一个绝对值足够小的差，使这个差成为基础码书中的一个码字。将这个码字在基础码书中的索引和在分裂表中分裂量的索引编码输出。

分裂表的编码见图 41 所示，在分裂表扩展处理过程中，包括两个扩展级编码方式：一级扩展编码、二级扩展编码。下面分别对两个扩展级的编码方法进行具体描述：

a)　一级扩展编码：

一级扩展编码采用等比特数来编码各维分裂量。一级扩展分裂码表见表 80 所示，分裂量取 0 或 4两种数值。表中第三列为用二进制编码的分裂量索引，即取分裂量 0 时，编码输出为索引“0”；取分裂量 4 时，编码输出为索引“1”。这样，对于一个八维矢量中的八个分量，每一个分量都需要 1 比特来表示其分裂量的索引，共 8 比特。

一级扩展编码输出格式如图 47 所示，包括码头、偶标志位(even_flag)、基础码书的索引 i 和分裂量的索引 k 组成，其中码头如表 81 所示：

表 80　一级扩展编码的分裂码表

分裂量索引	分裂量	分裂量索引编码(二进制)
0	0	0
1	4	1

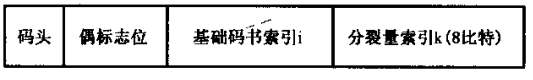


图 47　一级扩展编码的输出编码格式

表 81　一级扩展编码的码头定义表

码头编码(二进制)	码头编码(十六进制)	基础码书
11110	0x1E	Q_4
1111110	0x7E	Q_3
1111111110	0x3FE	inv_Q_4

b)　二级扩展编码：

二级扩展编码采用不等长比特数来编码各维分裂量。二级扩展编码的分裂码表如表 82 所示，列出了二级扩展使用的分裂级 k0 及 k2～k5 中的分裂量定义，k6、k7 依此类推。表中各个分裂量的大小为 4 的整数倍。

二级扩展使用的分裂表包括各个不同的级别，给各个级别的分裂表再定义一个码头信息，即

分裂量码头(split header),定义如表 83。

表 82　二级扩展编码的分裂码表

分裂量索引	k0 中分裂量	k0 中分裂量索引编码(二进制)	k2 中分裂量	k2 中分裂量索引编码(二进制)	k3 中分裂量	k3 中分裂量索引编码(二进制)	k4 中分裂量	k4 中分裂量索引编码(二进制)	k5 中分裂量	k5 中分裂量索引编码(二进制)
0	0	无	4	0	12	00	28	000	60	0000
1			8	1	16	01	32	001	64	0001
2					20	10	36	010	68	0010
3					24	11	40	011	72	0011
4							44	100	76	0100
5							48	101	80	0101
6							52	110	84	0110
7							56	111	88	0111
8									92	1000
9									96	1001
10									100	1010
11									104	1011
12									108	1100
13									112	1101
14									116	1110
15									120	1111

表 83　二级扩展编码的分裂量码头表

分裂量码头编码(二进制)	分裂量码头编码(十六进制)	分裂级
0	0x0	k0
10	0x2	k2
110	0x6	k3
1110	0xE	k4
11110	0x1E	k5
111110	0x3E	k6
1111110	0x7E	k7

若取分裂量为 0 时,即该维数据未进行分裂,则输出一个比特“0”作为分裂量码头来标识,且无需输出分裂量在 k0 中的索引;若取分裂量为 4 时,该分裂量在 k2 中,编码输出其分裂量码头索引“10”,以及其在 k2 中的索引“0”;若取分裂量为 20 时,该分裂量在 k3 中,编码输出其分裂量码头索引“110”,以及其在 k3 中的索引“10”。

二级扩展编码输出格式如图 48,包括码头、偶标志位(even_flag)、基础码书的索引、8 个分量的分裂量码头,以及对应的 8 个分裂量索引。二级扩展编码的码头定义如表 84 所示。

如果某一个分裂量码头为“0”时,其对应的索引不需要编码,占用 0 比特。分裂量索引的长度依据分裂量码头而定,分裂量在 k0,k2～k7 中时,所消耗的比特数如表 85 所示。

码头	偶标志位	基础码书索引i	第一个分量的分裂量码头	第一个分量的分裂量索引	第二个分量的分裂量码头	第二个分量的分裂量索引	……	第八个分量的分裂量码头	第八个分量的分裂量索引

图 48　二级扩展编码的输出编码格式

表 84　二级扩展编码的码头定义表

码头编码(二进制)	码头编码(十六进制)	基础码书
111110	0x3E	Q_4
11111110	0xFE	Q_3
1111111111	0x3FF	inv_Q_4

表 85　分裂量所消耗比特数表

分裂级	分裂量码头消耗比特数	分裂量索引消耗比特数	单个分裂量消耗的总比特数
k0	1	0	1
k2	2	1	3
k3	3	2	5
k4	4	3	7
k5	5	4	9
k6	6	5	11
k7	7	6	13

6.2.4.9　增益平衡

由于对不同频带样值采用了不同的缩放因子，重建信号时需要消除缩放因子的影响，即增益平衡。假设量化后输出的频谱样值为 $X_q(1,2,\cdots,N)$，则增益平衡后输出值为：

$$X_{\text{balence}}=\left[X_q(0,1,\cdots,N/2),\frac{g_2}{g_1}X_q(N/2+1,N/2+2,\cdots,N)\right] \quad \cdots\cdots(86)$$

6.2.4.10　峰值逆整形

峰值逆整形的原理见峰值预整形部分，算法流程见图 49 所示。

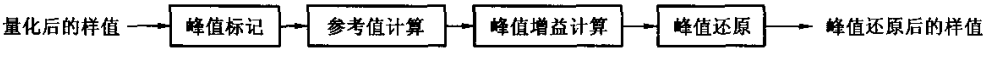


图 49　峰值逆整形示意图

处理步骤如下：

a)　标记频谱中的峰值集合$\{p_i\}$。p_i 定义为整形频谱段幅值的局部最大值；

b)　计算参考值 $ref_{\max}$。这里的参考值应该与峰值预整形中一致；

c)　计算峰值 p_i 的缩小因子 $R_i=ref_{\max}/p_i$；

d)　在峰值集合$\{p_i\}$中除去 $ref_{\max}$相关的峰值点(保证 $ref_{\max}$的不变性)。对剩余的峰值点 p_i 进行缩小 $p_i=p_i/R_i$。

6.2.4.11　逆时频变换

量化频谱 $\hat{X}(k)$进行逆变换得到时域量化信号 $\hat{x}(n)$，IDFT 变换定义如下：

$$\hat{x}(n)=\frac{1}{L_{\text{DFT}}}\sum_{k=0}^{L_{\text{DFT}}-1}\hat{X}(k)e^{\frac{j\pi kn}{L_{\text{DFT}}}} \quad \cdots\cdots(87)$$

式中：

L_{DFT}——IDFT 变换的长度，等于 288 个样本点。具体实现时可使用 IFFT。

6.2.4.12　计算和量化全局增益

最佳全局增益使得量化加权信号与原始加权信号之间的均方误差最小。假设原始加权信号为 x，

量化加权信号为 $\hat{x}$，则最佳全局增益如下：

$$g^* = \frac{\sum_{n=0}^{L_{DFT}-1} x(n)\hat{x}(n)}{\sum_{n=0}^{L_{DFT}-1} \hat{x}(n)\hat{x}(n)} \quad \cdots\cdots(88)$$

增益 g^* 使用对数量化为 7 比特索引值，具体过程如下：

a) 计算量化加权信号能量：$E=\sum_{n=0}^{L_{DFT}-1}\hat{x}^2(n)$；

b) 计算 RMS 值：$rms=4\sqrt{\frac{E}{L_{DFT}}}$；

c) 计算归一化的增益：$G=g^*\times rms$；

d) 计算索引：$index=floor(28\log_{10}G+0.5)$；

e) $if(index<0)index=0;if(index>127)index=127$。

在编码器和解码器端，量化的全局增益 $\hat{g}^*$ 计算如下：

$$\hat{g}^* = 10^{\frac{index}{28}}/rms \quad \cdots\cdots(89)$$

6.2.5 高频信号编码(BWE)

高频信号编码框图见图 50 所示，图中 I 表示每个音频超帧所包含子帧的个数，当音频超帧长度为 512 时，I=4。高频信号是指输入信号中频率大于 $Fs/4$ 的信号分量，高频信号的带宽取决于输入信号的采样频率。

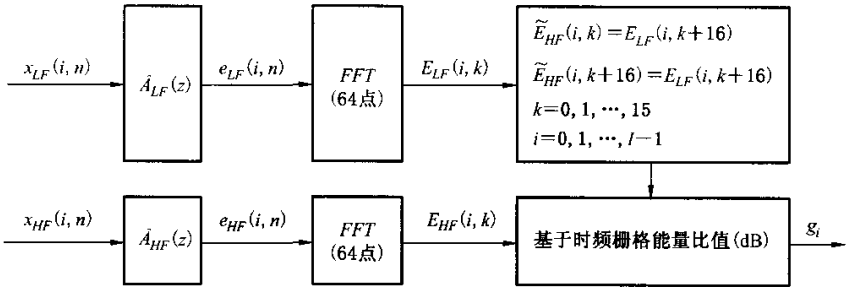


图 50 高频编码框图

编码步骤如下：

a) 低频和高频分析滤波器

低频信号的分析滤波器 $\hat{A}_{LF}(z)$直接从低频编码器获得。

高频信号的分析滤波器 $\hat{A}_{HF}(z)$：对高频信号 $x_{HF}(i,n)$的每帧(256 个样本)求取一组八阶的 LP 系数，将 LP 系数转换为 ISP 系数，ISP 系数又进一步变换成 ISF 系数并用 9 比特量化。每个子帧(64 个样本)按照 6.2.3.2.7 插值式(37)对 ISP 系数进行内插，这样就得到高频信号的分析滤波器 $\hat{A}_{HF}(z)$。

b) 残差计算

低频信号 $x_{LF}(i,n)$通过量化的分析滤波器 $\hat{A}_{LF}(z)$得到低频残差 $e_{LF}(i,n)$；

高频信号 $x_{HF}(i,n)$通过量化的分析滤波器 $\hat{A}_{HF}(z)$得到高频残差 $e_{HF}(i,n)$；

c) FFT 变换

低频残差 $e_{LF}(i,n)$经过 FFT 变换得到低频残差谱 $E_{LF}(i,k)$，FFT 变换长度采用 64 点。高频残差 $e_{HF}(i,n)$也经过 FFT 变换得到高频残差谱 $E_{HF}(i,k)$，FFT 变换长度也采用 64 点。

d) 频谱拷贝

低频信号在时间上以 64 点为一个子帧，在频率上将低频残差谱 $E_{LF}(i,k)$ 等分为低低频残差谱 $E_{LFL}(i,k)$ 和低高频残差谱 $E_{LFH}(i,k)$。因为低低频残差谱 $E_{LFL}(i,k)$ 存在很强的弦性，所以在高频拷贝中不用。而是将低高频残差谱 $E_{LFH}(i,k)$ 复制两次来得到估计的高频残差谱 $\widetilde{E}_{HF}(i,k)$。

e) 高频信号的时频栅格划分

当低频编码器使用 ACELP 时，高频信号的时频划分总是采用最高时间分辨率，即 64 点；相应的频率分辨率最小，其时频划分见图 51 所示。

当低频编码器使用 TAC 时，高频的时频栅格划分依赖于超帧长度。当超帧长度为 512 时，只能使用图 51 的时频栅格划分结构。

f) 增益计算

由低频残差信号代替高频残差信号，然后计算重建高频残差信号和原始高频残差信号之间增益因子：先分别计算两个信号所对应栅格覆盖的时频数据的能量，最后计算能量比值，并转换以 dB 为单位。这样每个超帧就得到 I 个增益（$g_0, g_1, g_2, \cdots, g_{I-1}$）。

g) 增益调整

对每个时频划分块，判断原始信号和拷贝信号的弦性，如果原始信号有弦而拷贝信号无弦，则对相应块的增益进行适当的减小，以防止噪声水平过分抬高。

h) 增益矢量量化编码

最后每个超帧的连续 4 个增益系数组成一个增益矢量，并用 7 比特进行量化。

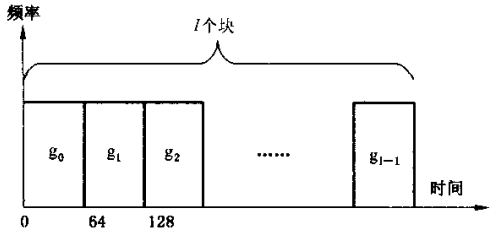


图 51 时频栅格划分图

每个超帧 BWE 参数的总比特消耗为：16×(I/4)，其中包括 I/4 组高频 LP 系数（每组 9 比特），以及 I/4 个增益矢量（每个矢量 7 比特）。

6.2.6 识别特征参数编码

6.2.6.1 识别特征参数提取

6.2.6.1.1 去直流偏置

信号经过高通滤波器，目的是为了滤掉信号中的直流分量。高通滤波器的传递函数见式(90)：

$$H_{h1}(z)=\frac{b_0-b_1z^{-1}+b_2z^{-2}}{1-a_1z^{-1}+a_2z^{-2}} \qquad \cdots\cdots(90)$$

式中：

$a_1=-1.988\ 89$；

$a_2=0.988\ 95$；

$b_0=0.994\ 46$；

$b_1=-1.988\ 92$；

$b_2=0.994\ 46$。

6.2.6.1.2 噪声消除

噪声消除模块主要作用是降低背景噪声，提高信号的信噪比。无论语音识别还是声纹识别算法，噪声对识别结果影响很大。因此在识别特征参数提取之前，应先对信号进行降噪处理。噪声消除算法描述参见附录 J。

6.2.6.1.3 波形预处理

6.2.6.1.3.1 基音周期搜索

波形预处理应用于降噪后的信号。降噪模块的输出(每帧160个样本)存储在包含480个样本的缓存中,波形预处理模块处理的窗大小 N_m 为缓存中前400个样本。图52描述了波形预处理模块的原理。

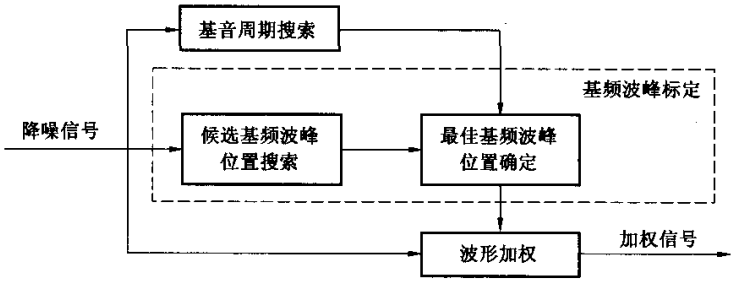


图52 波形预处理原理框图

利用信号的归一化自相关对降噪后的信号估计基音周期,归一化自相关计算见式(91):

$$corr=\frac{\sum_{n=0}^{239}s_{nr}(n)s_{nr}(n-delay)}{\sqrt{\sum_{n=0}^{239}s_{nr}^2(n)\sum_{n=0}^{239}s_{nr}^2(n-delay)}},40\leqslant delay\leqslant 160 \quad \cdots\cdots(91)$$

详细基音周期搜索算法描述见6.2.3.4.1。

6.2.6.1.3.2 候选基频波峰位置搜索

在波形预处理窗内搜索绝对值最大的M个样本点所对应的位置作为候选基频波峰位置,记为:Pc[i],i=1,2,…,M;按该位置所对应的信号样本绝对值从大到小顺序排列[见式(92)]。

$$Pc[1],\cdots,Pc[M]=\underset{n=0,\cdots,L-1}{\overset{\text{最大的M个值}}{\operatorname{argmax}}}(|s_{nr}(n)|) \quad \cdots\cdots(92)$$

式中:

$M=3$。

6.2.6.1.3.3 最佳基频波峰位置确定

该模块用来通过基音周期和候选基频波峰位置来确定最佳基频波峰位置,根据基音周期 Tp 在本帧范围内延拓出的所有对应位置上样本的平均幅度值,计算每个候选基频波峰位置[见式(93)]:

$$Mp\_avg[i]=\frac{\left|\sum_{j}s(P_c[i]+j\times T_p)\right|}{D},i=1,\cdots,M \quad \cdots\cdots(93)$$

式中:

D ——满足 $0\leqslant P_c[i]+j\times T_p\leqslant N_m-1$ 的 j 的个数;

$Mp\_avg[i]$——根据第 i 个候选基频波峰位置所计算的平均幅度值,M 为候选基频波峰位置个数。

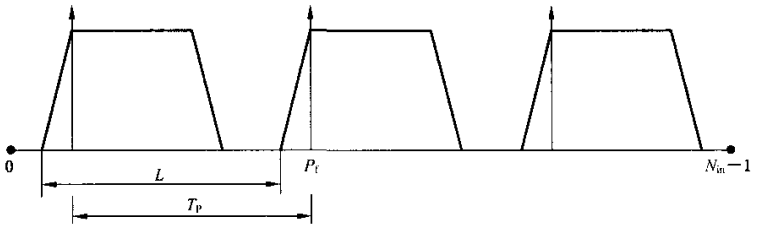
假设最佳的基频波峰位置为 Pf,则见式(94):

$$Pf=\underset{i=1,\cdots,M}{\operatorname{argmax}}(Mp\_avg[i]) \quad \cdots\cdots(94)$$

6.2.6.1.3.4 波形加权

波形加权就是将基频波峰位置附近的信号提升,其他位置的信号减小,从而起到提高信噪比同时增强基频的作用。方法是在该帧信号内的每个基音周期内用窗函数加权。波峰附近的权值大,其他位置

的权值小，同时窗函数要尽量保证处理后的信号连续。见图 53 所示。



L——窗函数的长度；

T_p——基音周期；

P_f——最佳的基频波峰位置。

图 53　波形加权处理框图

窗函数的定义为：

$$w_{swp}(n)=\begin{cases}0.8, & n=0\\ 1.0, & n=1\\ 1.2, & 2\leqslant n\leqslant 0.6T_p\\ 1.0, & n=0.6T_p+1\\ 0.8, & 0.6T_p+2\leqslant n\leqslant T_p-1\end{cases} \quad \cdots\cdots(95)$$

