ICS 13.310
A 91

中华人民共和国国家标准

GB/T 25724—2017
代替 GB/T 25724—2010

公共安全视频监控数字视音频编解码技术要求

Technical specifications for surveillance video and audio coding

2017-03-09 发布　　2017-06-01 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

本标准按照 GB/T 1.1—2009 给出的规则起草。

本标准代替 GB/T 25724—2010《安全防范监控数字视音频编解码技术要求》，与 GB/T 25724—2010 相比主要技术变化如下：

——增加了术语（见 3.1.93～3.1.95）；

——修改了编码单位结构（见 5.1.3，2010 年版的 5.1.3）；

——修改了码流的语法和语义（见 5.2.3、5.2.4，2010 年版的 5.2.3、5.2.4）；

——修改了安全参数集的语法和语义（见 5.2.3.2.5、5.2.4.4.4，2010 年版的 5.2.3.2.3、5.2.4.4.3）；

——修改了参考图像的选择方法（见 5.3.3.4，2010 年版的 5.3.3.4）；

——修改了帧内预测过程的内容（见 5.3.4，2010 年版的 5.3.4）；

——修改了帧间预测过程的内容（见 5.3.5，2010 年版的 5.3.5）；

——修改了变换量化与重建的内容（见 5.3.6，2010 年版的 5.3.6）；

——修改了去块效应滤波过程的内容（见 5.3.7，2010 年版的 5.3.7）；

——增加了样点自适应补偿（SAO）（见 5.3.8）；

——增加了样点滤波补偿（见 5.3.9）；

——修改了解析过程的内容（见 5.4，2010 年版的 5.4）；

——修改了附录 F，删除了变长码表，增加了智能分析数据描述（见附录 F，2010 年版的附录 F）。

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、杭州恒生数字设备科技有限公司、公安部安全与警用电子产品质量检测中心、山西中天信科技股份有限公司、千目聚云数码科技（上海）有限公司、北京欣博电子科技有限公司、杭州海康威视数字技术股份有限公司、湖南国科微电子股份有限公司、浙江大华技术股份有限公司、苏州科达科技股份有限公司、浙江宇视科技有限公司、天津天地伟业数码科技有限公司、北京联视神盾安防技术有限公司、北京智芯原动科技有限公司、上海熙菱信息技术有限公司。

本标准主要起草人：陈朝武、邓中翰、郅晨、邱嵩、余子龙、张韵东、董骞、咎劲文、欧阳甸、卢京辉、闫雪、林冬、施巨岭、查敏中、汪人瑞、梁敏学、黄麒麟、廖双龙、周文博、马莉、夏昌盛、曾娟鹃、李伟丽、卢玉华、胡建华、王磊、孙大瑞、俞海、段争志、刘文尧、吕卓逸、姜黎、卢虹、倪昕、马伟、王秦镜、章勇、邢培银、王大治、吴参毅。

本标准所代替标准的历次版本发布情况为：

——GB/T 25724—2010。

引　言

在GB/T 25724—2010《安全防范监控数字视音频编解码技术要求》(以下简称SVAC标准)发布之前,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,所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。

SVAC标准(2010年版)于2010年12月23日发布,2011年5月1日实施。该标准是具有我国自主知识产权的、专门应用于安全防范视频监控技术领域的数字视音频编解码技术标准。该标准发布实施以后,国家标准委、公安部、工信部等部门高度重视标准的推广应用,支持成立了北京安防视音频编解码技术产业联盟(以下简称SVAC联盟),业内科研院所和广大企业围绕着SVAC产业链积极开展技术研发和产品应用。

在标准实施过程中发现,SVAC标准在数据安全保护、提升压缩性能和编码效率、对智能化和大数据的支持等方面还有待补充和完善之处。为此,全国安全防范报警系统标准化技术委员会(代号SAC/TC 100)组织公安部第一研究所和北京中星微电子有限公司等单位对SVAC标准进行了修订,使标准更具有先进性和可操作性。

近年来,视频监控系统建设应用已经从安全防范行业扩展到公共安全各行业、领域,已经成为新形势下维护国家安全和社会稳定的重要手段,在打击犯罪、治安防范、社会管理、服务民生等方面发挥着积极作用。本次修订充分考虑了公共安全视频监控联网与应用建设的需要,标准内容普遍适用于公共安全各行业、领域,因此标准名称变更为《公共安全视频监控数字视音频编解码技术要求》。

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比特～12比特数据;

b) 支持多样化的帧内及帧间预测、变换量化、二进制算术编码等技术,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、智能分析信息等监控专用信息。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,规定了常用智能分析信息的携带方式,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,加强了对国密算法的支持,完善了安全参数集,增添了摘要、签名算法的标识等内容,并对密钥及数字证书相关信息的携带做了规范定义,支持视频数据加密、认证功能。

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，可能涉及到与 5.2.3.1、5.2.3.2、5.2.4.2、5.2.4.4、5.2.4.7、6.1.2、6.1.4、6.2.6.1.3、6.2.6.1.4.10、6.5.2.2 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保证，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)
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：李伟丽
通讯地址：北京海淀区首体南路 1 号
邮政编码：100048
电子邮件：lwl@zhongdun.com.cn
电话：010-68773553-6387
传真：010-68773553-6215

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。

公共安全视频监控数字视音频编解码技术要求

1 范围

本标准规定了公共安全视频监控应用的数字视音频压缩编码的解码过程。

本标准适用于公共安全领域的视音频实时压缩、传输、播放和存储等业务，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

下列文件对于本文件的应用是必不可少的。凡是注日期的引用文件，仅注日期的版本适用于本文件。凡是不注日期的引用文件，其最新版本(包括所有的修改单)适用于本文件。

rfc 3548 The Base16, Base32, and Base64 Data Encodings

3 术语、定义和缩略语

3.1 术语和定义

下列术语和定义适用于本文件。

3.1.1

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，必要时其中还包括散布的防伪字节。

3.1.2

NAL 单元流 NAL unit stream

由 NAL 单元组成的序列。

3.1.3

保留 reserved

某些语法元素的特定取值。

注：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。

3.1.4

闭环基音搜索 closed-loop pitch search

用于从加权输入信号和长时预测滤波器状态估计基音延迟。也称为自适应码书搜索。

3.1.5

比特流 bitstream

由编码视音频及其相关数据构成的比特序列。比特流既可用来表示 NAL 单元流，也可表示字节流。

3.1.6

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。

3.1.7

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。

3.1.8

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。

3.1.9

编码器　encoder

实现编码过程的实体，包括软件及硬件。

3.1.10

编码片　tile

一个矩形区域内部按照光栅扫描顺序排列的整数个树形编码单元。

3.1.11

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。

3.1.12

编码图像　coded picture

一幅图像的编码表示。

注：符合本标准的一个编码图像为一个编码帧。

3.1.13

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。

3.1.14

编码帧　coded frame

一个帧的编码表示。

3.1.15

残差　residual

样点或数据元素预测值与解码值之间的差值。

3.1.16

参考索引　reference index

参考图像的索引。

3.1.17

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。

3.1.18

参考帧　reference frame

一个标记为参考图像的帧，用于解码过程中的帧间预测。

3.1.19

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。

3.1.20

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、编码片层和编码单元层。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）。

3.1.21

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。

3.1.22

档次　profile

本标准中的一个特定语法子集。

3.1.23

电导频谱对　immittance spectral pair

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，指示该函数在单位圆上的根。用于线性预测系数的变换。

3.1.24

二进制位　bin

二进制位串中的 1 比特。

3.1.25

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。

3.1.26

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。

3.1.27

反变换　inverse transform

将变换系数矩阵转换为空域样点矩阵的过程。

3.1.28

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。

3.1.29

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。

3.1.30

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。

3.1.31

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。

3.1.32

功率谱　power spectrum

信号通过傅立叶变换后得到幅度谱的平方。

3.1.33

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、第三行等等。对于图像每行（由上到下）都是从左到右扫描的。

3.1.34

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。

3.1.35

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。

3.1.36

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。

3.1.37

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。

3.1.38

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

一幅帧内解码图像。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。

3.1.39

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。

3.1.40

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。

3.1.41

解码器　decoder

实现解码过程的实体，包括软件及硬件。

3.1.42

解码顺序　decoding order

解码过程中处理语法元素的顺序。

3.1.43

解码图像　decoded picture

通过解码一幅编码图像得到的图像。符合本标准的一幅解码图像应是一个解码帧。

3.1.44

解码图像缓存区　decoded picture buffer

保存解码图像的存储空间，用于附录 A 中规定的预测参考、输出重排序或输出延时等。

3.1.45

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。

3.1.46

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。

3.1.47

块　block

视频信号空间中的一个 $M\times N$（M 列 N 行）的样点矩阵，或者一个 $M\times N$ 的变换系数矩阵。

音频信号空间的一个一维矢量。

3.1.48

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。

3.1.49

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。

3.1.50

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。

3.1.51

美尔　mel

一种非线性的频率刻度，根据主观音高进行划分。

3.1.52

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。

3.1.53

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。

3.1.54

逆滤波器　inverse filter

去除信号短时相关性的滤波器。

3.1.55

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。

3.1.56

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。

3.1.57

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。

3.1.58

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。

3.1.59

二进制算术编码　binary arithmetic coding

一种熵编码方法，根据概率模型对二进制位进行编码，产生比特流。

3.1.60

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。

3.1.61

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。

3.1.62

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。

3.1.63

树形编码单元　coding tree unit

一个 $N\times N$ 的亮度样点块和相应的两个色度样点块。

3.1.64

图像　picture

源、编码或重构的图像数据的通称。符合本标准的一幅图像指一帧。

3.1.65

图像参数集　picture parameter set

一个语法结构，包含应用于一个或多个编码图像的语法元素。

3.1.66

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。

3.1.67

线性预测系数　LP coefficients

短时预测滤波器系数，也称为LPC系数。

3.1.68

序列参数集　sequence parameter set

一个语法结构，包含应用于一个或多个完整编码视频序列的语法元素。

3.1.69

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。

3.1.70

音频子帧　audio subframe

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

3.1.71

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。

3.1.72

预测值　predictor

以前解码的样点值或数据元素的线性组合。

3.1.73

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。

3.1.74

语法元素　syntax element

比特流中表示数据的元素。

3.1.75

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。

3.1.76

源　source

编码前视音频素材或者素材的某些属性。

3.1.77

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。

3.1.78

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

值为 1 的一比特，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。

3.1.79

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。

3.1.80

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图像层信息。

3.1.81

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。

3.1.82

帧间编码　inter coding

使用帧间预测对块或图像进行编码。

3.1.83

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。

3.1.84

帧间解码图像　inter decoded picture

使用帧内预测进行解码，或者根据先前解码的参考图像利用单前向预测、双前向预测或者双向预测进行解码的图像，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。

3.1.85

帧内编码　intra coding

使用帧内预测对块或图像进行编码。

3.1.86

帧内解码图像　intra decoded picture

I 图像

只使用帧内预测解码的图像。

3.1.87

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。

3.1.88

字节　byte

连续的 8 比特，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。

3.1.89

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。

3.1.90

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。

3.1.91

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。

3.1.92

直流偏置　DC-offset

音频信号的直流分量。

3.1.93

安全前端设备　secure front end device

具备安全密码部件（如安全芯片，安全 TF 卡等）的视频监控前端设备。该设备能够利用证书存储和管理、密钥存储和管理、数据签名验签、数据加密等技术实现设备身份认证、视频签名、视频加密等功能。

3.1.94

视频加密密钥　video encryption key

安全前端设备随机产生的对称密钥，按照一定的规律变化，用于直接加密视频内容，实现视频传输的机密性保护。

3.1.95

视频密钥加密密钥　video key encryption key

安全前端设备维护的对称密钥，按照一定的规律变化，用于加密视频加密密钥，实现其传输的机密性保护。

3.2　缩略语

下列缩略语适用于本文件。

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

ALF：样点滤波补偿(Adaptive Loopfilter)

BWE：带宽扩展(Bandwidth Extension)

CBR：恒定比特率(Constant Bit Rate)

CPB：编码图像缓存区(Coded Picture Buffer)

CRC：循环冗余校验码(Cyclic Redundancy Code)

CTU：树形编码单元(Coding Tree Unit)

DCT：离散余弦变换(Discrete Cosine Transform)

DFT：离散傅立叶变换(Discrete Fourier Transform)

DPB：解码图像缓存区(Decoded Picture Buffer)

FFT：快速傅立叶变换(Fast Fourier Transform)

FIR：有限冲击响应(Finite Impulse Response)

GOP：图像编码组(Group Of Pictures)

HRD：假设参考解码器(Hypothetical Reference Decoder)

IDCT：离散余弦逆变换(Inverse Discrete Cosine Transform)

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

IDR：即时解码刷新(Instantaneous Decoding Refresh)

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

ISF：电导谱频率(Immittance Spectral Frequency)

ISP：电导谱对(Immittance Spectral Pair)

LD：低延时(Low Delay)

LP：线性预测(Linear Prediction)

LPC：线性预测编码(Linear Predictive Coding)

LSB：最低有效位(Least Significant Bit)

LTP：长时预测(Long Term Predictor)

MA：滑动平均(Moving Average)

MB：宏块(Macroblock)

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

MSB：最高有效位(Most Significant Bit)

MSVQ：多级矢量量化(Multi-Stage Vector Quantization)

NAL：网络抽象层(Network Abstraction Layer)

OFB：输出反馈模式(Output Feedback)

PCM:脉冲编码调制(Pulse Code Modulation)
RA:随机访问(Random Access)
RBSP:原始字节序列负载(Raw Byte Sequence Payload)
ROI:感兴趣区域(Region Of Interest)
RPS:参考图像集(Reference Picture Set)
SAO:样点偏移补偿(Sample Adaptive Offset)
SEI:补充增强信息(Supplement Enhancement Information)
SNR:信噪比(Signal Noise Ratio)
SODB:数据比特串(String Of Data Bits)
SVC:可伸缩性视频编码(Scalable Video Coding)
TAC:变换域音频编码(Transform Audio Coding)
TVC:变换域矢量编码(Transform Vector Coding)
VAD:语音活动检测(Voice Activity Detection)
VBR:可变比特率(Variable Bit Rate)
VCL:视频编码层(Video Coding Layer)
VQ:矢量量化(Vector Quantization)
VUI:视频可用性信息(Video Usability Information)
WPP:波前并行处理(Wavefront Parallel Processing)

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	⊗	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:

a) 乘法和除法运算先于加法和减法运算;
b) 乘法和除法运算从左到右进行;
c) 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	~	按位“取反”运算。按位取反运算是单目运算,用来求一个位串信息按位的反,即为0的位,结果是1;为1的位,结果是0
4	^	按位“异或”运算。异或运算是求两个运算分量相应位值是否相异,相异的为1,相同的为0
5	x>>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
6	x<<y	将 x 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

$\mathrm{Clip1}Y(x)=\mathrm{Clip3}(0,(1<<\mathrm{BitDepth}Y)-1,x)$

$\mathrm{Clip1}C(x)=\mathrm{Clip3}(0,(1<<\mathrm{BitDepth}C)-1,x)$

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

$\cos(x)$ 表示 x 的余弦函数

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

$\exp(x)$ 表示 e 的 x 次幂

$\mathrm{Floor}(x)$ 取不大于 x 的最大整数

$\ln(x)$ 取以 e 为底的 x 的对数

$\log_{10}(x)$ 取以 10 为底的 x 的对数

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

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

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

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

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

$\sin(x)$ 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：

a) 如果下列所有条件为真，语句 0

——条件 0a

——条件 0b

b) 否则，如果下列任何一个条件为真，语句 1

——条件 1a

——条件 1b

c) ……

d) 否则，语句 n

正文中，一个以下列伪码描述的逻辑运算语句：

if(条件 0)

语句 0

if(条件 1)

语句 1

可描述如下：

——当条件 0 时，语句 0

——当条件 1 时，语句 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。

每个源或已解码帧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。

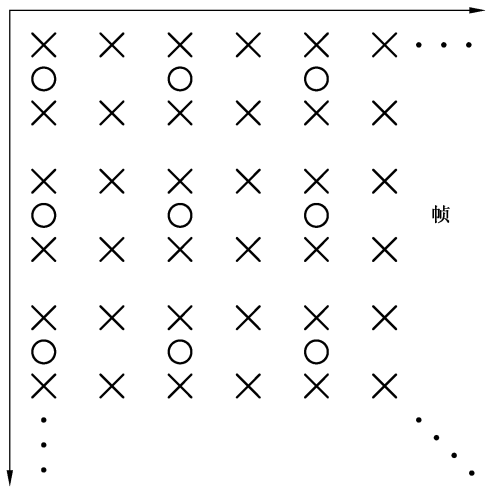
为了便于标记和命名，本标准不考虑实际使用的颜色表示方法，与这些阵列相关的变量和词语均指亮度和色度，亮度阵列用 Y 表示，两个色度阵列分别用 Cb 和 Cr 表示。

本标准支持的色彩格式有 4∶2∶0 和 4∶2∶2。

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。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。

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。

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应相同。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。



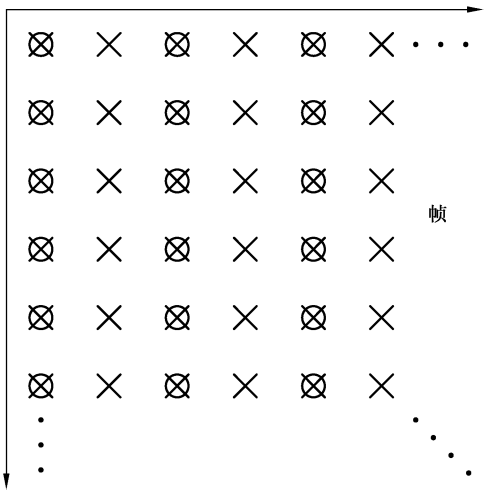
说明：

×——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

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，帧中的样点位置如图 2 所示。



说明：

×——亮度样点的位置；

○——色度样点的位置。

图 2　帧图像中 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。

5.1.3　图像的空间分割

5.1.3.1　编码片的划分

本条规定一幅图像如何分割为编码片(Tile)和树形编码单元(CTU)。tile_enable 等于 0 时,整帧图像只有一个编码片;tile_enable 等于 1 时,图像有可能被划分为多个编码片。编码片由一系列的树形编码单元组成。树形编码单元为编码的基本单元,每个树形编码单元包含一个亮度阵列及两个色度阵列。图像左上角的 CTU 的索引等于 0,CTU 在图像中的索引按照光栅扫描顺序递增。

将一幅图像从水平和垂直方向上分割为若干个矩形区域,每个矩形区域称为一个 Tile。每个 Tile 包含若干个 CTU,其可以并行独立编解码。

图像宽度大于等于 8 个最大树形编码单元尺寸时,可以在水平方向上划分为多个 Tile 列。图像宽度超过 64 个最大树形编码单元尺寸时,必须划分为多个 Tile 列。每个 Tile 在水平方向至少包含 4 个 CTU,最多包括 64 个 CTU。图像在垂直方向上可以划分为多个 Tile 行,Tile 行的个数可以等于 1,2 或者 4。

图 3 为图像的 Tile 划分示意图,图中图像被划分为 2 个 Tile 行和 4 个 Tile 列,每个 Tile 中包含 6×3 个树形编码单元。每个 Tile 的宽度和高度按照 Tile 的列数和行数决定。图像的宽度和高度分别按照 Tile 列数和行数均分,得到 Tile 的宽度和高度。当无法整除时,最右一列和最下一行的 Tile 与其他 Tile 宽度和高度不同。

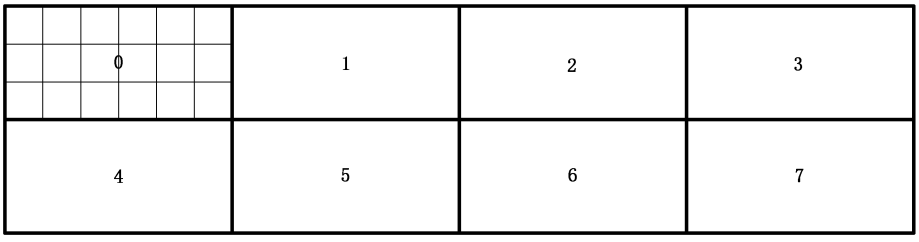


图 3　编码片划分与顺序示意图

在码流中，图像中的所有 Tile 应按照光栅扫描顺序进行传输，每个 Tile 中的 CTU 也应按照在 Tile 中的光栅扫描顺序进行传输。

5.1.3.2 树形编码单元(CTU)、预测单元(PU)、变换单元(TU)

本条规定树形编码单元如何进一步划分为预测单元、变换单元。树形编码单元之间不应重叠，树形编码单元左上角的样点不应超出图像边界，树形编码单元右下角的样点可超出图像边界。

当序列参数集中的 extended_sb_size_flag 的取值等于 1 时，树形编码单元 $2N\times2N$ 的尺寸为 128×128，当 extended_sb_size_flag 的取值等于 0 时，树形编码单元的尺寸为 64×64。树形编码单元可划分为一个或多个预测单元，划分方式由编码树决定。每级的 $2N\times2N$ 尺寸的预测单元，可进一步划分为 $2N\times2N$、$N\times N$、$2N\times N$、$N\times2N$ 等四种模式，如果该级划分为 $N\times N$，则进入下一级选择进一步的划分方式。对于 64×64 的树形编码单元，其编码树如图 4 所示，预测单元共有 13 种尺寸：64×64、32×64、64×32、32×32、16×32、32×16、16×16、8×16、16×8、8×8、4×8、8×4、4×4。对于 128×128 的树形编码单元，其预测单元共有 16 种尺寸：128×128、64×128、128×64、64×64、32×64、64×32、32×32、16×32、32×16、16×16、8×16、16×8、8×8、4×8、8×4、4×4。

编码树的扫描顺序如图 5 所示，矩形里的数字表示该块在编码时的处理顺序。

预测单元是帧内预测与帧间预测的基本单元。预测单元可进一步划分为一个或多个变换单元。变换单元是进行变换/量化的基本单元。变换单元共有 4 种尺寸：32×32、16×16、8×8、4×4。tx_mode 不等于 TX_MODE_SELECT 时，变换单元的尺寸为 tx_mode 允许的适合预测单元的最大尺寸；tx_mode 等于 TX_MODE_SELECT 时，变换单元的尺寸由 tx_size 确定。

如果是帧间预测，预测块尺寸与预测单元相同。当预测单元小于等于 64×64 时，变换单元在预测单元内依据从左到右，从上到下的光栅扫描顺序扫描；当预测单元大于 64×64 时，先将预测单元以 64×64 为单位按光栅扫描进一步划分，然后在每一个 64×64 的划分内部，变换单元按光栅扫描顺序依次扫描。

如果是帧内预测，预测单元可以进一步拆分为多个相同尺寸的预测块，每一个预测块对应一个变换单元。当预测单元小于等于 64×64 时，预测块及其对应的变换单元依据从左到右，从上到下的光栅扫描顺序；当预测单元大于 64×64 时，先将预测单元以 64×64 为单位按光栅扫描进一步划分，然后在每一个 64×64 的划分内部，预测块及其对应的变换单元按光栅扫描顺序依次扫描。

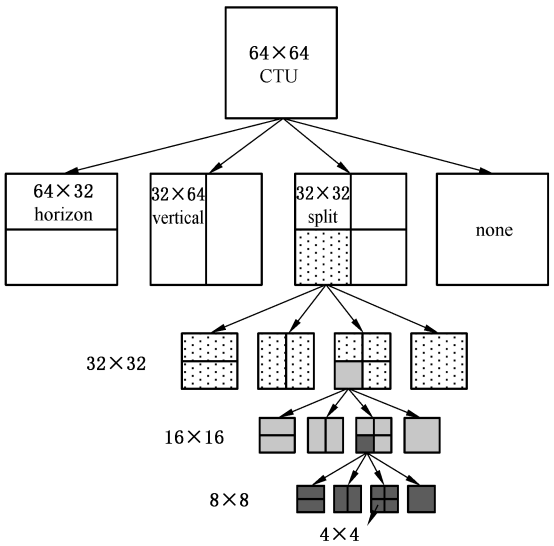


图 4 编码树划分方式

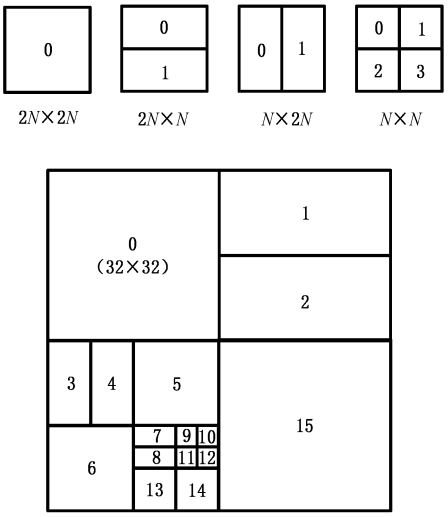


图 5　编码树的扫描顺序

5.1.3.3　相邻块可用性的推导过程

如图 6 所示，当前预测块大小为 $M\times N$，其参考样点按区域可分为 5 部分：左下(A)、左侧(B)、左上(C)、上方(D)和右上(E)，一共 $2\times(M+N)+1$ 个点。设当前块左上角样点在图像中的坐标是(x_0，y_0)，右下角样点在图像中的坐标是(x_1，y_1)，块 X(X 为 A、B、C、D 或 E)为表 6 中列出的样点所属的块。表 6 中坐标均为样点在帧图像中的位置坐标。

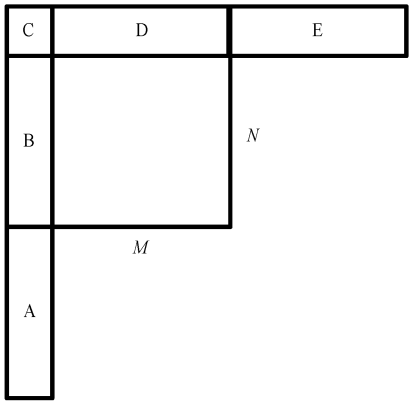


图 6　当前块与相邻块的空间位置关系

表 6　块 A、B、C、D 和 E 的位置

块 A 右上角样点坐标	块 B 右上角样点坐标	块 C 右下角样点坐标	块 D 左下角样点坐标	块 E 左下角样点坐标
(x_0-1，y_1+1)	(x_0-1，y_0)	(x_0-1，y_0-1)	(x_0，y_0-1)	(x_1+1，y_0-1)