按照每个基音周期进行加权：

$$s_{swp}(n)=w_{swp}(i)\times s_{ur}(n),\quad n=P_f\_n+i-2,\quad 0\leqslant i<T_p$$

$$P_f\_n+=T_p \quad \cdots\cdots(96)$$

第一个基音周期和最后一个基音周期边界处理，保证 $0\leqslant n<N_{in}$。P_f_n 表示按基音周期扩展的基频波峰位置，其初值通过下面式(97)获得：

$$P_f\_n=P_f$$

$$While(P_f\_n>0)\quad P_f\_n-=T_p \quad \cdots\cdots(97)$$

6.2.6.1.4　倒谱计算

6.2.6.1.4.1　对数能量计算

对波形加权后的信号 $s_{swp}(n)$的每帧能量参数取对数：

$$\ln E=\begin{cases}\ln(E_{swp}), & E_{swp}\geqslant E_{THRESH}\\ \ln(E_{THRESH}), & E_{swp}<E_{THRESH}\end{cases} \quad \cdots\cdots(98)$$

$$E_{THRESH}=\exp(-1),E_{swp}=\sum_{n=0}^{N_{in}-1}S_{swp}(n)\times S_{swp}(n) \quad \cdots\cdots(99)$$

6.2.6.1.4.2　预加重

预加重滤波器处理后的信号 $s_{swp\_pe}(n)$：

$$s_{swp\_pe}(n)=s_{swp}(n)-0.9\times s_{swp}(n-1) \quad \cdots\cdots(100)$$

式中：

$s_{swp\_pe}(-1)$——上一帧的最后一个样本，如果是第一帧，则其值为 0。

6.2.6.1.4.3　加窗

对预加重处理模块输出信号进行加窗处理，窗类型为长度 $N_{in}=400$ 的海明窗：

$$s_{swp\_w}(n)=\left[0.54-0.46\times\cos\left(\frac{2\pi\times(n+0.5)}{N_{in}}\right)\right]\times s_{swp\_pe}(n),0\leqslant n\leqslant N_{in}-1 \quad \cdots(101)$$

6.2.6.1.4.4　FFT 变换和功率谱估计

通过后面补零将 N_{in} 个样本扩展为 512 个样本。用长度 $N_{FFT}=512$ 的 FFT 计算出信号频谱 $X_{swp}(bin)$

$$X_{swp}(bin)=FFT\{s_{swp\_w}(n)\},0\leqslant bin\leqslant N_{FFT}/2 \quad \cdots\cdots(102)$$

相应的功率谱 $P_{swp}(bin)$ 为：

$$P_{swp}(bin)=|X_{swp}(bin)|^2,0\leqslant bin\leqslant N_{FFT}/2 \quad \cdots\cdots(103)$$

6.2.6.1.4.5 Mel 滤波

Mel 滤波模块将线性频率频谱表示为 Mel 刻度频谱。信号有效频带位于 f_{start} 与 $f_{samp}/2$ 之间，在 Mel 域分为 K_{FB} 个子带，每个子带对应一个三角形频率窗，相邻子带有 50%重叠。

定义 FFT 索引 $bin=N_{FFT}$ 对应的频率是 f_{samp}，则线性频率转换为 FFT 索引公式为：

$$index\{f\}=round\left\{\frac{f}{f_{samp}}\times N_{FFT}\right\} \quad \cdots\cdots(104)$$

Mel 函数计算公式如下：

$$Mel\{x\}=\Lambda\times\log_{10}\left(1+\frac{x}{\mu}\right)=\lambda\times\ln\left(1+\frac{x}{\mu}\right),\ \lambda=\frac{\Lambda}{\ln(10)} \quad \cdots\cdots(105)$$

Mel 反函数计算公式如下：

$$Mel^{-1}\{y\}=\mu\times\left(\exp\left(\frac{y}{\lambda}\right)-1\right) \quad \cdots\cdots(106)$$

为了将 Mel 域的频带等间隔划分，通过计算 Mel 函数得到滤波器的中心频率。图 54 给出了线性频率和 Mel 频率之间的映射关系。

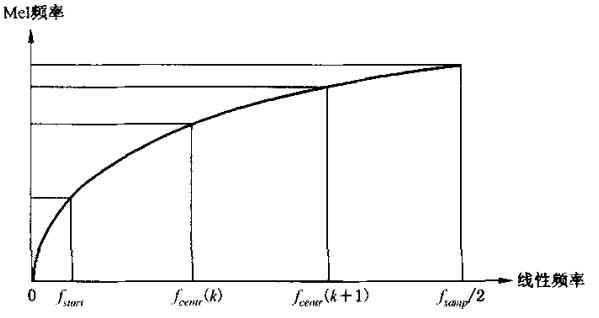


图 54 线性频率和 Mel 频率映射

$$f_{centr}(k)=Mel^{-1}\left\{Mel\{f_{start}\}+k\times\frac{Mel\{f_{samp}/2\}-Mel\{f_{start}\}}{K_{FB}+1}\right\},1\leqslant k\leqslant K_{FB} \quad \cdots\cdots(107)$$

式中：

$f_{start}=64$ Hz；

$f_{samp}=16\ 000$ Hz；

$\mu=700$；

$\Lambda=2595$；

$\lambda=1127$；

$K_{FB}=32$。

将 Mel 滤波器的中心频率表示为 FFT 索引：

$$bin_{centr}(k)=index\{f_{centr}(k)\}=round\left\{\frac{f_{centr}(k)}{f_{samp}}\times N_{FFT}\right\},\ 1\leqslant k\leqslant K_{FB} \quad \cdots\cdots(108)$$

对第 k 个 Mel 子带，频率窗可分两个部分。前一部分(频率为 $f_{centr}(k-1)<f<f_{centr}(k)$)宜增加权重，后一部分(频率为 $f_{centr}(k)<f<f_{centr}(k+1)$)宜降低权重，对功率谱 $P_{swp}(bin)$ 加上这些频率窗。根据每个子带中频谱线相对于中心频率的位置，计算每个子带的频率窗权重。

$$W_{left}(i,k)=\frac{i-bin_{centr}(k-1)+1}{bin_{centr}(k)-bin_{centr}(k-1)+1},$$

$$1 \leqslant k \leqslant K_{FB}, bin_{centr}(k-1) \leqslant i \leqslant bin_{centr}(k) \quad \cdots\cdots(109)$$

$$W_{right}(i,k) = 1 - \frac{i - bin_{centr}(k)}{bin_{centr}(k+1) - bin_{centr}(k) + 1},$$

$$1 \leqslant k \leqslant K_{FB}, bin_{centr}(k) < i \leqslant bin_{centr}(k+1)$$

其他情况下，权重值为 0。

Mel 滤波器的输出为每个子带的功率谱值 $P_{swp}(bin)$ 的加权和，公式如下：

$$E_{FB}(k) = \sum_{i=bin_{centr}(k-1)}^{bin_{centr}(k)} W_{left}(i,k) \times P_{swp}(i) + \sum_{i=bin_{centr}(k)+1}^{bin_{centr}(k+1)} W_{right}(i,k) \times P_{swp}(i),\ 1 \leqslant k \leqslant K_{FB} \quad \cdots\cdots(110)$$

6.2.6.1.4.6 高频聚合

6.2.6.1.4.5 求出了 32 个子带的 Mel 滤波后的输出，由于高频信号容易受到噪声的干扰，较多的子带划分影响了参数的鲁棒性，因此将高频的 9 个子带聚合成 3 个子带，聚合方法采用加权平均的方法：

$$E_{FB}(24) = \frac{\alpha E_{FB}(24) + \beta E_{FB}(25) + \gamma E_{FB}(26)}{\alpha + \beta + \gamma} \quad \cdots\cdots(111)$$

$$E_{FB}(25) = \frac{\alpha E_{FB}(27) + \beta E_{FB}(28) + \gamma E_{FB}(29)}{\alpha + \beta + \gamma} \quad \cdots\cdots(112)$$

$$E_{FB}(26) = \frac{\alpha E_{FB}(30) + \beta E_{FB}(31) + \gamma E_{FB}(32)}{\alpha + \beta + \gamma} \quad \cdots\cdots(113)$$

式中：

$\alpha = \beta = \gamma = 1$。

6.2.6.1.4.7 非线性变换(计算对数)

对 Mel 滤波器的输出取对数：

$$S_{FB}(k) = \ln(E_{FB}(k)), 1 \leqslant k \leqslant K_{FB} \quad \cdots\cdots(114)$$

式中：

$K_{FB} = 26$。

限制对数滤波器组的输出不能小于 -1。

6.2.6.1.4.8 DCT 变换

对非线性变换模块的输出作 DCT 得到 13 个倒谱系数。

$$c(i) = \sum_{k=1}^{K_{FB}} S_{FB}(k) \times \cos\left(\frac{i \times \pi}{K_{FB}} \times (k - 0.5)\right),\ 0 \leqslant i \leqslant 12 \quad \cdots\cdots(115)$$

式中：

$K_{FB} = 26$。

6.2.6.1.4.9 倒谱计算输出

倒谱计算得到的特征矢量由 14 个系数组成：1 个对数能量系数 $\ln E$ 和 13 个倒谱系数 $c(0)$ 到 $c(12)$。对数能量系数和倒谱系数 $c(0)$ 都表示信号短时能量。

6.2.6.1.4.10 去信道干扰

对于卷积特性的信道干扰，在倒谱域为相加。因此将信号倒谱系数减去信道倒谱系数，即可得到去除信道干扰后均衡的信号倒谱系数。

$$c_{eq}(i) = c(i) - TranCep(i),\ i = 1, \cdots, 12 \quad \cdots\cdots(116)$$

式中：

$c_{eq}(i)$——均衡的信号倒谱系数；

$TranCep(i)$——信道倒谱系数，初值为 0。

采用低通滤波的方法来估计信道倒谱分量 $TranCep(i)$，可采用一阶 IIR 滤波。

$$TranCep(i)=TranCep(i)(1-\alpha)+(c(i)-RefCep(i))\beta_1+c(i)\beta_2 \quad \cdots\cdots(117)$$

式中：

$RefCep(i)$——均衡信道下的语音信号倒谱特征矢量的统计均值。如下：

$RefCep(1)=-6.618\ 909$，$RefCep(2)=0.198\ 269$，$RefCep(3)=-0.740\ 308$，

$RefCep(4)=0.055\ 132$，$RefCep(5)=-0.227\ 086$，$RefCep(6)=0.144\ 280$，

$RefCep(7)=-0.112\ 451$，$RefCep(8)=-0.146\ 940$，$RefCep(9)=-0.327\ 466$，

$RefCep(10)=0.134\ 571$，$RefCep(11)=0.027\ 884$，$RefCep(12)=-0.114\ 905$，

α——IIR 低通滤波器的参数，且满足当信噪比 $SNR\geqslant 0$ dB 时，$\alpha=0.01$；否则，$\alpha=0.005$。

β_1 和 β_2——$\beta_1+\beta_2=\alpha$，且满足，当 SNR 高时，$\beta_1\gg\beta_2$；当 SNR 低时，$\beta_1\ll\beta_2$，见表 86 设置。

表 86　低通滤波器系数定义表

SNR(dB)	SNR>20	15≤SNR<20	10≤SNR<15	5≤SNR<10	0≤SNR<5	−10≤SNR<0	SNR<−10
β_1	100%α	90%α	80%α	70%α	50%α	20%α	0
β_2	0	10%α	20%α	30%α	50%α	80%α	100%α

SNR 由噪声消除模块提供，如果得不到 SNR，则将参数设置为：

$\alpha=0.01$；$\beta_1=\alpha$；$\beta_2=0$。

6.2.6.2　识别特征参数压缩

6.2.6.2.1　特征矢量

特征矢量就是 6.2.6.1 中从每个短时分析帧中提取的参数，包括 12 个 MFCC 系数：

$$C_{eq}(t)=[c_{eq}(1,t),c_{eq}(2,t),\cdots,c_{eq}(12,t)]^T \quad \cdots\cdots(118)$$

式中：

t——帧索引；

特征失量还包括 MFCC 系数 $c(0,t)$，对数能量系数 $\ln E(t)$ 和 VAD 标志位。

最终编码的特征矢量表示如下：

$$y(t)=\begin{bmatrix} C_{eq}(t) \\ VAD(t) \\ c(0,t) \\ \ln E(t) \end{bmatrix} \quad \cdots\cdots(119)$$

6.2.6.2.2　矢量量化

特征矢量 $y(t)$使用分裂的矢量量化。14 个系数[$c(1)\sim c(12)$，$c(0)$和 lnE]两个 1 组，被分成 7 组，每组都用独立的 VQ 码书进行量化，VAD 标志位作为 1 个独立比特进行传输。矢量量化所使用的量化失真度量是加权欧氏距离：

$$d_j^{i,i+1}=\begin{bmatrix} y_i(t) \\ y_{i+1}(t) \end{bmatrix}-q_j^{i,i+1} \quad \cdots\cdots(120)$$

$$idx^{i,i+1}(t)=\underset{0\leqslant j\leqslant(N^{i,i+1}-1)}{\operatorname{argmin}}\{(d_j^{i,i+1})^T W^{i,i+1}(d_j^{i,i+1})\},\quad i=\{0,2,4,\cdots,12\} \quad \cdots\cdots(121)$$

式中：

$q_j^{i,i+1}$——码书 $Q^{i,i+1}$的第 j 个码字；

$(N^{i,i+1}-1)$——码书大小；

$W^{i,i+1}$——码书 $Q^{i,i+1}$的加权矩阵；

$idx^{i,i+1}(t)$——量化$[y_i(t), y_{i+1}(t)]^T$所得到的码书索引。

加权矩阵$W^{12,13}$如下：

$$W^{12,13} = \begin{bmatrix} 0.000717185 & 0 \\ 0 & 1 \end{bmatrix} \quad \cdots\cdots(122)$$

其他加权矩阵为单位阵(即对角线元素为1,其他为0)。

识别特征编码提供两种编码模式,因此矢量量化的码表也需要两组:一组是直接量化特征矢量(见表87),另一组是量化残差矢量(见表88)。

表87 直接编码模式下量化码表

码书	码书大小	量化子矢量
$Q^{0,1}$	64	[c(1), c(2)]
$Q^{2,3}$	64	[c(3), c(4)]
$Q^{4,5}$	64	[c(5), c(6)]
$Q^{6,7}$	64	[c(7), c(8)]
$Q^{8,9}$	64	[c(9), c(10)]
$Q^{10,11}$	32	[c(11), c(12)]
$Q^{12,13}$	256	[c(0), lnE]

表88 预测编码模式下量化码表

码书	码书大小	量化子矢量
$Q^{0,1}$	16	[c(1), c(2)]
$Q^{2,3}$	16	[c(3), c(4)]
$Q^{4,5}$	16	[c(5), c(6)]
$Q^{6,7}$	16	[c(7), c(8)]
$Q^{8,9}$	16	[c(9), c(10)]
$Q^{10,11}$	8	[c(11), c(12)]
$Q^{12,13}$	64	[c(0), lnE]

6.2.7 打包格式

6.2.7.1 音频编码参数打包

512个样本点音频帧的编码参数被写入了1个二进制包,包的具体格式同编码模式相关,见图55所示。打包的编码参数包括:4比特的模式信息,5比特填充位(填充"11111"),1比特的感知加权标志,ACELP参数或TVC参数,以及高频编码参数。对于ACELP编码模式,模式位为0;对于TVC编码模式,模式位为1;其他模式位保留,以备将来扩展。

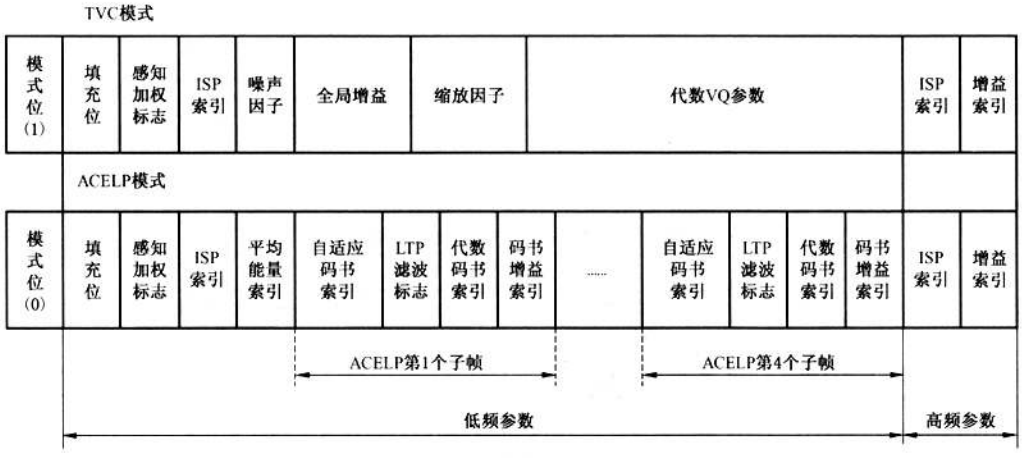


图 55　音频帧编码参数的数据格式

6.2.7.2　识别特征参数打包

在 6.2.6.2.2 对每帧得到的特征矢量进行矢量量化，只需传输码书索引，1 比特的 VAD 标志和 1 个CRC 校验和。对于直接编码模式，使用 4 比特的 CRC 校验和(CRC 生成多项式为 $g(X)=1+X+X^4$)；对于预测编码模式，使用 2 比特的 CRC 校验和(CRC 生成多项式为 $g(X)=1+X+X^2$)。每帧识别特征参数压缩后打包格式见图 56 和图 57 所示。

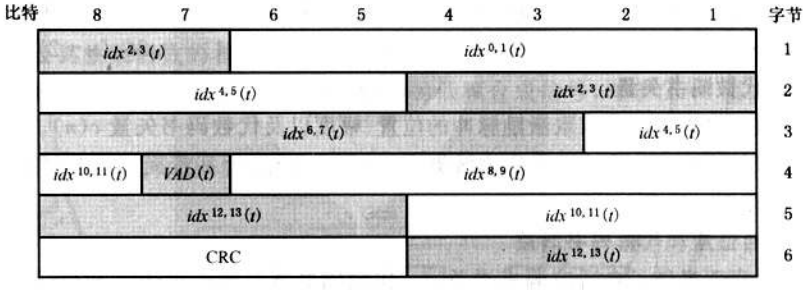


图 56　直接编码模式帧的数据格式(48 比特)

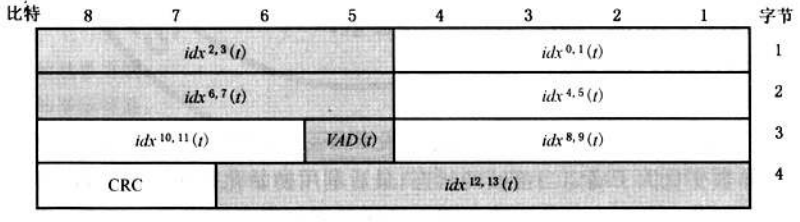


图 57　预测编码模式帧的数据格式(32 比特)

6.3　解码器功能描述

6.3.1　低频信号解码

6.3.1.1　ACELP 解码

6.3.1.1.1　ACELP 解码过程

解码器解码接收到的参数并合成得到重建信号，见图 58 所示。

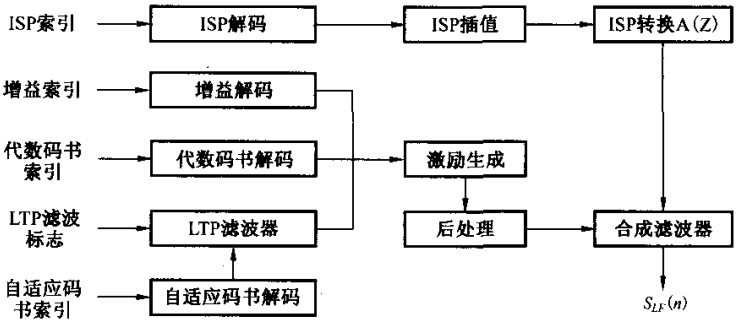


图 58 ACELP 解码合成框图

6.3.1.1.2 解码 LP 滤波器参数

接收到的 ISP 索引用来重构已量化的 ISP 系数，然后按照 6.2.3.2.7 中 ISP 系数的插值方法计算每个子帧的 ISP 系数。对于每个子帧，插值后 ISP 系数被转换为 LP 系数，用于 LP 合成滤波器来重建信号。

6.3.1.1.3 解码自适应码书矢量

接收到的自适应码书索引(基音延迟索引)用来搜索基音延迟的整数部分和小数部分。自适应码书矢量 $v(n)$使用过去的激励 $u(n)$通过 FIR 滤波器来生成。接收到的 LTP 滤波标志被用来确定解码的自适应码书矢量是 $v_1(n)=v(n)$还是 $v_2(n)=0.26v(n)+0.48v(n-1)+0.26v(n-2)$。

6.3.1.1.4 解码代数码书矢量

接收到的代数码书索引用来求取激励脉冲的位置、幅度以及代数码书矢量 $c(n)$。如果基音延迟的整数部分小于子帧长度 64，则通过自适应滤波器 $F(z)$对 $c(n)$进行基音周期锐化滤波。$F(z)$定义见 6.2.3.4.6中预滤波器。

6.3.1.1.5 解码自适应和代数码书增益

接收到的增益索引提供了自适应码书增益 $\hat{g}_p$，代数码书增益校正因子 $\hat{\gamma}$ 和代数码书的平均能量 $\overline{E}_s$。代数码书增益的重建过程为：

a) 按 6.2.3.4.9 中式(72)求得代数码书激励矢量的能量 E_i；

b) 按 6.2.3.4.9 中式(73)代数码书的预测增益 g_c'；

c) 得到量化的代数码书增益 $\hat{g}_c=\hat{\gamma}g_c'$。

6.3.1.1.6 代数码书增益平滑

代数码书增益平滑先计算当前帧的 ISF 参数变化因子，并对当前帧的代数码书增益进行初始化修正；然后按照 ISF 参数变化因子确定当前帧的状态；最后利用初始化修正后的代数码书增益以及当前帧的平滑因子，对当前帧的代数码书增益进行平滑。具体步骤如下：

a) 计算 ISF 变化因子 isf_diff，计算公式为：

$$isf\_diff=\frac{\sum_{i=0}^{14}(isf\_new_i-isf\_old_i)^2}{400\ 000} \quad \cdots\cdots(123)$$

式中：

isf_new——当前帧的 ISF；

isf_old——上一帧的 ISF。

b) 对代数码书增益 $\hat{g}_c$ 进行初始化修正

$$g_0=\begin{cases}\max(g_{-1},\hat{g}_c/1.06), & \hat{g}_c>g_{-1}\\ \min(g_{-1},\hat{g}_c\times 1.06), & \hat{g}_c\leqslant g_{-1}\end{cases} \quad \cdots\cdots(124)$$

式中：

g_0——当前帧初始化修正后的代数码书增益；

g_{-1}——前一帧初始化修正后的代数码书增益。

c) 根据 isf_diff 将当前帧分为两类状态，根据不同状态类型，用不同平滑因子进行的增益平滑。

$$\hat{g}_c=\begin{cases}0.17g_0+0.83\hat{g}_c, & isf\_diff>0.58\\ 0.83g_0+0.17\hat{g}_c, & isf\_diff\leqslant 0.58\end{cases} \quad \cdots\cdots(125)$$

6.3.1.1.7 激励信号生成

每个子帧的激励信号 $\hat{u}(n)$ 由式(126)得到：

$$\hat{u}(n)=\hat{g}_p v(n)+\hat{g}_c c(n),\ n=0,\cdots,63 \quad \cdots\cdots(126)$$

6.3.1.1.8 信号合成

每个子帧的重建信号通过式(127)计算：

$$\hat{s}(n)=\hat{u}(n)-\sum_{i=1}^{16}\hat{\alpha}_i\hat{s}(n-i),\quad n=0,\cdots,63 \quad \cdots\cdots(127)$$

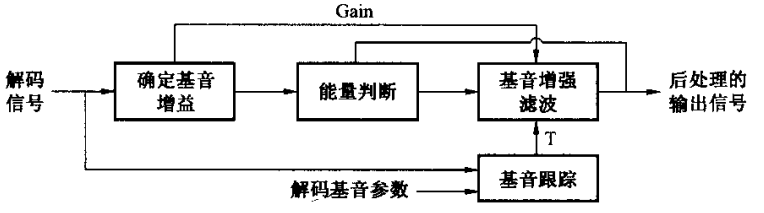
式中：

$\hat{\alpha}_i$—— 子帧的 LP 系数。

6.3.1.1.9 合成信号基音增强

在低码率下，解码音频信号仍会有一定的噪声，因此采取后处理滤波以抑制噪声。基音增强技术通过增强基频及其整数倍数的各谐波分量，抑制位于非谐波分量处的噪声，以提高解码信号的感知质量。基音增强后处理只针对 ACELP 模式解码的低频合成信号，对于 TAC 模式解码信号不进行此后处理。

该基音增强后处理算法见图 59 所示。



T——解码信号的基音周期；

Gain——解码信号的基音增益。

图 59 基音增强后处理算法框图

T 由该子帧解码的闭环基音周期给出。基音跟踪模块为避免出现跟踪倍数基音周期问题，需要计算延迟为 T/2 处信号的归一化自相关值。如果该归一化自相关值大于 0.95，则将 T/2 作为后处理中的基音周期值。

该后处理算法按每 64 个样点子帧对解码信号 $\hat{s}(n)$ 进行处理，具体过程如下：

a) 计算解码信号的基音增益 Gain：

确定相邻基音周期的信号能量比值，该比值的计算公式见式(128)：

$$E_c=\sqrt{\frac{\sum_{n=0}^{63}\hat{s}^2(n)}{\sum_{n=0}^{63}\hat{s}^2(n-T)}} \quad \cdots\cdots(128)$$

将该比值 E_c 与从码流中解码获得的基音增益进行比较，取其中较小的一个作为最终解码信号的基音增益 Gain。

b) 判断 a)中计算的能量比值 E_c 是否超过预定阈值 E_t。若超过，则执行 c)，进行基音增强滤波处理；否则不进行基音增强滤波处理，直接将解码信号 $\hat{s}(n)$作为最终的信号 $s_E(n)$输出。

$$s_E(n)=\begin{cases}\hat{s}(n), & E_c<E_t\\ \hat{s}(n)\otimes h(n), & E_c\geqslant E_t\end{cases} \qquad \cdots\cdots(129)$$

式中：

$h(n)$——基音增强滤波器的脉冲响应函数；

E_t——预定阈值，其值固定为 0.6。

c) 基音增强滤波

基音增强滤波器的传输函数如下：

$$H(z)=G_1(1+\lambda z^{-T}) \qquad \cdots\cdots(130)$$

式中：

G_1——滤波器的全局增益；

λ——局部调整因子，其值固定为 0.1。

全局增益 G_1 的计算公式为：

$$G_1=\frac{1}{1+\lambda\times Gain} \qquad \cdots\cdots(131)$$

式中：

$Gain$——a)中计算的基音增益。

基音增强滤波的输出信号：

$$s_E(n)=G_1(\hat{s}(n)+\lambda\times\hat{s}(n-T)) \qquad \cdots\cdots(132)$$

6.3.1.2 TVC 解码

6.3.1.2.1 TVC 解码过程

TVC 解码过程见图 60 所示。

解码器获得的码流信息：缩放因子，量化频谱值，全局增益和噪声因子。

TVC 的解码步骤简述如下：

a) 解包，获取 TVC 编码参数；

b) 量化频谱值进行基于变长分裂表的反矢量量化；

c) 增益平衡，消除缩放因子的影响；

d) 峰值逆整形；

e) 逆时频变换，信号由频域变换到时域，得到的时域信号与全局增益相乘；

f) 加自适应窗和重叠加，为下一帧保存重叠信号，自适应窗的窗长和窗型定义见 6.2.4.3；

g) 通过逆感知加权滤波器得到合成音频信号；

h) 如果上一帧使用 ACELP 模式编码，那么需要进行帧间平滑处理。

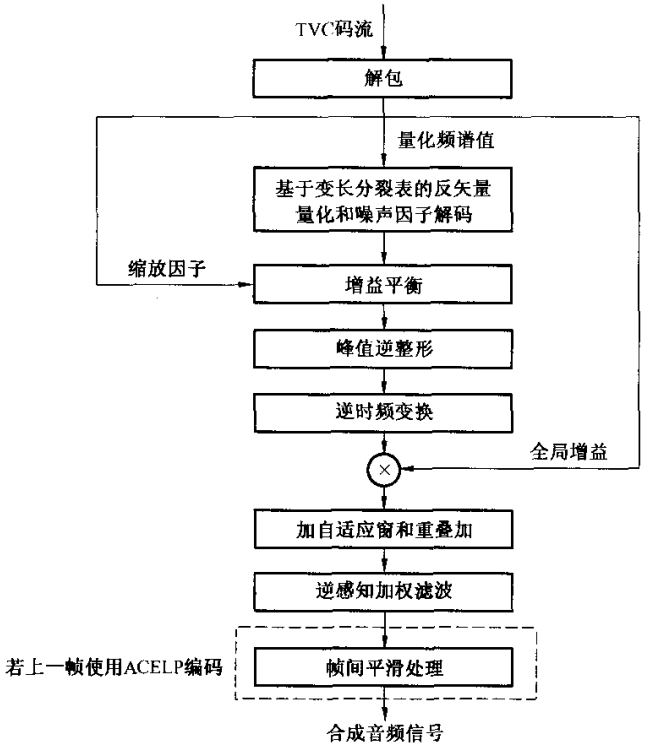


图 60　TVC 解码过程图

6.3.1.2.2　量化频谱解码

频谱的矢量量化，TVC 使用了基础码书编码、一级分裂扩展编码和二级分裂扩展编码三种方法，详见 6.2.4.8。因此在解码中，需要根据不同编码方法进行解码操作。首先解包码流中的编码参数，通过每组数据中参数 header 值来判断这组数据使用的是哪种编码方法。如果为基础码书编码，则根据 header 和索引 i 的值直接计算码字 y，然后由偶标志位来判断是否对码字进行加 1 的处理，并将其输出；如果为分裂表编码，则先计算相对应的基础码书中的码字 c，再加上分裂量还原为 y，然后由偶标志位判断是否进行加 1 的处理，最后输出 y。图 61 给出了量化频谱解码流程。

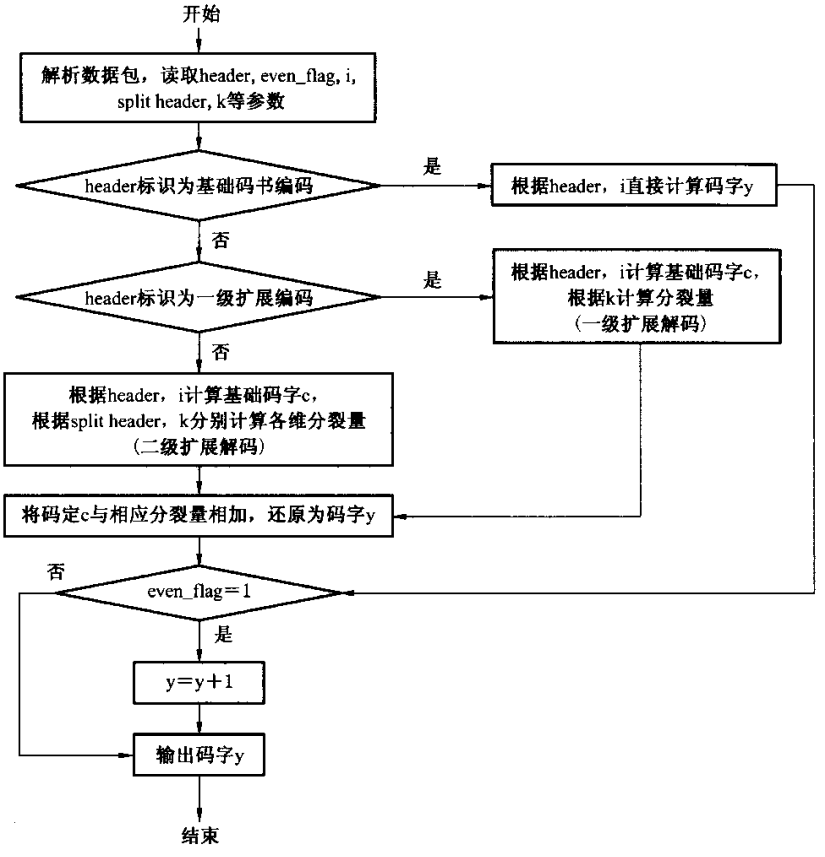


图 61　量化频谱解码过程图

6.3.1.2.3　解码噪声因子

噪声因子采用 3 比特量化，见 6.2.4.6。对接收到的 3 比特编码索引($0 \leqslant idx \leqslant 7$)，解码的噪声因子为 $\delta_{noise}=0.1\times(8-idx)$。对于 K/6≤k≤K 范围内的频谱矢量，如果解码的频谱矢量为零矢量，则使用下面的 8 维随机矢量代替：

$$\delta_{noise}\times[\cos\theta_1 \quad \sin\theta_1 \quad \cos\theta_2 \quad \sin\theta_2 \quad \cos\theta_3 \quad \sin\theta_3 \quad \cos\theta_4 \quad \sin\theta_4] \quad \cdots\cdots\cdots\cdots(133)$$

式中相位 θ_1，θ_2，θ_3 和 θ_4 是随机选取。

6.3.1.2.4　增益平衡及峰值逆整形

反量化频谱值通过增益平衡，即通过解码得到的两个缩放因子信息，消除缩放因子的影响，经峰值逆整形和逆时频变换后得到时域信号，最后乘以全局增益即可得到重建的时域信号，其对应解码框图见图 62 所示。增益平衡见 6.2.4.9，峰值逆整形见 6.2.4.10，逆时频变换见 6.2.4.11，全局增益因子的反量化见 6.2.4.12。

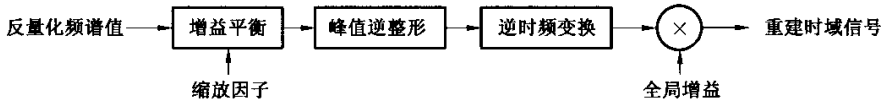


图 62　增益平衡和频谱逆整形的解码框图

6.3.1.2.5 帧间平滑处理

如果上一帧采用 ACELP 编码，为了消除编码模式切换的影响，则需要对上一帧 ACELP 解码得到的音频信号和当前帧 TVC 解码得到的音频信号的重叠部分进行加窗和重叠相加操作。重叠部分采用的三角窗如图 63，对 TVC 帧重叠的音频信号用 $w_1(n)$ 加权，对 ACELP 帧重叠的音频信号用 $w_2(n)$ 加权，其定义如下所示：

$$\begin{aligned} w_1(n) &= n/L_1, \quad n=0,\cdots,L_1-1 \\ w_2(n) &= (L_1-n)/L_1, \quad n=0,\cdots,L_1-1 \end{aligned} \qquad \cdots\cdots(134)$$

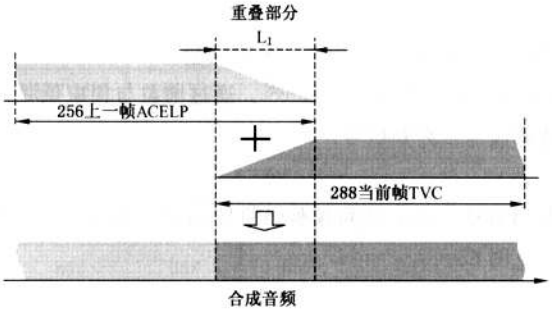


图 63 帧间平滑处理示意图

6.3.2 高频信号解码

6.3.2.1 高频信号解码过程

高频信号解码使用一种带宽扩展机制，并且需要用到低频信号解码中的一些数据，高频信号解码过程见图 64 所示。高频信号解码分为两步：计算高频激励信号和合成滤波。解码过程为：对低频激励信号进行增益调整得到高频的激励信号，然后通过高频 LP 合成滤波器得到合成信号。

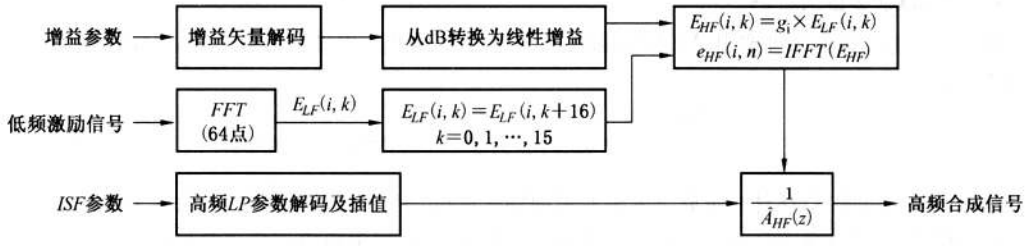


图 64 高频解码过程

6.3.2.2 高频参数解码

高频解码的参数包括：ISF 参数和增益参数。ISF 参数用来生成合成滤波器 $1/\hat{A}_{HF}(z)$，增益参数用来对低频激励信号进行整形。

ISF 矢量 isf_hf_q 编码使用的是基于预测的 MSVQ。定义 2 比特索引 i_1 表示第一级码书索引，7 比特索引 i_2 表示第二级码书，则：

$$\text{isf\_hf\_q} = \text{cb1}(i_1) + \text{cb2}(i_2) + \text{mean\_isf\_hf} + \mu_{\text{isf\_hf}} \times \text{mem\_isf\_hf} \qquad \cdots\cdots(135)$$