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

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;否则标记为可用。

5.1.3.4 感兴趣区域(ROI)的划分

一个图像中可划分出若干感兴趣区域(ROI),ROI 的最小单位为 8×8,并且同一预测单元内的样点应属于同一 ROI 区域。如果图像中存在 ROI 划分,对应图像参数集中的参数 segmentation_enable 的取值等于 1,如果图像中不存在 ROI 划分,对应图像参数集中的参数 segmentation_enable 的取值应等于 0。一个图像中的 ROI 区域可分为 8 个不同等级,该等级由样点所在块的 segment_id 指示。图像中不存在 ROI 划分时,所有样点所在块的 segment_id 应等于 0。

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 7 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 7 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	

表 7（续）

伪代码描述语言	描述符
/* “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)	
/* “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比特的位置。具体要求如下：

byte_aligned( )的规定：

——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。

get_left_ae_bits( )的规定：

——熵解码器中的计数器 count 加上 8 然后对 8 求模，如果等于 0，则通过固定概率 128 继续解析出 16 比特；

——如果不等于 0，则通过固定概率 128 继续解析出求模后的值加 8 比特。

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不在本标准中规定。

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。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：

——ae(v)：二进制算术编码语法元素。该描述符的解析过程在 5.4.2 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；

——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；

——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 5.4.3 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 5.4.3 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 8。

表 8 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)
NumBytesInPayload = 0	
for(i=1; 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)

表 8（续）

nal_unit(NumBytesInNALunit) {	描述符
}	
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 9。

表 9 序列参数集 RBSP 语法表

seq_parameter_set_rbsp( ) {	描述符
profile_id	u(8)
level_id	u(8)
ldp_mode_flag	u(1)
frame_width_minus_1	u(16)
frame_height_minus_1	u(16)
chroma_format_idc	u(2)
bit_depth	u(2)
refs_per_frame	u(3)
frame_rate	u(3)
extended_sb_size_flag	u(1)
tile_enable	u(1)
wpp_enable	u(1)
sao_enable	u(1)
alf_enable	u(1)
roi_flag	u(1)
temporal_svc_flag	
if(temporal_svc_flag)	
layer_num minus_1	u(2)
spatial_svc_flag	u(2)
if(spatial_svc_flag){	
svc_ratio	u(3)
svc_mode	u(1)

表 9（续）

seq_parameter_set_rbsp( ) {	描述符
}	
if (frame_rate >=4){	
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 10。

表 10 图像参数集 RBSP 语法表

pic_parameter_set_rbsp( ) {	描述符
frame_num	u(8)
if(temporal_svc_flag)	
layer_id	u(3)
if(spatial_svc_flag){	
if(roi_flag && svc_mode)	
svc_roi_flag	u(1)
if(svc_roi_flag == 1){	
svc_top_left	u(16)
svc_bottom_right	u(16)
}	
}	
frame_type	u(1)
if(! roi_flag)	
ctu_dqp_enable	u(1)
if(ctu_dqp_enable){	
min_dqp_partition_size	u(3)
}	
if(frame_type ! = 0){	
refresh_frame_flags	u(5)
update_rps_flag	u(1)

表 10（续）

pic_parameter_set_rbsp( ) {	描述符
rps_idx	u(6)
if (update_rps_flag){	
for(i=0;i< refs_per_frame;i++){	
if (i == 2)	
opt_minus_flag	u(1)
delta_poc[i]	u(6)
}	
}	
refresh_pictures_num	u(3)
for(i=0;i< refresh_pictures_num;i++)	
delta_poc[i]	u(6)
allow_high_precision_mv	u(1)
interp_filter_switchable	u(1)
if(! interp_filter_switchable)	
interp_filter	u(3)
}	
filter_level	u(6)
sharpness_level	u(3)
lf_delta_enable	u(1)
if(lf_delta_enable){	
lf_delta_update	u(1)
if(lf_delta_update){	
for(i=0;i<6;i++){	
lf_ref_delta_enable[i]	u(1)
if(lf_ref_delta_enable[i]){	
lf_ref_deltas[i]	u(6)
lf_ref_deltas_sign[i]	u(1)
}	
}	
for(i=0;i<2;i++){	
lf_mode_delta_enable[i]	u(1)
if(lf_mode_delta_enable[i]){	
lf_mode_deltas[i]	u(6)
lf_mode_deltas_sign[i]	u(1)

表 10（续）

pic_parameter_set_rbsp( ) {	描述符
}	
}	
}	
}	
if(sao_enable)	
for (compIdx=0; compIdx<3; compIdx++)	
picture_sao_enable[compIdx]	u(1)
if(alf_enable)	
read_alf ( )	
base_qindex	u(8)
y_dc_delta_q_update_flag	u(1)
if(y_dc_delta_q_update_flag){	
y_dc_delta_q	u(4)
y_dc_delta_q_sign	u(1)
}	
uv_dc_delta_q_update_flag	u(1)
if(uv_dc_delta_q_update_flag){	
uv_dc_delta_q	u(4)
uv_dc_delta_q_sign	u(1)
}	
uv_ac_delta_q_update_flag	u(1)
if(uv_ac_delta_q_update_flag){	
uv_ac_delta_q	u(4)
uv_ac_delta_q_sign	u(1)
}	
if(roi_flag)	
segmentation_enable	u(1)
if(segmentation_enable){	
segmentation_update_map	u(1)
if(segmentation_update_map){	
for(i=0;i<7;i++){	
seg_tree_flag[i]	u(1)
if(seg_tree_flag)	
seg_tree_probs[i]	u(8)

表 10（续）

pic_parameter_set_rbsp( ) {	描述符
}	
seg_temporal_update	u(1)
if(seg_temporal_update){	
for (i=0;i<3;i++){	
seg_pred_flag[i]	u(1)
if(seg_pred_flag)	
seg_pred_probs[i]	u(8)
}	
}	
}	
seg_update_data	u(1)
if(seg_update_data){	
seg_abs_delta	u(1)
for(i=0;i<8;i++){	
for(j=0;j<4;j++){	
feature_enable[i][j]	u(1)
if(feature_enable){	
seg_feature_data[i][j]	u(v)
if(is_segfeature_signed[j])	
seg_feature_data_sign[i][j]	u(1)
}	
}	
}	
}	
}	
if(tile_enable){	
tile_cols_log2 = minLog2TileCols	
while(tile_cols_log2 < maxLog2TileCols){	
increment_tile_cols_log2	u(1)
if(increment_tile_cols_log2 ==1)	
tile_cols_log2++	
else	
break;	
}	

表 10（续）

pic_parameter_set_rbsp( ) {	描述符
tile_rows_log2	u(1)
if(tile_rows_log2 ==1){	
tile_rows_delta	u(1)
tile_rows_log2+= tile_rows_delta	
}	
}	
while(byte_aligned( ) == FALSE){	
reserved_bit	u(1)
}	
tx_mode	ae(v)
if(tx_mode==ALLOW_TX_32×32)	
tx_mode_delta	ae(v)
if(tx_mode==TX_MODE_SELECT){	
for(i=0;i<2;i++)	
diff_update_prob_8×8	ae(v)
for(i=0;i<2;i++)	
for(j=0;j<2;j++)	
diff_update_prob_16×16	ae(v)
for(i=0;i<2;i++)	
for(j=0;j<3;j++)	
diff_update_prob_32×32	ae(v)
}	
for(t=TX_4×4;t<= MAX_TX_SIZE;t++){	
coef_update_prob_flag	ae(v)
if(coef_update_prob_flag){	
for(i=0;i<2;i++)	
for(j=0;j<2;j++)	
for(k=0;k<6;k++)	
for(l=0; l< (k == 0 ? 3 : 6); l++)	
for(m=0;m<3; m++)	
diff_update_prob_coef	ae(v)
}	
}	
for (k =0; k<3; k++)	

表 10（续）

pic_parameter_set_rbsp( ) {	描述符
diff_update_prob_skip	ae(v)
if(alf_enable ==1){	
for(compIdx=0;compIdx<3;compIdx++)	
if(picture_alf_enable [compIdx])	
for (k =0; k<4; k++)	
diff_update_prob_alf	ae(v)
}	
if (frame_type == 1){	
for(i=0;i<7;++i)	
diff_update_prob_newmv	ae(v)
for(i=0;i<2;++i)	
diff_update_prob_zeromv	ae(v)
for(i=0;i<8;++i)	
diff_update_prob_refmv	ae(v)
if(interp_filter_switchable)	
for(j=0;j<5;++j)	
for(i=0;i<3;++i)	
diff_switchable_interp_probs	ae(v)
for(i=0;i<4;i++)	
diff_intrainter_update_prob	ae(v)
not_single_ref	ae(v)
if(not_single_ref)	
not_compound_ref	ae(v)
if(frame_reference_mode==REFERENCE_MODE_SELECT)	
for(i=0;i<5;++i)	
diff_inter_update_prob	ae(v)
if(frame_reference_mode! = COMPOUND_REFERENCE){	
for(i=0;i<5;++i){	
for(j=0;j<4;++j)	
diff_single_ref_update_prob	ae(v)
}	
}	
if(frame_reference_mode! =SINGLE_REFERENCE){	
for(i=0;i<5;++i)	

表 10（续）

pic_parameter_set_rbsp( ) {	描述符
for(j=0;j<3;++j)	
diff_comp_ref_update_prob	ae(v)
}	
for(j=0;j<20;j++)	
for(i=0;i<3;++i)	
diff_partition_update_prob	ae(v)
for(j=0;j<2;j++)	
for(i=0;i<7;++i)	
diff_delta_q_update_prob	ae(v)
read_mv_prob()	
}	
get_left_ae_bits( )	
rbsp_trailing_bits( )	
}	

read_mv_prob()语法定义见表 11。

表 11　read_mv_prob 语法表

read_mv_prob(){	描述符
for(j=0;j<3;j++)	
mv_joint_probs[j]	ae(v)
for(i=0;i<2;i++){	
mv_sign_prob[i]	ae(v)
for(j=0;j<10;j++)	
mv_class_probs[i][j]	ae(v)
mv_class0_bit_prob[i]	ae(v)
for(j=0;j<10;j++)	
mv_bits_prob[i][j]	ae(v)
}	
for(i=0;i<2;i++){	
for(j=0;j<2;j++)	
for(k=0;k<2;k++)	

表 11（续）

read_mv_prob(){	描述符
mv_class0_fr_probs[i][j][k]	ae(v)
for(k=0;k<2;k++)	
mv_fr_probs[i][k]	ae(v)
}	
if(allow_high_precision_mv){	
for(i=0;i<2;i++){	
mv_class0_hp_prob[i]	ae(v)
mv_hp_prob[i]	ae(v)
}	
}	
}	

read_alf ( )语法定义见表 12。

表 12 **read_alf 语法表**

read_alf ( ) {	描述符
if(alf_enable ==1){	
for(compIdx=0;compIdx<3;compIdx++)	
picture_alf_enable[compIdx]	u(1)
if(picture_alf_enable[0]==1 || picture_alf_enable[1]==1 || picture_alf_enable[2]==1)	
alf_parameter_set( )	
}	
}	

alf_parameter_set()语法定义见表 13。

表 13 **alf_parameter_set 语法表**

alf_parameter_set( ){	描述符
if(picture_alf_enable[0]==1){	
alf_filter_num_minus1	u(4)
for(i=0;i<alf_filter_num_minus1+1;i++){	
if(i>0 && alf_filter_num_minus1 ! = 15)	
alf_region_distance[i]	u(4)
for(j=0;j<10;j++)	
alf_coeff_luma[i][j]	se(v)

表 13（续）

alf_parameter_set( ){	描述符
}	
}	
if(picture_alf_enable[1]==1)	
for(j=0;j<10;j++)	
alf_coeff_chroma[0][j]	se(v)
if(picture_alf_enable[2]==1)	
for(j=0;j<10;j++)	
alf_coeff_chroma[1][j]	se(v)
}	

5.2.3.2.3 安全参数集 RBSP 语法

安全参数集 RBSP 语法见表 14。

表 14 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)
vek_flag	u(1)
iv_flag	u(1)
if(vek_flag) {	
vek_encryption_type	u(4)
evek_length_minus1	u(8)
evek	f(n)
vkek_version_length_minus1	u(8)
vkek_version	f(n)
}	
if(iv_flag) {	
iv_length_minus1	u(8)
iv	f(n)
}	
}	
if(authentication_flag) {	

表 14（续）

sec_parameter_set_rbsp( ) {	描述符
hash_type	u(2)
hash_discard_p_pictures	u(1)
signature_type	u(2)
successive_hash_pictures_minus1	u(8)
camera_idc	f(152)
}	
if(vek_flag || authentication_flag)	
camera_id	f(160)
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 15。

表 15 补充增强信息 RBSP 语法表

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

补充增强信息消息语法见表 16。

表 16 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	

表 16（续）

sei_message( ) {	描述符
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 编码片 RBSP 语法

编码片 RBSP 语法见表 17。

表 17 编码片 RBSP 语法表

tile_data_rbsp( ) {	描述符
if(tile_enable)	
tile_idx	ae(v)
for(n=0;n<ctu_num_in_tile;n++){	
if(wpp_enable && (0 == n % SbCols))	
substream_len	u(32)
if(sao_enable && (picture_sao_enable [0] || picture_sao_enable [1] || picture_sao_enable [2])){	
if(MergeUpAvail || MergeLeftAvail)	
sao_merge_flag	ae(v)
if(sao_merge_flag){	
if(MergeUpAvail && MergeLeftAvail)	
sao_merge_type	ae(v)
else{	
for (compIdx =0; compIdx<3; compIdx++){	
if(picture_sao_enable[compIdx]){	
sao_mode[compIdx]	ae(v)
if(sao_mode[compIdx] ! = 0){	
sao_type[compIdx]	ae(v)
if(sao_type[compIdx] == 0){	
sao_start_band[compIdx]	ae(v)

表 17（续）

tile_data_rbsp( ) {	描述符
for(j=0;j<4;j++){	
sao_offset_abs [compIdx][j];	ae(v)
if(sao_offset_abs [compIdx][j]! =0)	
sao_offset_sign [compIdx][j]	ae(v)
}	
}	
else {	
sao_edge_type[compIdx]	ae(v)
sao_edge_offset[compIdx][0]	ae(v)
sao_edge_offset[compIdx][1]	ae(v)
sao_edge_offset[compIdx][2]	ae(v)
sao_edge_offset[compIdx][3]	ae(v)
}	
}	
}	
}	
}	
}	
if (alf_enable)	
for (compIdx =0; compIdx <3; compIdx++)	
alf_ctu_enable[compIdx]	ae(v)
coding_tree_unit(n)	
}	
get_left_ae_bits( )	
rbsp_trailing_bits( )	
}	

5.2.3.2.6 认证数据 RBSP 语法

认证数据 RBSP 语法见表 18。

表 18　认证数据 RBSP 语法表

authentication_data_rbsp(){	描述符
frame_num	u(8)
if(spatial_svc_flag)	
spatial_el_flag	u(8)
authentication_data_length_minus1	u(8)
for(i=0;i< authentication_data_length_minus1+1;i++)	
authentication_data[i]	u(8)
rbsp_trailing_bits( )	
}	

5.2.3.2.7　流结尾 RBSP 语法

流结尾 RBSP 语法见表 19。

表 19　流结尾 RBSP 语法表

end_of_stream_rbsp( ) {	描述符
}	

5.2.3.2.8　RBSP 尾比特语法

RBSP 尾比特语法见表 20。

表 20　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)
}	

5.2.3.3　CTU 语法

CTU 语法见表 21。

表 21　CTU 语法表

coding_tree_unit( ) {	描述符
if(hasCols || hasRows)	
partition	ae(v)

表 21（续）

coding_tree_unit( ) {	描述符
if(ctu_dqp_enable && bsize>= min_dqp_partition_size)	
iscoded = 0	
if (subsize < BLOCK_8×8)	
block(0)	
else{	
switch(partition){	
case PARTITION_NONE:	
block(0)	
break	
case PARTITION_HORZ:	
block(0)	
block(1)	
break	
case PARTITION_VERT:	
block(0)	
block(1)	
break	
case PARTITION_SPLIT:	
coding_tree_unit(0)	
coding_tree_unit(1)	
coding_tree_unit(2)	
coding_tree_unit(3)	
break	
}	
}	
}	

5.2.3.4 块语法

块语法见表 22。

表 22 块语法

block(j) {	描述符
if(frame_type == 0){	
if (segmentation_enable && segmentation_update_map)	

表 22（续）

block(j) {	描述符
segment_id	ae(v)
if(segmentation_enable && FeatureEnabled[segment_id][SEG_LVL_SKIP])	
skip_flag = 1	
else	
skip_flag	ae(v)
if(tx_mode == TX_MODE_SELECT && bsize >= BLOCK_8×8)	
tx_size	ae(v)
switch (bsize) {	
case BLOCK_4×4:{	
for(i=0;i<4;i++)	
read_block_intra_luma_mode(i)	
break	
}	
case BLOCK_4×8:{	
read_block_intra_luma_mode(0)	
read_block_intra_luma_mode(1)	
break	
}	
case BLOCK_8×4:{	
read_block_intra_luma_mode(0)	
read_block_intra_luma_mode(1)	
break	
}	
default:	
read_block_intra_luma_mode(0)	
}	
read_block_intra_chroma_mode()	
}	
else{	
if (segmentation_enable) {	
if (segmentation_update_map) {	
if (segmentation_temporal_update){	
seg_id_predicted	ae(v)
if (! seg_id_predicted)	

表 22（续）

block(j) {	描述符
segment_id	ae(v)
}	
else	
segment_id	ae(v)
}	
}	
if(segmentation_enable && FeatureEnabled[segment_id][SEG_LVL_SKIP])	
skip_flag = 1	
else	
skip_flag	ae(v)
if(! segmentation_enable || ! FeatureEnabled[segment_id][SEG_LVL_REF_FRAME])	
inter_block	ae(v)
if((! skip_flag || ! inter_block) && tx_mode == TX_MODE_SELECT && MiSize >= BLOCK_8×8)	
tx_size	ae(v)
if(inter_block){	
if(! ref_segfeature_active){	
if(frame_reference_mode == REFERENCE_MODE_SELECT)	
block_reference_mode	ae(v)
if(ref_per_frame>1)	
ref_frame	ae(v)
}	
if(! skip_segfeature_active){	
if (bsize >= BLOCK_8×8){	
mv_mode	ae(v)
if (mv_mode == NEWMV){	
for (ref = 0; ref < 1 + is_compound; ++ref){	
mv_joint	ae(v)
if(mv_joint == MV_JOINT_HZVNZ || mv_joint == MV_JOINT_HNZVNZ){	
mvd_sign_0	ae(v)
mvd_value_0	ae(v)
}	
if (mv_joint == MV_JOINT_HNZVZ || mv_joint == MV_JOINT_HNZVNZ){	
mvd_sign_1	ae(v)

表 22（续）

block(j) {	描述符
mvd_value_1	ae(v)
}	
}	
}	
}	
if (bsize<BLOCK_8×8){	
for(i=0;i<2;i++){	
for(j=0;j<2;j++){	
mv_mode	ae(v)
if (mv_mode == NEWMV){	
for (ref = 0; ref < 1 + is_compound; ++ref){	
mv_joint	ae(v)
if(mv_joint == MV_JOINT_HZVNZ || mv_joint == MV_JOINT_HNZVNZ){	
mvd_sign_0	ae(v)
mvd_value_0	ae(v)
}	
if (mv_joint == MV_JOINT_HNZVZ || mv_joint == MV_JOINT_HNZVNZ){	
mvd_sign_1	ae(v)
mvd_value_1	ae(v)
}	
}	
}	
}	
}	
}	
if(interp_filter_switchable)	
interp_filter_mode	ae(v)
}	
}	
else{// Intra block	
switch (bsize) {	
case BLOCK_4×4:{	
for(i=0;i<4;i++)	
read_block_intra_luma_mode (i)	

表 22（续）

block(j) {	描述符
break	
}	
case BLOCK_4×8:{	
read_block_intra_luma_mode (0)	
read_block_intra_luma_mode (1)	
break	
}	
case BLOCK_8×4:{	
read_block_intra_luma_mode (0)	
read_block_intra_luma_mode (1)	
break	
}	
default:	
read_block_intra_luma_mode ( )	
}	
read_block_intra_chroma_mode ( )	
}	
}	
if (ctu_dqp_enable &&! iscoded) {	
if(! skip_flag){	
dqp_abs	ae(v)
if(dqp_abs)	
dqp_sign	ae(v)
iscoded=1	
}	
}	
for (plane=0;plane<3;++plane){	
if (! skip_flag){	
i=0	
while (i < max_eob) {	
coeff_value[i]	ae(v)
if(coeff_value[i]==EOB)	
break	
else{	

表 22（续）

block(j) {	描述符
while(coeff_value[i] == 0){	
i++	
coeff_value[i]	ae(v)
}	
coeff_sign[i]	ae(v)
i++	
}	
}	
}	
}	
}	

read_block_intra_luma_mode ( )语法定义见表 23。

表 23　read_block_intra_luma_mode 语法表

read_block_intra_luma_mode ( ){	
prev_intra_luma_pred_flag	ae(v)
if(prev_intra_luma_pred_flag){	
mpm_idx0	ae(v)
if(mpm_idx0)	
mpm_idx1	ae(v)
}	
else	
rem_pred_intra_mode	ae(v)
}	

read_block_intra_chroma_mode( )语法定义见表 24。

表 24　read_block_intra_chroma_mode 语法表

read_block_intra_chroma_mode( ){	
uv_fllow_y_flag	ae(v)
if(uv_fllow_y_flag)	
chroma_intra_mode	ae(v)
}	

5.2.3.5 监控扩展数据单元语法

5.2.3.5.1 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)==0x04)	
time_extension( )	
else if(next_bits(8)==0x10)	
gis_extension( )	
else if(next_bits(8)==0x11)	
analysis_ extension( )	
else if(next_bits(8)==0x12)	
osd_extension( )	
else	
reserved_extension( )	
}	
surveillance_extension_stop_byte	f(8)
}	

5.2.3.5.2 绝对时间信息扩展语法

绝对时间信息扩展语法见表 26。

表 26 绝对时间信息扩展语法表

time_extension( ) {	描述符
extension_id	u(8)
extension_length	u(8)
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)
}	
}	

5.2.3.5.3 智能分析信息扩展语法

智能分析信息扩展语法见表 27。

表 27 智能分析信息扩展语法表

analysis_ extension( ){	描述符
extension_id	u(8)
extension_length	u(16)
camera_id	f(160)
analysis_num	u(6)
for(i=0;i<analysis_num;i++){	
analysis_id[i]	u(8)
description_type[i]	u(2)
data_length[i]	u(16)
for(j=0;j<data_length;j++)	
analysis_data[i][j]	u(8)
}	
while(byte_align( )==FALSE)	
reserved_bit	u(1)
}	

5.2.3.5.4 OSD 信息扩展语法

OSD 信息扩展语法见表 28。

表 28 OSD 信息扩展语法表

osd_extension( ) {	描述符
extension_id	u(8)
extension_length	u(8)
sub_type	u(8)
code_type	u(8)
align_type	u(8)
char_size	u(8)
char_type	u(8)
top_low8	u(8)
top_high8	u(8)
left_low8	u(8)
left_high8	u(8)

表 28（续）

osd_extension( ) {	描述符
len	u(8)
res	u(24)
osd_data	u(n)
}	

5.2.3.5.5 地理信息扩展语法

地理信息扩展语法见表 29。

表 29 地理信息扩展语法表

gis_extension( ){	描述符
extension_id	u(8)
extension_length	u(8)
longitude_type	u(1)
longitude_degree	u(8)
longitude_fraction_bits	u(20)
latitude_type	u(1)
latitude_degree	u(8)
latitude_fraction_bits	u(20)
height	i(15)
speed	u(8)
yaw_degree	u(9)
reserverd	u(6)
}	

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 指示视频流支持的 SVAC 标准的版本。forbidden_zero_bit 应等于 1。

forbidden_zero_bit 等于 0 表示该视频流支持 GB/T 25724—2010 标准。

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，或一个参考图像的编码片。当一个编码图像的一个编码片 NAL 单元的 nal_ref_idc 等于 0 时，该编码图像的所有编码片 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 1：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 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 的 NAL 单元可以被解码器丢弃。

注 2：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。

当一个编码片 NAL 单元的 nal_unit_type 值等于 2 时，编码同一图像的其他所有编码片 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编码片 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 图像的编码片 tile_data_rbsp( )
2	IDR 图像的编码片 tile_data_rbsp( )
3	非 IDR 图像的 SVC 增强层编码片 tile_data_rbsp( )
4	IDR 图像的 SVC 增强层编码片 tile_data_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( )

表 30（续）

nal_unit_type	NAL 单元中 RBSP 语法结构的内容
10	认证数据 authentication_data_rbsp( )
11	流结尾 end_of_stream_rbsp( )
12	保留
13	本标准第 6 章使用
14	保留
15	SVC 增强层图像参数集 pic_parameter_set_rbsp( )

encryption_idc 指示 NAL 单元是否加密。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，且 RBSP 的最后一个字节不加密。

authentication_idc 指示 NAL 单元是否认证。authentication_idc 等于 0 表示该 NAL 单元未经认证，authentication_idc 等于 1 表示该 NAL 单元以安全参数集中指定的认证方法认证。

当 authentication_idc 等于 1 时，编码比特流中必须携带绝对时间扩展信息，用于标识认证时间。

payload_byte[i] 为一个 NAL 单元负载的第 i 个字节，等于 rbsp_byte[j]。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)。

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：

a) 如果 SODB 为空(长度为 0 比特)，RBSP 也为空；

b) 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：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，后面应存在一个或多个 rbsp_alignment_zero_bit 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 3：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 4：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 5：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 封装为 NAL 单元

使用 emulation_prevention_three_byte 将一个 RBSP 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 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 中查找字节对齐的下面二进制比特位串‘00000000 00000000 000000xx’，其中 xx 代表任意 2 比特位串‘00’‘01’‘10’或‘11’，并且在其中插入一个等于 0x03 的字节，形成‘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 单元的单元头部分即形成了整个 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 的顺序及生效

一个序列参数集 RBSP 包括的参数可以被一个或多个图像或者包含缓存周期 SEI 消息的 SEI NAL 单元使用。每个序列参数集 RBSP 在被解码器收到的同时生效，并且会导致先前有效的序列参数集 RBSP(如果有的话)失效。至多一个序列参数集 RBSP 在解码过程中的指定时刻为有效的。