式中：

$\text{cb1}(i_1)$——第一级码书的第 i_1 个码矢量；

cb2(i_2)——第二级码书的第 i_2 个码矢量；

mean_isf_hf——ISF 矢量的平均值；

$\mu_{isf\_hf}=0.5$——预测系数；

mem_isf_hf——ISF 预测器的存储器，它的更新如下：

$$mem\_isf\_hf = isf\_hf\_q - mean\_isf\_hf \quad (mem\_isf\_hf 初始值为 0)$$

解码的 ISF 参数先转换为 ISP 系数，再按照 6.2.3.2.7 中 ISP 系数的插值方法计算每个子帧 ISP 系数，然后把插值后 ISP 系数转换为 LP 系数，用于 LP 合成滤波器来重建信号。

根据 4 维的高频残差增益 VQ 码表的 7 比特索引，残差增益解码公式见式(136)：

$$(g_0, g_1, g_2, g_3) = cb\_gain\_hf(idx) \quad \cdots\cdots(136)$$

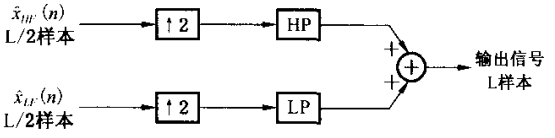
式中：

cb_gain_hf(idx)——码书 cb_gain_hf 的第 idx 个码矢量。

最终每个子帧的高频增益转换线性表示为 $10^{g_i/20}$。通过增益与相应低频拷贝频谱乘积获得高频残差分量，最后通过高频合成滤波输出高频信号。

6.3.3 解码后处理

将 $Fs/2$ 采样的低频解码信号 $\hat{x}_{LF}(n)$和高频解码信号 $\hat{x}_{HF}(n)$经过上采样恢复到 Fs 采样频率，然后相加就得到全带输出信号，见图 65。



L——音频超帧长度

图 65 低频和高频信号合成

如果输出信号的采样频率不同于 Fs，则需要进行采样频率转换。

6.3.4 识别特征参数的解码

从编码码流中提取码书索引，VAD 标志位以及 CRC 校验和。根据码书索引，从 VQ 码书中查找得到估计矢量。

$$\begin{bmatrix}\hat{y}_i(t)\\ \hat{y}_{i+1}(t)\end{bmatrix} = q^{i,i+1}_{idx^{i,i+1}(t)}, i=\{0,2,4\cdots,12\} \quad \cdots\cdots(137)$$

在直接编码模式下，估计矢量就是解码的识别特征参数。在预测编码模式下，估计矢量只是残差矢量，需要对音频解码器输出的重建音频信号提取特征矢量，最后同解码得到的残差矢量相加得到解码的识别特征参数。

6.4 比特分配描述

6.4.1 音频编码器的比特分配描述

表 89 和表 90 给出了 ACELP 和 TVC 编码模式下音频码流的比特分配表，这些表给出了音频编码器产生的比特流的顺序，在输出比特时编码器先输出每个编码参数的 MSB。

表 89 ACELP 编码模式下的比特分配表

单位为比特每帧

	比特(MSB-LSB)							
描述	488	424	392	344	312	280	248	216
模式位	b0-b3	b0-b3	b0-b3	b0-b3	b0-b3	b0-b3	b0-b3	b0-b3
填充位	b4-b8	b4-b8	b4-b8	b4-b8	b4-b8	b4-b8	b4-b8	b4-b8
感知加权标志位	b9	b9	b9	b9	b9	b9	b9	b9
第 1 个 ISP 子矢量	b10-b19	b10-b19	b10-b19	b10-b19	b10-b19	b10-b19	b10-b19	b10-b19
第 2 个 ISP 子矢量	b20-b28	b20-b28	b20-b28	b20-b28	b20-b28	b20-b28	b20-b28	b20-b28
第 3 个 ISP 子矢量	b29-b37	b29-b37	b29-b37	b29-b37	b29-b37	b29-b37	b29-b37	b29-b37
第 4 个 ISP 子矢量	b38-b46	b38-b46	b38-b46	b38-b46	b38-b46	b38-b46	b38-b46	b38-b46
第 5 个 ISP 子矢量	b47-b55	b47-b55	b47-b55	b47-b55	b47-b55	b47-b55	b47-b55	b47-b55
平均能量索引	b56-b57	b56-b57	b56-b57	b56-b57	b56-b57	b56-b57	b56-b57	b56-b57
子帧 1								
自适应码书索引	b58-b66	b58-b66	b58-b66	b58-b66	b58-b66	b58-b66	b58-b66	b58-b66
LTP 滤波标志	b67	b67	b67	b67	b67	b67	b67	b67
代数码书索引	b68-b155	b68-b139	b68-b131	b68-b119	b68-b111	b68-b103	b68-b95	b68-b87
码书增益索引	b156-b162	b140-b146	b132-b138	b120-b126	b112-b118	b104-b110	b96-b102	b88-b94
子帧 2								
自适应码书索引	b163-b168	b147-b152	b139-b144	b127-b132	b119-b124	b111-b116	b103-b108	b95-b100
LTP 滤波标志	b169	b153	b145	b133	b125	b117	b109	b101
代数码书索引	b170-b257	b154-b225	b146-b209	b134-b185	b126-b169	b118-b153	b110-b137	b102-b121
码书增益索引	b258-b264	b226-b232	b210-b216	b186-b192	b170-b176	b154-b160	b138-b144	b122-b128
子帧 3								
自适应码书索引	b265-b273	b233-b241	b217-b225	b193-b201	b177-b185	b161-b169	b145-b153	b129-b137
LTP 滤波标志	b274	b242	b226	b202	b186	b170	b154	b138
代数码书索引	b275-b362	b243-b314	b227-b290	b203-b254	b187-b230	b171-b206	b155-b182	b139-b158
码书增益索引	b363-b369	b315-b321	b291-b297	b255-b261	b231-b237	b207-b213	b183-b189	b159-b165
子帧 4								
自适应码书索引	b370-b375	b322-b327	b298-b303	b262-b267	b238-b243	b214-b219	b190-b195	b166-b171
LTP 滤波标志	b376	b328	b304	b268	b244	b220	b196	b172
代数码书索引	b377-b464	b329-b400	b305-b368	b269-b320	b245-b288	b221-b256	b197-b224	b173-b192
码书增益索引	b465-b471	b401-b407	b369-b375	b321-b327	b289-b295	b257-b263	b225-b231	b193-b199
带宽扩展								
高频 ISP 索引	b472-b480	b408-b416	b376-b384	b328-b336	b296-b304	b264-b272	b232-b240	b200-b208
高频增益索引	b481-b487	b417-b423	b385-b391	b337-b343	b305-b311	b273-b279	b241-b247	b209-b215

表 90 TVC 编码模式下的比特分配表

单位为比特每帧

	比特(MSB-LSB)							
描述	488	424	392	344	312	280	248	216
模式位	b0-b3	b0-b3	b0-b3	b0-b3	b0-b3	b0-b3	b0-b3	b0-b3
填充位	b4-b8	b4-b8	b4-b8	b4-b8	b4-b8	b4-b8	b4-b8	b4-b8
感知加权标志位	b9	b9	b9	b9	b9	b9	b9	b9
第 1 个 ISP 子矢量	b10-b19	b10-b19	b10-b19	b10-b19	b10-b19	b10-b19	b10-b19	b10-b19
第 2 个 ISP 子矢量	b20-b28	b20-b28	b20-b28	b20-b28	b20-b28	b20-b28	b20-b28	b20-b28
第 3 个 ISP 子矢量	b29-b37	b29-b37	b29-b37	b29-b37	b29-b37	b29-b37	b29-b37	b29-b37
第 4 个 ISP 子矢量	b38-b46	b38-b46	b38-b46	b38-b46	b38-b46	b38-b46	b38-b46	b38-b46
第 5 个 ISP 子矢量	b47-b55	b47-b55	b47-b55	b47-b55	b47-b55	b47-b55	b47-b55	b47-b55
噪声因子	b56-b58	b56-b58	b56-b58	b56-b58	b56-b58	b56-b58	b56-b58	b56-b58
全局增益	b59-b65	b59-b65	b59-b65	b59-b65	b59-b65	b59-b65	b59-b65	b59-b65
缩放因子	b66-b72	b66-b72	b66-b72	b66-b72	b66-b72	b66-b72	b66-b72	b66-b72
代数 VQ 参数	b73-b471	b73-b407	b73-b375	b73-b327	b73-b295	b73-b263	b73-b231	b73-b199
带宽扩展								
高频 ISP 索引	b472-b480	b408-b416	b376-b384	b328-b336	b296-b304	b264-b272	b232-b240	b200-b208
高频增益索引	b481-b487	b417-b423	b385-b391	b337-b343	b305-b311	b273-b279	b241-b247	b209-b215

6.4.2 识别特征参数编码的比特分配描述

见 6.2.7.2 的描述。

6.5 存储、传输接口格式

6.5.1 编码模式和码率

表 91 规定了音频编码的编码模式和码率。

表 91 编码模式和码率

编码模式	每帧比特数	25.6 kHz 码率/kbps
0	216	10.8
1	248	12.4
2	280	14.0
3	312	15.6
4	344	17.2
5	392	19.6
6	424	21.2
7	488	24.4

每个音频帧由 512 个样本组成，但时间长度却依赖于内部采样频率 Fs，表 92 列出了音频编码器支持的内部采样频率。

表 92 内部采样频率和对应的每帧时长

内部采样频率索引	内部采样频率/Hz	帧长度/ms	内部采样频率因子
0	保留	保留	保留
1	12 800	40	1/2
2	16 000	32	5/8
3	24 000	21.33	15/16
4	25 600	20	1
5	32 000	16	5/4
6	38 400	13.33	3/2

表 92 中的最后一列采样频率因子＝内部采样频率/25 600。内部采样频率索引在码流中用来指明每帧使用了哪种内部采样频率。

表 93 列出的音频编码的帧类型用来表示每个音频帧的编码模式和字节数。编码帧类型和内部采样频率这两个参数共同决定了编码码率。

表 93 音频编码的帧类型

帧类型	编码模式	25.6kHz 码率/kbps	每帧字节数
0	0	10.8	27
1	1	12.4	31
2	2	14.0	35
3	3	15.6	39
4	4	17.2	43
5	5	19.6	49
6	6	21.2	53
7	7	24.4	61
8～15	保留		

识别特征参数编码提供两种编码模式：直接编码模式和预测编码模式。直接编码模式码率为 4.8kbps，对应每帧数据 48 比特，而预测编码模式码率为 3.2 kbps，对应每帧数据 32 比特。

码流数据打包时，为了节省帧头开销，每个帧头后紧跟一个音频超帧数据。每个音频超帧由若干音频帧组成，目前本标准规定音频超帧中只包含一个音频帧。在每个音频超帧数据中，先打包所有的音频帧码流，再打包所有的识别特征参数帧码流。由于识别特征参数帧的时间长度和音频帧的时间长度并不相同，因此在帧头中编码 1 比特标志位来表示当前音频超帧包括了 N 帧的识别特征参数码流还是 N＋1帧的识别特征参数码流。N 取值同音频编码的内部采样频率和音频超帧的长度有关，计算如下：

$$N=\mathrm{Floor}\left(\frac{\text{超帧的样本点数}}{\text{内部采样频率(Hz)}\times 0.01}\right)$$

音频支持两种码流格式：RAW 格式和 NAL 格式。NAL 格式是在 RAW 格式上增加了 NAL 层封装。字节流 NAL 单元语法见 5.2.3.1。

6.5.2 音频数据 RAW 格式

6.5.2.1 音频数据单元格式

音频数据单元 audio_data_unit()定义见表 94 所示，一个音频数据单元打包一个音频超帧数据。

表 94 音频数据单元的格式

audio_data_unit( ){	描述符	说明
audio_frame_header( )		帧头,包括编码参数和码流标志位
if(extension_flag)		
audio_frame_extension_header( )		扩展帧头,包括编码扩展信息
audio_frame_data( )		音频超帧数据,包括音频码流和识别特征参数码流
}		

帧头 audio_frame_header()定义见表 95 所示。

表 95 帧头格式

audio_frame_header( ){	描述符	说明
version_id	u(1)	编码器的版本号,0 表示 1.0,其他值保留
profile-id	u(2)	码流的档次,档次定义见表 G.1
level_id	u(4)	码流的级别,级别定义见表 G.2
bitstream_type	u(1)	码流的内容:0 表示只有音频码流,1 表示有音频码流和识别特征参数码流
isf_index	u(3)	音频编码采用的内部采样频率索引,具体索引定义见表 92
frame_type	u(4)	音频编码的帧类型,具体帧类型定义见表 93
extension_flag	u(1)	扩展帧头标志位:0 表示不编码,1 表示编码
}		

扩展帧头 audio_frame_extension_header()定义见表 96 所示。

表 96 扩展帧头格式

audio_frame_extension_header( ){	描述符	说明
feature_type	u(2)	编码的识别特征参数:0 表示 MFCC,其他值保留
feature_mode	u(1)	识别特征参数的编码模式:0 表示直接编码,1 表示预测编码
feature_bs_flag	u(1)	音频超帧中识别特征参数码流帧数的标志位:0 表示 N 帧识别特征参数码流,1 表示 N+1 帧识别特征参数码流
ref_time_flag	u(1)	编码绝对时间信息标志位:0 表示不编码,1 表示编码。规定绝对时间表示音频超帧的第 1 个样本的时间
event_flag	u(1)	编码异常声音事件类型标志位:0 表示不编码,1 表示编码
direction_flag	u(1)	编码声源定向信息标志位:0 表示不编码,1 表示编码
reserve_bit	u(1)	保留位,固定为 0
if(ref_time_flag){		
hour_bits	u(5)	小时信息,取值范围在 0～23 之间(包括 0 和 23)
minute_bits	u(6)	分钟信息,取值范围在 0～59 之间(包括 0 和 59)
second_bits	u(6)	秒信息,取值范围在 0～59 之间(包括 0 和 59)
second_fraction_bits	u(14)	秒的分数信息,以 1/16 384 s 为单位,取值范围在 0～16 383 之间(包括 0 和 16 383)
ref_date_flag	u(1)	编码日期信息标志位:0 表示不编码,1 表示编码

表 96（续）

	描述符	说明
if(ref_date_flag){		
year_minus2000_bits	u(7)	加 2000 表示年份信息，Year = year_minus2000_bits + 2000，取值范围在 0～127 之间(包括 0 和 127)
month_minus1_bits	u(4)	加 1 表示月份信息，Month = month_minus1_bits + 1，取值范围在 0～11 之间(包括 0 和 11)
date_minus1_bits	u(5)	加 1 表示日期信息，Date = date_minus1_bits + 1，取值范围在 0～30 之间(包括 0 和 30)
}		
}		
if(event_flag)		
abnormal_sound_event_type	u(8)	异常声音事件类型，具体类型定义见表 H.1，取值范围应在 0～255 之间(包括 0 和 255)
if(direction_flag){		
azimuth	u(8)	声源定向的水平方向角，编码格式见 6.5.2.2
elevation	u(8)	声源定向的仰角，编码格式见 6.5.2.2
}		

6.5.2.2 声源定向信息编码格式

声源定向信息包括水平方向的方位角 α 和垂直方向的仰角 β，角度定义见图 66 所示，角度有效范围为 0°～359°(360°等效于 0°)。方位角和仰角分别使用 8 比特编码，有效范围 0～255，编码角度的分辨率为 1.406 25°。

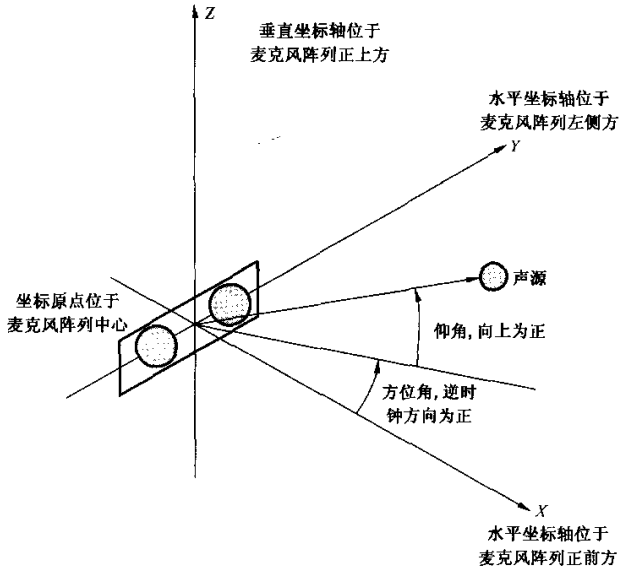


图 66 麦克风阵列的坐标系统

编码时角度量化公式为：

$$\hat{\alpha}=\text{round}(\alpha/1.406\ 25)\qquad \hat{\beta}=\text{round}(\beta/1.406\ 25)$$

解码时角度的反量化公式为：

$$\alpha'=\text{round}(\hat{\alpha}\times 1.406\ 25)\qquad \beta'=\text{round}(\hat{\beta}\times 1.406\ 25)$$

如果前后相邻两帧的声源定向信息没有变化，则置当前帧的扩展帧头中 direction_flag 为零，即当前帧的扩展帧头中不编码声源定向信息，当前帧的声源定向信息同前一帧。

6.5.3 音频数据 NAL 格式

6.5.3.1 音频数据 RBSP 单元

字节流 NAL 单元语法见 5.2.3.1，定义 nal_unit_type 等于 13 的 NAL 单元负载为音频数据 RBSP 单元 audio_data_rbsp( )。

音频数据 RBSP 单元 audio_data_rbsp( )定义见表 97。

表 97 音频数据 RBSP 单元

audio_data_rbsp( ){	描述符	说明
for(i=0; i<audio_unit_num; i++){		audio_unit_num 表示一个音频数据 RBSP 单元包括的音频数据单元个数，取值范围应该在 1～256 之间(包括 1 和 256)
audio_data_unit( )		音频数据单元
}		
rbsp_trailing_bits( )		表示 RBSP 字节流的结束
}		

6.5.3.2 音频数据的绝对时间扩展单元

音频数据采用 NAL 格式后，为了同视频部分保持一致，绝对时间信息将采用单独的 NAL 单元来传输。音频数据单元 audio_data_unit()中不包括绝对时间信息，即 audio_frame_ extension _header() 中 ref_time_flag 置为零。

基于 5.2.3.8 中定义的监控扩展数据单元 surveillance_extension_rbsp( )，定义扩展单元标识(extension_id)等于 5 的监控扩展数据单元包含音频数据的绝对时间信息。

音频数据的绝对时间扩展单元 audio_time_extension( )定义见表 98，其中绝对时间信息中各语法元素定义同表 96。

表 98 音频数据的绝对时间扩展单元

audio_time_extension( ){	描述符	说明
extension_id	u(8)	固定为 5
hour_bits	u(5)	
minute_bits	u(6)	
second_bits	u(6)	
second_fraction_bits	u(14)	
ref_date_flag	u(1)	
if(ref_date_flag) {		
year_minus2000_bits	u(7)	
month_minus1_bits	u(4)	
date_minus1_bits	u(5)	

表 98（续）

	描述符	说明
}		
frame_num	u(8)	绝对时间所对应的音频数据 RBSP 单元中的音频数据单元的序号，规定绝对时间表示所对应音频数据单元的第 1 个样本的时间，取值范围应该在 0～255 之间（包括 0 和 255）
}		

在音频 NAL 单元传输和存储时，规定音频绝对时间的 NAL 单元同后面最靠近的音频数据的 NAL 单元保持对应关系，不允许改变这两个 NAL 单元的先后顺序。

附 录 A
（规范性附录）
假设参考解码器（HRD）

A.1 概述

本附录定义了假设参考解码器（以下简称 HRD）。每一组 HRD 参数确定一个 HRD 操作模式。HRD 包含一个编码图像缓存区（CPB），一个瞬时解码器和一个解码图像缓存区（DPB），见图 A.1。

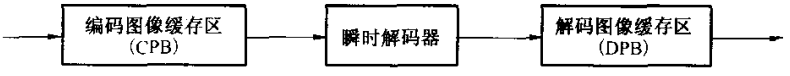


图 A.1 假设参考解码器（HRD）

CPB 大小（比特数）为 CpbSize[SchedSelIdx]，DPB 的大小（帧数）为 Max(1, max_dec_frame_buffering)。HRD 可被某缓存周期 SEI 消息进行初始化。一旦初始化后，随后的缓存周期 SEI 消息不能再初始化 HRD。如果序列参数集改变，则 HRD 需重新初始化。在初始化后，CPB 与 DPB 均为空。编码图像数据按规定的到达时间表注入 CPB 中。图像序号以 n 表示，从 0 开始，在解码过程中，每当一个图像解码完毕，图像序号加 1。当在 CPB 移除时间到达时，图像数据被移除并由实时解码过程进行实时解码，并放入到 DPB 中。如果该图像需要在 CPB 移除时间时输出，并且为非参考图像，则不放入 DPB 中。DPB 可包含多个帧缓存，每个帧缓存可能包括一个解码帧或一帧的两个解码场，或单个解码场，这些图像或作为后续解码图像的参考图像，或保留等待以后输出（重排序或延时）。对于放入 DPB 的图像，在其从 DPB 输出时，或不再作为参考图像时，从 DPB 中移除。CPB 的操作在 A.2 中规定，DPB 的操作在 A.3 中规定。

本附录中所有的操作都为实数操作，不存在舍入误差，例如 HRD 缓存区中的比特数可以不是整数值。HRD 使用两种的时间基准为：一个是 90 kHz 时钟，只在 HRD 收到缓存周期 SEI 消息时使用；另一个是序列参数集中定义的时钟，其时钟基准定义为，tc = num_units_in_tick ÷ time_scale，可以认为是图像采样的最短时间间隔。

注：缓存周期 SEI 消息，及其周期内的编码图像数据，应关联于相同的序列参数集。如果码流中包括多序列参数集，其 HRD 参数应一致。

A.2 编码图像缓存区（CPB）

A.2.1 编码图像到达时间

图像 n 的第一个比特进入 CPB 的时间定义为起始到达时间 $t_{ai}(n)$，最后一个比特进入 CPB 的时间定义为最终到达时间 $t_{af}(n)$。图像 n 的第一个比特最早进入 CPB 的时间定义为最早到达时间 $t_{ai,earliest}(n)$。

$t_{ai}(n)$的计算如下：

如果为第一幅图像，即图像 0，则 $t_{ai}(0) = 0$；

否则（为图像 n，n>0），按照如下规则：

——如果 cbr_flag[SchedSelIdx]等于 1，即恒定比特率情况下，

$t_{ai}(n) = t_{af}(n-1)$

——否则（cbr_flag[SchedSelIdx]等于 0），变比特率情况下，

$t_{ai}(n) = \mathrm{Max}(t_{af}(n-1), t_{ai,earliest}(n))$

$$t_{ai,earliest}(n) = t_c \times \sum_{m=0}^{n-1} \text{cpb\_removal\_delay}[m]$$

其中cpb_removal_delay(n)见 D.2.2 定义。

$t_{af}(n)$的计算如下：

$t_{af}(0)=b(0)\div BitRate[SchedSelIdx]$

$t_{af}(n)=t_{ai}(n)+b(n)\div BitRate[SchedSelIdx]$

其中 b(n)为图像 n 的码字长度。

如果图像 n 与图像 n-1 的序列参数集不同，HRD 参数需要重新初始化。如果图像 n 重新初始化了 HRD 参数，则有：

——BitRate[SchedSelIdx]在 $t_{ai}(n)$时更新；

——对于 CpbSize[SchedSelIdx]，如果 CpbSize[SchedSelIdx]的新值超过原 CPB 的大小，在 $t_{ai}(n)$时更新；

——否则 CpbSize[SchedSelIdx]的新值在 $t_r(n)$时更新。

A.2.2 编码图像移除时间

图像 n 从 CPB 中移除的标定时间定义为标定移除时间 $t_{r,n}(n)$，从 CPB 中移除的实际时间定义为实际移除时间 $t_r(n)$。

$t_{r,n}(n)$的计算如下：

对于图像 0，

$t_{r,n}(0)=$initial_cpb_removal_delay[SchedSelIdx]$\div 90\ 000$

对于图像 n，

$t_{r,n}(n)=t_{r,n}(n-1)+t_c\times$cpb_removal_delay(n)

$t_{r,n}(n-1)$为当前缓存周期前一个图像的标定移除时间，cpb_removal_delay(n)为与图像 n 关联的图像定时 SEI 消息中规定的 cpb_removal_delay 的值。如果图像 n 为未初始化 HRD 的缓存周期的第一个图像，则上式中的 $t_{r,n}(n-1)$为前一个缓存周期的最后一个图像的标定移除时间。

$t_r(n)$的计算如下：

如果low_delay_hrd_flag 等于 0 或 $t_{r,n}(n)>=t_{af}(n)$，

$t_r(n)=t_{r,n}(n)$

否则(low_delay_hrd_flag 等于 1 或 $t_{r,n}(n)<t_{af}(n)$)，

$t_r(n)=t_{r,n}(n)+t_c\times Ceil((t_{af}(n)-t_{r,n}(n))\div t_c)$

后一种情况防止 b(n)很大时的非正常移除。

A.3 解码图像缓存区(DPB)

A.3.1 图像的输出和移除

图像n 解码后，其 DPB 输出时间 $t_{o,dpb}(n)$如下：

$t_{o,dpb}(n)=t_r(n)+t_c\times$dpb_output_delay(n)

如果无延迟，$t_{o,dpb}(n)=t_r(n)$，当前图像直接输出。如果为参考图像，则还需将当前图像存储于 DPB 中；否则($t_{o,dpb}(n)>t_r(n)$)，当前图像延迟输出，并存储于 DPB。

当满足如下两个条件时，DPB 中的某图像 m 将被移除：

a) 图像 m 不作为后续解码图像的参考图像(当图像的一帧或两场均不作为参考图像，才能认为不作为参考图像)；

b) 图像 m 的 DPB 输出时间小于或等于 CPB 中当前图像 n 的移除时间，即 $t_{o,dpb}(m)<=t_r(n)$。

当帧缓存里所有数据从 DPB 里移除后，DPB 填充度减 1。

A.3.2 IDR 图像的插入

如果解码图像为 IDR 图像，有如下操作：

a) 所有在 DPB 中的图像均不作为后续解码图像的参考图像；

b) 当其不是第一幅 IDR 图像，且序列参数集中的 PicWidthInMbs，PicHeightInMbs，max_dec_frame_buffering 的值与前面图像对应的序列参数集中定义的值不同时，DPB 中所有的帧缓存清空且不输出，DPB 填充度重置为 0。

A.3.3 图像的标记和存储

如果当前图像为参考图像，或者为非参考图像但 $t_{o,dpb}(n)>t_r(n)$，则需要把当前图像存储于 DPB 中。否则不需要存储。如果需要存储，则有如下操作：

a) 如果当前解码图像为一帧图像的第二场(在解码顺序中)，并且第一场仍在 DPB 中，则和第一场存储在同一帧缓存中；

b) 否则当前解码图像存在一个空的帧缓存中，DPB 填充度加 1。

附 录 B
（规范性附录）
字节流的格式

B.1 概述

本附录规定了字节流格式的语法与语义，用于将 NAL 单元流生成有序的字节流，并且 NAL 单元边界位置应能够从字节流中被识别。对于面向比特的传送，字节流中的比特顺序起始于第一个字节的 MSB，处理至第一个字节的 LSB，然后为第二个字节的 MSB，以此类推。

字节流格式由一系列字节流 NAL 单元语法结构组成。每个字节流 NAL 单元语法结构包含有一个起始码前缀，后面跟随一个 NAL 单元。字节流 NAL 单元语法结构中可能包含一个额外的zero_byte 语法元素，也可能包含一个或多个额外的 trailing_zero_8bits 语法元素。第一个字节流 NAL 单元语法结构中还可能包含一个或多个额外的 leading_zero_8bits 语法元素。

B.2 字节流 NAL 单元语法与语义

B.2.1 字节流 NAL 单元语法

字节流 NAL 单元语法见表 B.1。

表 B.1 字节流 NAL 单元语法表

byte_stream_nal_unit(NumBytesInNALunit) {	描述符
while(next_bits(24) ! =0x000001 && next_bits(32) ! =0x00000001)	
leading_zero_8bits /* 应等于 0x00 */	f(8)
if(next_bits(24) ! = 0x000001)	
zero_byte /* 应等于 0x00 */	f(8)
start_code_prefix_one_3bytes /* 应等于 0x000001 */	f(24)
nal_unit(NumBytesInNALunit)	
while(more_data_in_byte_stream( ) && next_bits(24) ! = 0x000001 && next_bits(32) ! = 0x00000001)	
trailing_zero_8bits /* 应等于 0x00 */	f(8)
}	

B.2.2 字节流 NAL 单元语义

字节流 NAL 单元中 NAL 单元的顺序应遵循 NAL 单元解码顺序，见 5.2.4.3.2。

leading_zero_8bits 应等于 0x00。

注：leading_zero_8bits 语法元素只能在流的第一个字节流 NAL 单元里出现。

zero_byte 应等于 0x00。

当下述任一个条件满足时，应有 zero_byte 语法元素。

——NAL 单元中的 nal_unit_type 等于 7(序列参数集)或 8(图像参数集)。

——字节流 NAL 单元中包含一个基本编码图像的的第一个 NAL 单元，见 5.2.4.3.2.3。

start_code_prefix_one_3bytes 为一个 3 字节的固定值序列，等于 0x000001，该语法元素称为起始码前缀。

trailing_zero_8bits 应等于 0x00。

B.3 字节流 NAL 单元解码过程

本过程的输入为字节流，该字节流由一系列字节流 NAL 单元语法结构组成。

本过程的输出为一系列的 NAL 单元。

在解码过程开始时，解码器把其当前位置初始化为字节流的起始位置。然后提取并丢弃每一个 leading_zero_8bits 语法元素（如果存在的话），并相应移动当前位置，直到当前位置紧接的四个字节为 0x00000001。

解码器此时重复执行下述按步骤的过程，对字节流中每一个 NAL 单元语法结构进行提取与解码，直到字节流中最后一个 NAL 单元也已经解码：

a) 当字节流里的紧接的四个字节构成四字节序列 0x00000001，对比特流中下一个字节（为 zero_byte 语法元素）进行提取并丢弃时，字节流的当前位置设为紧接被丢弃的字节的字节位置；

b) 提取与丢弃比特流中下一个三字节序列（为 start_code_prefix_one_3bytes），且比特流当前位置设为此紧接被丢弃的 3 字节序列的字节的位置；

c) NumBytesInNALunit 设为自当前字节位置起至满足下述任一条件的位置的最后一个字节，且包括最后一个字节的编号；

   1) 一个三字节序列的排列等于 0x000000，或

   2) 一个三字节序列的排列等于 0x000001，或

   3) 字节流的结束。

d) 该 NumBytesInNALunit 字节从比特流中移除，字节流的当前位置前移 NumBytesInNALunit 字节。这个字节序列为 nal_unit(NumBytesInNALunit)，并用 NAL 单元解码过程进行解码；

e) 当字节流中的当前位置不为字节流的结尾，且字节流中一个字节不是等于 0x000001 开始的三字节序列，也不是等于 0x00000001 开始的四字节序列，解码器提取并丢弃每一个 trailing_zero_8bits 语法元素，并相应移动当前位置，直到字节流里的当前位置接下的四个字节构成四字节的序列 0x00000001 或已至字节流的结尾。

注：字节流的结束的判断方法不在本标准中规定。

附 录 C
（规范性附录）
视频档次与级别

C.1 概述

档次与级别规定了对比特流的限制，因此也限制了比特流解码所需的能力。每一个档次定义了一个算法特征的子集，并限定所有与该档次一致的解码器都应支持。每一个级别定义了对本标准中的语法元素取值的限制集合。相同的级别定义集合用于所有的档次，但单独的应用对所支持的档次可能支持不同的级别。一般来说，对于特定的一个档次，不同的级别对应于对解码器负荷和存储器容量的不同要求。

本附录描述了视频不同档次和级别所对应的各种限制。所有未被限定的语法元素和参数可以取任何本标准所允许的值。如果一个解码器能对某个档次和级别所规定的语法元素的所有允许值正确解码，则称此解码器在这个档次和级别上符合本标准。如果一个比特流中不存在某个档次和级别所不允许的语法元素，并且其所含有的语法元素的值不超过此档次和级别所允许的范围，则认为此比特流在这个档次和级别上符合本标准。

profile_id 和 level_id 定义了比特流的档次和级别。

注：解码器不应因为 profile_id 或 level_id 的取值落在本标准所规定的值之间，就推定这个值所代表的能力处于规定的档次与级别之间。

C.2 视频档次

C.2.1 视频档次的定义

视频档次的定义见表 C.1。

表 C.1 视频档次

profile_id	档次
0x00	禁止
0x11	简单档次
0x22	主要档次
0x33	高级档次
其他	保留

C.2.2 简单档次

比特流若与简单档次相一致，应遵循如下限制：

——profile_id 的值应为 0x11；

——只可能有 I 和 P 条带类型存在；

——NAL 单元流中不应包含 nal_unit_type 的取值为 3、4 和 9 的 NAL 单元；

——序列参数集中的参数 progressive_seq_flag 的值应为 1；

——序列参数集中的参数 chroma_format_idc 的取值应为 1；

——序列参数集中的参数 bit_depth_luma_minus8 的取值应为 0；

——序列参数集中的参数 bit_depth_chroma_minus8 的取值应为 0；

——序列参数集中的参数 roi_flag 的取值应为 0；

——序列参数集中的参数 svc_flag 的取值应为 0；

——图像参数集中的参数 entropy_coding_mode_flag 的取值应为 0；

——图像参数集中的参数 weighting_pred_flag 的取值应为 0；

——语法元素 pic_init_qp_for_roi，bottom_right[i]，top_left[i]，non_roi_skip_flag，scalable_non_roi_skip_flag，num_roi 不应在图像参数集中出现。

profile_id 的取值为 0x11 时，比特流与简单档次相一致。与简单档次某一级别相一致的解码器应可以对所有满足 profile_id 等于 0x11，且 level_id 所表示的级别小于或等于该级别的流进行解码。

C.2.3 主要档次

比特流若与主要档次相一致，应遵循如下限制：

——profile_id 的值应为 0x22；

——只可能有 I、P 和 B 条带类型存在；

——NAL 单元流中不应包含 nal_unit_type 的取值为 3 和 4 的 NAL 单元；

——序列参数集中的参数 progressive_seq_flag 的值应为 0 或 1；

——序列参数集中的参数 chroma_format_idc 的取值应为 0 或 1；

——序列参数集中的参数 bit_depth_luma_minus8 的取值应在 0～2 之间，包括 0 和 2；

——序列参数集中的参数 bit_depth_chroma_minus8 的取值应在 0～2 之间，包括 0 和 2；

——序列参数集中的参数 roi_flag 的取值应为 0 或 1；

——序列参数集中的参数 svc_flag 的取值应为 0；

——图像参数集中的参数 entropy_coding_mode_flag 的取值应为 0 或 1；

——图像参数集中的参数 weighting_pred_flag 的取值应为 0 或 1；

——语法元素 scalable_non_roi_skip_flag 不应在图像参数集中出现。

profile_id 的取值为 0x22 时，比特流与主要档次相一致。与主要档次某一级别相一致的解码器应可以对所有满足 profile_id 等于 0x22，且 level_id 所表示的级别小于或等于该级别的流进行解码。

C.2.4 高级档次

比特流若与高级档次相一致，应遵循如下限制：

——profile_id 的值应为 0x33；

——只可能有 I、P 和 B 条带类型存在；

——序列参数集中的参数 progressive_seq_flag 的值应为 0 或 1；

——序列参数集中的参数 chroma_format_idc 的取值应在 0～2 之间，包括 0 和 2；

——序列参数集中的参数 bit_depth_luma_minus8 的取值应在 0～2 之间，包括 0 和 2；

——序列参数集中的参数 bit_depth_chroma_minus8 的取值应在 0～2 之间，包括 0 和 2；

——序列参数集中的参数 roi_flag 的取值应为 0 或 1；

——序列参数集中的参数 svc_flag 的取值应为 0 或 1；

——图像参数集中的参数 entropy_coding_mode_flag 的取值应为 0 或 1；

——图像参数集中的参数 weighting_pred_flag 的取值应为 0 或 1。

profile_id 的取值为 0x33 时，比特流与高级档次相一致。与高级档次某一级别相一致的解码器应可以对所有满足 profile_id 等于 0x33，且 level_id 所表示的级别小于或等于该级别的流进行解码。

C.3 视频级别

C.3.1 视频级别的定义

视频级别的定义见表 C.2。

表 C.2 视频级别

level_id	级别
0x00	禁止
0x10	2.0
0x11	2.1
0x20	4.0
0x22	4.2
0x30	5.0
0x32	5.2
0x40	6.0
0x42	6.2
其他	保留