当一个序列参数集 RBSP 被一个包含缓存周期 SEI 消息的 SEI NAL 单元使用时，该 SEINAL 单元应位于序列参数集 RBSP 之后。

图像参数集 RBSP 包括的参数可以被一个编码图像的编码片 NAL 单元使用。每个图像参数集 RBSP 在被解码器收到的同时生效，并且会导致先前有效的图像参数集 RBSP(如果有的话)失效。对 SVC 的某一层图像而言，至多一个图像参数集 RBSP 在解码过程中的指定时刻为有效的。

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，仅针对有效序列参数集和有效图像参数集。

在解码过程中，有效图像参数集和有效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 的生效

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生效，并且会导致先前有效的安全参数集 RBSP(如果有的话)失效。当序列参数集的 ldp_mode_flag 等于 1 时，安全参数集应只出现在 IDR 图像前。至多一个安全参数集 RBSP 在解码过程中的指定时刻为有效的。

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 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：

——一个图像的编码片顺序应按编码片的第一个 CTU 索引递增的顺序；

——一个图像的基本层图像参数集应位于一个编码图像的第一个 VCL NAL 单元之前；

——一个图像的增强层图像参数集与增强层 VCL NAL 单元应在对应的基本层图像的 VCL NAL 单元之后；

——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 及 14 的 NAL 单元不能位于编码图像的第一个 VCL NAL 单元之前。

nal_unit_type 等于 7，8 及 9 的 NAL 单元可作为编码图像的分界。

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

序列参数集 RBSP 语义如下：

profile_id 标识比特流支持的档次，取值定义见附录 C。

level_id 标识比特流支持的级别，取值定义见附录 C。

ldp_mode_flag 等于 1 表示比特流为低延时模式，编码图像的编码顺序与显示顺序一致，帧间预测只使用前向预测；等于 0 表示比特流为 RA 模式，编码图像的编码顺序与显示顺序可能不一致，帧间预测可能使用前向预测和双向预测。

frame_width_minus_1 加 1 等于图像宽度的样点数。

frame_height_minus_1 加 1 等于图像高度的样点数。

图像尺寸参数计算如下：

FrameWidth=frame_width_minus_1+1

FrameHeight=frame_height_minus_1+1

MiCols=(FrameWidth+7) 〉〉 3

MiRows=(FrameHeight+7) 〉〉 3

SbCols=(MiCols+(1〈〈(3+extended_sb_size_flag))−1) 〉〉(3+extended_sb_size_flag)

SbRows=(MiRows+(1〈〈(3+extended_sb_size_flag))−1) 〉〉(3+extended_sb_size_flag)

当 spatial_svc_flag 等于 1 时，FrameWidth 和 FrameHeight 应为增强层图像宽度和高度的样点数。

chroma_format_idc 等于 1 表示编码图像的色度采样格式为 4∶2∶0；等于 2 表示编码图像的色度采样格式为 4∶2∶2。

bit_depth 表示图像样点的比特精度。bit_depth 等于 0 表示图像样点比特精度 BitDepth 为 8；bit_depth 等于 1 表示图像样点比特精度 BitDepth 为 10；bit_depth 等于 2 表示图像样点比特精度 BitDepth

为 12，其余取值保留。

refs_per_frame 表示当前帧参考帧个数，refs_per_frame 的值应为 1～5。

frame_rate 为 3 比特无符号数，用于标识编码图像的帧率，取值见表 31。

表 31　frame_rate 取值对应的帧率

frame_rate	帧率 fps
0	25
1	30
2	50
3	60
4	由 VUI 参数决定
5～7	保留

extended_sb_size_flag 等于 1 表示树形编码单元尺寸为 128×128；等于 0 表示树形编码单元尺寸为 64×64。

tile_enable 等于 0 表示图像不会分割为多个编码片进行解码；等于 1 表示图像有可能会分割为多个编码片进行解码。

wpp_enable 等于 1 表示码流可支持 WPP 解码模式；等于 0 表示码流不支持 WPP 解码模式。当 tile_enable 等于 1 时，wpp_enable 应等于 0。

sao_enable 等于 0 表示不开启样点偏移补偿；等于 1 表示开启样点偏移补偿。

alf_enable 等于 0 表示不开启样点滤波补偿；等于 1 表示开启样点滤波补偿。

roi_flag 等于 1 表示编码视频序列中的图像允许进行分段编码；等于 0 表示编码视频序列中的图像不进行分段编码。

temporal_svc_flag 等于 0 表示不支持时域可分级编码；等于 1 表示支持时域可分级编码。

layer_num minus_1 在 temporal_svc_flag 等于 1 时有效。layer_num minus_1＋1 表示时域分级编码的增强层的层级数目，layer_num minus_1 值应为 0～3。

spatial_svc_flag 等于 1 表示支持空域可分级编码，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；等于 0 表示不支持空域可分级编码，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。

svc_ratio 为 3 比特无符号整数，在 spatial_svc_flag 等于 1 时有效，用于指示增强层与基本层的宽度和高度比，如表 32 所示。

表 32　svc_ratio 取值和增强层与基本层的比例对应关系

svc_ratio	增强层与基本层的比例
0	4∶3
1	2∶1
2	4∶1
3	6∶1
4	8∶1
5～7	保留

svc_mode 等于 0 表示 SVC 空域增强层不使用层间预测进行编码；等于 1 表示 SVC 空域增强层使用层间预测进行编码。

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；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

图像参数集 RBSP 语义描述如下：

frame_num 表示当前图像的图像顺序号的低 8 位。

layer_id 表示当前帧所属时域 SVC 的层级号，取值范围应为 0～4。高层级帧可参考低层级帧，低层级帧不能参考高层级帧。基本层的帧的 layer_id 应为 0。如果码流中不存在 layer_id，默认其取值等于 0。

svc_roi_flag 等于 1 表示支持增强层使用 ROI 编码；等于 0 表示不支持增强层使用 ROI 编码。如果码流中不存在 svc_roi_flag，默认其取值等于 0。当 svc_mode 等于 0 时，svc_roi_flag 应等于 0。

svc_top_left 在 svc_roi_flag 等于 1 时有效，表示增强层 ROI 区域的左上角的坐标索引，以 8×8 块为单位。

svc_bottom_right 在 svc_roi_flag 等于 1 时有效，表示增强层 ROI 区域的右下角的坐标索引，以 8×8 块为单位。

frame_type 等于 0 表示当前解码帧为帧内预测帧；等于 1 表示当前解码帧为帧间预测帧。

ctu_dqp_enable 等于 0 表示 CTU 单元内部不能调整量化参数；等于 1 表示 CTU 单元内部允许调整量化参数。如果 ctu_dqp_enable 在码流中不存在，默认其值等于 0。

min_dqp_partition_size 表示允许调整量化参数的预测单元的最小尺寸，见表 33。

表 33 允许调整量化参数的预测单元的最小尺寸与 min_dqp_partition_size 的对应关系

min_dqp_partition_size	extended_sb_size_flag=0	extended_sb_size_flag=1
0	BLOCK_64×64	BLOCK_128×128
1	BLOCK_32×32	BLOCK_64×64
2	BLOCK_16×16	BLOCK_32×32
3	BLOCK_8×8	BLOCK_16×16
4	N/A	BLOCK_8×8

refresh_frame_flags 表示参考帧缓冲区更新标识，refresh_frame_flags 的值应为 0～31。

update_rps_flag 表示是否更新 RPS 队列。update_rps_flag 等于 0 表示当前帧的 RPS 在 RPS 列表中，直接从 RPS 列表获取；等于 1 表示当前帧的 RPS 不在 RPS 列表中，需要从码流中读取。

rps_idx 表示 RPS 的索引，当 update_rps_flag 等于 1 时表示从码流中读取的 RPS 存储在 RPS 队列中的位置，当 update_rps_flag 等于 0 时表示当前帧的 RPS 在 RPS 列表中的索引。rps_idx 的值应为 0～63。

opt_minus_flag 标识参考帧 OPTIONAL_REF 所对应的 delta_poc[2]的符号。opt_minus_flag 等于 1 表示 delta_poc[2]为负；等于 0 表示 delta_poc[2]的符号为正。当 ldp_mode_flag 等于 1 时，opt_minus_flag 应等于 0。

delta_poc[i] 表示对应参考帧与当前帧的图像顺序号的差值，delta_poc[i]的值应为 0～63。

refresh_pictures_num 表示需要刷新的参考帧数目。

allow_high_precision_mv 等于 0 表示亮度运动补偿为四分之一样点精度；等于 1 表示亮度运动补

偿为八分之一样点精度。

interp_filter_switchale 等于 0 表示整帧图像使用固定的运动补偿插值滤波器；等于 1 表示在 CTU 内部确定使用的运动补偿插值滤波器。

interp_filter 表示使用的插值滤波器索引。

filter_level 表示环路滤波级别。

sharpness_level 表示环路滤波平滑度级别。

lf_delta_enable 等于 0 表示关闭环路滤波参数更新；等于 1 表示开启环路滤波参数更新。

lf_delta_update 表示环路滤波参数差值更新。

lf_ref_delta_enable[i] 表示参考帧相关环路滤波参数差值更新使能，等于 0 表示关闭参考帧相关环路滤波参数差值更新，等于 1 表示开启参考帧相关环路滤波参数差值更新。

lf_ref_deltas[i] 表示参考帧相关环路滤波参数差值。

lf_ref_deltas_sign[i] 表示参考帧相关环路滤波参数差值符号。

$$ref\_deltas[i]=lf\_ref\_deltas[i]\times lf\_ref\_deltas\_sign[i]$$

lf_mode_delta_enable[i] 表示模式相关环路滤波参数差值更新使能，等于 0 表示关闭模式相关环路滤波参数差值更新，等于 1 表示开启模式相关环路滤波参数差值更新。

lf_mode_deltas[i] 表示模式相关环路滤波参数差值。

lf_mode_deltas_sign[i] 表示模式相关环路滤波参数差值符号。

$$mode\_deltas[i] = lf\_mode\_deltas[i]\times lf\_mode\_deltas\_sign[i]$$

picture_sao_enable[i]表示当前图像的亮度与色度分量是否开启样点自适应补偿。picture_sao_enable[i]等于 0 表示不开启，等于 1 表示开启。其中 i 等于 0 表示亮度分量；i 等于 1 或 2 表示色度分量。

picture_alf_enable[i] 图像样点滤波补偿允许标志。表示当前图像的亮度与色度分量是否开启样点滤波补偿。picture_alf_enable[i]等于 0 则表示当前图像的第 i 个分量不应使用样点滤波补偿；等于 1 则表示当前图像的第 i 个分量使用样点滤波补偿。其中 i 等于 0 表示亮度分量；i 等于 1 或 2 表示色度分量。

alf_filter_num_minus1 的值加 1 表示当前图像亮度分量样点滤波补偿滤波器的个数。

alf_filter_num_minus1 的值应为 0～15。

alf_region_distance[i]表示亮度分量第 i 个样点滤波补偿滤波区域基本单元起始标号与第 i－1 个样点滤波补偿滤波区域基本单元起始标号间的差值。alf_region_distance[i]的取值范围应为 1～15。

如果比特流中不存在 alf_region_distance[i]，当 i 等于 0 时，则 alf_region_distance[i]的值为 0，当 i 不等于 0 且 alf_filter_num_minus1 的值为 15 时，则 alf_region_distance[i]的值为 1。比特流应满足 alf_region_distance[i](i=0～alf_filter_num_minus1)之和应小于等于 15。

alf_coeff_luma[i][j]表示亮度分量第 i 个样点滤波补偿滤波器的第 j 个系数。从比特流中解码得到的 alf_coeff_luma [i][j](j=0～8)的取值范围应为－64～63，alf_coeff_luma [i][9]的取值范围应为－1 088～1 071。

alf_coeff_chroma[0][j]表示 Cb 分量第 j 个样点滤波补偿滤波器的系数。alf_coeff_chroma[1][j]表示 Cr 分量第 j 个样点滤波补偿滤波器的系数。AlfCoeffChroma[0][j]的值等于 alf_coeff_chroma[0][j]的值，AlfCoeffChroma[1][j]的值等于 alf_coeff_chroma[1][j]的值。AlfCoeffChroma[i][j](i=0～1，j=0～7)的取值范围应为－64～63，AlfCoeffChroma[i][9](i=0～1)的取值范围应为－1 088～1 071。

base_qindex 表示量化参数索引。base_qindex 的值应为 1～255。

y_dc_delta_q_update_flag 表示亮度直流系数量化参数差值更新标志。y_dc_delta_q_update_flag 等于 0 表示关闭亮度直流系数量化参数差值更新；等于 1 表示开启亮度直流系数量化参数差值更新。

y_dc_delta_q 表示亮度直流系数量化参数差值。

y_dc_delta_q_sign 表示亮度直流系数量化参数差值符号。

uv_dc_delta_q_update_flag 表示色度直流系数量化参数差值更新标志。uv_dc_delta_q_update_flag 等于 0 表示关闭色度直流系数量化参数差值更新;等于 1 表示开启色度直流系数量化参数差值更新。

uv_dc_delta_q 表示色度直流系数量化参数差值。

uv_dc_delta_q_sign 表示色度直流系数量化参数差值符号。

uv_ac_delta_q_update_flag 表示色度交流系数量化参数差值更新标志。uv_ac_delta_q_update_flag 等于 0 表示关闭色度交流系数量化参数差值更新;等于 1 表示开启色度交流系数量化参数差值更新。

uv_ac_delta_q 表示色度交流系数量化参数差值。

uv_ac_delta_q_sign 表示色度交流系数量化参数差值符号。

segmentation_enable 表示图像分段使能。segmentation_enable 等于 0 表示关闭图像分段;等于 1 表示开启图像分段。如果码流中不存在 segmentation_enable,默认其值等于 0。

segmentation_update_map 表示更新图像分段图。

seg_tree_flag[i]等于 1 表示图像分段参数概率模型将更新;等于 0 表示图像分段参数将不更新。

seg_tree_probs[i]表示图像分段参数概率模型更新后的值。如果 seg_tree_probs[i]在码流中不存在,默认其值等于 255。

seg_temporal_update 表示时域图像分段更新。

seg_pred_flag[i]等于 1 表示图像分段预测概率模型将更新;等于 0 表示图像分段预测概率模型将不更新。

seg_pred_probs[i] 表示图像分段预测概率模型更新后的值。如果 seg_pred_probs[i]在码流中不存在,默认其值等于 255。

seg_update_data 表示图像分段更新数据。

seg_abs_delta 表示图像分段数据差值。

feature_enable[i][j] 表示图像分段特征使能。

seg_feature_data[i][j] 表示图像分段特征数据。

seg_feature_data_sign[i][j] 表示图像分段特征数据符号。如果 seg_feature_data_sign 在码流中不存在,默认其值等于 0。

图像分段特征数据 features_data[8][4]的计算方法如下:

```
is_segfeature_signed[4]={ 1, 1, 0, 0 };
for(i=0;i<8;i++){
  for(j=0;j<4;j++){
    if(feature_enable[i][j]){
      sign=1;
      if(is_segfeature_signed(j) && seg_feature_data_sign[i][j])
        sign=-1
    }
    features_data[i][j]=sign×seg_feature_data[i][j]
  }
}
```

increment_tile_cols_log2 表示是否增加 Tile 的列数,用于计算 tile_cols_log2。

tile_rows_log2 为 Tile 的行数标识,用于计算 TileRows。

log2_tile_rows_flag 表示 Tile 行标志。

tile_rows_delta 用于计算 tile_rows_log2。

图像划分的 Tile 的相关参数计算如下：

```
minLog2TileCols=0
while ((SbCols >> minLog2TileCols) >64)
  minLog2TileCols++
maxLog2TileCols=0
while ( (SbCols >> maxLog2 TileCols) >=8)
  maxLog2TileCols++
TileCols=1 << tile_cols_log2
TileRows=1 << tile_rows_log2
```

对第 c 个 Tile 列(c=0..TileCols−1)，其水平方向起始样点位置计算如下：

```
tile_col_start[c]=((c×SbCols) >> tile_cols_log2)<<(6+extended_sb_size_flag)
```

对第 r 个 Tile 行(r=0..TileRows−1)，其垂直方向起始样点位置计算如下：

```
tile_row_start[r]=((r×SbRows) >> tile_rows_log2)<<(6+extended_sb_size_flag)
```

当 tile_enable 等于 0 时，TileCols 和 TileRows 应等于 1。

reserved_bit 保留位，其值应等于 0。

tx_mode 表示变换模式。tx_mode 等于 0 表示变换模式为 TX_4×4，只支持 4×4 块变换，最大变换尺寸 MAX_TX_SIZE 为 TX_4×4；tx_mode 等于 1 表示变换模式为 ALLOW_TX_8×8，最大变换尺寸 MAX_TX_SIZE 为 TX_8×8；tx_mode 等于 2 表示变换模式为 ALLOW_TX_16×16，最大变换尺寸 MAX_TX_SIZE 为 TX_16×16；tx_mode 等于 3 表示变换模式为 ALLOW_TX_32×32，最大变换尺寸 MAX_TX_SIZE 为 TX_32×32。

tx_mode_delta 表示变换模式差值。如果 tx_mode_delta 在码流中不存在，默认其值等于 0。如果 tx_mode_delta 等于 1，则 tx_mode 等于 4，变换模式为 TX_MODE_SELECT。

diff_update_prob_8×8 表示 8×8 变换概率模型更新差值。

diff_update_prob_16×16 表示 16×16 变换概率模型更新差值。

diff_update_prob_32×32 表示 32×32 变换概率模型更新差值。

coef_update_prob_flag 表示残差系数概率模型更新标志。

diff_update_prob_coef 表示残差系数概率模型更新差值。

diff_update_prob_skip 表示跳过模式概率模型更新差值。

diff_update_prob_alf 表示自适应环路滤波使能概率模型更新差值。

diff_update_prob_newmv 表示 NEWMV 概率模型更新差值。

diff_update_prob_zeromv 表示 ZEROMV 概率模型更新差值。

diff_update_prob_refmv 表示 REFMV 概率模型更新标志差值。

diff_switchable_interp_probs 表示帧间运动补偿插值滤波器概率模型差值。

diff_intrainter_update_prob 表示帧内帧间概率模型差值。

not_single_ref 等于 0 表示本帧使用前向单参考帧，frame_reference_mode 取值为 SINGLE_REFERENCE；not_single_ref 等于 1 表示本帧可能使用双向参考模式。当 refs_per_frame 小于 3 时，not_single_ref 应等于 0，frame_reference_mode 取值应为 SINGLE_REFERENCE。

not_compound_ref 在 not_single_ref 等于 1 时有效。not_compound_ref 等于 0 表示本帧使用双向参考模式，frame_reference_mode 取值为 COMPOUND_REFERENCE，not_compound_ref 等于 1 表示本帧的参考帧模式由块级的 block_reference_mode 决定，frame_reference_mode 取值为 REFERENCE_MODE_SELECT。

diff_inter_update_prob 表示帧间编码概率模型更新差值。

diff_single_ref_update_prob 表示单向参考帧概率模型更新差值。

diff_comp_ref_update_prob 表示双向参考帧概率模型更新差值。

diff_mode_update_prob 表示模式概率模型更新差值。

diff_partition_update_prob 表示块划分概率模型更新差值。

diff_delta_q_update_prob 表示量化差值概率模型更新差值。

mv_joint_probs 表示运动矢量分量编码类型的概率模型。

mv_sign_prob 表示运动矢量符号的概率模型。

mv_class_probs 表示 mv_class 的概率模型。

mv_class0_bit_prob 表示 mv_class0_bit 的概率模型。

mv_bits_prob 表示 mv_bits 的概率模型。

mv_class0_fr_probs 表示 mv_class0_fr 的概率模型。

mv_fr_probs 表示 mv_fr 的概率模型。

mv_class0_hp_prob 表示 mv_class0_hp 的概率模型。

mv_hp_prob 表示 mv_hp 的概率模型。

上述概率模型的更新过程见 5.4.2.2.2。

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片，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 表示支持对整帧图像数据内容进行认证，进行认证的 NAL 单元包括编码片，以及在该帧传输的序列参数集、图像参数集、安全参数集以及扩展数据单元。当支持对上述数据内容进行认证时，编码比特流中必须携带绝对时间扩展信息，且携带在编码比特流中的认证数据应经过 Base64 编码。认证数据通过 nal_unit_type 等于 10 的 NAL 单元传输。如果图像中存在 authentication_idc 等于 1 的编码片、序列参数集、图像参数集、安全参数集、扩展数据单元等，对一个图像中 authentication_idc 为 1 的 NAL 单元按解码顺序排列后进行认证产生该图像的摘要数据。authentication_flag 等于 0 表示不支持对图像进行认证，编码比特流中不应包含 nal_unit_type 等于 10 的 NAL 单元。

如果 spatial_svc_flag 等于 1，对同一帧图像的基本层图像和增强层图像分别进行认证，增强层的图像参数集与编码片按增强层进行认证，序列参数集、安全参数集以及扩展数据单元与基本层的图像参数集及编码片一起按基本层进行认证。

encryption_type 指示加密所采用的算法，具体对应关系见表 34。

表 34 encryption_type 与具体加密算法的对应关系

encryption_type	加密算法
0	SM1
1	SM4
2～15	保留

vek_flag 等于 1 表示携带 vek；等于 0 表示不携带 vek。

iv_flag 等于 1 表示携带 iv；等于 0 表示不携带 iv。

camera_id 为 20 个字节的字符串，表示图像来源的摄像机 ID。

vek_encryption_type 指示密钥加密采用的算法，具体对应关系同 encryption_type。

evek_length_minus1 为加密后的密钥长度减 1，以字节为单位。

evek 为加密后的密钥，用于加密计算，长度为 evek_length_minus1 加 1 字节。

vkek_version length_minus1 为加密密钥版本号长度减 1，以字节为单位。

vkek_version 为加密密钥版本号，长度为 vkek_version_length_minus1 加 1 字节。

iv_length_minus1 为初始向量长度减 1，以字节为单位。

iv 为初始向量，用于分组加密，长度为 iv_length_minus1 加 1 字节。

hash_type 表示进行认证所采用的算法，见表 35。

表 35　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	SM3	32
1～3	保留	保留

hash_discard_p_pictures 等于 1 表示对非 IDR 图像不进行认证；等于 0 表示对非 IDR 图像进行认证。如果 hash_discard_p_pictures 不在码流中，默认其值等于 1。不进行认证的图像中每个 NAL 单元的 authentication_idc 均应等于 0。

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，且这些连续图像仅限在一个 IDR 图像间隔中。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 7 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。

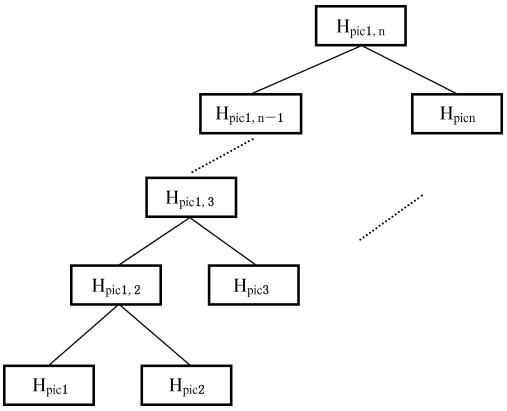


图 7　树型形摘要示意图

SuccessiveHashPictures＝successive_hash_pictures_minus1＋1

注：安全参数集激活后的第一个进行认证的图像应为连续 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。如果一个 IDR 图像间隔中进行认证的图像不足 SuccessiveHashPictures，签名数据对应的摘要数据为前一个 IDR 图像间隔包含的所有的图像摘要数据。

如果 successive_hash_pictures_minus1 等于 0，对每一个进行认证的图像的摘要数据进行数字签名。

signature_type 表示对图像的摘要数据进行数字签名的算法，见表 36。

表 36　signature_type 与具体签名算法的对应关系

signature_type	签名算法
0	SM2
1～3	保留

camera_idc 为 19 个字节的字符串，用于表示图像来源的摄像机的证书标识。

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　编码片 RBSP 语义

tile_idx 表示当前 Tile 在图像中的索引。

substream_len 表示当前 CTU 行的编码码流长度，以字节为单位。

sao_merge_flag 表示当前 CTU 的 SAO 参数融合标志。sao_merge_flag 等于 0 表示参数不融合；等于 1 表示参数融合，此时其 SAO 参数与其左侧相邻或者上方相邻的 CTU 的 SAO 参数相同。

sao_merge_type 等于 1 表示当前 CTU 的 SAO 参数使用左侧相邻的 CTU 的 SAO 参数；等于 0 表示当前 CTU 的 SAO 参数使用上方相邻的 CTU 的 SAO 参数。

sao_mode[compIdx]等于 0 表示当前 CTU 中第 compIdx 个分量的 SAO 模式为 SAO_OFF；等于 1 表示当前 CTU 中第 compIdx 个分量的 SAO 模式由 sao_type[compIdx]确定。

sao_type[compIdx]等于 0 表示当前 CTU 中第 compIdx 个分量的 SAO 模式为 SAO_BO；等于 1 表示当前 CTU 中第 compIdx 个分量的 SAO 模式为 SAO_EO。

sao_start_band[compIdx]表示当前 CTU 中第 compIdx 个分量在 SAO_BO 模式下的起始补偿区间，取值应为 0～31。

sao_offset_sign[compIdx][j]表示 SAO_BO 模式下 sao_offset[compIdx][j]的符号。sao_offset_sign[compIdx][j]等于 0 表示对应的 sao_offset[compIdx][j]的取值为正，等于 1 表示对应的 sao_offset[compIdx][j]的取值为负。

sao_offset_abs[compIdx][j]表示 SAO_BO 模式下的补偿值 sao_offset[compIdx][j]的绝对值，取值应为 0～(1<<(Min(bit_depth,10)－5))－1。

sao_edge_type[compIdx]表示当前 CTU 中第 compIdx 个分量在 SAO_EO 模式下的角度方向。sao_edge_type[compIdx]等于 0 表示 EO_0°；等于 1 表示 EO_90°；等于 2 表示 EO_135°；等于 3 表示 EO_45°。

sao_edge_offset[compIdx][j]表示 SAO_EO 模式下的对应的补偿值。

alf_ctu_enable[compIdx]等于 0 表示当前 CTU 的第 compIdx 个分量不进行样点滤波补偿，alf_ctu_enable[compIdx]等于 1 表示当前 CTU 的第 compIdx 个分量进行样点滤波补偿。