C.3.2 级别的参数限制

不同级别对应的参数限制见表 C.3～表 C.6。

表 C.3 级别的参数限制

参数	级别	
	2.0	2.1
每行最大样点数	352	352
每帧最大行数	288	288
每秒最大帧数	30	30
亮度样点速率	3,041,280	3,041,280
最大比特率 MaxBR /bps	1,000,000	2,500,000
每帧最大宏块个数	396	396
每秒最大宏块个数	11,880	11,880
最大垂直运动矢量范围(帧编码)	[−128, +127.75]	[−128, +127.75]
最大垂直运动矢量范围(场编码)	—	—
最大水平运动矢量范围	[−2 048, +2 047.75]	[−2 048, +2 047.75]
图像格式	4∶0∶0 或 4∶2∶0	4∶0∶0 或 4∶2∶0
DPB 最大容量 MaxDPB /kB	594.0	594.0
CPB 最大容量 MaxCPB /kB	2,000	2,500

表 C.4 级别的参数限制

参数	级别	
	4.0	4.2
每行最大样点数	720	720
每帧最大行数	576	576
每秒最大帧数	30	30
亮度样点速率	12,441,600	12,441,600

表 C.4（续）

参数	级别	
	4.0	4.2
最大比特率 MaxBR /bps	6,000,000	15,000,000
每帧最大宏块个数	1,620	1,620
每秒最大宏块个数	48,600	48,600
最大垂直运动矢量范围(帧编码)	[−256, +255.75]	[−256, +255.75]
最大垂直运动矢量范围(场编码)	[−128, +127.75]	[−128, +127.75]
最大水平运动矢量范围	[−2 048, +2 047.75]	[−2 048, +2 047.75]
图像格式	4:0:0 或 4:2:0	4:2:0 或 4:2:2
DPB 最大容量 MaxDPB /kB	2,430.0	2,430.0
CPB 最大容量 MaxCPB /kB	10,000	20,000

表 C.5 级别的参数限制

参数	级别	
	5.0	5.2
每行最大样点数	1,280	1,280
每帧最大行数	720	720
每秒最大帧数	30	30
亮度样点速率	921,600	921,600
最大比特率 MaxBR /bps	15,000,000	30,000,000
每帧最大宏块个数	3,600	3,600
每秒最大宏块个数	108,000	108,000
最大垂直运动矢量范围(帧编码)	[−512, +511.75]	[−512, +511.75]
最大垂直运动矢量范围(场编码)	[−256, +255.75]	[−256, +255.75]
最大水平运动矢量范围	[−2 048, +2 047.75]	[−2 048, +2 047.75]
图像格式	4:0:0 或 4:2:0	4:2:0 或 4:2:2
DPB 最大容量 MaxDPB /kB	5,400.0	5,400.0
CPB 最大容量 MaxCPB /kB	20,000	40,000

表 C.6 级别的参数限制

参数	级别	
	6.0	6.2
每行最大样点数	1,920	1,920
每帧最大行数	1,088	1,088
每秒最大帧数	30	30
亮度样点速率	62,668,800	62,668,800
最大比特率 MaxBR /bps	30,000,000	50,000,000

表 C.6（续）

参数	级别	
	6.0	6.2
每帧最大宏块个数	8,160	8,160
每秒最大宏块个数	244,800	244,800
最大垂直运动矢量范围(帧编码)	[−512，+511.75]	[−512，+511.75]
最大垂直运动矢量范围(场编码)	[−256，+255.75]	[−256，+255.75]
最大水平运动矢量范围	[−2 048，+2 047.75]	[−2 048，+2 047.75]
图像格式	4∶0∶0 或 4∶2∶0	4∶2∶0 或 4∶2∶2
DPB 最大容量 MaxDPB /kB	12,240.0	12,240.0
CPB 最大容量 MaxCPB /kB	40,000	60,000

注：表 C.3～表 C.6 中 DPB 最大容量(MaxDPB)以 4∶2∶0 格式 8 比特样点为例。其中 1 kB 等于 1 024 字节。

附 录 D
（规范性附录）
视频可用性信息(VUI)

D.1 视频可用性信息(VUI)语法

D.1.1 视频可用性信息(VUI)参数语法

VUI 参数语法见表 D.1。

表 D.1 VUI 参数语法表

vui_parameters( ) {	描述符
timing_info_present_flag	u(1)
if(timing_info_present_flag) {	
num_units_in_tick	u(32)
time_scale	u(32)
fixed_frame_rate_flag	u(1)
}	
hrd_parameters_present_flag	u(1)
if(hrd_parameters_present_flag)	
hrd_parameters( )	
if(hrd_parameters_present_flag)	
low_delay_hrd_flag	u(1)
max_dec_frame_buffering	ue(v)
}	

D.1.2 假设参考解码器(HRD)参数语法

HRD 参数语法见表 D.2。

表 D.2 HRD 参数语法表

hrd_parameters( ) {	描述符
cpb_cnt_minus1	ue(v)
bit_rate_scale	u(4)
cpb_size_scale	u(4)
for(SchedSelIdx = 0; SchedSelIdx <= cpb_cnt_minus1; SchedSelIdx++) {	
bit_rate_value_minus1[SchedSelIdx]	ue(v)
cpb_size_value_minus1[SchedSelIdx]	ue(v)
cbr_flag [SchedSelIdx]	u(1)
}	
initial_cpb_removal_delay_length_minus1	u(5)
cpb_removal_delay_length_minus1	u(5)
dpb_output_delay_length_minus1	u(5)
}	

D.2 视频可用性信息(VUI)语义

D.2.1 视频可用性信息(VUI)参数语义

timing_info_present_flag 等于 1 表示 num_units_in_tick,time_scale 和 fixed_frame_rate_flag 在码流中存在,timing_info_present_flag 等于 0 表示不存在上述参数。

num_units_in_tick 表示在频率为 time_scale Hz 的 clock tick 计数器的时间单元的个数。该值应大于 0,一个时钟 tick 为编码数据表示的最小时间间隔。例如,如果视频信号的频率为(30 000÷1001) Hz,time_scale 可以是 30 000,num_units_in_tick 可以是 1001。

time_scale 表示一秒钟内时间单元的总数。该值应大于 0。例如,一个时钟频率为 27 MHz 的定时系统的 time_scale 应为 27 000 000。

fixed_frame_rate_flag 等于 1,表示连续图像的 HRD 输出时间的时域间隔是固定的,该间隔为 num_units_in_tick ÷ time_scale。fixed_frame_rate_flag 等于 0 则表示无此限制。

hrd_parameters_present_flag 等于 1 表示码流中后面紧跟 HRD 参数。否则表示后续码流中不存在 HRD 参数。另外,两个临时变量 HrdBpPresentFlag(用于缓存周期 SEI 消息)和 CpbDpbDelaysPresentFlag(用于图像定时 SEI 消息)的取值应与 hrd_parameters_present_flag 相同。

low_delay_hrd_flag 定义了 HRD 的一种操作模式。当 fixed_frame_rate_flag 等于 1 时,low_delay_hrd_flag 应等于 0。

max_dec_frame_buffering 定义了 HRD DPB 的帧缓存的最大个数。其取值范围从 0 到 MaxDPB,如果码流中没有该语法元素,则为 MaxDPB(见附录 C)。

D.2.2 假设参考解码器(HRD)参数语义

cpb_cnt_minus1 加 1 定义了可供选择的 CPB 参数的个数。cpb_cnt_minus1 的取值范围为 0～31(包括 0 和 31)。如果 low_delay_hrd_flag 等于 1,cpb_cnt_minus1 应等于 0。如果码流中没有 cpb_cnt_minus1,则默认其值等于 0。

bit_rate_scale,bit_rate_value_minus1[SchedSelIdx]两个参数一起定义了第 SchedSelIdx 个 CPB 的最大输入码率。bit_rate_value_minus1[SchedSelIdx]的取值范围 0～($2^{32}-2$)(包括 0 和 $2^{32}-2$),并且随索引号 SchedSelIdx 的变大而变大。码率公式为:

$$BitRate[SchedSelIdx]=(bit\_rate\_value\_minus1[SchedSelIdx]+1)\times 2^{(6+bit\_rate\_scale)}$$

如果 bit_rate_value_minus1[SchedSelIdx]不存在,则默认其值等于 1 200×MaxBR。

cpb_size_scale,cpb_size_value_minus1[SchedSelIdx]两个参数一起定义了第 SchedSelIdx 个 CPB 的 CPB 的大小。cpb_size_value_minus1[SchedSelIdx]的取值范围 0～($2^{32}-2$)(包括 0 和 $2^{32}-2$),并且随索引号 SchedSelIdx 的变大而变小。CPB 大小公式为:

$$CpbSize[SchedSelIdx]=(cpb\_size\_value\_minus1[SchedSelIdx]+1)\times 2^{(4+cpb\_size\_scale)}$$

如果 cpb_size_value_minus1[SchedSelIdx]不存在,则默认其值等于 1 200×MaxCPB。

cbr_flag[SchedSelIdx]定义了第 SchedSelIdx 个 CPB 参数定义的 HRD 工作模式。如果其值等于 0,则为可变比特率模式(VBR),如果其值等于 1,则为恒定比特率模式(CBR)。如果码流中不存在 cbr_flag[SchedSelIdx],则默认其值等于 0。

initial_cpb_removal_delay_length_minus1 加 1 等于缓存周期 SEI 消息中的语法元素 initial_cpb_removal_delay[SchedSelIdx]的码字长度。如果码流中不存在 initial_cpb_removal_delay_length_minus1,则默认其值等于 23。

cpb_removal_delay_length_minus1 加 1 等于图像定时 SEI 消息中的语法元素 cpb_removal_delay 的码字长度。如果码流中不存在 cpb_removal_delay_length_minus1,则默认其值等于 23。

dpb_output_delay_length_minus1 加 1 等于图像定时 SEI 消息中的语法元素 dpb_output_delay 的码字长度。如果码流中不存在 dpb_output_delay_length_minus1,则默认其值等于 23。

附 录 E
（规范性附录）
补充增强信息(SEI)

E.1 补充增强信息(SEI)语法

E.1.1 补充增强信息(SEI)负载语法

SEI 负载语法见表 E.1。

表 E.1 SEI 负载语法表

sei_payload(PayloadType,PayloadSize){	描述符
if(PayloadType==0)	
buffering_period(PayloadSize)	
else if(PayloadType==1)	
pic_timing(PayloadSize)	
else	
reserved_sei_message(PayloadSize)	
if(! byte_aligned( )){	
bit_equal_to_one /* 应等于 1 */	f(1)
while(! byte_aligned( ))	
bit_equal_to_zero /* 应等于 0 */	f(1)
}	
}	

E.1.2 缓存周期 SEI 消息语法

缓存周期 SEI 消息语法见表 E.2。

表 E.2 缓存周期 SEI 消息语法表

buffering_period(PayloadSize){	描述符
seq_parameter_set_id	ue(v)
if(HrdBpPresentFlag){	
for(SchedSelIdx=0;SchedSelIdx<=cpb_cnt_minus1;SchedSelIdx++)	
initial_cpb_removal_delay[SchedSelIdx]	u(v)
}	
}	

E.1.3 图像定时 SEI 消息语法

图像定时 SEI 消息语法见表 E.3。

表 E.3 图像定时 SEI 消息语法表

pic_timing(PayloadSize){	描述符
if(CpbDpbDelaysPresentFlag){	
cpb_removal_delay	u(v)
dpb_output_delay	u(v)
}	
}	

E.2 补充增强信息(SEI)负载语义

E.2.1 缓存周期 SEI 消息语义

如果应用需要,缓存周期 SEI 消息应与 IDR 图像一起出现。该消息提供 HRD 初始化信息。

seq_parameter_set_id 指示包含相关 HRD 参数的序列参数集。

initial_cpb_removal_delay[SchedSelIdx]定义了在 HRD 初始化以后第一个缓存周期的第 SchedSelIdx 个 CPB 的初始延迟,即从图像的第一个 bit 到达 CPB 的时间,到该图像的数据开始从 CPB 移除的时间。其码字长度为 initial_cpb_removal_delay_length_minus1+1,本文件规定以 90 kHz 为单位。

该值应大于 0,并小于 90 000×CpbSize[SchedSelIdx]÷BitRate[SchedSelIdx]。

E.2.2 图像定时 SEI 消息定义

cpb_removal_delay 定义了该 SEI 消息关联的图像从 CPB 中移除前等待的时间。该值还可以用来计算图像数据到达 CPB 的最早的可能时间。该码字长度为 cpb_removal_delay_length_minus1+1。

对于码流中的第一个图像,cpb_removal_delay 应为 0。

dpb_output_delay 定义了图像数据从 CPB 中移除后到解码图像从 DPB 输出的等待时间,用来计算图像的 DPB 输出时间。其码字长度为 dpb_output_delay_length_minus1+1。

附 录 F
（规范性附录）
变 长 码 表

表 F.1～表 F.20 的内容应依序符合GB/T 20090.2—2006《信息技术 先进音视频编码 第 2 部分：视频》附录 A 中表 A.1～表 A.20 的规定。

表 F.1～表 F.20 中 Run 表示量化系数游程，Level 表示量化系数值，EOB 栏及 Level 栏的各列数据表示语法元素 trans_coefficient 的值。解析时根据 trans_coefficient 可以查表得到 Level 和 Run。表中 EOB 栏对应的数字表示代表 EOB 的 trans_coefficient 的值。

附 录 G
（规范性附录）
音频档次和级别

G.1 概述

本附录描述了不同档次和级别所对应的各种限制。档次与级别规定了对比特流的限制，因此也限制了比特流解码所需的能力。每个档次定义了一个算法特征的子集，并限定所有与该档次一致的解码器都应支持。每个级别定义了对本标准中的语法要素取值的限制集合。相同的级别定义集合用于所有的档次，但单独的应用对所支持的档次可能支持不同的级别。一般来说，对于特定的一个档次，不同的级别对应于对解码器负载和存储器容量的不同要求。

如果一个解码器能对某个档次和级别所规定的语法元素正确解码，则称此解码器在这个档次和级别上符合本标准。如果比特流中不存在某个档次和级别所不允许的语法元素，并且其所含有的语法元素的值不超过此档次和级别所允许的范围，则认为此比特流在这个档次和级别上符合本文件。

profile_id 和 level_id 定义了比特流的档次和级别。

注：解码器不宜因为 profile_id 或 level_id 的取值落在本标准所规定的值之间，就推定这个值所代表的能力处于规定好的档次与级别之间。

G.2 音频档次

音频档次主要定义编码器所包括的主要编码工具，目前分 3 个档次：简单档次、主要档次和高级档次。profile_id 采用 2 比特表示，0 表示禁止，1 表示简单档次、2 表示主要档次，3 表示高级档次。具体定义见表 G.1。

表 G.1 音频档次定义

编 码 工 具	简单档次	主要档次	高级档次
ACELP	支持	支持	支持
BWE	支持	支持	支持
TVC	不支持	不支持	支持
识别特征参数的直接编码模式	不支持	支持	支持
识别特征参数的预测编码模式	不支持	不支持	支持

G.3 音频级别

音频级别主要限制编码参数取值和编解码延迟，level_id 采用 4 比特表示，共 16 个级别，见表 G.2 和表 G.3。

表 G.2 音频级别定义

level_id	级 别
0	禁止
1	1.0
2	1.1
3	1.2
4～15	保留

表 G.3 音频级别 1.0～1.2 参数限制

参数	级别		
	1.0	1.1	1.2
内部采样频率/kHz	12.8 和 16	24 和 25.6	32 和 38.4
音频超帧样本点数	512	512	512
最大编解码延迟/ms	60	40	30
最大比特率/(bit/s)	23 050	34 000	48 610

注 1：编解码延迟包括：1 个超帧长度＋LPC 分析窗前瞻样本＋其他延迟(如采样频率转换等)。

注 2：比特率指 RAW 格式码率，包括帧头和扩展帧头开销。

附 录 H
（规范性附录）
异常声音事件类型定义

异常声音事件类型定义见表 H.1。

表 H.1 异常声音事件类型定义

事件类型	事件描述
0	正常声音事件(包括闹市噪声，汽车噪声，高斯噪声和正常人说话声等)
1	人的尖叫声、救命声
2	枪声
3	爆炸声
4	报警声
5	玻璃破碎声
6～255	保留

附 录 I
（资料性附录）
VAD 检测

I.1 概述

VAD 检测将输入的音频信号分为两类：语音和非语音（噪声或静音）。识别特征参数提取时需要检测每帧信号的类别，将来模式识别模块会根据信号的类别，将非语音帧信号的识别特征参数丢掉。识别模块在对识别特征参数进行后处理时，需要连续的帧参数计算识别特征参数的一阶导数和二阶导数，因此识别特征参数提取时需要保留非语音帧信号的识别特征参数。

I.2 VAD 检测介绍

VAD 检测包含两个阶段：第一阶段是基于帧的检测阶段，内部包含三种检测方式；第二阶段为决策阶段。第一阶段的每种检测结果都存储在循环缓冲中，用来分析并得出语音的似然值。第二阶段的最终决策结果需要参考缓冲中的最初几帧，所以该阶段提供了预测机制。同时，此阶段还提供了延迟释放机制，延迟释放的持续时间同语音的似然值相关。

I.3 检测阶段

VAD 检测采用的参数是语音开始时的能量加速度，该参数具有较好噪声鲁棒性。这种加速度可以通过以下三种方法来计算：

方法 1：全带频谱检测法

全带频谱测量法采用的参数，是通过两阶段维纳滤波器中第一阶段所得出的 Mel 域维纳滤波器系数（见 J.8）。对 Mel 域维纳滤波器系数求和后再平方的值作为输入值 input。

每帧的处理步骤如下所示：

a) $if(frame<15\&\&Acceleration<2.5)trac\ \mathrm{ker}=MAX(trac\ \mathrm{ker},input)$；

b) $if(input<trac\ \mathrm{ker}\times UpperBound\ \&\&\ input)trac\ \mathrm{ker}\times LowerBound)$

$trac\ \mathrm{ker}=a\times trac\ \mathrm{ker}+(1-a)\times input$；

c) $if(input<trac\ \mathrm{ker}\times Floor)$

$trac\ \mathrm{ker}=b\times trac\ \mathrm{ker}+(1-b)\times input$；

d) $if(input)trac\ \mathrm{ker}\times threshold)$

$return\ true$；

$else$

$return\ false$；

式中：

$a=0.8$；

$b=0.97$；

$UpperBound=1.5$；

$LowerBound=0.75$；

$Floor=0.5$；

$threshold=1.65$；

$tracker$——噪声能量估计值；

$Acceleration$——测量的加速度，可以通过连续输入的二阶差分来估计，但本检测算法通过

跟踪连续输入的两个平均数 $0\times mean+1\times input$ 和 $((frame-1)\times mean+1\times input)/frame$ 的比率来估计。

方法 2:子带频谱检测法

子带频谱检测法的输入是由方法 1 中产生的第二、第三和第四 Mel 域维纳滤波器系数的平均值。检测器对每帧的处理步骤如下所示:

a) $input=p\times currentInput+(1-p)\times \Pr eviousInput$;

b) $if(Frame<15)trac\ \ker=MAX(trac\ \ker,input)$;

c) $if(input<trac\ \ker\times UpperBound\ \&\&\ input>trac\ \ker\times LowerBound)$

$trac\ \ker=a\times trac\ \ker+(1-a)\times input$;

d) $if(input<trac\ \ker\times Floor)$

$trac\ \ker=b\times trac\ \ker+(1-b)\times input$;

e) $if(input)trac\ \ker\times threshold)$

$return\ true$;

$else$

$return\ false$;

式中:

$p=0.75$;

threshold=3.25,其他参数和方法 1 中相同。

方法 3:频谱方差检测法

频谱方差检测法的输入部分是由每帧全带范围内线性频率维纳滤波器系数的方差构成。方差的计算公式为:

$$\frac{1}{N_{SPEC}}\sum_{bin=0}^{N_{SPEC}-1}(H_2(bin))^2-\left(\sum_{bin=0}^{N_{SPEC}-1}H_2(bin)\right)^2/N_{SPEC}^2 \qquad \cdots\cdots\cdots\cdots(\text{I.1})$$

式中:

$N_{SPEC}=N_{FFT}/4$;

$H_2(bin)$—— 线性频率维纳滤波器系数。

方法 3 的第一步同方法 2 的 b);第二步到第四步同方法 1 的 b)~d),其中 LowerBound=0.85,Floor=0.25,其他参数不变。

I.4 决策阶段

VAD 决策算法输入为 I.3 讨论的三种方法输出的结果。这三种方法得到结果 true 或 false(T 或 F),并将其存储在缓冲中。连续帧得出的结果会不断填充缓冲。该过程提供了缓冲模式的上下文分析。只有缓冲填满了有效结果之后,VAD 决策算法才会进行输出。该过程导致了值为缓冲长度减一的帧延迟。

对于一个 N=7 帧的缓冲,最新的结果存放在第 N 个位置。有后续结果进入时,缓冲中的值向左平移,见图 I.1 所示。

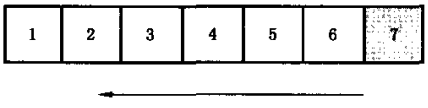


图 I.1 VAD 缓冲区示意图

VAD 决策算法处理步骤如下所示:

步骤 1:$V_N=Measurement\ 1\ or\ Measuremen\ 2\ or\ Measurement\ 3$

由全带频谱检测法、子带频谱检测法和频谱方差检测法得出的值有一个为 true 时,V_N 的结

果也是 true，并将 V_N 存储到位置为 N 的缓冲中。

步骤 2：

$$M=\mathrm{MAX}\left\{\begin{matrix}C++,V_i=true\\C=0,V_i=false\end{matrix},1<i<N\right\}\Big|_C \quad \cdots\cdots\cdots\cdots\cdots\cdots(\mathrm{I}.2)$$

决策算法分析缓冲中的结果，并寻找出缓冲内值为 true 的最长连续序列。在寻找过程中，如果下一个值为 true 时，算子 C 加 1，反之下一个值为 false 时，C 清零。整个缓冲扫描结束后，将算子 C 的最大值赋给 M。例如，序列 T T F T T T F 扫描后，M 的值为 3。

步骤 3：$if(M \geqslant S_p \ \&\ \&\ T < L_s)$

$T=L_s$；

S_p 为“可能是语音”的阈值，对应着第二步得出的 true 值连续序列最大值 $M \geqslant 3$ 的情况。如果延迟释放计时器 T 小于 L_s，则给 T 赋值短延迟释放时间（$L_s=5$ 帧）。

步骤 4：$if(M \geqslant S_L \ \&\ \&\ F > F_s)$

$T=L_M$；

$else\ if(M \geqslant S_L)$

$T=L_L$；

S_L 为“近似为语音”的阈值，对应着第二步得出的 true 值连续序列最大值 M 大于或等于 4 的情况。如果当前帧序号 F 在初始导入安全周期 F_S（35 帧）之外，则给 T 赋值中延迟释放时间 T（$L_M=23$ 帧）；否则，给 T 赋值长延迟释放时间 T（$L_L=50$ 帧），这样做的目的是，防止语音过早出现引起检测器的初始化噪声估计值太大。

步骤 5：$if(M < S_p \ \&\ \&\ T > 0)$

T--；

如果 M 没有达到阈值 S_p 时，将 T 减一。因此 T 只有在语音不存在的情况才会下减少。

步骤 6：$if(T>0)$

return true；

else

return false；

如果 T 大于 0 时，输出为 true（语音帧）；否则，输出为 false（非语音帧）。

步骤 7：在下一帧到达之前，缓冲区左移以接受新的输入帧。

从上面的 VAD 决策过程来看，输出的语音或非语音判决应用于即将离开缓冲区的帧，相应的预测机制如图 I.2 所示。

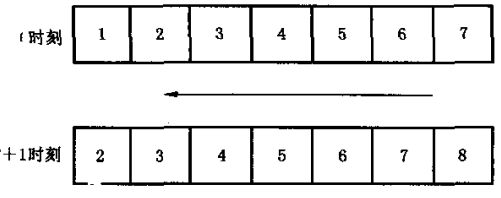


图 i.2　预测机制下缓冲区示意图

在 t 时刻时，缓冲区已经被 7 帧数据填满，第 6 帧和第 7 帧的 V_N 为 true。基于上述决策算法，第一帧的决策结果为 false（非语音帧）。在 $t+1$ 时刻时，缓冲区左移并移出第一帧，新的第 8 帧 V_N 结果为 true。应用决策算法，第二帧的决策结果为 true（语音帧）。当有新帧到达时，对于第 3，4，5 帧也会得到同样的结果。这样就形成了一个 4 帧的短时预测（使用第 6、7、8 帧的结果对第 2、3、4、5 帧进行预测）。

附 录 J
（资料性附录）
噪 声 消 除

J.1 概述

噪声消除算法主要作用是降低背景噪声，提高信号的信噪比。无论语音识别还是声纹识别算法，噪声对识别结果影响很大。因此在识别特征参数提取之前，应先对信号进行降噪处理。

J.2 Mel 域两阶段维纳(Wiener)滤波器

基于维纳滤波器的噪声消除算法由两阶段组成，见图 J.1。输入信号通过第一阶段的降噪处理后，在第二阶段根据处理后信号的信噪比进行噪声消除。

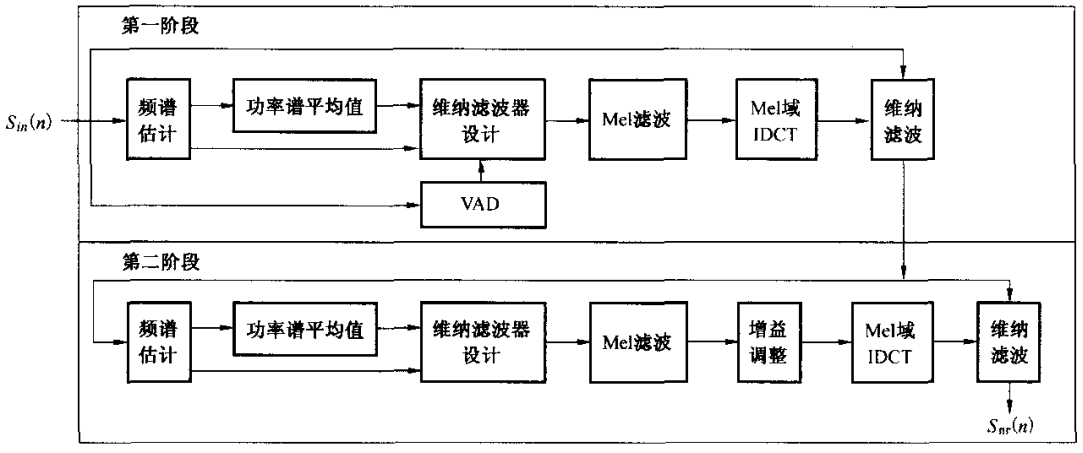


图 J.1 噪声消除流程图

输入信号首先按照帧的大小进行切分，然后在频谱估计模块中计算出每帧的线性频谱估计。在功率谱平均值模块内，对信号频谱按帧进行平滑处理。在维纳滤波器设计模块中，根据当前帧的频谱估计和噪声谱估计(噪声谱估计是通过 VAD 检测的噪声帧进行估计)，计算出频域维纳滤波器系数。线性维纳滤波器系数经过 Mel 滤波器组进行滤波处理，得到了 Mel 域维纳滤波器。然后通过 Mel 域 IDCT 计算 Mel 域维纳滤波器的脉冲响应。最后，将每阶段的输入信号通过维纳滤波器进行滤波。在图 J.1 中，第二阶段的输入信号就是第一阶段的输出信号。此外，第二阶段中增益调整模块的主要功能是对噪声消除的增益进行控制。

J.3 缓冲

噪声消除模块中输入信号以帧为单位，每帧长度为 10 ms(160 个样本)。噪声消除过程中的每阶段都需要一个大小为四帧的缓冲区(frame0～frame3)。当有新帧输入时，这两个缓冲区依次移动一帧。新输入帧被放置在第一个缓冲区的 frame3 位置上。

首先，对第一个缓冲区中的 frame1(样本 160～319)进行降噪，并把降噪后的帧放在第二个缓冲区中的 frame3 位置上。然后，对第二个缓冲区中的 frame1 作降噪处理，此帧是噪声消除模块的输出。因此，噪声消除的每阶段都有 2 帧(20 ms)的延迟。在每阶段中，频谱估计的窗长为 25 ms(样本 120～519)。图 J.2 给出两阶段噪声消除过程的缓冲区示意图。

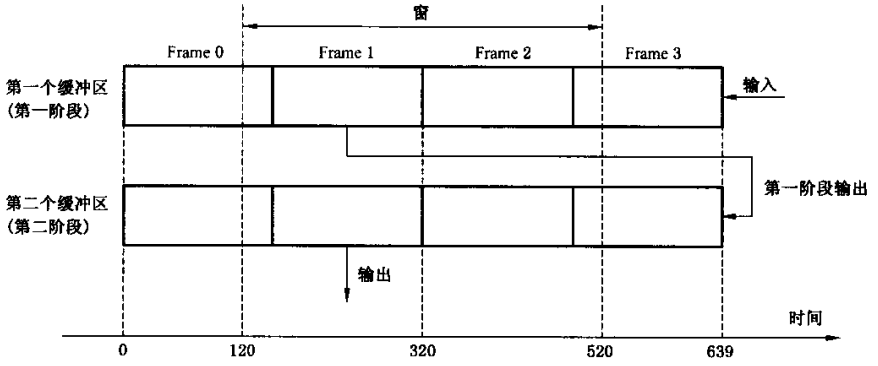


图 J.2　两阶段噪声消除过程的缓冲区示意图

J.4　频谱估计

输入信号被分成 N_{in} 个样本的重叠帧，帧长为 25 ms(N_{in}=400)并且有 10 ms(160 个样本)的帧移。每帧 $s_{in}(n)$ 都要作加窗处理，采用长为 N_{in} 的汉宁(Hanning)窗，见式(J.1)所示。

$$s_w(n)=s_{in}(n)\times w_{Hann}(n),0\leqslant n\leqslant N_{in}-1 \quad \cdots\cdots(J.1)$$

$$w_{Hann}(n)=0.5-0.5\times\cos\left(\frac{2\times\pi\times(n+0.5)}{N_{in}}\right) \quad \cdots\cdots(J.2)$$

N_{in} 和 $N_{FFT}-1$ 之间的样本补零，N_{FFT}=512 是 FFT 长度：

$$s_{FFT}(n)=\begin{cases}s_w(n),0\leqslant n\leqslant N_{in}-1\\0,\quad N_{in}\leqslant n\leqslant N_{FFT}-1\end{cases} \quad \cdots\cdots(J.3)$$

对 $s_{FFT}(n)$ 进行 FFT，以求出频谱：

$$X(bin)=FFT\{s_{FFT}(n)\} \quad \cdots\cdots(J.4)$$

式中：

bin——FFT 频率索引。

计算功率谱 $P(bin)$：

$$P(bin)=|X(bin)|^2,0\leqslant bin\leqslant N_{FFT}/2 \quad \cdots\cdots(J.5)$$

再对功率谱 $P(bin)$ 进行平滑处理：

$$P_{in}(bin)=\frac{P(2\times bin)+P(2\times bin+1)}{2},0\leqslant bin\leqslant N_{FFT}/4 \quad \cdots\cdots(J.6)$$

$$P_{in}(N_{FFT}/4)=P(N_{FFT}/2)$$

平滑处理后，功率谱的长度缩短为 $N_{SPEC}=N_{FFT}/4+1$。

J.5　功率谱平均值

对连续 T_{PSD} 帧求其功率谱 $P_{in}(bin)$ 的平均值，见图 J.3。

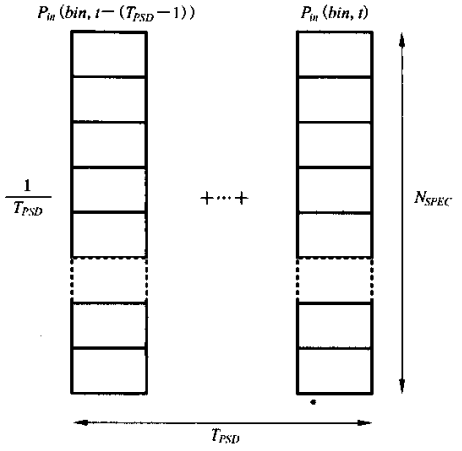


图 J.3　功率谱平均值

功率谱平均值为：

$$P_{in\_PSD}(bin,t)=\frac{1}{T_{PSD}}\sum_{i=0}^{T_{PSD}-1}P_{in}(bin,t-i),0\leqslant bin\leqslant N_{SPEC}-1 \quad \cdots\cdots\cdots\cdots(J.7)$$

式中：

$T_{PSD}=2$；

t——帧索引。

J.6　噪声估计的 VAD

根据帧索引 t，计算出每帧的遗忘因子 $lambdaLTE$：

$$\begin{aligned}&if(t<NB\_FRAME\_THRESHOLD\_LTE)\\&\quad lambdalTE=1-1/t;\\&else\\&\quad lambdaLTE=LAMBDA\_LTE;\end{aligned} \quad \cdots\cdots\cdots\cdots(J.8)$$

式中：

$NB\_FRAME\_THRESHOLD\_LTE=10$；

$LAMBDA\_LTE=0.97$。

输入信号 $s_{in}(n)$的连续 M 个(M=160)样本的对数能量 $frameEn$ 为：

$$frameEn=0.5+\frac{16}{\ln 2}\times\ln\left[\frac{\left(64+\sum_{i=0}^{M-1}s_{in}(n)^2\right)}{64}\right] \quad \cdots\cdots\cdots\cdots(J.9)$$

用 $frameEn$ 更新 $meanEn$：

$$\begin{aligned}&if((frameEn-meanEn)<SNR\_THRESHOLD\_UPD\_LTE)\ ||\ (t<MIN\_FRAME))\\&\{\\&\quad if((frameEn<meanEn)\ ||\ (t<MIN\_FRAME))\\&\quad\quad meanEn=meanEn+(1-lambdaLTE)\times(frameEn-meanEn);\\&\quad else\\&\quad\quad meanEn=meanEn+(1-lambdaLTEhigherE)\times(frameEn-meanEn);\\&\quad if(meanEn<ENERGY\_FLOOR)\\&\quad\quad meanEn=ENERGY\_FLOOR;\\&\}\end{aligned} \quad \cdots\cdots(J.10)$$

式中：

$SNR\_THRESHOLD\_UPD\_LTE=20$；

$ENERGY\_FLOOR=80$；

$MIN\_FRAME=10$；

$lambdaLTEhigherE=0.99$。

根据 $frameEn$ 和 $meanEn$ 这两个参数，确定当前帧是语音帧（$flagVAD_{Nest}=1$）还是噪声帧（$flagVAD_{Nest}=0$）：

```
if(t > 4)
{
    if((frameEn − meanEn) > SNR_THRESHOLD_VAD)
      {
        flagVAD_Nest = 1;
        nbSpeechFrame = nbSpeechFrame + 1;
        }
        else
        {
        if(nbSpeechFrame > MIN_SPEECH_FRAME_HANGOVER)
          hangOver = HANGOVER;                                    ……(J.11)
        nbSpeechFrame = 0;
        if(hangOver! =0)
          {
          hangOver = hangOver − 1;
          flagVAD_Nest = 1;
          }
          else
            flagVAD_Nest = 0;
          }
}
```

式中：

$SNR\_THRESHOLD\_VAD=15$；

$MIN\_SPEECH\_FRAME\_HANGOVER=4$

$HANGOVER=15$；

$nbSpeechFrame$，$meanEn$，$flagVAD_{Nest}$，$hangOver$ 初始化为 0。

J.7 维纳滤波器设计

根据帧索引 t，得到每帧的遗忘因子 $lambdaNSE$：

```
if(t < NB_FRAME_THRESHOLD_NSE)
  lambdaNSE = 1 − 1/t;
else
  lambdaNSE = LAMBDA_NSE;
```

式中：

$NB\_FRAME\_THRESHOLD\_NSE=100$；

$LAMBDA\_NSE=0.99$。

已知 VAD 中的 $flagVAD_{Nest}$，通过下面的公式得出第一阶段的噪声谱估计：

$$\begin{cases} P_{noise}^{1/2}(bin,t_n)=\max(lambdaNSE\times P_{noise}^{1/2}(bin,t_n-1)+ \\ \qquad (1-lambdaNSE)\times P_{in\_PSD}^{1/2}(bin,t_n),EPS) \\ P_{noise}^{1/2}(bin,t)=P_{noise}^{1/2}(bin,t_n) \end{cases} \quad \cdots\cdots(J.12)$$