5.2.4.4.6 认证数据 RBSP 语义

frame_num 表示应包含认证数据的图像，该图像为在认证数据 NAL 单元之前最临近的 frame_num 与认证数据的 frame_num 相同的图像。successive_hash_pictures_minus1 等于 0 时，frame_num 指示认证数据对应的图像；大于 0 时，frame_num 指示连续 SuccessiveHashPictures 个图像的最后一个。

spatial_el_flag 等于 1 表示该认证数据为增强层签名数据，spactial_el_flag 等于 0 表示该认证数据为基本层签名数据。如果 spactial_el_flag 在码流中不存在，默认其值等于 0。

authentication_data_length_minus1 加 1 表示签名数据的长度，以字节为单位，取值应为 0～255。

authentication_data[i]为一个签名数据的第 i 个字节。

签名数据应经过 Base64 编码，Base64 编码方法见 rfc3548。

5.2.4.4.7 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。

5.2.4.4.8 RBSP 尾比特语义

rbsp_stop_one_bit 应等于 1。

rbsp_alignment_zero_bit 应等于 0。

5.2.4.5 CTU 语义

partition 表示当前块划分类型，取值可能为 PARTITION_NONE，PARTITION_HORZ，PARTITION_VERT 和 PARTITION_SPLIT 中的一个。当前块在水平方向上属于图像内的部分小于等于块宽度的一半时，hasCols 等于 0，否则 hasCols 等于 1；当前块在垂直方向上属于图像内的部分小于等于块高度的一半时，hasRows 等于 0，否则 hasRows 等于 1。当 hasCols 等于 1 或 hasRows 等于 1 时，partition 通过码流解析获得；当 hasCols 和 hasRows 均等于 0 时，默认 partition 取值等于 PARTITION_SPLIT。

5.2.4.6 块语义

segment_id 表示当前段的编码，取值范围为 0～7。

seg_id_predicted 表示段的编码是否采用预测编码。

inter_block 表示当前块是否是帧间编码块。

skip_flag 表示当前块是否是跳过编码。

coeff_value 表示块系数的取值。

coeff_sign 表示块系数的符号。

tx_size 表示当前块采用的变换矩阵尺寸。tx_size 等于 0 表示变换矩阵为 TX_4×4；等于 1 表示变换矩阵为 TX_8×8；等于 2 表示变换矩阵为 TX_16×16；等于 3 表示变换矩阵为 TX_32×32。

prev_intra_luma_pred_flag 表示亮度帧内预测模式是否位于帧内预测模式预测列表中，预测列表含 5 个最有可能的预测模式。

mpm_idx0 等于 0 为表示当前亮度预测模式为预测列表中的第一个模式；等于 1 表示当前亮度预测模式不是预测列表中的第一个模式。

mpm_idx1 当 mpm_idx0 为 1 时，mpm_idx1＋1 表示当前预测模式位于预测列表中的位置。mpm_idx1 取值应为 0～3。

rem_pred_intra_mode 表示当前亮度预测模式在除预测列表中的 5 个预测模式外，剩余 32 个预测模式中的索引。rem_pred_intra_mode 取值应为 0～31。

uv_fllow_y_flag 等于 1 表示色度帧内预测模式与其对应位置的亮度帧内预测模式一致，uv_fllow_y_flag 等于 0 表示色度帧内预测模式与其对应位置的亮度帧内预测模式不一致。

chroma_intra_mode 表示色度帧内预测模式索引。

block_reference_mode 表示当前块的参考帧模式，取值为 SINGLE_REFERENCE 或者 COMPOUND_REFERENCE。如果 block_reference_mode 在码流中不存在，则 block_reference_mode 取值等于 frame_reference_mode。如果 block_reference_mode 等于 COMPOUND_REFERENCE，则 is_compound 等于 1，否则 is_compound 等于 0。

ref_frame 表示当前预测块参考帧索引。当 block_reference_mode 等于 SINGLE_REFERENCE 时，ref_frame 有 5 种可能的取值，分别为 DYNAMIC_REF，STATIC_REF，OPTIONAL_REF，DYNAMIC_REF_1 和 DYNAMIC_REF_2。当 block_reference_mode 等于 COMPOUND_REFERENCE 时，ref_frame 有 4 种可能的取值，分别为 DYNAMIC_REF，STATIC_REF，DYNAMIC_REF_1 和 DYNAMIC_REF_2。

mv_mode 表示运动矢量模式。mv_mode 有 4 种可能的取值，分别为 NEARESTMV，NEARMV，ZEROMV，NEWMV。

mv_joint 表示运动矢量分量编码类型。

mvd_sign_0、**mvd_sign_1** 表示运动矢量与预测运动矢量差值的符号。

mvd_value_0、**mvd_value_1** 表示运动矢量与预测运动矢量的差值。当 allow_high_precision_mv 等于 1 且对应的 PMV 的水平分量与垂直分量均小于 8 时，usehp 等于 1，此时 mvd_value0 和 mvd_value1 为 1/8 样点精度，否则 mvd_value0 和 mvd_value1 为 1/4 样点精度。

interp_filter_mode 表示帧间运动补偿滤波器模式。interp_filter_mode 取值为 0～3，分别对应滤波器模式 Regular、Smooth-1、Sharp 和 Smooth-2。

dqp_abs 表示量化系数差值的绝对值。如果 dqp_abs 在码流中不存在，默认取值为 0。

dqp_sign 表示量化系数差值的符号。0 代表负，1 代表正。如果 dqp_sign 在码流中不存在，默认取值为 0。

5.2.4.7 监控扩展数据单元语义

5.2.4.7.1 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。

5.2.4.7.2 绝对时间信息扩展语义

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

hour_bits 表示小时信息。hour_bits 取值应为 0～23。

minute_bits 表示分钟信息。minute_bits 取值应为 0～59。

second_bits 表示秒信息。second_bits 取值应为 0～59。

second_fraction_bits 表示秒的分数信息，以 1/16 384 秒为单位。second_fraction_bits 取值应为 0～16 383。

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 取值应为 0～127，加 2000 表示年份信息 Year。计算如下：

$$Year = year\_minus2000\_bits + 2000$$

month_bits 表示月份信息。month_bits 取值应为 1～12。

day_bits 表示日期信息。date_bits 取值应为 1～31。

5.2.4.7.3 智能分析信息扩展语义

extension_id 为 8 位无符号整数，智能分析信息扩展的标号 extension_id 应等于 0x11。

extension_length 为 16 位无符号整数，表示 extension_length 之后的本扩展语法元素长度，以字节为单位。

camera_id 为摄像机标识。

analysis_num 为图像分析结果的数量。

analysis_id[i] 为第 i 项图像分析结果的分析功能标识，其定义见表 37。

表 37 anlysis_id 的取值定义

analysis_id	定义
0x01	图像分析规则
0x02	运动目标检测
0x03	人员属性分析
0x04	机动车特征分析
0x05	人脸比对
0x06	车牌识别
0x07	绊线检测
0x08	入侵检测
0x09	逆行检测
0x0A	徘徊检测
0x0B	遗留物检测
0x0C	目标移除检测
0x0D	目标数量统计
0x0E-0xFF	保留

description_type[i] 为第 i 项图像分析结果的描述形式，其取值定义见表 38。

表 38 description_type 的取值定义

description_type	定义
0x00	保留
0x01	根据附录 F 表示
0x02	自定义表示
0x03	保留

data_length[i] 为第 i 项图像分析结果描述内容的字节长度。

analysis_data[i][j] 为第 i 项图像分析结果描述内容的第 j 个字节。

5.2.4.7.4 OSD 信息扩展语义

extension_id 为 8 位无符号整数，OSD 信息扩展的标号 extension_id 应等于 0x12。

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

sub_type 为 8 位无符号整数，表示 OSD 扩展信息子类型。可为不同的 OSD 信息分配不同的子类型，sub_type 等于 32 时，表示时间 OSD 信息；sub_type 等于 33 时，表示摄像机名称 OSD 信息；sub_type 等于 34 时，表示地点标注 OSD 信息。

一个 NAL 单元中不应出现两个及以上的 sub_type 取值相同的 OSD 扩展信息。

注：当多个 sub_type 取值相同的 OSD 扩展信息出现在同一个 NAL 单元时，最后一个扩展信息有效。

code_type 为 8 位无符号整数，表示 OSD 字符的编码格式。code_type 的值为 0 时，表示使用 UTF-8 编码。

align_type 为 8 位无符号整数，表示 OSD 字符的对齐格式。align_type 的值等于 0 为左对齐；等于 1 为右对齐。

char_size 为 8 位无符号整数，表示 OSD 字符字体大小，以样点为单位表示。

char_type 为 8 位无符号整数，表示 OSD 字符字符格式。char_type 等于 0 为白底黑边；等于 1 为黑底白边；等于 2 为白色；等于 3 为黑色；等于 4 为自动反色。

top_low8 和 **top_high8** 组成一个 16 位无符号整数 top，表示 OSD 字符信息上边界在图像画面中的位置，以样点为单位表示。top 的取值计算如下：

$$top=(top\_high8\langle\langle 8)+top\_low8$$

left_low8 和 **top_high8** 组成一个 16 位无符号整数 left，表示 OSD 字符信息左边界在图像画面中的位置，以样点为单位表示。left 的取值计算如下：

$$left=(left\_high8\langle\langle 8)+left\_low8$$

len 为 8 位无符号整数，表示 osd_data 占用的字节长度，取值应为 0～243。

res 为 8 位无符号整数，取值应为 0～255 之间。

osd_data OSD 字符数据。其中，定义'\n'为换行符，'\0'为结束符。osd_data 的长度为 len 字节。

5.2.4.7.5 地理信息扩展语义

extension_id 为 8 位无符号整数，地理信息扩展的标号 extension_id 应等于 0x10。

longitude_type 等于 0 表示东经；等于 1 表示西经。

longitude_degree 为 8 位无符号整数，表示经度的度数。

longitude_fraction_bits 表示度的分数信息，以 1/1 048 576 度为单位，取值应为 0～1 048 575。

latitude_type 等于 0 表示北纬；等于 1 表示南纬。

latitude_degree 为 8 位无符号整数，表示纬度的度数。

latitude_fraction_bits 表示度的分数信息，以 1/1 048 576 度为单位，取值应为 0～1 048 575。

height 表示高度，单位为米。

speed 表示速度，单位为米/秒。

yaw_degree 表示方向角度数，0 表示方向为正北，角度数沿顺时针方向递增。

reserverd 为保留比特位。

5.3 解码过程

5.3.1 视频解码器

视频解码流程示例见图 8。视频解码器接收编码比特流，对图像中的树形编码单元，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'。重建图像经去块效应滤波，样点偏移补偿(SAO)、样点滤波补偿(ALF)后产生最终的解码图像。

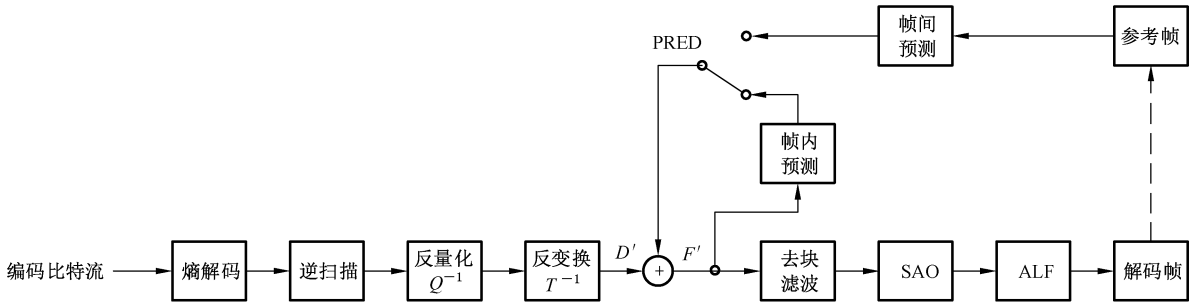


图 8　解码流程示例

5.3.2　NAL 单元解码过程

NAL 单元解码过程的输入为 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，则从 NAL 单元中提取 RBSP 语法结构时需对加密的 RBSP 进行解密处理，得到未加密的 RBSP。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 单元的帧内预测过程，见 5.3.4；

——nal_unit_type 的值为 1 时 NAL 单元中的的帧间预测过程，见 5.3.5；

——nal_unit_type 的值为 1 和 2 时 NAL 单元中的树形编码单元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 1，2，3 和 4 时 NAL 单元的重建图像的样点偏移补偿过程，见 5.3.8；

——nal_unit_type 的值为 1，2，3 和 4 时 NAL 单元的重建图像的样点滤波补偿过程，见 5.3.9；

——nal_unit_type 的值为 3 和 4 时 NAL 单元中的树形编码单元在去块效应滤波之前的解码过程，见 5.3.10；

——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。有效的序列参数集、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 0，12，14 和 15 的 NAL 单元的解码过程不在本标准规定。

5.3.3　图像解码过程

5.3.3.1　图像的分类和对应关系

编码视频序列中可能解码出以下三种图像：

——帧内解码图像(I 图像)；

——即时解码刷新图像(IDR 图像)；

——帧间解码图像。

所有图像为帧图像。

符合本标准的 I 图像同时应为 IDR 图像。IDR 图像解码之后，解码顺序上所有后续的编码图像均不用根据任何在该 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

应为 IDR 图像。可通过 nal_unit_type 来识别一幅图像是否为 IDR 图像。

帧内解码图像和帧间解码图像均可作为参考图像。

编码图像序列以图像编码组(GOP)组成。连续编码的两帧图像的图像顺序号的差值应不大于 16。参考图像支持分层与不分层结构。当 temporal_svc_flag 等 0 时,表示当前编码图像不支持时域分层编码;当 temporal_svc_flag 等 1 时,表示当前编码支持时域分层编码,此时在码流中的 layer_id 表示当前帧的时域 SVC 层级。处于同一层级的图像帧应按照显示顺序进行编码。低层级的帧(layer_id 较小的帧)不能参考高层级的帧(layer_id 较大的帧)。

RA 模式下典型的分层结构如图 9 所示。

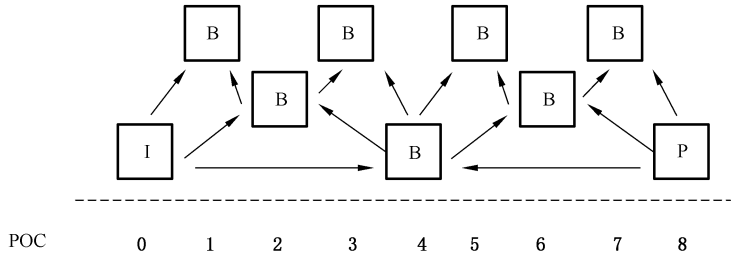


图 9　RA 模式下的时域 SVC 分层结构模型

RA 模式下的解码顺序为 0→8→4→2→6→1→3→5→7,层级结构可设为第 0、8 帧为第 0 层级(layer_id 为 0),第 4 帧为第 1 层级(layer_id 为 1),第 2、6 帧为第 2 层级(layer_id 为 2)第 1、3、5、7 帧为第 3 层级(layer_id 为 3)。第 8 帧参考第 0 帧,第 4 帧参考第 0 和第 8 帧,第 2 帧参考第 0 和第 4 帧,第 6 帧参考第 4 和第 8 帧,第 1、3、5、7 帧属于最高层级可以参考之前已经解码的所有帧。

LD 模式下典型的分层结构如图 10 所示。

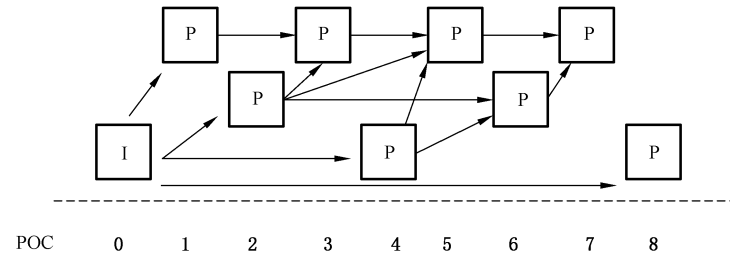


图 10　LD 模式下的时域 SVC 分层结构模型

LD 模式下解码顺序为 0→1→2→3→4→5→6→7→8,层级结构可设 0、8 帧为第 0 层级(layer_id 为 0),第 4 帧为第 1 层级(layer_id 为 1),第 2、6 帧为第 2 层级(layer_id 为 2),第 1、3、5、7 为第 3 层级(layer_id 为 3)。参考帧配置可以为:第 1 帧参考之前已经解码的所有帧,这里参考第 0 帧;第 2 帧参考之前已经解码的所有图像顺序号为偶数的帧,这里参考第 0 帧;第 3 帧参考之前已经解码的所有帧,这里参考第 0、1、2 帧;第 4 帧参考之前已经解码的所有图像顺序号为 4 的倍数的帧,这里参考第 0 帧;第 5 帧可以参考之前已经解码的所有帧,这里参考第 2、3、4 帧;第 6 帧参考之前已经解码的所有图像顺序号为 2 的倍数的帧,这里参考第 0、2、4 帧;第 7 帧参考之前已经解码的所有帧,这里参考第 4、5、6 帧;第 8 帧参考之前已经解码的所有图像顺序号为 8 的倍数的帧,这里参考第 0 帧。

5.3.3.2　图像顺序号的解码过程

图像顺序号按照图像的显示顺序依次递增。当前帧的图像顺序号 cur_frame_poc 计算如下:

cur_frame_poc=poc_msb+poc_lsb

其中,poc_lsb 等于当前图像的 frame_num。poc_msb 在解码开始或者当前帧为 IDR 帧时置为 0,

否则，当前帧的 poc_msb 计算如下：

```
if((last_poc_lsb－poc_lsb)＞＝128)
{
    poc_msb＝poc_msb＋256
}
else if((poc_lsb－last_poc_lsb)＞＝128)
{
    poc_msb＝poc_msb－256
    if(poc_msb＜0)
      poc_msb＝0
}
else
{
    poc_msb＝poc_msb
}
```

其中 last_poc_lsb 为编码顺序上前一帧图像的 poc_lsb。

5.3.3.3 编码片的解码过程

编码片解码开始时，根据 FrameWidth、FrameHeight、TileCols、TileRows、tile_idx 可以计算出编码片起始的图像样点位置以及本编码片中包含的 CTU 个数(记作 ctu_num_in_tile)。依次解码编码片中的 CTU，解码 ctu_num_in_tile 个 CTU 后，编码片解码结束。

5.3.3.4 参考图像选择

5.3.3.4.1 参考图像集

参考图像集(RPS)用于表示当前帧对应的各参考帧图像，每个 RPS 中的参考帧数量等于 refs_per_frame。

每帧图像具有唯一的图像顺序标志 poc。当前帧所用的第 i 个参考帧对应的 poc 计算如下：

poc_refframe[i]＝cur_frame_poc－sign_ref[i]×delta_poc[i]

sign_ref[i]＝1　(i＝0,1,3,4)

sign_ref[i]＝Sign(0－opt_minus_flag)　(i＝2)

i 等于 0～4 分别对应参考帧 DYNAMIC_REF、STATIC_REF、OPTIONAL_REF、DYNAMIC_REF_1 和 DYNAMIC_REF_2，其中只有 OPTIONAL_REF 参考帧有可能是后向参考帧，其他都应为前向参考帧。

当 delta_poc[i](i＞0)等于 delta_poc[0]时，表示 RPS 中的第 i 个参考帧无效，

不同图像帧可以采用相同的 RPS，因此采用队列来存储 RPS，其中队列最大存储 RPS 个数为 64 个。当 update_rps_flag 等于 1，表示当前帧的 RPS 需要更新，应从码流中读取 delta_poc 计算出 RPS，并存入 RPS 队列，参考索引值为 rps_idx。当 update_rps_flag 等于 0 表示当前帧的 RPS 在 RPS 队列，直接从 RPS 队列中获取，参考索引值为 rps_idx。

5.3.3.4.2 参考帧缓冲区的刷新

参考帧缓冲区内最多可以存在 8 个参考帧。从码流中获取 refresh_frame_flags，refresh_flag [i]表示 refresh_frame_flags 从低位算起的第 i 个比特，refresh_flag [i]对应 RPS 中第 i 个参考帧的刷新状

态。如果 refresh_flag [i]等于 1,则当前帧解码完成后,对应的参考帧将从参考帧缓冲区中移除;如果 refresh_flag [i]等于 0,则解码完当前帧后,对应的参考帧将保留在参考帧缓冲区中。解码当前帧后,根据从码流获取的 refresh_pictures_num 以及之后的 delta_poc,计算对应参考帧的图像顺序号,然后将对应参考帧从参考帧缓冲区中移除。

5.3.4 帧内预测过程

5.3.4.1 帧内预测模式

帧内预测模式共 37 种,取值为 0～36,如图 11 所示。luma_intra_mode 与 chroma_intra_mode 的取值与帧内预测模式的对应关系见表 39 和表 40。

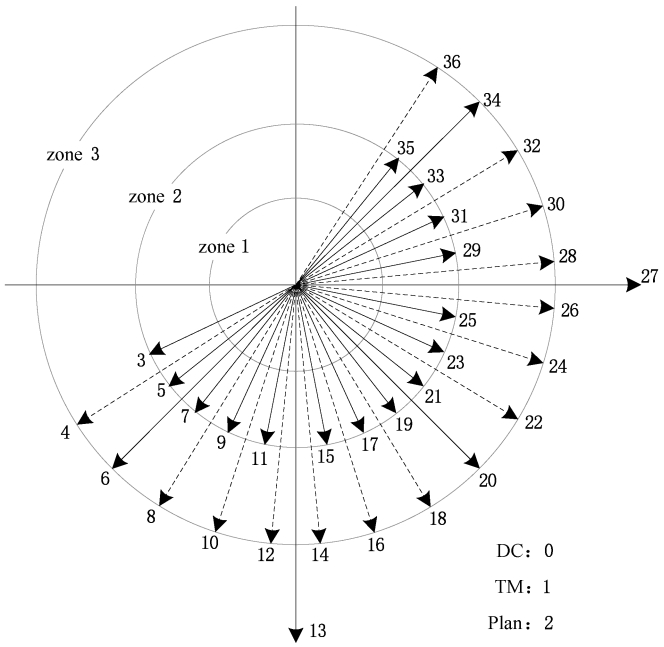


图 11 帧内预测模式

表 39 亮度块帧内预测模式

luma_intra_mode	帧内预测模式
0	DC_PRED
1	TM_PRED
2	PLAN_PRED
3～12	ANG_PRED
13	Vertical_PRED
14～26	ANG_PRED
27	Horizontal_PRED
28～36	ANG_PRED

表 40　色度块帧内预测模式

chroma_intra_mode	帧内预测模式
0	DC_PRED
1	PLAN_PRED
2	Vertical_PRED
3	Horizontal_PRED
4	DM_PRED

表 40 中，DM_PRED 表示当前色度块对应位置亮度块的预测模式，当对应的亮度块包含多个预测块时，为最后一个预测块的预测模式。当 DM_PRED 等于 DC_PRED、PLAN_PRED、Vertical_PRED、Horizontal_PRED 其中的一种，则将表 40 中对应索引的预测模式用 TM_PRED 代替，而 DM_PRED 本身不变。

5.3.4.2　帧内预测模式的确定

5.3.4.2.1　亮度块的帧内预测模式

亮度块的帧内预测模式计算如下：

a)　根据当前块的左边块和上边块的预测模式构建预测列表 intra_candlis [i]，i=0～4：

如果左边块存在并且是帧内预测块，则将左边块的帧内预测模式赋值给 intraPredModeLeft；否则 intraPredModeLeft 等于 0(DC_PRED)。

如果上边块存在并且是帧内预测块，则将上边块的帧内预测模式赋值给 intraPredModeAbove；否则 intraPredModeAbove 等于 0(DC_PRED)。

令 PredMode0 等于 min(intraPredModeLeft，intraPredModeAbove)，PredMode1 等于 max(intraPredModeLeft，intraPredModeAbove)。

1)　当 PredMode0 等于 PredMode1 时，

——如果 PredMode0 大于 PLAN _PRED，预测列表的构建按如下方式进行：

intra_candlis[0]＝PredMode0

intra_candlis[1]＝DC_PRED

intra_candlis[2]＝TM_PRED

intra_candlis[3]＝((PredMode0＋31) % 34)＋2

intra_candlis[4]＝((PredMode0－1) % 34)＋2

并且令 mpm_contex_idx＝0。

——否则，按如下方式进行：

intra_candlis[0]＝PredMode0

intra_candlis[1]＝PredMode0＝＝DC_PRED ? TM _PRED：DC_PRED

intra_candlis[2]＝PredMode0＞ TM _PRED ? TM _PRED：PLAN _PRED

intra_candlis[3]＝V_PRED

intra_candlis[4]＝H_PRED

并且令 mpm_contex_idx＝1。

2)　当 PredMode0 不等于 PredMode1 时，

——如果 PredMode0 大于 PLAN_PRED(即上边块和左边块都为角度预测时)：

if(PredMode1－PredMode0 ＜4){

```
        intra_candlis[0]=(PredMode0+PredMode1)>>1
        intra_candlis[1]=intra_candlis[0]-2
        intra_candlis[2]=intra_candlis[0]-1
        intra_candlis[3]=intra_candlis[0]+1
        intra_candlis[4]=PredMode1==ANG_36 ? H_PRED: intra_candlis[0]+2
        mpm_contex_idx=2
      }
      else if(PredMode1-PredMode0<12){
        intra_candlis[0]=(PredMode0+PredMode1)>>1
        intra_candlis[1]=PredMode0
        intra_candlis[2]=PredMode0+1
        intra_candlis[3]=PredMode1
        intra_candlis[4]=PredMode1-1
        mpm_contex_idx=3
      }
      else {
        intra_candlis[0]=PredMode0
        intra_candlis[1]=TM_PRED
        intra_candlis[2]=PredMode0+1
        intra_candlis[3]=PredMode1
        intra_candlis[4]=PredMode1-1
        mpm_contex_idx=4
      }
```

——否则，预测列表的构建按如下方式进行：

```
    if(PredMode1 > ANG_3) {
      intra_candlis[0]=PredMode0
      intra_candlis[1]=PredMode0==TM_PRED? DC_PRED:TM_PRED
      intra_candlis[2]=PredMode1-1
      intra_candlis[3]=PredMode1
      intra_candlis[4]=PredMode1==ANG_36 ? H_PRED:(((PredMode1-1) %35)+2)
      mpm_contex_idx=5
    }
    else {
      intra_candlis[0]=DC _PRED
      intra_candlis[1]=TM _PRED
      intra_candlis[2]=PLAN_PRED
      intra_candlis[3]=PredMode1>PLAN _PRED? PredMode1:V_PRED
      intra_candlis[4]=H_PRED
      mpm_contex_idx=6
    }
```

b) 计算 luma_intra_mode 的值：

如果码流中解析的 prev_intra_luma_pred_flag 的值为 1，则继续解析码流中的 mpm_idx0，如果 mpm_idx0 为 0，则 luma_intra_mode 等于 intra_candlis[0]，如果 mpm_idx0 为 1，则继续解

析码流中的 mpm_idx1，luma_intra_mode 等于 intra_candlis[1+mpm_idx1]。

如果码流中解析的 prev_intra_luma_pred_flag 的值为 0，则从码流中解析出 rem_pred_intra_mode，并对其值作如下调整：

遍历 i=0～4，如果 rem_pred_intra_mode 大于 intra_candlis[i]，则将 rem_pred_intra_mode 加 1，最后令 luma_intra_mode 等于 rem_pred_intra_mode。

根据 luma_intra_mode 的值，查表 39 得到亮度块帧内预测模式。

5.3.4.2.2 色度块的帧内预测模式

如果码流中解析的 uv_fllow_y_flag 的值为 1，则 chroma_intra_mode 等于 4；否则，从码流中解析得到 chroma_intra_mode。

根据 chroma_intra_mode 的值，查表 40 得到色度块帧内预测模式。

5.3.4.3 帧内预测时相邻块参考样点可用性判断

对于帧内预测，预测块尺寸与变换块尺寸绑定，因变换只有 N×N 的变换，因而预测块尺寸也为 N×N。相邻块可用性参见 5.1.3.3。

5.3.4.4 亮度参考样点的获取

对于 $N\times N$ 的亮度块，当前块上边的参考样点记为 $r[i]$，左边的参考样点记为 $c[j]$，其中 $r[0]=c[0]$。

用 I 表示当前块所在图像的补偿后(也就是滤波前)样点的亮度样值矩阵。

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

a) 初始化 $r[i]$，$c[j]$为 $2^{\text{bitdepth-1}}$，$i=0\sim 2N$，$j=0\sim 2N$；

b) 如果上边块可用，则 $r[i]=I[x_0+i-1,y_0-1]$，$i=1\sim N$，$r[i]$可用；否则 $r[i]$不可用；

c) 如果右上块可用，则 $r[i]=I[x_0+i-1,y_0-1]$，$i=N+1\sim 2N$，$r[i]$可用；否则 $r[i]$等于 $r[N]$，$r[i]$是否可用取决于 $r[N]$是否可用；

d) 如果左边块可用，则 $c[j]=I[x_0-1,y_0+j-1]$，$j=1\sim N$，$c[j]$可用；否则 $c[j]$不可用；

e) 如果左下块可用，则 $c[j]=I[x_0-1,y_0+j-1]$，$j=N+1\sim 2N$，$c[j]$可用；否则 $c[j]$等于 $c[N]$，$c[j]$是否可用取决于 $c[N]$是否可用；

f) 如果坐标为(x_0-1,y_0-1)的样点可用，则 $r[0]=I[x_0-1,y_0-1]$，$r[0]$可用，否则 $r[0]$不可用。

5.3.4.5 色度参考样点的获取

色度参考样点与亮度参考样点的获取方法一致，只是亮度块变为对应的色度块。

5.3.4.6 预测样点的计算

各帧内预测模式下的亮度块和色度块的预测样点矩阵 predMatrix 导出如下：

a) Horizontal_PRED

predMatrix[x,y]=c[y+1] (x=0～N−1,y=0～N−1)

b) Vertical_PRED

predMatrix[x,y]=r[x+1] (x=0～N−1,y=0～N−1)

c) TM_PRED

predMatrix[x,y]=clip(r[x+1]+c [y+1]−r [0]) (x=0～N−1,y=0～N−1)

d) DC_PRED

1) 如果 r[i],c[j](i=1～N,j=1～N)均可用,则

$predMatrix[x,y]=(\sum_{i=1}^{N}r[i]+\sum_{j=1}^{N}c[j]+N))/2N$ 其中($x=0\sim N-1, y=0\sim N-1$)

2) 否则,如果 r[i]可用(i 等于 1～N),则 $predMatrix[x,y]=(\sum_{i=1}^{N}r[i]+(N\gg 1))\gg \log_2(N)$ 其中($x=0\sim N-1, y=0\sim N-1$)

3) 否则,如果 c[j]可用(j 等于 1～N),则 $predMatrix[x,y]=(\sum_{i=1}^{N}c[j]+(N\gg 1))\gg \log_2(N)$ 其中($x=0\sim N-1, y=0\sim N-1$)

4) 否则, $predMatrix[x,y]=2^{\mathrm{BitDepth}-1}$ 其中($x=0\sim N-1, y=0\sim N-1$)

e) PLAN_PRED

ib_mult[5]={13,17,5,11,23}

ib_shift[5]={7,10,11,15,19}

ia=(r[N]+c[N])≪4

imh=ib_mult[$\log_2$(N)−2]

ish=ib_shift[$\log_2$(N)−2]

imv=ib_mult[$\log_2$(N)−2]

isv=ib_shift[$\log_2$(N)−2]

$ih=(\sum_{i=0}^{(N\gg 1)-1}(i+1)\times(r[(N\gg 1)+1+i]-r[(N\gg 1)-1-i])$

$iv=(\sum_{j=0}^{(N\gg 1)-1}(j+1)\times(c[(N\gg 1)+1+j]-c[(N\gg 1)-1-j])$

$ib=((ih\ll 5)\times imh+(1\ll(ish-1)))\gg ish$

$ic=((iv\ll 5)\times imv+(1\ll(isv-1)))\gg isv$

$predMatrix[x,y]=Clip1((ia+(x-(N\gg 1)+1)\times ib+(y-(N\gg 1)+1)\times ic+16)\gg 5)$

($x=0\sim N-1, y=0\sim N-1$)

f) ANG_PRED

初始化 r[−1]=c[1],r[−2]=c[2],c[−1]=r[1],c[−2]=r[2]。根据 luma_intra_mode 的值查表 41 得到角度预测模式的预测方向 dx、dy 以及 xyaxis、xysign、imx、isx、imy 和 isy。

dx=dirDx[luma_intra_mode]

dy=dirDy[luma_intra_mode]

xyaxis=dirXYflag[luma_intra_mode]

xysign=dirXYsign[luma_intra_mode]

imx=div_dxy[luma_intra_mode][0]

isx=div_dxy[luma_intra_mode][1]

imy=div_dyx[luma_intra_mode][0]

isy=div_dyx[luma_intra_mode][1]

1) 如果 xysign 小于 0,则

——如果 xyaxis 等于 0,则有

$offset=(((y+1)\times imx\times 32)\gg isx)-((y+1)\times imx\gg isx)$

$iX=x+((y+1)\times imx\gg isx)$

$iY=-1$

——如果 xyaxis 等于 1,则有

$offset=(((x+1)\times imy\times 32)\gg isy)-((x+1)\times imy\gg isy)$

$iX=-1$

$iY=y+((x+1)\times imy\rangle\rangle isy)$

2) 如果 xysign 大于 0,则

$offsetx=(((y+1)\times imx\times 32)\rangle\rangle isx)-((y+1)\times imx\rangle\rangle isx)$

$iXx=x((y+1)\times imx\rangle\rangle isx)$

$offsety=(((x+1)\times imy\times 32)\rangle\rangle isy)-((x+1)\times im\rangle\rangle isy)$

$iYy=y-((x+1)\times imy\rangle\rangle isy)$

——如果 iYy 小于或等于−1,offset=offsetx,iX=iXx,iY=−1;

——否则,offset=offsety,iX=−1,iY=iYy;

3) 如果 iY 等于−1:

——如果 dx×dy 小于 0,则 iXn=iX+1,iXnP2=iX+2,iXnN1=iX−1;

——如果 dx×dy 大于 0,则 iXn=iX−1,iXnP2=iX−2,iXnN1=iX+1;

$predMatrix[x,y]=(r[iXnN1+1]\times(32-offset)+r[iX+1]\times(64-offset)+r[iXn+1]\times(32+offset)+r[iXnP2+1]\times offset+64)\rangle\rangle 7,(x=0\sim N-1,y=0\sim N-1)$

4) 如果 iX 等于−1:

——如果 dx×dy 小于 0,则 iYn=iY+1,iYnP2=iY+2,iYnN1=iY−1;

——如果 dx×dy 大于 0,则 iYn=iY−1,iYnP2=iY−2,iYnN1=iY+1;

$predMatrix[x,y]=(c[iYnN1+1]\times(32-offset)+c[iY+1]\times(64-offset)+c[iYn+1]\times(32+offset)+c[iYnP2+1]\times offset+64)\rangle\rangle 7,(x=0\sim N-1,y=0\sim N-1)$

表 41 角度预测模式对应的方向参数

luma_intra_mode	dirDx	dirDy	dirXYflag	dirXYsign	div_dxy	div_dyx
DC	—	—	—	—	—	—
TM	—	—	—	—	—	—
PLAN	—	—	—	—	—	—
3	11	−4	0	−1	{11,2}	{93,8}
4	2	−1	0	−1	{2,0}	{1,1}
5	11	−8	0	−1	{11,3}	{93,7}
6	1	−1	0	−1	{1,0}	{1,0}
7	13	−16	0	−1	{13,4}	{315,8}
8	5	−8	0	−1	{5,3}	{13,3}
9	7	−16	0	−1	{7,4}	{9,2}
10	5	−16	0	−1	{5,4}	{205,6}
11	3	−16	0	−1	{3,4}	{171,5}
12	3	−32	0	−1	{3,5}	{171,4}
Ver	—	—	—	—	—	—
14	3	32	0	1	{3,5}	{171,4}
15	3	16	0	1	{3,4}	{171,5}
16	5	16	0	1	{5,4}	{205,6}
17	7	16	0	1	{7,4}	{9,2}
18	5	8	0	1	{5,3}	{13,3}

表 41（续）

luma_intra_mode	dirDx	dirDy	dirXYflag	dirXYsign	div_dxy	div_dyx
19	13	16	0	1	{13,4}	{315,8}
20	1	1	0	1	{1,0}	{1,0}
21	16	13	0	1	{315,8}	{13,4}
22	8	5	0	1	{13,3}	{5,3}
23	16	7	0	1	{9,2}	{7,4}
24	16	5	0	1	{205,6}	{5,4}
25	16	3	0	1	{171,5}	{3,4}
26	32	3	0	1	{171,4}	{3,5}
Hor	—	—	—	—	—	—
28	32	−3	1	−1	{171,4}	{3,5}
29	16	−3	1	−1	{171,5}	{3,4}
30	16	−5	1	−1	{205,6}	{5,4}
31	16	−7	1	−1	{9,2}	{7,4}
32	8	−5	1	−1	{13,3}	{5,3}
33	16	−13	1	−1	{315,8}	{13,4}
34	1	−1	1	−1	{1,0}	{1,0}
35	8	−11	1	−1	{93,7}	{11,3}
36	1	−2	1	−1	{1,1}	{2,0}

5.3.5 帧间预测过程

5.3.5.1 概述

本过程的输出为当前预测单元的帧间预测样点矩阵 predMatrix，包含一个亮度矩阵 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 候选亮度运动矢量集导出

候选运动矢量 PMV[0]和 PMV[1]初始设置为{0,0}。

按下述步骤进行搜索，满足条件的运动矢量加入候选运动矢量集。其中，第一个满足条件的 MV（mvx，mvy）为候选运动矢量 PMV[0]，第二个满足条件且不同于 PMV[0]的运动矢量为 PMV[1]。当 PMV[1]得到或下述所有步骤执行完成后，本过程结束。

第一步，按顺序搜索候选位置（Xc(i)，Yc(i)）(i=0～7)的 8×8 块，如果该位置在图像内、且该块使用的某一参考帧与当前块的参考帧相同，则该块对应参考帧的 MV 加入候选运动矢量集。当所有候选位置均被搜完后，进入第二步。

其中，候选块相对于当前块（X0，Y0）的位置根据当前块的 sub_size 查 mv_ref_blocks 得到。

mv_ref_blocks[BLOCK_SIZES][MVREF_NEIGHBOURS]=

```
{
    {{-1, 0}, {0,-1}, {-1,-1}, {-2, 0}, {0,-2}, {-2,-1}, {-1,-2}, {-2,-2}},
    {{-1, 0}, {0,-1}, {-1,-1}, {-2, 0}, {0,-2}, {-2,-1}, {-1,-2}, {-2,-2}},
    {{-1, 0}, {0,-1}, {-1,-1}, {-2, 0}, {0,-2}, {-2,-1}, {-1,-2}, {-2,-2}},
    {{-1, 0}, {0,-1}, {-1,-1}, {-2, 0}, {0,-2}, {-2,-1}, {-1,-2}, {-2,-2}},
    {{0,-1}, {-1, 0}, {1,-1}, {-1,-1}, {0,-2}, {-2, 0}, {-2,-1}, {-1,-2}},
    {{-1, 0}, {0,-1}, {-1, 1}, {-1,-1}, {-2, 0}, {0,-2}, {-1,-2}, {-2,-1}},
    {{-1, 0}, {0,-1}, {-1, 1}, {1,-1}, {-1,-1}, {-3, 0}, {0,-3}, {-3,-3}},
    {{0,-1}, {-1, 0}, {2,-1}, {-1,-1}, {-1, 1}, {0,-3}, {-3, 0}, {-3,-3}},
    {{-1, 0}, {0,-1}, {-1, 2}, {-1,-1}, {1,-1}, {-3, 0}, {0,-3}, {-3,-3}},
    {{-1, 1}, {1,-1}, {-1, 2}, {2,-1}, {-1,-1}, {-3, 0}, {0,-3}, {-3,-3}},
    {{0,-1}, {-1, 0}, {4,-1}, {-1, 2}, {-1,-1}, {0,-3}, {-3, 0}, {2,-1}},
    {{-1, 0}, {0,-1}, {-1, 4}, {2,-1}, {-1,-1}, {-3, 0}, {0,-3}, {-1, 2}},
    {{-1, 3}, {3,-1}, {-1, 4}, {4,-1}, {-1,-1}, {-1, 0}, {0,-1}, {-1, 6}}
}
```

候选块的位置计算如下：

(Xc(i),Yc(i)) =(X0,Y0)+ mv_ref_blocks[sub_size][i]×8；

其中，如果 i 小于 2 且当前块的 sub_size 小于 3，则取候选位置块中对应位置的块的 MV 作为候选 MV。

第二步，如果解码顺序上前一帧中，与当前块相同位置的块使用的某一参考帧，与当前块的参考帧相同，则该块对应参考帧的 MV 加入候选运动矢量集。

第三步，按顺序搜索候选位置(Xc(i)，Yc(i))(i=0～7)的 8×8 块，如果该位置在图像内且该块使用的某一参考帧与当前块的参考帧不同，则该块对应参考帧的 MV 加入候选运动矢量集。如果该参考帧与当前块参考帧的方向不同，则该块的 MV 符号取反后(－mvx，－mvy)加入候选运动矢量集。当所有候选位置均被搜完后，进入第四步。其中，候选位置与第一步相同。

第四步，如果解码顺序上前一帧中，与当前块相同位置的块使用的某一参考帧，与当前块的参考帧不同，则该块对应参考帧的 MV 加入候选运动矢量集。如果该参考帧与当前块参考帧的方向不同，则该块的 MV 符号取反后(－mvx，－mvy)加入候选运动矢量集。

5.3.5.2.2 亮度运动矢量导出

如果当前块的 skip_flag 等于 1，则当前块的 MV 为{0，0}，对应参考帧为 DYNAMIC_REF。否则：

如果当前块的 block_reference_mode 等于 SINGLE_REFERENCE，则：

a) 如果当前块的 mv_mode 为 ZEROMV，则当前块的 MV 为{0，0}；

b) 如果当前块的 mv_mode 为 NEARESTMV，则当前块的 MV 为 PMV[0]；

c) 如果当前块的 mv_mode 为 NEARMV，则当前块的 MV 为 PMV[1]；

d) 如果当前块的 mv_mode 为 NEWMV，则当前块的 MV 为 MVP[0]＋MVD[0]。

如果当前块的 block_reference_mode 等于 COMPOUND_REFERENCE，则当前块处于双向预测模式，帧间预测有两个参考帧，其中第一参考帧从码流中读取，第二参考帧固定为 OPTIONAL_REF。两个参考帧中依次导出两个运动矢量，分别记为 MV[0]，MV[1]，计算如下：

a) 如果当前块的 mv_mode 为 ZEROMV，则 MV[0]和 MV[1]均为{0，0}；

b) 如果当前块的 mv_mode 为 NEARESTMV，则 MV[0]和 MV[1]均为 PMV[0]；

c) 如果当前块的 mv_mode 为 NEARMV，则 MV[0]和 MV[1]均为 PMV[1]；

d) 如果当前块的 mv_mode 为 NEWMV，则 MV[0]和 MV[1]分别为：

$$MV[0] = PMV[0] + MVD[0]$$
$$MV[1] = PMV[1] + MVD[1]$$

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

图 12 给出了参考图像亮度分量整数样点、1/2 样点、1/4 样点和 1/8 样点的位置示意图，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 1/2、1/4 和 1/8 样点位置。亮度分量分数样点的值由 8 抽头滤波器生成。滤波器分为四种类型，分别为 Regular，Smooth-1，Sharp，Smooth-2，其对应的抽头系数由表 42～表 45 给出。当 interp_filter_switchale 等于 0 时，亮度样点插值采用的滤波器类型为 interp_filter；当 interp_filter_switchale 等于 1 时，亮度样点插值采用的滤波器类型由 interp_filter_mode 的取值决定。

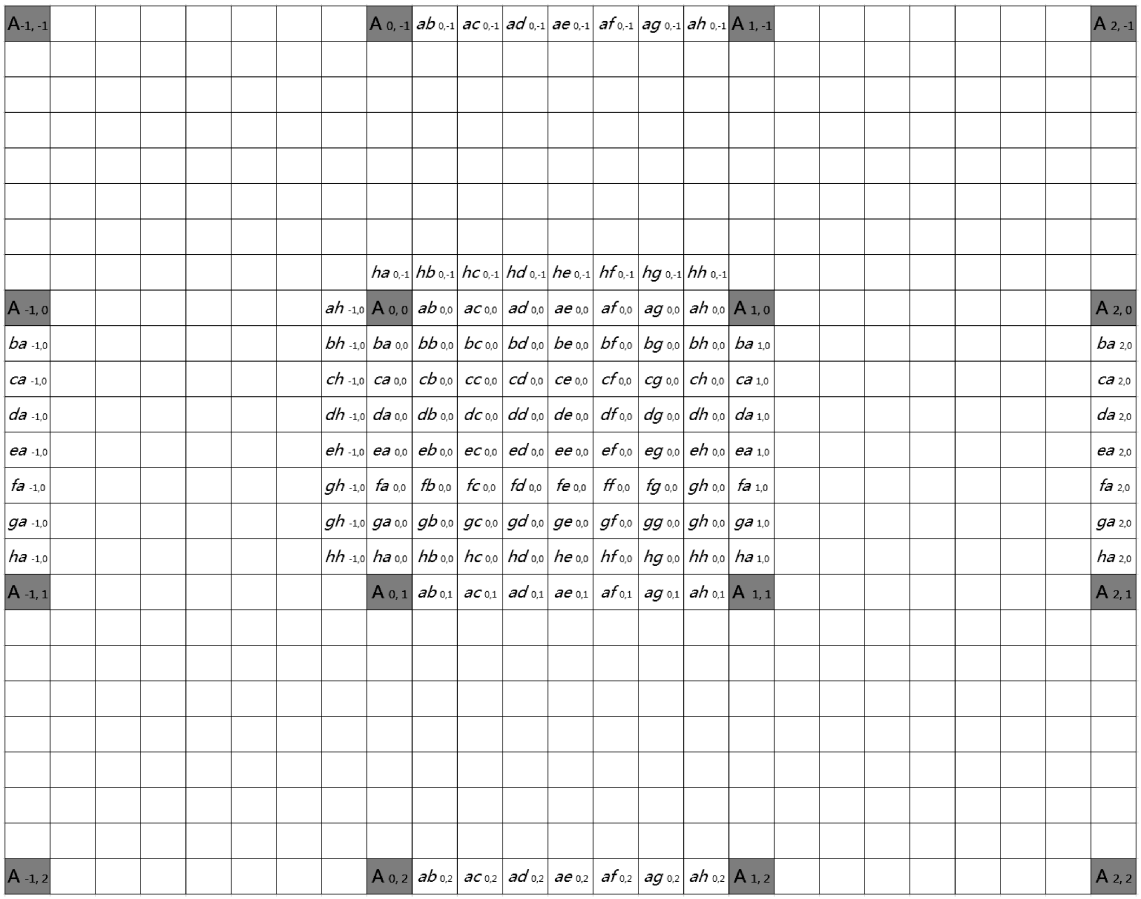


图 12 亮度分量整数样点、1/2、1/4 和 1/8 样点的位置

表 42 Regular 亮度插值滤波器抽头系数

分数样点位置	抽头系数(Regular 类型)
1/8	−1， 3，−9，121， 19，−7， 2， 0
2/8	−1， 4，−15，111， 38，−12， 4，−1
3/8	−1， 5，−18， 96， 59，−17， 5，−1

表 42（续）

分数样点位置	抽头系数(Regular 类型)
4/8	−1， 6,−19， 78， 78,−19， 6,−1
5/8	−1， 5,−17， 59， 96,−18， 5,−1
6/8	−1， 4,−12， 38，111,−15， 4,−1
7/8	0， 2，−7， 19，121，−9， 3,−1

表 43　Smooth-1 亮度插值滤波器抽头系数

分数样点位置	抽头系数(Smooth1 类型)
1/8	−2,−1， 28， 62， 42， 1,−2， 0
2/8	−2,−2， 23， 61， 47， 4,−3， 0
3/8	−1,−3， 17， 58， 52， 8,−3， 0
4/8	−1,−3， 13， 55， 55，13,−3,−1
5/8	0,−3， 8， 52， 58，17,−3,−1
6/8	0,−3， 4， 47， 61，23,−2,−2
7/8	0,−2， 1， 42， 62，28,−1,−2

表 44　Sharp 亮度插值滤波器抽头系数

分数样点位置	抽头系数(Sharp 类型)
1/8	−2， 5,−14，126， 18， −7， 3,−1
2/8	−4， 9,−21，116， 38,−14， 6,−2
3/8	−4，10,−25，101， 60,−20， 9,−3
4/8	−4，11,−24， 81， 81,−24， 11,−4
5/8	−3， 9,−20， 60，101,−25， 10,−4
6/8	−2， 6,−14， 38，116,−21， 9,−4
7/8	−1， 3， −7， 18，126,−14， 5,−2

表 45　Smooth-2 亮度插值滤波器抽头系数

分数样点位置	抽头系数(Smooth2 类型)
1/8	−1， 7，29，46,36，12， 0,−1
2/8	−1， 5，26，46，38，14， 1,−1
3/8	−1， 4，23，44，41，17， 1,−1
4/8	−1， 2，20，43，43，20， 2,−1
5/8	−1， 1，17，41，44，23， 4,−1
6/8	−1， 1，14，38，46，26， 5,−1
7/8	−1， 0，12，36，46，29， 7,−1

运动补偿插值滤波器包含四种不同类型的系数，其插值的方法都是相同的，因此下面均参照Regular类型抽头系数来描述，具体计算过程如下：

亮度分量样点位置 $ab_{0,0}$，$ac_{0,0}$，$ad_{0,0}$，$ae_{0,0}$的预测值由水平方向距离插值点最近的8个整数值滤波得到，计算方法如下：

$$ab'_{0,0}=-A_{-3,0}+3\times A_{-2,0}-9\times A_{-1,0}+121\times A_{0,0}+19\times A_{1,0}-7\times A_{2,0}+2\times A_{3,0}$$

$$ac'_{0,0}=-A_{-3,0}+4\times A_{-2,0}-15\times A_{-1,0}+111\times A_{0,0}+38\times A_{1,0}-12\times A_{2,0}+4\times A_{3,0}-A_{4,0}$$

$$ad'_{0,0}=-A_{-3,0}+5\times A_{-2,0}-18\times A_{-1,0}+96\times A_{0,0}+59\times A_{1,0}-17\times A_{2,0}+5\times A_{3,0}-A_{4,0}$$

$$ae'_{0,0}=-A_{-3,0}+6\times A_{-2,0}-19\times A_{-1,0}+78\times A_{0,0}+78\times A_{1,0}-19\times A_{2,0}+6\times A_{3,0}-A_{4,0}$$

$ab_{0,0}$，$ac_{0,0}$，$ad_{0,0}$，$ae_{0,0}$最终预测值计算方法如下：

$$ab_{0,0}=\text{Clip1}((ab'_{0,0}+64)\gg 7)$$

$$ac_{0,0}=\text{Clip1}((ac'_{0,0}+64)\gg 7)$$

$$ad_{0,0}=\text{Clip1}((ad'_{0,0}+64)\gg 7)$$

$$ae_{0,0}=\text{Clip1}((ae'_{0,0}+64)\gg 7)$$

其中，样点位置 $af_{0,0}$，$ag_{0,0}$，$ah_{0,0}$的预测方式同与样点位置 $ab_{0,0}$，$ac_{0,0}$，$ad_{0,0}$的预测方法类似，均采用对应位置的插值系数计算获得。

亮度分量样点位置 $ba_{0,0}$，$ca_{0,0}$，$da_{0,0}$，$ea_{0,0}$的预测值由垂直方向距离插值点最近的8个整数值滤波得到，计算方法如下：

$$ba'_{0,0}=-A_{0,-3}+3\times A_{0,-2}-9\times A_{0,-1}+121\times A_{0,0}+19\times A_{0,1}-7\times A_{0,2}+2\times A_{0,3}$$

$$ca'_{0,0}=-A_{0,-3}+4\times A_{0,-2}-15\times A_{0,-1}+111\times A_{0,0}+38\times A_{0,1}-12\times A_{0,2}+4\times A_{0,3}-A_{0,4}$$

$$da'_{0,0}=-A_{0,-3}+5\times A_{0,-2}-18\times A_{0,-1}+96\times A_{0,0}+59\times A_{0,1}-17\times A_{0,2}+5\times A_{0,3}-A_{0,4}$$

$$ea'_{0,0}=-A_{0,-3}+6\times A_{0,-2}-19\times A_{0,-1}+78\times A_{0,0}+78\times A_{0,1}-19\times A_{0,2}+6\times A_{0,3}-A_{0,4}$$