式中：

$EPS=\exp(-10.0)$；

t_n——上一个非语音帧的索引；

$P_{in\_PSD}(bin,t_n)$——功率谱平均值；

$P_{in_PSD}^{1/2}(bin,-1)$初始化为 EPS。

第二阶段中，通过以下步骤得到噪声谱估计：

```
if(t < 11)
{
  lambdaNSE = 1 - 1/t;
  P_noise(bin,t) = lambdaNSE × P_noise(bin,t-1) + (1 - lambdaNSE) × P_in_PSD(bin,t);
}
else
{
  upDate = 0.9 + 0.1 × P_in_PSD(bin,t)/(P_in_PSD(bin,t) + P_noise(bin,t-1))
           × (1 + 1/(1 + 0.1 × P_in-PSD(bin,t)/P_noise(bin,t-1)));
  P_noise(bin,t) = P_noise(bin,t-1) × upDate;
}
if(P_noise^1/2(bin,t) < EPS)
  P_noise^1/2(bin,t) = EPS;
```

降噪信号谱估计：

$$P_{den}^{1/2}(bin,t)=BETA\times P_{den3}^{1/2}(bin,t-1)+(1-BETA)\times T[P_{in\_PSD}^{1/2}(bin,t)-P_{noise}^{1/2}(bin,t)] \quad \cdots\cdots(J.13)$$

式中，$P_{den}^{1/2}(bin,-1)$初始化为 0，$BETA$ 等于 0.98，阈值函数 T 为：

$$T[z(bin,t)]=\begin{cases} z(bin,t), & z(bin,t)>0 \\ 0, & z(bin,t)\leqslant 0 \end{cases} \quad \cdots\cdots(J.14)$$

因此，先验信噪比 $\eta(bin,t)$为：

$$\eta(bin,t)=\frac{P_{den}(bin,t)}{P_{noise}(bin,t)} \quad \cdots\cdots(J.15)$$

滤波器传递函数 $H(bin,t)$：

$$H(bin,t)=\frac{\sqrt{\eta(bin,t)}}{1+\sqrt{\eta(bin,t)}} \quad \cdots\cdots(J.16)$$

已知 $H(bin,t)$，便可对降噪信号谱估计进行更新：

$$P_{den2}^{1/2}(bin,t)=H(bin,t)P_{in\_PSD}^{1/2}(bin,t) \quad \cdots\cdots(J.17)$$

更新后的先验信噪比 $\eta_2(bin,t)$为：

$$\eta_2(bin,t)=\max\left(\frac{P_{den2}(bin,t)}{P_{noise}(bin,t)},\eta_{TH}^2\right) \quad \cdots\cdots(J.18)$$

式中：

$\eta_{TH}=0.0794\ 328\ 23$(对应的 SNR 为−22 dB)。

相应地，滤波器传递函数 $H_2(bin,t)$更新为：

$$H_2(bin,t)=\frac{\sqrt{\eta_2(bin,t)}}{1+\sqrt{\eta_2(bin,t)}},0\leqslant bin\leqslant N_{SPEC}-1 \quad \cdots\cdots(J.19)$$

根据 $H_2(bin,t)$得出降噪信号谱 $P_{den3}^{1/2}(bin,t)$：

$$P_{den3}^{1/2}(bin,t)=H_2(bin,t)P_{in}^{1/2}(bin,t) \quad \cdots\cdots(J.20)$$

J.8 Mel 滤波

首先对线性频率维纳滤波器系数 $H_2(bin)$，$0\leqslant bin\leqslant N_{SPEC}-1$ 作平滑处理，之后转化为 Mel 频率刻度。通过对 $H_2(bin)$作半重叠三角形频率窗处理后，估计出 Mel 域维纳滤波器系数 $H_{2\_mel}(k)$。为了得出 Mel 子带的中心频率 $bin_{centr}(k)$，线性频率表 f_{lin} 通过下面的公式转化为 Mel 刻度：

$$MEL\{f_{lin}\}=2\ 595\times\log_{10}(1+f_{lin}/700) \quad \cdots\cdots(J.21)$$

第 k 子带的中心频率 $f_{mel}(k)$：

$$f_{centr}(k)=700\times(10^{f_{mel}(k)/2\ 595}-1),1\leqslant k\leqslant K_{FB} \quad \cdots\cdots(J.22)$$

式中：

$K_{FB}=32$，并且：

$$f_{mel}(k)=k\times\frac{MEL\{f_{lin\_sump}/2\}}{K_{FB}+1} \quad \cdots\cdots(J.23)$$

式中：

$f_{lin\_spmp}=16$ kHz——采样频率。

两个边缘子带的中心频率 $f_{centr}(0)$和 $f_{centr}(K_{FB}+1)=f_{lin\_sump}/2$ 加在 $K_{FB}=32$ 子带上。因此，一共要计算 $K_{FB}+2=34$ 个 Mel 域维纳滤波器系数。中心频率所对应的 FFT 频率为：

$$bin_{centr}(k)=round\left[\frac{f_{centr}(k)}{f_{lin\_sump}}\times 2\times(N_{SPEC}-1)\right] \quad \cdots\cdots(J.24)$$

下面计算三角形频率窗 $W(k,i)$。

$1\leqslant k\leqslant K_{FB}$的窗函数计算如下：

$$W(k,i)=\frac{i-bin_{centr}(k-1)}{bin_{centr}(k)-bin_{centr}(k-1)},bin_{centr}(k-1)+1\leqslant i\leqslant bin_{centr}(k) \quad \cdots\cdots(J.25)$$

$$W(k,i)=1-\frac{i-bin_{centr}(k)}{bin_{centr}(k+1)-bin_{centr}(k)},bin_{centr}(k)+1\leqslant i\leqslant bin_{centr}(k+1) \quad \cdots(J.26)$$

i 取其他值时 $W(k,i)=0$。

k=0 的窗函数计算如下：

$$W(0,i)=1-\frac{i}{bin_{centr}(1)-bin_{centr}(0)},0\leqslant i\leqslant bin_{centr}(1)-bin_{centr}(0)-1 \quad \cdots\cdots(J.27)$$

i 取其他值时 $W(0,i)=0$。

$k=K_{FB}+1$ 的窗函数计算如下：

$$W(K_{FB}+1,i)=\frac{i-bin_{centr}(K_{FB})}{bin_{centr}(K_{FB}+1)-bin_{centr}(K_{FB})},bin_{centr}(K_{FB})+1\leqslant i\leqslant bin_{centr}(K_{FB}+1) \quad \cdots(J.28)$$

i 取其他值时 $W(K_{FB}+1,i)=0$。

Mel 域维纳滤波器系数 $H_{2_mel}(k)$在 $0\leqslant k\leqslant K_{FB}+1$ 时，计算公式如下：

$$H_{2\_mel}(k)=\frac{1}{\sum_{i=0}^{N_{SPEC}-1}W(k,i)}\sum_{i=0}^{N_{SPEC}-1}W(k,i)\times H_2(i) \quad \cdots\cdots(J.29)$$

J.9 增益调整

第一阶段降噪处理中，根据降噪信号的功率谱 $P_{den3}(bin,t)$ 计算降噪信号的能量 $E_{den}(t)$ 如下：

$$E_{den}(t)=\sum_{bin=0}^{N_{SPEC}-1}P_{den3}^{1/2}(bin,t) \qquad \text{(J.30)}$$

第二阶段降噪处理中，根据噪声功率谱 $P_{noise}(bin,t)$ 计算噪声能量：

$$E_{noise}(t)=\sum_{bin=0}^{N_{SPEC}-1}P_{noise}^{1/2}(bin,t) \qquad \text{(J.31)}$$

通过连续三帧的降噪信号的能量和噪声能量可估计出平滑后的信噪比：

$$Ratio=\frac{E_{den}(t-2)\times E_{den}(t-1)\times E_{den}(t)}{E_{noise}(t)\times E_{noise}(t)\times E_{noise}(t)};$$

$$if(Ratio>0.000\ 01) \qquad \text{(J.32)}$$

$$SNR_{over}(t)=20/3\times\log_{10}(Ratio);$$

$$else$$

$$SNR_{over}(t)=-100/3;$$

为了估计第二阶段的降噪增益，低信噪比跟踪值 SNR_{low_track} 计算如下：

$$if(((SNR_{over}(t)-SNR_{lower\_track}(t-1))<10\ ||\ (t<10))$$

$$SNR_{low\_track(t)}=\lambda_{SNR}(t)\times SNR_{lrw\_track}(t-1)+(1-\lambda_{SNR}(t))\times SNR_{over}(t); \qquad \text{(J.33)}$$

$$else$$

$$SNR_{low\_track}(t)=SNR_{low\_track}(t-1);$$

式中：

SNR_{low_track}——初始化为 0；

$\lambda_{SNR}(t)$——遗忘因子，计算如下：

$$if(t<10)$$

$$\lambda_{SNR}(t)=1-1/t;$$

$$else \qquad \text{(J.34)}$$

$$if(SNR_{over}(t)<SNR_{low\_track}(t))$$

$$\lambda_{SNR}(t)=0.95;$$

$$else$$

$$\lambda_{SNR}(t)=0.99;$$

增益调整主要目的在于：当处理纯噪声帧时，需采用相对较大的降噪增益；而处理包含语音的噪声帧时，需要采用相对较小的降噪增益。对当前信噪比估计 $SNR_{over}(t)$ 和低信噪比跟踪值 $SNR_{low\_track}(t)$ 进行比较，同时更新维纳滤波器增益调整系数 $\alpha_{GF}(t)$，计算如下：

```
if(E_den(t)>100)
{
  if(SNR_over(t)<(SNR_low_track(t)+3.5))
  {
    α_GF(t)=α_GF(t-1)+0.15;
    if(α_GF(t)>0.8)
      α_GF(t)=0.8;
  }
```

```
    else
    {
      α_GF(t)=α_GF(t−1)−0.3;
      if(α_GF(t)<0.1)
        α_GF(t)=0.1;
    }
  }
```

……（J.35）

式中：

$\alpha_{GF}(0)=0.8$。

第二阶段的维纳滤波器系数乘以增益调整系数 $\alpha_{GF}(t)$：

$$H_{2\_mel\_GF}(k,t)=(1-\alpha_{GF}(t))+\alpha_{GF}(t)\times H_{2\_mel}(k,t),\quad 0\leqslant k\leqslant K_{FB}+1 \qquad \cdots\cdots(J.36)$$

式中，系数 $\alpha_{GF}(t)$ 的取值在 0.1～0.8 之间。

J.10 Mel 域 IDCT

维纳滤波器的时域脉冲响应 $h_{WF}(n)$ 通过 Mel 域维纳滤波器系数 $H_{2\_mel}(k)$［第二阶段为 $H_{2\_mel\_GF}(k)$，见式(J.36)］进行 Mel 域 IDCT 得到：

$$h_{WF}(n)=\sum_{k=0}^{K_{FB}+1}H_{2\_mel}(k)\times IDCT_{mel}(k,n),\quad 0\leqslant n\leqslant K_{FB}+1 \qquad \cdots\cdots(J.37)$$

式中：

$IDCT_{mel}(k,n)$——Mel 域 IDCT 函数。

具体推导如下：

首先，$1\leqslant k\leqslant K_{FB}$ 频带的各自中心频率为：

$$f_{centr}(k)=\frac{1}{\sum_{i=0}^{N_{SPEC}-1}W(k,i)}\sum_{i=0}^{N_{SPEC}-1}W(k,i)\times i\times\frac{f_{samp}}{2\times(N_{SPEC}-1)} \qquad \cdots\cdots(J.38)$$

式中：

$f_{samp}=16$ kHz——采样频率；

$f_{centr}(0)=0$ kHz；

$f_{centr}(K_{FB}+1)=f_{samp}/2$。

则，$IDCT_{mel}(k,n)$ 为：

$$IDCT_{mel}(k,n)=\cos\left(\frac{2\times\pi\times n\times f_{centr}(k)}{f_{samp}}\right)\times df(k),0\leqslant k\leqslant K_{FB}+1,\quad 0\leqslant n\leqslant K_{FB}+1 \qquad \cdots\cdots(J.39)$$

式中：

$f_{centr}(k)$——Mel 子带 k 所对应的中心频率。

$df(k)$ 为：

$$\begin{aligned}df(k)&=\frac{f_{centr}(k+1)-f_{centr}(k-1)}{f_{samp}},1\leqslant k\leqslant K_{FB}\\ df(0)&=\frac{f_{centr}(1)-f_{centr}(0)}{f_{samp}}\\ df(K_{FB}+1)&=\frac{f_{centr}(K_{FB}+1)-f_{centr}(K_{FB})}{f_{samp}}\end{aligned} \qquad \cdots\cdots(J.40)$$

维纳滤波器的脉冲响应扩展到 $0\leqslant k\leqslant 2\times(K_{FB}+1)$：

$$h_{WF\_mirr}(n)=\begin{cases}h_{WF}(n), & 0\leqslant n\leqslant K_{FB}+1\\ h_{WF}(2\times(K_{FB}+1)+1-n), & k_{FB}+2\leqslant n\leqslant 2\times(K_{FB}+1)\end{cases} \quad \cdots\cdots(\text{J.41})$$

J.11 维纳滤波

根据 $h_{WF_mirr}(n)$得出因果的脉冲响应 $h_{WF\_caus}(n,t)$：

$$\begin{cases}h_{WF\_caus}(n,t)=h_{WF\_mirr}(n+K_{FB}+1,t), & n=0,\cdots,K_{FB}\\ h_{WF\_caus}(n,t)=h_{WF\_mirr}(n-K_{FB}-1,t), & n=K_{FB}+1,\cdots,2\times(K_{FB}+1)\end{cases} \quad \cdots\cdots(\text{J.42})$$

截断后脉冲响应 $F_{WF_trunc(n,t)}$为：

$$h_{WF\_trunc}(n,t)=h_{WF\_caus}(n+K_{FB}+1-(FL-1)/2,t),\quad n=0,\cdots,FL-1 \quad \cdots\cdots(\text{J.43})$$

滤波器长度 FL 等于 17。截断后脉冲响应加汉宁窗处理：

$$h_{WF\_w}(n,t)=\left\{0.5-0.5\times\cos\left(\frac{2\times\pi\times(n+0.5)}{FL}\right)\right\}\times h_{WF\_trunc}(n,t),\quad 0\leqslant n\leqslant FL-1 \quad \cdots\cdots(\text{J.44})$$

这样，输入信号 s_{in}经过脉冲响应 $h_{WF_w}(n,t)$的维纳滤波器后就得到降噪信号 s_{nr}：

$$s_{nr}(n)=\sum_{i=-(FL-1)/2}^{(FL-1)/2}h_{WF\_w}(i+(FL-1)/2)\times s_{in}(n-i),\quad 0\leqslant n\leqslant M-1 \quad \cdots\cdots(\text{J.45})$$

式中：

$FL=17$——滤波器长度；

$M=160$——帧移样本数。

参 考 文 献

[1] ISO/IEC IS 14496-1. Information Technology—Generic coding of audio-visual objects. Part1:Systems. Nov. 1998

[2] ISO/IEC IS 14496-2. Information Technology—Generic coding of audio-visual objects. Part2:Visual. Nov. 1998

[3] ISO/IEC JCT1/SC29 WG11 N3342. Overview of MPEG-7 standard. Maui,1999

[4] ISO/IEC JCT1/SC29 WG11 and ITU-T SG26 Q. 6 (JVT-K051, Version3 of ISO/IEC 14496-10E). 12th Meeting Redmond, U. S. A, Jul. 2004

[5] 3GPP TS 26.290, Version 7.0.0, "Extended Adaptive Multi-Rate - Wideband (AMR－WB+) codec; Transcoding functions", Mar. 2007

[6] 3GPP TS 26.190, Version 7.0.0, " AMR Wideband speech codec; Transcoding functions ", Jun. 2007

[7] ETSI ES 202 050, Version 1.1.5, "Distributed Speech Recognition; Advanced Front-end Feature Extraction Algorithm; Compression Algorithm", Jan. 2007

[8] ETSI ES 202 212, Version 1.1.2, "Distributed Speech Recognition; Extended Advanced Front-end Feature Extraction Algorithm; Compression Algorithm, Back-end Speech Reconstruction Algorithm", Nov. 2005

[9] 姚天任,孙洪. 现代数字信号处理. 武汉：华中理工大学出版社,1999

[10] A. V. 奥本海姆. R. W. 谢弗. 离散时间信号处理. 2 版. 刘树棠,黄建国,译. 西安：西安交通大学出版社,2001

[11] 鲍长春. 低比特率数字语音通信编码基础. 北京：北京工业大学出版社,2001

[12] 张贤达. 现代信号处理. 2 版. 北京：清华大学出版社,2002

[13] 赵力. 语音信号处理. 北京：机械工业出版社,2003

[14] 夸特尔瑞. 离散时间语音信号处理——原理与应用. 赵胜辉等,译. 北京：电子工业出版社,2004

[15] 王炳锡,王洪. 变速率语音编码. 西安:西安电子科技大学出版社,2004

[16] 胡航. 语音信号处理. 3 版. 哈尔滨：哈尔滨工业大学出版社,2005

[17] 程佩清. 数字信号处理教程. 3 版. 北京：清华大学出版社,2007

[18] 吴家安. 现代语音编码技术. 北京：科学出版社,2007

[19] ITU-T Draft Recommendation and Final Draft International Standard of Joint Video Specification (ITU-T Rec. H. 264 | ISO/IEC 14496-10 AVC). 7th Meeting: Pattaya, Thailand, 7-14 Mar. 2003

[20] Abdul H. Sadka. Compressed Video Communications. Hohn Wiley & Sons, Ltd. England. 2002

[21] Kenneth. R. Castleman. Digital Image Processing. 北京. 清华大学出版社. 1998

[22] Iain E. G. Richardson. H. 264 and MPEG-4 Video Compression. Hohn Wiley & Sons, Ltd. England. 2002

[23] J. H. Conway and N. J. A. Sloane, "A fast encoding method for lattice codes and quantizers," IEEE Trans. Inform. Theory, vol. IT-29, no. 6, pp. 820-824, Nov. 1983

[24] F. Jabloun and A. E. Cetin, "The Teager Energy Based Feature Parameters for Robust Speech Recognition in Noise," Proc. IEEE Int. Conf. Acoustics, Speech, and Signal Processing, Mar. 1999

[25] L. B. Almeida and F. M. Silva, "Variable-Frequency Synthesis: An Improved Harmonic Coding Scheme," Proc. IEEE Int. Conf. Acoustics, Speech, and Signal Processing, San Diego, CA, May 1984

[26] M. Xie and J. P. Adoul, "Embedded algebraic vector quantization (EAVQ) with application to wideband audio coding," IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP), Atlanta, GA, U. S. A, vol. 1, pp. 240-243, 1996

[27] T. Ganchev, N. Fakotakis, and G. Kokkinakis, "Comparative evaluation of various MFCC implementations on the speaker verification task," 10th International Conference on Speech and Computer (SPECOM), Vol. 1, pp. 191-194, 2005

[28] GB/T 20090.10 信息技术 先进音视频编码 第10部分:移动语音与音频编码