$ba_{0,0}$，$ca_{0,0}$，$da_{0,0}$，$ea_{0,0}$的最终预测值应通过下列式子得到：

$$ba_{0,0}=\text{Clip1}((ba'_{0,0}+64)\gg 7)$$

$$ca_{0,0}=\text{Clip1}((ca'_{0,0}+64)\gg 7)$$

$$da_{0,0}=\text{Clip1}((da'_{0,0}+64)\gg 7)$$

$$ea_{0,0}=\text{Clip1}((ea'_{0,0}+64)\gg 7)$$

其中，样点位置 $fa_{0,0}$，$ga_{0,0}$，$ha_{0,0}$的预测方式同与样点位置 $ba_{0,0}$，$ca_{0,0}$，$da_{0,0}$的预测方式类似，均采用对应位置的插值系数计算获得。

其他亮度分量的分数样点，如：$bb_{0,0}$，$bc_{0,0}$…$bh_{0,0}$，…$hb_{0,0}$，$hc_{0,0}$，…$hh_{0,0}$，需要使用整行的分数样点值（$ab'_{0,0}$，$ac'_{0,0}$，$ad'_{0,0}$，$ae'_{0,0}$，$af'_{0,0}$，$ag'_{0,0}$，$ah'_{0,0}$）来计算得出，计算方法如下：

$$bb'_{0,0}=-ab'_{0,-3}+3\times ab'_{0,-2}-9\times ab'_{0,-1}+121\times ab'_{0,0}+19\times ab'_{0,1}-7\times ab'_{0,2}+2\times ab'_{0,3}$$

$$cc'_{0,0}=-ac'_{0,-3}+4\times ac'_{0,-2}-15\times ac'_{0,-1}+111\times ac'_{0,0}+38\times ac'_{0,1}-12\times ac'_{0,2}+4\times ac'_{0,3}-ac'_{0,4}$$

$$ee'_{0,0}=-ae'_{0,-3}+6\times ae'_{0,-2}-19\times ae'_{0,-1}+78\times ae'_{0,0}+78\times ae'_{0,1}-19\times ae'_{0,2}+6\times ae'_{0,3}-ae'_{0,4}$$

$$hh'_{0,0}=2\times ah'_{0,-2}-7\times ah'_{0,-1}+19\times ah'_{0,0}+121\times ah'_{0,1}-9\times ah'_{0,2}+3\times ah'_{0,3}-ah'_{0,4}$$

$bb_{0,0}$，$cc_{0,0}$，$ee_{0,0}$，$hh_{0,0}$最终预测值计算方法如下：

$$bb_{0,0}=\text{Clip1}((bb'+8192)\gg 14)$$

$$cc_{0,0}=\text{Clip1}((cc'+8192)\gg 14)$$

$$ee_{0,0}=\text{Clip1}((ee'+8192)\gg 14)$$

$$hh_{0,0}=\text{Clip1}((hh'+8192)\gg 14)$$

同样，其余亮度样点位置的预测方式也类似，均采用对应位置的插值系数计算获得。

5.3.5.3.2 色度样点插值过程

图13给出了参考图像色度分量整数样点、1/2样点、1/4样点、1/8和1/16样点的位置示意图。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 1/2、1/4、1/8 和 1/16 样点位置。色度分量分数样点的值由 8 抽头滤波器生成。滤波器分为四种类型，分别为 Regular，Smooth-1，Sharp，Smooth-2，其对应的抽头系数由表 46～表 49 给出。色度样点插值采用的滤波器类型与对应位置的亮度样点插值滤波器相同。

	pa 0,-1	pb 0,-1	pc 0,-1	pd 0,-1	pe 0,-1	pf 0,-1	pg 0,-1	ph 0,-1	pi 0,-1	pj 0,-1	pk 0,-1	pl 0,-1	pm 0,-1	pn 0,-1	po 0,-1	pp 0,-1	
ap -1,0	B 0, 0	ab 0,0	ac 0,0	ad 0,0	ae 0,0	af 0,0	ag 0,0	ah 0,0	ai 0,0	aj 0,0	ak 0,0	al 0,0	am 0,0	an 0,0	ao 0,0	ap 0,0	B 1, 0
bp -1,0	ba 0,0	bb 0,0	bc 0,0	bd 0,0	be 0,0	bf 0,0	bg 0,0	bh 0,0	bi 0,0	bj 0,0	bk 0,0	bl 0,0	bm 0,0	bn 0,0	bo 0,0	bp 0,0	ba 1,0
cp -1,0	ca 0,0	cb 0,0	cc 0,0	cd 0,0	ce 0,0	cf 0,0	cg 0,0	ch 0,0	ci 0,0	cj 0,0	ck 0,0	cl 0,0	cm 0,0	cn 0,0	co 0,0	cp 0,0	ca 1,0
dp -1,0	da 0,0	db 0,0	dc 0,0	dd 0,0	de 0,0	df 0,0	dg 0,0	dh 0,0	di 0,0	dj 0,0	dk 0,0	dl 0,0	dm 0,0	dn 0,0	do 0,0	dp 0,0	da 1,0
ep -1,0	ea 0,0	eb 0,0	ec 0,0	ed 0,0	ee 0,0	ef 0,0	eg 0,0	eh 0,0	ei 0,0	ej 0,0	ek 0,0	el 0,0	em 0,0	en 0,0	eo 0,0	ep 0,0	ea 1,0
fp -1,0	fa 0,0	fb 0,0	fc 0,0	fd 0,0	fe 0,0	ff 0,0	fg 0,0	fh 0,0	fi 0,0	fj 0,0	fk 0,0	fl 0,0	fm 0,0	fn 0,0	fo 0,0	fp 0,0	fa 1,0
gp -1,0	ga 0,0	gb 0,0	gc 0,0	gd 0,0	ge 0,0	gf 0,0	gg 0,0	gh 0,0	gi 0,0	gj 0,0	gk 0,0	gl 0,0	gm 0,0	gn 0,0	go 0,0	gp 0,0	ga 1,0
hp -1,0	ha 0,0	hb 0,0	hc 0,0	hd 0,0	he 0,0	hf 0,0	hg 0,0	hh 0,0	hi 0,0	hj 0,0	hk 0,0	hl 0,0	hm 0,0	hn 0,0	ho 0,0	hp 0,0	ha 1,0
ip -1,0	ia 0,0	ib 0,0	ic 0,0	id 0,0	ie 0,0	if 0,0	ig 0,0	ih 0,0	ii 0,0	ij 0,0	ik 0,0	il 0,0	im 0,0	in 0,0	io 0,0	ip 0,0	ia 1,0
jp -1,0	ja 0,0	jb 0,0	jc 0,0	jd 0,0	je 0,0	jf 0,0	jg 0,0	jh 0,0	ji 0,0	jj 0,0	jk 0,0	jl 0,0	jm 0,0	jn 0,0	jo 0,0	jp 0,0	ja 1,0
kp -1,0	ka 0,0	kb 0,0	kc 0,0	kd 0,0	ke 0,0	kf 0,0	kg 0,0	kh 0,0	ki 0,0	kj 0,0	kk 0,0	kl 0,0	km 0,0	kn 0,0	ko 0,0	kp 0,0	ka 1,0
lp -1,0	la 0,0	lb 0,0	lc 0,0	ld 0,0	le 0,0	lf 0,0	lg 0,0	lh 0,0	li 0,0	lj 0,0	lk 0,0	ll 0,0	lm 0,0	ln 0,0	lo 0,0	lp 0,0	la 1,0
mp -1,0	ma 0,0	mb 0,0	mc 0,0	md 0,0	me 0,0	mf 0,0	mg 0,0	mh 0,0	mi 0,0	mj 0,0	mk 0,0	ml 0,0	mm 0,0	mn 0,0	mo 0,0	mp 0,0	ma 1,0
np -1,0	na 0,0	nb 0,0	nc 0,0	nd 0,0	ne 0,0	nf 0,0	ng 0,0	nh 0,0	ni 0,0	nj 0,0	nk 0,0	nl 0,0	nm 0,0	nn 0,0	no 0,0	np 0,0	na 1,0
op -1,0	oa 0,0	ob 0,0	oc 0,0	od 0,0	oe 0,0	of 0,0	og 0,0	oh 0,0	oi 0,0	oj 0,0	ok 0,0	ol 0,0	om 0,0	on 0,0	oo 0,0	op 0,0	oa 1,0
pp -1,0	pa 0,0	pb 0,0	pc 0,0	pd 0,0	pe 0,0	pf 0,0	pg 0,0	ph 0,0	pi 0,0	pj 0,0	pk 0,0	pl 0,0	pm 0,0	pn 0,0	po 0,0	pp 0,0	pa 1,0
	B 0, 1	ab 0,1	ac 0,1	ad 0,1	ae 0,1	af 0,1	ag 0,1	ah 0,1	ai 0,1	aj 0,1	ak 0,1	al 0,1	am 0,1	an 0,1	ao 0,1	ap 0,1	B 1, 1

图 13　色度分量整数样点、1/2、1/4、1/8 和 1/16 样点的位置

表 46　色度插值滤波器抽头系数-1

分数样点位置	抽头系数(Regular 类型)
1/16	0，　1，　−4，125，　9，　−4，　1，　0
2/16	−1，　3，　−9，121，19，　−7，　2，　0
3/16	−1，　4，−12，117，28，−10，　3，−1
4/16	−1，　4，−15，111，38，−12，　4，−1
5/16	−1，　5，−17，104，49，−15，　4，−1
6/16	−1，　5，−18，　96，59，−17，　5，−1
7/16	−1，　6，−18，　87，69，−19，　5，−1
8/16	−1，　6，−19，　78，78，−19，　6，−1
9/16	−1，　5，−19，　69，87，−18，　6，−1
10/16	−1，　5，−17，　59，96，−18，　5，−1
11/16	−1，　4，−15，　49，104，−17，　5，−1
12/16	−1，　4，−12，　38，111，−15，　4，−1

表 46（续）

分数样点位置	抽头系数(Regular 类型)
13/16	−1， 3,−10，28,117,−12， 4,−1
14/16	0， 2，−7，19,121，−9， 3,−1
15/16	0， 1，−4， 9,125，−4， 1， 0

表 47　色度插值滤波器抽头系数-2

分数样点位置	抽头系数(Smooth1 类型)
1/16	−3,−1， 31， 63， 39， 1,−2， 0
2/16	−2,−1， 28， 62， 42， 1,−2， 0
3/16	−2,−1， 25， 62， 44， 3,−3， 0
4/16	−2,−2， 23， 61， 47， 4,−3， 0
5/16	−2,−2， 20， 59， 50， 6,−3， 0
6/16	−1,−3， 17， 58， 52， 8,−3， 0
7/16	−1,−3， 15， 56， 54，11,−3,−1
8/16	−1,−3， 13， 55， 55，13,−3,−1
9/16	−1,−3， 11， 54， 56，15,−3,−1
10/16	0,−3， 8， 52， 58，17,−3,−1
11/16	0,−3， 6， 50， 59，20,−2,−2
12/16	0,−3， 4， 47， 61，23,−2,−2
13/16	0,−3， 3， 44， 62，25,−1,−2
14/16	0,−2， 1， 42， 62，28,−1,−2
15/16	0,−2， 1， 39， 63，31,−1,−3

表 48　色度插值滤波器抽头系数-3

分数样点位置	抽头系数(Sharp 类型)
1/16	−1， 3， −8，127， 10， −4， 1， 0
2/16	−2， 5,−14，126， 18， −7， 3,−1
3/16	−3， 7,−18，122， 28，−11， 5,−2
4/16	−4， 9,−21，116， 38，−14， 6,−2
5/16	−4，10,−24，109， 49，−17， 8,−3
6/16	−4，10,−25，101， 60，−20， 9,−3
7/16	−4，11,−25， 91， 71，−22，10,−4
8/16	−4，11,−24， 81， 81，−24，11,−4
9/16	−4，10,−22， 71， 91，−25，11,−4

表 48（续）

分数样点位置	抽头系数(Sharp 类型)
10/16	−3， 9,−20， 60，101，−25，10,−4
11/16	−3， 8,−17， 49，109，−24，10,−4
12/16	−2， 6,−14， 38，116，−21， 9,−4
13/16	−2， 5,−11， 28，122，−18， 7,−3
14/16	−1， 3，−7， 18，126，−14， 5,−2
15/16	0， 1，−4， 10，127， −8， 3,−1

表 49　色度插值滤波器抽头系数-4

分数样点位置	抽头系数(Smooth2 类型)
1/16	−1， 8,31,47,34,10， 0,−1
2/16	−1， 7,29,46,36,12， 0,−1
3/16	−1， 6,28,46,37,13， 0,−1
4/16	−1， 5,26,46,38,14， 1,−1
5/16	−1， 4,25,45,39,16， 1,−1
6/16	−1， 4,23,44,41,17， 1,−1
7/16	−1， 3,21,44,42,18， 2,−1
8/16	−1， 2,20,43,43,20， 2,−1
9/16	−1， 2,18,42,44,21， 3,−1
10/16	−1， 1,17,41,44,23， 4,−1
11/16	−1， 1,16,39,45,25， 4,−1
12/16	−1， 1,14,38,46,26， 5,−1
13/16	−1， 0,13,37,46,28， 6,−1
14/16	−1， 0,12,36,46,29， 7,−1
15/16	−1， 0,10,34,47,31， 8,−1

色度分量样点运动补偿插值方法与亮度分量类似，也是先对整数样点所在的行或列进行插值，再利用其结果对其余分数样点位置进行插值。同亮度插值一样，下面均参照 Regular 类型抽头系数来描述。例如，图 13 中的 $ah_{0,0}$，$ha_{0,0}$，$hh_{0,0}$ 的计算方法如下：

$ah'_{0,0}=-B_{-3,0}+6\times B_{-2,0}-18\times B_{-1,0}+87\times B_{0,0}+69\times B_{1,0}-19\times B_{2,0}+5\times B_{3,0}-B_{4,0}$

$ha'_{0,0}=-B_{0,-3}+6\times B_{0,-2}-18\times B_{0,-1}+87\times B_{0,0}+69\times A_{0,1}-19\times B_{0,2}+5\times B_{0,3}-B_{0,4}$

$ah_{0,0}$，$ha_{0,0}$ 最终预测值计算方法如下：

$ah_{0,0}=\mathrm{Clip1}((ah'_{0,0}+64)\gg 7)$

$ha_{0,0}=\mathrm{Clip1}((ha'_{0,0}+64)\gg 7)$

其他色度分量的分数样点，如：$bb_{0,0}$，$bc_{0,0}$…$bh_{0,0}$，…$hb_{0,0}$，$hc_{0,0}$，…$hh_{0,0}$，需要使用第一步中计算的整数样点位置的行的分数样点值（$ab'_{0,0}$，$ac'_{0,0}$，$ad'_{0,0}$，$ae'_{0,0}$，$af'_{0,0}$，$ag'_{0,0}$，$ah'_{0,0}$）来计算得出，计算方法如下：

$hh'_{0,0}=-ah'_{0,-3}+6\times ah'_{0,-2}-19\times ah'_{0,-1}+78\times ah'_{0,0}+78\times ah'_{0,1}-19\times ah'_{0,2}+6\times ah'_{0,3}-ah'_{0,4}$

$hh_{0,0}$的最终预测值计算方法如下：

$hh_{0,0}=\mathrm{Clip1}((hh'_{0,0}+8192)\gg 14)$

同样，其余色度样点位置的预测方式也类似，均采用对应位置的插值系数计算获得。

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

本过程在去块滤波过程之前进行，包括块系数解析，逆扫描，反量化，反变换和重建。

逆扫描的输入为由块解析生成的数组 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，当前块的量化参数 QP，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 或 16×16 或 32×32 变换系数矩阵 CoeffMatrix，输出为 4×4 或 8×8 或 16×16 或 32×32 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 块系数解析

设置 max_eob 为当前块中的样点个数，块系数数组 QuantCoeffArray 初始化为全 0，块系数索引 i 等于 0。块解析按如下步骤进行：

第一步，解析 coeff_value[i]，其过程见 5.4.2，如果该值为 EOB，则索引值大于等于 i 的块系数均为 0，块解析过程结束，否则进入下一步；

第二步，如果 coeff_value[i]等于 0，则 QuantCoeffArray[i]等于 0，块系数索引 i 增加 1，继续解析下一个 coeff_value[i]，直至解析到一个 coeff_value[i]大于 0 后进入下一步；

第三步，获得 coeff_sign[i]，如果 coeff_sign[i]等于 0，块系数 QuantCoeffArray[i]的取值等于 coeff_value[i]；如果 coeff_sign[i]等于 1，块系数 QuantCoeffArray[i]的取值等于－coeff_value[i]；

第四步，块系数索引 i 增加 1。如果当前块中的所有系数均已解析出，块解析过程结束，否则回到第一步。

5.3.6.3 逆扫描

设 QuantCoeffArray 数组中某元素的地址为 k，通过逆扫描找到地址为 k 的单元对应的列号 i 和行号 j，然后将 QuantCoeffArray[k]赋给 QuantCoeffMatrix[i,j]。根据块尺寸及编码模式的不同，扫描方式定义如图 14。

当编码块采用帧间编码或编码块为色度块时，根据块尺寸，分别采用 4×4 扫描、8×8 扫描、16×16 扫描和 32×32 扫描。

当编码块采用帧内编码且为亮度块时：

1) 若 luma_intra_mode 的取值大于等于 9 小于等于 17 时，根据块尺寸分别采用 4×4 行扫描、8×8 行扫描、16×16 行扫描和 32×32 扫描；
2) 若 luma_intra_mode 的取值大于等于 24 小于等于 31 时，根据块尺寸分别采用 4×4 列扫描、8×8 列扫描、16×16 列扫描和 32×32 扫描；
3) 否则，根据块尺寸，分别采用 4×4 扫描、8×8 扫描、16×16 扫描和 32×32 扫描。

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

a） 4×4 扫描

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

b） 4×4 行扫描

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

c） 4×4 列扫描

图 14 4×4、8×8、16×16、32×32 块的逆扫描

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

d） 8×8 扫描

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

e） 8×8 行扫描

图 14（续）

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

f） 8×8 列扫描

j \ i	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	0	16	1	32	17	2	48	33	18	3	64	34	49	19	65	80
1	50	4	35	66	20	81	96	51	5	36	82	97	67	112	21	52
2	98	37	83	113	6	68	128	53	22	99	114	84	7	129	38	69
3	100	115	144	130	85	54	23	8	145	39	70	116	101	131	160	146
4	55	86	24	71	132	117	161	40	9	102	147	176	162	87	56	25
5	133	118	177	148	72	103	41	163	10	192	178	88	57	134	149	119
6	26	164	73	104	193	42	179	208	11	135	89	165	120	150	58	194
7	180	27	74	209	105	151	136	43	90	224	166	195	181	121	210	59
8	12	152	106	167	196	75	137	225	211	240	182	122	91	28	197	13
9	226	168	183	153	44	212	138	107	241	60	29	123	198	184	227	169
10	242	76	213	154	45	92	14	199	139	61	228	214	170	185	243	108
11	77	155	30	15	200	229	124	215	244	93	46	186	171	201	109	140
12	230	62	216	245	31	125	78	156	231	47	187	202	217	94	246	141
13	63	232	172	110	247	157	79	218	203	126	233	188	248	95	173	142
14	219	111	249	234	158	127	189	204	250	235	143	174	220	205	159	251
15	190	221	175	236	237	191	206	252	222	253	207	238	223	254	239	255

g） 16×16 扫描

图 14（续）

j \ i	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	0	1	2	16	3	17	4	18	32	5	33	19	6	34	48	20
1	49	7	35	21	50	64	8	36	65	22	51	37	80	9	66	52
2	23	38	81	67	10	53	24	82	68	96	39	11	54	83	97	69
3	25	98	84	40	112	55	12	70	99	113	85	26	41	56	114	100
4	13	71	128	86	27	115	101	129	42	57	72	116	14	87	130	102
5	144	73	131	117	28	58	15	88	43	145	103	132	146	118	74	160
6	89	133	104	29	59	147	119	44	161	148	90	105	134	162	120	176
7	75	135	149	30	60	163	177	45	121	91	106	164	178	150	192	136
8	165	179	31	151	193	76	122	61	137	194	107	152	180	208	46	166
9	167	195	92	181	138	209	123	153	224	196	77	168	210	182	240	108
10	197	62	154	225	183	169	211	47	139	93	184	226	212	241	198	170
11	124	155	199	78	213	185	109	227	200	63	228	242	140	214	171	186
12	156	229	243	125	94	201	244	215	216	230	141	187	202	79	172	110
13	157	245	217	231	95	246	232	126	203	247	233	173	218	142	111	158
14	188	248	127	234	219	249	189	204	143	174	159	250	235	205	220	175
15	190	251	221	191	206	236	207	237	252	222	253	223	238	239	254	255

h） 16×16 行扫描

图 14（续）

j \ i	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	0	16	32	48	1	64	17	80	33	96	49	2	65	112	18	81
1	34	128	50	97	3	66	144	19	113	35	82	160	98	51	129	4
2	67	176	20	114	145	83	36	99	130	52	192	5	161	68	115	21
3	146	84	208	177	37	131	100	53	162	224	69	6	116	193	147	85
4	22	240	132	38	178	101	163	54	209	117	70	7	148	194	86	179
5	225	23	133	39	164	8	102	210	241	55	195	118	149	71	180	24
6	87	226	134	165	211	40	103	56	72	150	196	242	119	9	181	227
7	88	166	25	135	41	104	212	57	151	197	120	73	243	182	136	167
8	213	89	10	228	105	152	198	26	42	121	183	244	168	58	137	229
9	74	214	90	153	199	184	11	106	245	27	122	230	169	43	215	59
10	200	138	185	246	75	12	91	154	216	231	107	28	44	201	123	170
11	60	247	232	76	139	13	92	217	186	248	155	108	29	124	45	202
12	233	171	61	14	77	140	15	249	93	30	187	156	218	46	109	125
13	62	172	78	203	31	141	234	94	47	188	63	157	110	250	219	79
14	126	204	173	142	95	189	111	235	158	220	251	127	174	143	205	236
15	159	190	221	252	175	206	237	191	253	222	238	207	254	223	239	255

i） 16×16 列扫描

图 14（续）

j \ i	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
0	0	32	1	64	33	2	96	65	34	128	3	97	66	160	129	35	98	4	67	130	161	192	36	99	224	5	162	193	68	131	37	100
1	225	194	256	163	69	132	6	226	257	288	195	101	164	38	258	7	227	289	133	320	70	196	165	290	259	228	39	321	102	352	8	197
2	71	134	322	291	260	353	384	229	166	103	40	354	323	292	135	385	198	261	72	9	416	167	386	355	230	324	104	293	41	417	199	136
3	262	387	448	325	356	10	73	418	231	168	449	294	388	105	419	263	42	200	357	450	137	480	74	326	232	11	389	169	295	420	106	451
4	481	358	264	327	201	43	138	512	482	390	296	233	170	421	75	452	359	12	513	265	483	328	107	202	514	544	422	391	453	139	44	234
5	484	297	360	171	76	515	545	266	329	454	13	423	203	108	546	485	576	298	235	140	361	330	172	547	45	455	267	577	486	77	204	362
6	608	14	299	578	109	236	487	609	331	141	579	46	15	173	610	363	78	205	16	110	237	611	142	47	174	79	206	17	111	238	48	143
7	80	175	112	207	49	18	239	81	113	19	50	82	114	51	83	115	640	516	392	268	144	20	672	641	548	517	424	393	300	269	176	145
8	52	21	704	673	642	580	549	518	456	425	394	332	301	270	208	177	146	84	53	22	736	705	674	643	612	581	550	519	488	457	426	395
9	364	333	302	271	240	209	178	147	116	85	54	23	737	706	675	613	582	551	489	458	427	365	334	303	241	210	179	117	86	55	738	707
10	614	583	490	459	366	335	242	211	118	87	739	615	491	367	243	119	768	644	520	396	272	148	24	800	769	676	645	552	521	428	397	304
11	273	180	149	56	25	832	801	770	708	677	646	584	553	522	460	429	398	336	305	274	212	181	150	88	57	26	864	833	802	771	740	709
12	678	647	616	585	554	523	492	461	430	399	368	337	306	275	244	213	182	151	120	89	58	27	865	834	803	741	710	679	617	586	555	493
13	462	431	369	338	307	245	214	183	121	90	59	866	835	742	711	618	587	494	463	370	339	246	215	122	91	867	743	619	495	371	247	123
14	896	772	648	524	400	276	152	28	928	897	804	773	680	649	556	525	432	401	308	277	184	153	60	29	960	929	898	836	805	774	712	681
15	650	588	557	526	464	433	402	340	309	278	216	185	154	92	61	30	992	961	930	899	868	837	806	775	744	713	682	651	620	589	558	527
16	496	465	434	403	372	341	310	279	248	217	186	155	124	93	62	31	993	962	931	869	838	807	745	714	683	621	590	559	497	466	435	373
17	342	311	249	218	187	125	94	63	994	963	870	839	746	715	622	591	498	467	374	343	250	219	126	95	995	871	747	623	499	375	251	127
18	900	776	652	528	404	280	156	932	901	808	777	684	653	560	529	436	405	312	281	188	157	964	933	902	840	809	778	716	685	654	592	561
19	530	468	437	406	344	313	282	220	189	158	996	965	934	903	872	841	810	779	748	717	686	655	624	593	562	531	500	469	438	407	376	345
20	314	283	252	221	190	159	997	966	935	873	842	811	749	718	687	625	594	563	501	470	439	377	346	315	253	222	191	998	967	874	843	750
21	719	626	595	502	471	378	347	254	223	999	875	751	627	503	379	255	904	780	656	532	408	284	936	905	812	781	688	657	564	533	440	409
22	316	285	968	937	906	844	813	782	720	689	658	596	565	534	472	441	410	348	317	286	1000	969	938	907	876	845	814	783	752	721	690	659
23	628	597	566	535	504	473	442	411	380	349	318	287	1001	970	939	877	846	815	753	722	691	629	598	567	505	474	443	381	350	319	1002	971
24	878	847	754	723	630	599	506	475	382	351	1003	879	755	631	507	383	908	784	660	536	412	940	909	816	785	692	661	568	537	444	413	972
25	941	910	848	817	786	724	693	662	600	569	538	476	445	414	1004	973	942	911	880	849	818	787	756	725	694	663	632	601	570	539	508	477
26	446	415	1005	974	943	881	850	819	757	726	695	633	602	571	509	478	447	1006	975	882	851	758	727	634	603	510	479	1007	883	759	635	511
27	912	788	664	540	944	913	820	789	696	665	572	541	976	945	914	852	821	790	728	697	666	604	573	542	1008	977	946	915	884	853	822	791
28	760	729	698	667	636	605	574	543	1009	978	947	885	854	823	761	730	699	637	606	575	1010	979	886	855	762	731	638	607	1011	887	763	639
29	916	792	668	948	917	824	793	700	669	980	949	918	856	825	794	732	701	670	1012	981	950	919	888	857	826	795	764	733	702	671	1013	982
30	951	889	858	827	765	734	703	1014	983	890	859	766	735	1015	891	767	920	796	952	921	828	797	984	953	922	860	829	798	1016	985	954	923
31	892	861	830	799	1017	986	955	893	862	831	1018	987	894	863	1019	895	924	956	925	988	957	926	1020	989	958	927	1021	990	959	1022	991	1023

j） 32×32 扫描

图 14（续）

5.3.6.4 系数反量化

5.3.6.4.1 量化参数

量化参数 QPIndex 的取值应为 0～255。帧解码开始时，QPIndex 初始设置为 base_qindex。每个解码块的 QPIndex 计算如下：

QPIndex＝QPIndex＋dqp_sign ? dqp_abs:－dqp_abs

亮度 DC 系数量化参数 QPY_dc 以 QPIndex、y_dc_delta_q 与 BitDepthY 为索引查表 50～表 52 得到，QPY_dc 为查表得到的 QP_dc，其中 index＝Clip3(0,255,QPIndex＋y_dc_delta_q)。亮度 AC 系数量化参数 QPY_ac 以 QPIndex、y_ac_delta_q 与 BitDepthY 为索引查表 53～表 55 得到，QPY_ac 为查表得到的 QP_ac，其中 index＝Clip3(0,255,QPIndex)。色度 DC 系数量化参数 QPC_dc 以 QPIndex、uv_dc_delta_q 与 BitDepthY 为索引查表 50～表 52 得到，QPC_dc 为查表得到的 QP_dc，其中 index＝Clip3(0,255,QPIndex＋uv_dc_delta_q)。色度 AC 系数量化参数 QPC_ac 以 QPIndex、uv_ac_delta_q 与 BitDepthY 为索引查表 53～表 55 得到，QPC_ac 为查表得到的 QP_ac，其中 index＝Clip3(0,255,QPIndex＋uv_ac_delta_q)。

表 50　BitDepth＝8 时，QP_dc 与 index＝QPIndex＋dc_delta_q 的映射关系

index	0	1	2	3	4	5	6	7
QP_dc	4	8	8	9	10	11	12	12
index	8	9	10	11	12	13	14	15
QP_dc	13	14	15	16	17	18	19	19
index	16	17	18	19	20	21	22	23
QP_dc	20	21	22	23	24	25	26	26
index	24	25	26	27	28	29	30	31
QP_dc	27	28	29	30	31	32	32	33
index	32	33	34	35	36	37	38	39
QP_dc	34	35	36	37	38	38	39	40
index	40	41	42	43	44	45	46	47
QP_dc	41	42	43	43	44	45	46	47
index	48	49	50	51	52	53	54	55
QP_dc	48	48	49	50	51	52	53	53
index	56	57	58	59	60	61	62	63
QP_dc	54	55	56	57	57	58	59	60
index	64	65	66	67	68	69	70	71
QP_dc	61	62	62	63	64	65	66	66
index	72	73	74	75	76	77	78	79
QP_dc	67	68	69	70	70	71	72	73

表 50（续）

index	80	81	82	83	84	85	86	87
QP_dc	74	74	75	76	77	78	78	79
index	88	89	90	91	92	93	94	95
QP_dc	80	81	81	82	83	84	85	85
index	96	97	98	99	100	101	102	103
QP_dc	87	88	90	92	93	95	96	98
index	104	105	106	107	108	109	110	111
QP_dc	99	101	102	104	105	107	108	110
index	112	113	114	115	116	117	118	119
QP_dc	111	113	114	116	117	118	120	121
index	120	121	122	123	124	125	126	127
QP_dc	123	125	127	129	131	134	136	138
index	128	129	130	131	132	133	134	135
QP_dc	140	142	144	146	148	150	152	154
index	136	137	138	139	140	141	142	143
QP_dc	156	158	161	164	166	169	172	174
index	144	145	146	147	148	149	150	151
QP_dc	177	180	182	185	187	190	192	195
index	152	153	154	155	156	157	158	159
QP_dc	199	202	205	208	211	214	217	220
index	160	161	162	163	164	165	166	167
QP_dc	223	226	230	233	237	240	243	247
index	168	169	170	171	172	173	174	175
QP_dc	250	253	257	261	265	269	272	276
index	176	177	178	179	180	181	182	183
QP_dc	280	284	288	292	296	300	304	309
index	184	185	186	187	188	189	190	191
QP_dc	313	317	322	326	330	335	340	344
index	192	193	194	195	196	197	198	199
QP_dc	349	354	359	364	369	374	379	384
index	200	201	202	203	204	205	206	207
QP_dc	389	395	400	406	411	417	423	429

表 50（续）

index	208	209	210	211	212	213	214	215
QP_dc	435	441	447	454	461	467	475	482
index	216	217	218	219	220	221	222	223
QP_dc	489	497	505	513	522	530	539	549
index	224	225	226	227	228	229	230	231
QP_dc	559	569	579	590	602	614	626	640
index	232	233	234	235	236	237	238	239
QP_dc	654	668	684	700	717	736	755	775
index	240	241	242	243	244	245	246	247
QP_dc	796	819	843	869	896	925	955	988
index	248	249	250	251	252	253	254	255
QP_dc	1 022	1 058	1 098	1 139	1 184	1 232	1 282	1 336

表 51　BitDepth＝10 时，QP_dc 与 index＝QPIndex＋dc_delta_q 的映射关系

index	0	1	2	3	4	5	6	7
QP_dc	4	9	10	13	15	17	20	22
index	8	9	10	11	12	13	14	15
QP_dc	25	28	31	34	37	40	43	47
index	16	17	18	19	20	21	22	23
QP_dc	50	53	57	60	64	68	71	75
index	24	25	26	27	28	29	30	31
QP_dc	78	82	86	90	93	97	101	105
index	32	33	34	35	36	37	38	39
QP_dc	109	113	116	120	124	128	132	136
index	40	41	42	43	44	45	46	47
QP_dc	140	143	147	151	155	159	163	166
index	48	49	50	51	52	53	54	55
QP_dc	170	174	178	182	185	189	193	197
index	56	57	58	59	60	61	62	63
QP_dc	200	204	208	212	215	219	223	226
index	64	65	66	67	68	69	70	71
QP_dc	230	233	237	241	244	248	251	255

表 51（续）

index	72	73	74	75	76	77	78	79
QP_dc	259	262	266	269	273	276	280	283
index	80	81	82	83	84	85	86	87
QP_dc	287	290	293	297	300	304	307	310
index	88	89	90	91	92	93	94	95
QP_dc	314	317	321	324	327	331	334	337
index	96	97	98	99	100	101	102	103
QP_dc	343	350	356	362	369	375	381	387
index	104	105	106	107	108	109	110	111
QP_dc	394	400	406	412	418	424	430	436
index	112	113	114	115	116	117	118	119
QP_dc	442	448	454	460	466	472	478	484
index	120	121	122	123	124	125	126	127
QP_dc	490	499	507	516	525	533	542	550
index	128	129	130	131	132	133	134	135
QP_dc	559	567	576	584	592	601	609	617
index	136	137	138	139	140	141	142	143
QP_dc	625	634	644	655	666	676	687	698
index	144	145	146	147	148	149	150	151
QP_dc	708	718	729	739	749	759	770	782
index	152	153	154	155	156	157	158	159
QP_dc	795	807	819	831	844	856	868	880
index	160	161	162	163	164	165	166	167
QP_dc	891	906	920	933	947	961	975	988
index	168	169	170	171	172	173	174	175
QP_dc	1 001	1 015	1 030	1 045	1 061	1 076	1 090	1 105
index	176	177	178	179	180	181	182	183
QP_dc	1 120	1 137	1 153	1 170	1 186	1 202	1 218	1 236
index	184	185	186	187	188	189	190	191
QP_dc	1 253	1 271	1 288	1 306	1 323	1 342	1 361	1 379
index	192	193	194	195	196	197	198	199
QP_dc	1 398	1 416	1 436	1 456	1 476	1 496	1 516	1 537

表 51（续）

index	200	201	202	203	204	205	206	207
QP_dc	1 559	1 580	1 601	1 624	1 647	1 670	1 692	1 717
index	208	209	210	211	212	213	214	215
QP_dc	1 741	1 766	1 791	1 817	1 844	1 871	1 900	1 929
index	216	217	218	219	220	221	222	223
QP_dc	1 958	1 990	2 021	2 054	2 088	2 123	2 159	2 197
index	224	225	226	227	228	229	230	231
QP_dc	2 236	2 276	2 319	2 363	2 410	2 458	2 508	2 561
index	232	233	234	235	236	237	238	239
QP_dc	2 616	2 675	2 737	2 802	2 871	2 944	3 020	3 102
index	240	241	242	243	244	245	246	247
QP_dc	3 188	3 280	3 375	3 478	3 586	3 702	3 823	3 953
index	248	249	250	251	252	253	254	255
QP_dc	4 089	4 236	4 394	4 559	4 737	4 929	5 130	5 347

表 52　BitDepth＝12 时，QP_dc 与 index＝QPIndex＋dc_delta_q 的映射关系

index	0	1	2	3	4	5	6	7
QP_dc	4	12	18	25	33	41	50	60
index	8	9	10	11	12	13	14	15
QP_dc	70	80	91	103	115	127	140	153
index	16	17	18	19	20	21	22	23
QP_dc	166	180	194	208	222	237	251	266
index	24	25	26	27	28	29	30	31
QP_dc	281	296	312	327	343	358	374	390
index	32	33	34	35	36	37	38	39
QP_dc	405	421	437	453	469	484	500	516
index	40	41	42	43	44	45	46	47
QP_dc	532	548	564	580	596	611	627	643
index	48	49	50	51	52	53	54	55
QP_dc	659	674	690	706	721	737	752	768
index	56	57	58	59	60	61	62	63
QP_dc	783	798	814	829	844	859	874	889

表 52（续）

index	64	65	66	67	68	69	70	71
QP_dc	904	919	934	949	964	978	993	1 008
index	72	73	74	75	76	77	78	79
QP_dc	1 022	1 037	1 051	1 065	1 080	1 094	1 108	1 122
index	80	81	82	83	84	85	86	87
QP_dc	1 136	1 151	1 165	1 179	1 192	1 206	1 220	1 234
index	88	89	90	91	92	93	94	95
QP_dc	1 248	1 261	1 275	1 288	1 302	1 315	1 329	1 342
index	96	97	98	99	100	101	102	103
QP_dc	1 368	1 393	1 419	1 444	1 469	1 494	1 519	1 544
index	104	105	106	107	108	109	110	111
QP_dc	1 569	1 594	1 618	1 643	1 668	1 692	1 717	1 741
index	112	113	114	115	116	117	118	119
QP_dc	1 765	1 789	1 814	1 838	1 862	1 885	1 909	1 933
index	120	121	122	123	124	125	126	127
QP_dc	1 957	1 992	2 027	2 061	2 096	2 130	2 165	2 199
index	128	129	130	131	132	133	134	135
QP_dc	2 233	2 267	2 300	2 334	2 367	2 400	2 434	2 467
index	136	137	138	139	140	141	142	143
QP_dc	2 499	2 532	2 575	2 618	2 661	2 704	2 746	2 788
index	144	145	146	147	148	149	150	151
QP_dc	2 830	2 872	2 913	2 954	2 995	3 036	3 076	3 127
index	152	153	154	155	156	157	158	159
QP_dc	3 177	3 226	3 275	3 324	3 373	3 421	3 469	3 517
index	160	161	162	163	164	165	166	167
QP_dc	3 565	3 621	3 677	3 733	3 788	3 843	3 897	3 951
index	168	169	170	171	172	173	174	175
QP_dc	4 005	4 058	4 119	4 181	4 241	4 301	4 361	4 420
index	176	177	178	179	180	181	182	183
QP_dc	4 479	4 546	4 612	4 677	4 742	4 807	4 871	4 942
index	184	185	186	187	188	189	190	191
QP_dc	5 013	5 083	5 153	5 222	5 291	5 367	5 442	5 517

表 52（续）

index	192	193	194	195	196	197	198	199
QP_dc	5 591	5 665	5 745	5 825	5 905	5 984	6 063	6 149
index	200	201	202	203	204	205	206	207
QP_dc	6 234	6 319	6 404	6 495	6 587	6 678	6 769	6 867
index	208	209	210	211	212	213	214	215
QP_dc	6 966	7 064	7 163	7 269	7 376	7 483	7 599	7 715
index	216	217	218	219	220	221	222	223
QP_dc	7 832	7 958	8 085	8 214	8 352	8 492	8 635	8 788
index	224	225	226	227	228	229	230	231
QP_dc	8 945	9 104	9 275	9 450	9 639	9 832	10 031	10 245
index	232	233	234	235	236	237	238	239
QP_dc	10 465	10 702	10 946	11 210	11 482	11 776	12 081	12 409
index	240	241	242	243	244	245	246	247
QP_dc	12 750	13 118	13 501	13 913	14 343	14 807	15 290	15 812
index	248	249	250	251	252	253	254	255
QP_dc	16 356	16 943	17 575	18 237	18 949	19 718	20 521	21 387

表 53　BitDepth＝8 时，QP_ac 与 index＝QPIndex＋ac_delta_q 的映射关系

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

表 53（续）

index	56	57	58	59	60	61	62	63
QP_ac	63	64	65	66	67	68	69	70
index	64	65	66	67	68	69	70	71
QP_ac	71	72	73	74	75	76	77	78
index	72	73	74	75	76	77	78	79
QP_ac	79	80	81	82	83	84	85	86
index	80	81	82	83	84	85	86	87
QP_ac	87	88	89	90	91	92	93	94
index	88	89	90	91	92	93	94	95
QP_ac	95	96	97	98	99	100	101	102
index	96	97	98	99	100	101	102	103
QP_ac	104	106	108	110	112	114	116	118
index	104	105	106	107	108	109	110	111
QP_ac	120	122	124	126	128	130	132	134
index	112	113	114	115	116	117	118	119
QP_ac	136	138	140	142	144	146	148	150
index	120	121	122	123	124	125	126	127
QP_ac	152	155	158	161	164	167	170	173
index	128	129	130	131	132	133	134	135
QP_ac	176	179	182	185	188	191	194	197
index	136	137	138	139	140	141	142	143
QP_ac	200	203	207	211	215	219	223	227
index	144	145	146	147	148	149	150	151
QP_ac	231	235	239	243	247	251	255	260
index	152	153	154	155	156	157	158	159
QP_ac	265	270	275	280	285	290	295	300
index	160	161	162	163	164	165	166	167
QP_ac	305	311	317	323	329	335	341	347
index	168	169	170	171	172	173	174	175
QP_ac	353	359	366	373	380	387	394	401
index	176	177	178	179	180	181	182	183
QP_ac	408	416	424	432	440	448	456	465

表 53（续）

index	184	185	186	187	188	189	190	191
QP_ac	474	483	492	501	510	520	530	540
index	192	193	194	195	196	197	198	199
QP_ac	550	560	571	582	593	604	615	627
index	200	201	202	203	204	205	206	207
QP_ac	639	651	663	676	689	702	715	729
index	208	209	210	211	212	213	214	215
QP_ac	743	757	771	786	801	816	832	848
index	216	217	218	219	220	221	222	223
QP_ac	864	881	898	915	933	951	969	988
index	224	225	226	227	228	229	230	231
QP_ac	1 007	1 026	1 046	1 066	1 087	1 108	1 129	1 151
index	232	233	234	235	236	237	238	239
QP_ac	1 173	1 196	1 219	1 243	1 267	1 292	1 317	1 343
index	240	241	242	243	244	245	246	247
QP_ac	1 369	1 396	1 423	1 451	1 479	1 508	1 537	1 567
index	248	249	250	251	252	253	254	255
QP_ac	1 597	1 628	1 660	1 692	1 725	1 759	1 793	1 828

表 54　BitDepth＝10 时，QP_ac 与 index＝QPIndex＋ac_delta_q 的映射关系

index	0	1	2	3	4	5	6	7
QP_ac	4	9	11	13	16	18	21	24
index	8	9	10	11	12	13	14	15
QP_ac	27	30	33	37	40	44	48	51
index	16	17	18	19	20	21	22	23
QP_ac	55	59	63	67	71	75	79	83
index	24	25	26	27	28	29	30	31
QP_ac	88	92	96	100	105	109	114	118
index	32	33	34	35	36	37	38	39
QP_ac	122	127	131	136	140	145	149	154
index	40	41	42	43	44	45	46	47
QP_ac	158	163	168	172	177	181	186	190

表 54（续）

index	48	49	50	51	52	53	54	55
QP_ac	195	199	204	208	213	217	222	226
index	56	57	58	59	60	61	62	63
QP_ac	231	235	240	244	249	253	258	262
index	64	65	66	67	68	69	70	71
QP_ac	267	271	275	280	284	289	293	297
index	72	73	74	75	76	77	78	79
QP_ac	302	306	311	315	319	324	328	332
index	80	81	82	83	84	85	86	87
QP_ac	337	341	345	349	354	358	362	367
index	88	89	90	91	92	93	94	95
QP_ac	371	375	379	384	388	392	396	401
index	96	97	98	99	100	101	102	103
QP_ac	409	417	425	433	441	449	458	466
index	104	105	106	107	108	109	110	111
QP_ac	474	482	490	498	506	514	523	531
index	112	113	114	115	116	117	118	119
QP_ac	539	547	555	563	571	579	588	596
index	120	121	122	123	124	125	126	127
QP_ac	604	616	628	640	652	664	676	688
index	128	129	130	131	132	133	134	135
QP_ac	700	713	725	737	749	761	773	785
index	136	137	138	139	140	141	142	143
QP_ac	797	809	825	841	857	873	889	905
index	144	145	146	147	148	149	150	151
QP_ac	922	938	954	970	986	1 002	1 018	1 038
index	152	153	154	155	156	157	158	159
QP_ac	1 058	1 078	1 098	1 118	1 138	1 158	1 178	1 198
index	160	161	162	163	164	165	166	167
QP_ac	1 218	1 242	1 266	1 290	1 314	1 338	1 362	1 386
index	168	169	170	171	172	173	174	175
QP_ac	1 411	1 435	1 463	1 491	1 519	1 547	1 575	1 603

表 54（续）

index	176	177	178	179	180	181	182	183
QP_ac	1 631	1 663	1 695	1 727	1 759	1 791	1 823	1 859
index	184	185	186	187	188	189	190	191
QP_ac	1 895	1 931	1 967	2 003	2 039	2 079	2 119	2 159
index	192	193	194	195	196	197	198	199
QP_ac	2 199	2 239	2 283	2 327	2 371	2 415	2 459	2 507
index	200	201	202	203	204	205	206	207
QP_ac	2 555	2 603	2 651	2 703	2 755	2 807	2 859	2 915
index	208	209	210	211	212	213	214	215
QP_ac	2 971	3 027	3 083	3 143	3 203	3 263	3 327	3 391
index	216	217	218	219	220	221	222	223
QP_ac	3 455	3 523	3 591	3 659	3 731	3 803	3 876	3 952
index	224	225	226	227	228	229	230	231
QP_ac	4 028	4 104	4 184	4 264	4 348	4 432	4 516	4 604
index	232	233	234	235	236	237	238	239
QP_ac	4 692	4 784	4 876	4 972	5 068	5 168	5 268	5 372
index	240	241	242	243	244	245	246	247
QP_ac	5 476	5 584	5 692	5 804	5 916	6 032	6 148	6 268
index	248	249	250	251	252	253	254	255
QP_ac	6 388	6 512	6 640	6 768	6 900	7 036	7 172	7 312

表 55　BitDepth＝12 时，QP_ac 与 index＝QPIndex＋ac_delta_q 的映射关系

index	0	1	2	3	4	5	6	7
QP_ac	4	13	19	27	35	44	54	64
index	8	9	10	11	12	13	14	15
QP_ac	75	87	99	112	126	139	154	168
index	16	17	18	19	20	21	22	23
QP_ac	183	199	214	230	247	263	280	297
index	24	25	26	27	28	29	30	31
QP_ac	314	331	349	366	384	402	420	438
index	32	33	34	35	36	37	38	39
QP_ac	456	475	493	511	530	548	567	586

表 55（续）

index	40	41	42	43	44	45	46	47
QP_ac	604	623	642	660	679	698	716	735
index	48	49	50	51	52	53	54	55
QP_ac	753	772	791	809	828	846	865	884
index	56	57	58	59	60	61	62	63
QP_ac	902	920	939	957	976	994	1 012	1 030
index	64	65	66	67	68	69	70	71
QP_ac	1 049	1 067	1 085	1 103	1 121	1 139	1 157	1 175
index	72	73	74	75	76	77	78	79
QP_ac	1 193	1 211	1 229	1 246	1 264	1 282	1 299	1 317
index	80	81	82	83	84	85	86	87
QP_ac	1 335	1 352	1 370	1 387	1 405	1 422	1 440	1 457
index	88	89	90	91	92	93	94	95
QP_ac	1 474	1 491	1 509	1 526	1 543	1 560	1 577	1 595
index	96	97	98	99	100	101	102	103
QP_ac	1 627	1 660	1 693	1 725	1 758	1 791	1 824	1 856
index	104	105	106	107	108	109	110	111
QP_ac	1 889	1 922	1 954	1 987	2 020	2 052	2 085	2 118
index	112	113	114	115	116	117	118	119
QP_ac	2 150	2 183	2 216	2 248	2 281	2 313	2 346	2 378
index	120	121	122	123	124	125	126	127
QP_ac	2 411	2 459	2 508	2 556	2 605	2 653	2 701	2 750
index	128	129	130	131	132	133	134	135
QP_ac	2 798	2 847	2 895	2 943	2 992	3 040	3 088	3 137
index	136	137	138	139	140	141	142	143
QP_ac	3 185	3 234	3 298	3 362	3 426	3 491	3 555	3 619
index	144	145	146	147	148	149	150	151
QP_ac	3 684	3 748	3 812	3 876	3 941	4 005	4 069	4 149
index	152	153	154	155	156	157	158	159
QP_ac	4 230	4 310	4 390	4 470	4 550	4 631	4 711	4 791
index	160	161	162	163	164	165	166	167
QP_ac	4 871	4 967	5 064	5 160	5 256	5 352	5 448	5 544

表 55（续）

index	168	169	170	171	172	173	174	175
QP_ac	5 641	5 737	5 849	5 961	6 073	6 185	6 297	6 410
index	176	177	178	179	180	181	182	183
QP_ac	6 522	6 650	6 778	6 906	7 034	7 162	7 290	7 435
index	184	185	186	187	188	189	190	191
QP_ac	7 579	7 723	7 867	8 011	8 155	8 315	8 475	8 635
index	192	193	194	195	196	197	198	199
QP_ac	8 795	8 956	9 132	9 308	9 484	9 660	9 836	10 028
index	200	201	202	203	204	205	206	207
QP_ac	10 220	10 412	10 604	10 812	11 020	11 228	11 437	11 661
index	208	209	210	211	212	213	214	215
QP_ac	11 885	12 109	12 333	12 573	12 813	13 053	13 309	13 565
index	216	217	218	219	220	221	222	223
QP_ac	13 821	14 093	14 365	14 637	14 925	15 213	15 502	15 806
index	224	225	226	227	228	229	230	231
QP_ac	16 110	16 414	16 734	17 054	17 390	17 726	18 062	18 414
index	232	233	234	235	236	237	238	239
QP_ac	18 766	19 134	19 502	19 886	20 270	20 670	21 070	21 486
index	240	241	242	243	244	245	246	247
QP_ac	21 902	22 334	22 766	23 214	23 662	24 126	24 590	25 070
index	248	249	250	251	252	253	254	255
QP_ac	25 551	26 047	26 559	27 071	27 599	28 143	28 687	29 247

5.3.6.4.2 反量化

根据量化参数 QP(QPY_dc 或 QPY_ac 或 QPC_dc 或 QPC_ac)，将 4×4、8×8、16×16、32×32 量化系数矩阵 QuantCoeffMatrix 转换为 4×4、8×8、16×16、32×32 变换系数矩阵 CoeffMatrix。

变换系数矩阵 CoeffMatrix 的元素 w_{ji} 由量化系数矩阵 QuantCoeffMatrix 的元素 val_{ji} 经下式得到：

$w_{ji}=(val_{ji}\times QP)\gg dq\_shift$，

其中 tx_size=4×4 时，dq_shift=0，i，j=0…3；

tx_size=8×8 时，dq_shift=0，i，j=0…7；

tx_size=16×16，时 dq_shift=0，i，j=0…15；

tx_size=32×32，时 dq_shift=1，i，j=0…31。

5.3.6.5 反变换

5.3.6.5.1 4×4 反变换

4×4 反变换分为行反变换和列反变换，行/列反变换均可为 dct 的反变换或者 adst 的反变换。其组合形式有：{idct_idct}、{idct_iadst}、{iadst_idct}、{iadst_iadst}。

其中，4×4IADST 变换核矩阵为：

$$DST_4 = \begin{bmatrix} 5283 & 9929 & 13377 & 15212 \\ 13377 & 13377 & 0 & -13377 \\ 15212 & -5238 & -13377 & 9929 \\ 9929 & -15212 & 13377 & -5283 \end{bmatrix}$$

4×4IDCT 变换核矩阵为：

$$DCT_4 = \begin{bmatrix} 11585 & 11585 & 11585 & 11585 \\ 15137 & 6270 & -6270 & -15137 \\ 11585 & -11585 & -11585 & 11585 \\ 6270 & -15137 & 15137 & -6270 \end{bmatrix}$$

4×4 反变换步骤如下：

对变换系数矩阵进行水平反变换，如果反变换形式为 iadst，$T_4 = DST_4$；如果反变换形式为 idct，则 $T_4 = DCT_4$；

$$H' = (CoeffMatrix \times T_4{}^T + 2^{13}) \gg 14$$

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。对矩阵 H 进行垂直反变换，如果反变换形式为 iadst，则 $T_4 = DST_4$；如果反变换形式为 idct，则 $T_4 = DCT_4$；

$$H = (T_4 \times H' + 2^{13}) \gg 14$$

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 $-2^{4+BitDepth} \sim (2^{4+BitDepth} - 1)$。

残差样点矩阵 ResidueMatrix 的元素 r_{ji} 计算如下：

$$r_{ji} = (h_{ji} + 2^3) \gg 4 \qquad \cdots\cdots (i,j = 0 \ldots 3)$$

其中，h_{ji} 为 H 矩阵的元素。

5.3.6.5.2 8×8 反变换

8×8 反变换分为行反变换和列反变换分别进行，行/列反变换都可以为 dct 的反变换或者 adst 的反变换。其组合形式有：{idct_idct}、{idct_iadst}、{iadst_idct}、{iadst_iadst}。

其中 8×8IADST 变换核矩阵为：

$$DST_8 = \begin{bmatrix} 1606 & 4756 & 7723 & 10394 & 12665 & 14449 & 15679 & 16305 \\ 4756 & 12665 & 16305 & 14449 & 7723 & -1606 & -10394 & -15679 \\ 7723 & 16305 & 10394 & -4756 & -15679 & -12665 & 1606 & 14449 \\ 10394 & 14449 & -4756 & -16305 & -1606 & 15679 & 7723 & -12665 \\ 12665 & 7723 & -15679 & -1606 & 16305 & -4756 & -14449 & 10394 \\ 14449 & -1606 & -12665 & 15679 & -4756 & -10394 & 16305 & -7723 \\ 15679 & -10394 & 1606 & 7723 & -14449 & 16305 & -12665 & 4756 \\ 16305 & -15679 & 14449 & -12665 & 10394 & -7723 & 4756 & -1606 \end{bmatrix}$$

8×8IDCT 变换核矩阵为：

$$DCT_8 = \begin{bmatrix} 11585 & 11585 & 11585 & 11585 & 11585 & 11585 & 11585 & 11585 \\ 16069 & 13623 & 9102 & 3196 & -3196 & -9102 & -13623 & -16069 \\ 15137 & 6270 & -6270 & -15137 & -15137 & -6270 & 6270 & 15137 \\ 13623 & -3196 & -16069 & -9102 & 9102 & 16069 & 3196 & -13623 \\ 11585 & -11585 & -11585 & 11585 & 11585 & -11585 & -11585 & 11585 \\ 9102 & -16069 & 3196 & 13623 & -13623 & -3196 & 16069 & -9102 \\ 6270 & -15137 & 15137 & -6270 & -6270 & 15137 & -15137 & 6270 \\ 3196 & -9102 & 13623 & -16069 & 16069 & -13623 & 9102 & -3196 \end{bmatrix}$$

8×8 反变换步骤如下：

a) 对变换系数矩阵进行水平反变换，如果反变换形式为 iadst，$T_8 = DST_8$；如果反变换形式为 idct，则 $T_8 = DCT_8$；

$$H' = (CoeffMatrix \times T_8^{T} + 2^{13}) \gg 14$$

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。

b) 对矩阵 H′进行垂直反变换，如果反变换形式为 iadst，则 $T_8 = DST_8$；如果反变换形式为 idct，则 $T_8 = DCT_8$；

$$H = (T_8 \times H' + 2^{13}) \gg 14$$

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 $-2^{5+BitDepth} \sim (2^{5+BitDepth} - 1)$。

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

$$r_{ji} = (h_{ji} + 2^4) \gg 5 \qquad \cdots\cdots (i,j = 0 \ldots 7)$$

其中，h_{ji} 为 H 矩阵的元素。

5.3.6.5.3 16×16 反变换

16×16 反变换分为行反变换和列反变换分别进行，行/列反变换均可为 dct 的反变换或者 adst 的反变换。其组合形式有：{idct_idct}、{idct_iadst}、{iadst_idct}、{iadst_iadst}。

其中 16×16IADST 变换核矩阵为：

$$DST_{16} = \begin{bmatrix} 804 & 2404 & 3981 & 5520 & 7005 & 8423 & 9760 & 11003 & 12140 & 13160 & 14053 & 14811 & 15426 & 15893 & 16207 & 16364 \\ 2404 & 7005 & 11003 & 14053 & 15893 & 16364 & 15426 & 13160 & 9760 & 5520 & 804 & -3981 & -8423 & -12140 & -14811 & -16207 \\ 3981 & 11003 & 15426 & 16207 & 13160 & 7005 & -804 & -8423 & -14053 & -16364 & -14811 & -9760 & -2404 & 5520 & 12140 & 15893 \\ 5520 & 14053 & 16207 & 11003 & 804 & -9760 & -15893 & -14811 & -7005 & 3981 & 13160 & 16364 & 12140 & 2404 & -8423 & -15426 \\ 7005 & 15893 & 13160 & 804 & -12140 & -16207 & -8423 & 5520 & 15426 & 14053 & 2404 & -11003 & -16364 & -9760 & 3981 & 14811 \\ 8423 & 16364 & 7005 & -9760 & -16207 & -5520 & 11003 & 15893 & 3981 & -12140 & -15426 & -2404 & 13160 & 14811 & 804 & -14053 \\ 9760 & 15426 & -804 & -15893 & -8423 & 11003 & 14811 & -2404 & -16207 & -7005 & 12140 & 14053 & -3981 & -16364 & -5520 & 13160 \\ 11003 & 13160 & -8423 & -14811 & 5520 & 15893 & -2404 & -16364 & -804 & 16207 & 3981 & -15426 & -7005 & 14053 & 9760 & -12140 \\ 12140 & 9760 & -14053 & -7005 & 15426 & 3981 & -16207 & -804 & 16364 & -2404 & -15893 & 5520 & 14811 & -8423 & -13160 & 11003 \\ 13160 & 5520 & -16364 & 3981 & 14053 & -12140 & -7005 & 16207 & -2404 & -14811 & 11003 & 8423 & -15893 & 804 & 15426 & -9760 \\ 14053 & 804 & -14811 & 13160 & 2404 & -15426 & 12140 & 3981 & -15893 & 11003 & 5520 & -16207 & 9760 & 7005 & -16364 & 8423 \\ 14811 & -3981 & -9760 & 16364 & -11003 & -2404 & 14053 & -15426 & 5520 & 8423 & -16207 & 12140 & 804 & -13160 & 15893 & -7005 \\ 15426 & -8423 & -2404 & 12140 & -16364 & 13160 & -3981 & -7005 & 14811 & -15893 & 9760 & 804 & -11003 & 16207 & -14053 & 5520 \\ 15893 & -12140 & 5520 & 2404 & -9760 & 14811 & -16364 & 14053 & -8423 & 804 & 7005 & -13160 & 16207 & -15426 & 11003 & -3981 \\ 16207 & -14811 & 12140 & -8423 & 3981 & 804 & -5520 & 9760 & -13160 & 15426 & -16364 & 15893 & -14053 & 11003 & -7005 & 2404 \\ 16364 & -16207 & 15893 & -15426 & 14811 & -14053 & 13160 & -12140 & 11003 & -9760 & 8423 & -7005 & 5520 & -3981 & 2404 & -804 \end{bmatrix}$$

16×16idct 变换核矩阵为：

$$DCT_{16}=\begin{bmatrix}
11585 & 11585 & 11585 & 11585 & 11585 & 11585 & 11585 & 11585 & 11585 & 11585 & 11585 & 11585 & 11585 & 11585 & 11585 & 11585\\
16305 & 15679 & 14449 & 12665 & 10394 & 7723 & 4756 & 1606 & -1606 & -4756 & -7723 & -10394 & -12665 & -14449 & -15679 & -16305\\
16069 & 13623 & 9102 & 3196 & -3196 & -9102 & -13623 & -16069 & -16069 & -13623 & -9102 & -3196 & 3196 & 9102 & 13623 & 16069\\
15679 & 10394 & 1606 & -7723 & -14449 & -16305 & -12665 & -4756 & 4756 & 12665 & 16305 & 14449 & 7723 & -1606 & -10394 & -15679\\
15137 & 6270 & -6270 & -15137 & -15137 & -6270 & 6270 & 15137 & 15137 & 6270 & -6270 & -15137 & -15137 & -6270 & 6270 & 15137\\
14449 & 1606 & -12665 & -15679 & -4756 & 10394 & 16305 & 7723 & -7723 & -16305 & -10394 & 4756 & 15679 & 12665 & -1606 & -14449\\
13623 & -3196 & -16069 & -9102 & 9102 & 16069 & 3196 & -13623 & -13623 & 3196 & 16069 & 9102 & -9102 & -16069 & -3196 & 13623\\
12665 & -7723 & -15679 & 1606 & 16305 & 4756 & -14449 & -10394 & 10394 & 14449 & -4756 & -16305 & -1606 & 15679 & 7723 & -12665\\
11585 & -11585 & -11585 & 11585 & 11585 & -11585 & -11585 & 11585 & 11585 & -11585 & -11585 & 11585 & 11585 & -11585 & -11585 & 11585\\
10394 & -14449 & -4756 & 16305 & -1606 & -15679 & 7723 & 12665 & -12665 & -7723 & 15679 & 1606 & -16305 & 4756 & 14449 & -10394\\
9102 & -16069 & 3196 & 13623 & -13623 & -3196 & 16069 & -9102 & -9102 & 16069 & -3196 & -13623 & 13623 & 3196 & -16069 & 9102\\
7723 & -16305 & 10394 & 4756 & -15679 & 12665 & 1606 & -14449 & 14449 & -1606 & -12665 & 15679 & -4756 & -10394 & 16305 & -7723\\
6270 & -15137 & 15137 & -6270 & -6270 & 15137 & -15137 & 6270 & 6270 & -15137 & 15137 & -6270 & -6270 & 15137 & -15137 & 6270\\
4756 & -12665 & 16305 & -14449 & 7723 & 1606 & -10394 & 15679 & -15679 & 10394 & -1606 & -7723 & 14449 & -16305 & 12665 & -4756\\
3196 & -9102 & 13623 & -16069 & 16069 & -13623 & 9102 & -3196 & -3196 & 9102 & -13623 & 16069 & -16069 & 13623 & -9102 & 3196\\
1606 & -4756 & 7723 & -10394 & 12665 & -14449 & 15679 & -16305 & 16305 & -15679 & 14449 & -12665 & 10394 & -7723 & 4756 & -1606
\end{bmatrix}$$

16×16 反变换步骤如下：

a) 对变换系数矩阵进行水平反变换，如果反变换形式为 iadst，则 $T_{16}=DST_{16}$；如果反变换形式为 idct，则 $T_{16}=DCT_{16}$：

$$H'=(CoeffMatrix\times T_{16}{}^{T}+2^{13})\rangle\rangle 14$$

其中，T_{16} 为 16×16 反变换矩阵，$T_{16}{}^{T}$ 为 T_{16} 的转置矩阵，H' 表示水平反变换后的中间结果。

b) 对矩阵 H' 进行垂直反变换，如果反变换形式为 iadst，则 $T_{16}=DST_{16}$；如果反变换形式为 idct，则 $T_{16}=DCT_{16}$：

$$H=(T_{16}\times H'+2^{13})\rangle\rangle 14$$

其中，H 表示反变换后的 16×16 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 $-2^{6+BitDepth}\sim(2^{6+BitDepth}-1)$。

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

$$r_{ji}=(h_{ji}+2^{5})\rangle\rangle 6 \qquad (i,j=0...15)$$

5.3.6.5.4 32×32 反变换

32×32 反变换分为行反变换和列反变换分别进行，行/列反变换都只能为 dct 的反变换

32×32IDCT 变换核矩阵为

$$DCT_{32}=[DCT_{L16\times 32}\ DCT_{R16\times 32}]$$

$$DCT_{L16\times32} = \begin{bmatrix}
11585 & 11585 & 11585 & 11585 & 11585 & 11585 & 11585 & 11585 & 11585 & 11585 & 11585 & 11585 & 11585 & 11585 & 11585 & 11585 \\
16364 & 16207 & 15893 & 15426 & 14811 & 14053 & 13160 & 12140 & 11003 & 9760 & 8423 & 7005 & 5520 & 3981 & 2404 & 804 \\
16305 & 15679 & 14449 & 12665 & 10394 & 7723 & 4756 & 1606 & -1606 & -4756 & -7723 & -10394 & -12665 & -14449 & -15679 & -16305 \\
16207 & 14811 & 12140 & 8423 & 3981 & -804 & -5520 & -9760 & -13160 & -15426 & -16364 & -15893 & -14053 & -11003 & -7005 & -2404 \\
16069 & 13623 & 9102 & 3196 & -3196 & -9102 & -13623 & -16069 & -16069 & -13623 & -9102 & -3196 & 3196 & 9102 & 13623 & 16069 \\
15893 & 12140 & 5520 & -2404 & -9760 & -14811 & -16364 & -14053 & -8423 & -804 & 7005 & 13160 & 16207 & 15426 & 11003 & 3981 \\
15679 & 10394 & 1606 & -7723 & -14449 & -16305 & -12665 & -4756 & 4756 & 12665 & 16305 & 14449 & 7723 & -1606 & -10394 & -15679 \\
15426 & 8423 & -2404 & -12140 & -16364 & -13160 & -3981 & 7005 & 14811 & 15893 & 9760 & -804 & -11003 & -16207 & -14053 & -5520 \\
15137 & 6270 & -6270 & -15137 & -15137 & -6270 & 6270 & 15137 & 15137 & 6270 & -6270 & -15137 & -15137 & -6270 & 6270 & 15137 \\
14811 & 3981 & -9760 & -16364 & -11003 & 2404 & 14053 & 15426 & 5520 & -8423 & -16207 & -12140 & 804 & 13160 & 15893 & 7005 \\
14449 & 1606 & -12665 & -15679 & -4756 & 10394 & 16305 & 7723 & -7723 & -16305 & -10394 & 4756 & 15679 & 12665 & -1606 & -14449 \\
14053 & -804 & -14811 & -13160 & 2404 & 15426 & 12140 & -3981 & -15893 & -11003 & 5520 & 16207 & 9760 & -7005 & -16364 & -8423 \\
13623 & -3196 & -16069 & -9102 & 9102 & 16069 & 3196 & -13623 & -13623 & 3196 & 16069 & 9102 & -9102 & -16069 & -3196 & 13623 \\
13160 & -5520 & -16364 & -3981 & 14053 & 12140 & -7005 & -16207 & -2404 & 14811 & 11003 & -8423 & -15893 & -804 & 15426 & 9760 \\
12665 & -7723 & -15679 & 1606 & 16305 & 4756 & -14449 & -10394 & 10394 & 14449 & -4756 & -16305 & -1606 & 15679 & 7723 & -12665 \\
12140 & -9760 & -14053 & 7005 & 15426 & -3981 & -16207 & 804 & 16364 & 2404 & -15893 & -5520 & 14811 & 8423 & -13160 & -11003 \\
11585 & -11585 & -11585 & 11585 & 11585 & -11585 & -11585 & 11585 & 11585 & -11585 & -11585 & 11585 & 11585 & -11585 & -11585 & 11585 \\
11003 & -13160 & -8423 & 14811 & 5520 & -15893 & -2404 & 16364 & -804 & -16207 & 3981 & 15426 & -7005 & -14053 & 9760 & 12140 \\
10394 & -14449 & -4756 & 16305 & -1606 & -15679 & 7723 & 12665 & -12665 & -7723 & 15679 & 1606 & -16305 & 4756 & 14449 & -10394 \\
9760 & -15426 & -804 & 15893 & -8423 & -11003 & 14811 & 2404 & -16207 & 7005 & 12140 & -14053 & -3981 & 16364 & -5520 & -13160 \\
9102 & -16069 & 3196 & 13623 & -13623 & -3196 & 16069 & -9102 & -9102 & 16069 & -3196 & -13623 & 13623 & 3196 & -16069 & 9102 \\
8423 & -16364 & 7005 & 9760 & -16207 & 5520 & 11003 & -15893 & 3981 & 12140 & -15426 & 2404 & 13160 & -14811 & 804 & 14053 \\
7723 & -16305 & 10394 & 4756 & -15679 & 12665 & 1606 & -14449 & 14449 & -1606 & -12665 & 15679 & -4756 & -10394 & 16305 & -7723 \\
7005 & -15893 & 13160 & -804 & -12140 & 16207 & -8423 & -5520 & 15426 & -14053 & 2404 & 11003 & -16364 & 9760 & 3981 & -14811 \\
6270 & -15137 & 15137 & -6270 & -6270 & 15137 & -15137 & 6270 & 6270 & -15137 & 15137 & -6270 & -6270 & 15137 & -15137 & 6270 \\
5520 & -14053 & 16207 & -11003 & 804 & 9760 & -15893 & 14811 & -7005 & -3981 & 13160 & -16364 & 12140 & -2404 & -8423 & 15426 \\
4756 & -12665 & 16305 & -14449 & 7723 & 1606 & -10394 & 15679 & -15679 & 10394 & -1606 & -7723 & 14449 & -16305 & 12665 & -4756 \\
3981 & -11003 & 15426 & -16207 & 13160 & -7005 & -804 & 8423 & -14053 & 16364 & -14811 & 9760 & -2404 & -5520 & 12140 & -15893 \\
3196 & -9102 & 13623 & -16069 & 16069 & -13623 & 9102 & -3196 & -3196 & 9102 & -13623 & 16069 & -16069 & 13623 & -9102 & 3196 \\
2404 & -7005 & 11003 & -14053 & 15893 & -16364 & 15426 & -13160 & 9760 & -5520 & 804 & 3981 & -8423 & 12140 & -14811 & 16207 \\
1606 & -4756 & 7723 & -10394 & 12665 & -14449 & 15679 & -16305 & 16305 & -15679 & 14449 & -12665 & 10394 & -7723 & 4756 & -1606 \\
804 & -2404 & 3981 & -5520 & 7005 & -8423 & 9760 & -11003 & 12140 & -13160 & 14053 & -14811 & 15426 & -15893 & 16207 & -16364
\end{bmatrix}$$

$$DCT_{R16\times32}=\begin{bmatrix}
11585 & 11585 & 11585 & 11585 & 11585 & 11585 & 11585 & 11585 & 11585 & 11585 & 11585 & 11585 & 11585 & 11585 & 11585 & 11585\\
-804 & -2404 & -3981 & -5520 & -7005 & -8423 & -9760 & -11003 & -12140 & -13160 & -14053 & -14811 & -15426 & -15893 & -16207 & -16364\\
-16305 & -15679 & -14449 & -12665 & -10394 & -7723 & -4756 & -1606 & 1606 & 4756 & 7723 & 10394 & 12665 & 14449 & 15679 & 16305\\
2404 & 7005 & 11003 & 14053 & 15893 & 16364 & 15426 & 13160 & 9760 & 5520 & 804 & -3981 & -8423 & -12140 & -14811 & -16207\\
16069 & 13623 & 9102 & 3196 & -3196 & -9102 & -13623 & -16069 & -16069 & -13623 & -9102 & -3196 & 3196 & 9102 & 13623 & 16069\\
-3981 & -11003 & -15426 & -16207 & -13160 & -7005 & 804 & 8423 & 14053 & 16364 & 14811 & 9760 & 2404 & -5520 & -12140 & -15893\\
-15679 & -10394 & -1606 & 7723 & 14449 & 16305 & 12665 & 4756 & -4756 & -12665 & -16305 & -14449 & -7723 & 1606 & 10394 & 15679\\
5520 & 14053 & 16207 & 11003 & 804 & -9760 & -15893 & -14811 & -7005 & 3981 & 13160 & 16364 & 12140 & 2404 & -8423 & -15426\\
15137 & 6270 & -6270 & -15137 & -15137 & -6270 & 6270 & 15137 & 15137 & 6270 & -6270 & -15137 & -15137 & -6270 & 6270 & 15137\\
-7005 & -15893 & -13160 & -804 & 12140 & 16207 & 8423 & -5520 & -15426 & -14053 & -2404 & 11003 & 16364 & 9760 & -3981 & -14811\\
-14449 & -1606 & 12665 & 15679 & 4756 & -10394 & -16305 & -7723 & 7723 & 16305 & 10394 & -4756 & -15679 & -12665 & 1606 & 14449\\
8423 & 16364 & 7005 & -9760 & -16207 & -5520 & 11003 & 15893 & 3981 & -12140 & -15426 & -2404 & 13160 & 14811 & 804 & -14053\\
13623 & -3196 & -16069 & -9102 & 9102 & 16069 & 3196 & -13623 & -13623 & 3196 & 16069 & 9102 & -9102 & -16069 & -3196 & 13623\\
-9760 & -15426 & 804 & 15893 & 8423 & -11003 & -14811 & 2404 & 16207 & 7005 & -12140 & -14053 & 3981 & 16364 & 5520 & -13160\\
-12665 & 7723 & 15679 & -1606 & -16305 & -4756 & 14449 & 10394 & -10394 & -14449 & 4756 & 16305 & 1606 & -15679 & -7723 & 12665\\
11003 & 13160 & -8423 & -14811 & 5520 & 15893 & -2404 & -16364 & -804 & 16207 & 3981 & -15426 & -7005 & 14053 & 9760 & -12140\\
11585 & -11585 & -11585 & 11585 & 11585 & -11585 & -11585 & 11585 & 11585 & -11585 & -11585 & 11585 & 11585 & -11585 & -11585 & 11585\\
-12140 & -9760 & 14053 & 7005 & -15426 & -3981 & 16207 & 804 & -16364 & 2404 & 15893 & -5520 & -14811 & 8423 & 13160 & -11003\\
-10394 & 14449 & 4756 & -16305 & 1606 & 15679 & -7723 & -12665 & 12665 & 7723 & -15679 & -1606 & 16305 & -4756 & -14449 & 10394\\
13160 & 5520 & -16364 & 3981 & 14053 & -12140 & -7005 & 16207 & -2404 & -14811 & 11003 & 8423 & -15893 & 804 & 15426 & -9760\\
9102 & -16069 & 3196 & 13623 & -13623 & -3196 & 16069 & -9102 & -9102 & 16069 & -3196 & -13623 & 13623 & 3196 & -16069 & 9102\\
-14053 & -804 & 14811 & -13160 & -2404 & 15426 & -12140 & -3981 & 15893 & -11003 & -5520 & 16207 & -9760 & -7005 & 16364 & -8423\\
-7723 & 16305 & -10394 & -4756 & 15679 & -12665 & -1606 & 14449 & -14449 & 1606 & 12665 & -15679 & 4756 & 10394 & -16305 & 7723\\
14811 & -3981 & -9760 & 16364 & -11003 & -2404 & 14053 & -15426 & 5520 & 8423 & -16207 & 12140 & 804 & -13160 & 15893 & -7005\\
6270 & -15137 & 15137 & -6270 & -6270 & 15137 & -15137 & 6270 & 6270 & -15137 & 15137 & -6270 & -6270 & 15137 & -15137 & 6270\\
-15426 & 8423 & 2404 & -12140 & 16364 & -13160 & 3981 & 7005 & -14811 & 15893 & -9760 & -804 & 11003 & -16207 & 14053 & -5520\\
-4756 & 12665 & -16305 & 14449 & -7723 & -1606 & 10394 & -15679 & 15679 & -10394 & 1606 & 7723 & -14449 & 16305 & -12665 & 4756\\
15893 & -12140 & 5520 & 2404 & -9760 & 14811 & -16364 & 14053 & -8423 & 804 & 7005 & -13160 & 16207 & -15426 & 11003 & -3981\\
3196 & -9102 & 13623 & -16069 & 16069 & -13623 & 9102 & -3196 & -3196 & 9102 & -13623 & 16069 & -16069 & 13623 & -9102 & 3196\\
-16207 & 14811 & -12140 & 8423 & -3981 & -804 & 5520 & -9760 & 13160 & -15426 & 16364 & -15893 & 14053 & -11003 & 7005 & -2404\\
-1606 & 4756 & -7723 & 10394 & -12665 & 14449 & -15679 & 16305 & -16305 & 15679 & -14449 & 12665 & -10394 & 7723 & -4756 & 1606\\
16364 & -16207 & 15893 & -15426 & 14811 & -14053 & 13160 & -12140 & 11003 & -9760 & 8423 & -7005 & 5520 & -3981 & 2404 & -804
\end{bmatrix}$$

32×32 反变换步骤如下：

a) 对变换系数矩阵进行水平反变换，$T_{32}=DCT_{32}$：

$$H'=(CoeffMatrix\times {T_{32}}^{T}+2^{13})\rangle\rangle 14$$

其中，T_{32}为 32×32 反变换矩阵，${T_{32}}^{T}$ 为 T_{32}的转置矩阵，H'表示水平反变换后的中间结果。

b) 对矩阵 H'进行垂直反变换，$T_{32}=DCT_{32}$：

$$H=(T_{32}\times H'+2_{13})\rangle\rangle 14$$

其中，H 表示反变换后的 32×32 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$-2^{6+BitDepth}\sim(2^{6+BitDepth}-1)$。

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

$$r_{ji}=(h_{ji}+2^{5})\rangle\rangle 6 \qquad (i,j=0...31)$$

5.3.6.6 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。

如果当前块为 Intra 块类型，重建样点矩阵 RecMatrix 计算如下：

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

如果当前块为 Inter 块类型，采用 Single 预测模式，重建样点矩阵 RecMatrix 计算如下：

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

如果当前块为 Inter 块类型，采用 Compound 预测模式，并且有两帧参考图像，重建样点矩阵 RecMatrix 计算如下：

RecMatrix[x,y]=Clip1(((predMatrix0[x,y]+predMatrix1[x,y]+1) ››1)+ResidueMatrix[x,y])

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 filter_level 大于 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编码单元为单位对图像中的所有的编码单元按照光栅扫描的顺序进行。否则将不进行调用去块效应滤波过程。

在一个编码单元内，对亮度和色度分别做去块效应滤波，去块效应滤波的单位是滤波块。对于每个树形编码单元以及每个分量按滤波块尺寸进行划分进行滤波，亮度滤波块的尺寸是 8×8，色度滤波块的尺寸是 8×8。每个滤波块包括一条垂直边界和一条水平边界。先滤波纵向的边界，从编码单元的左侧的边界开始，按照从左到右的顺序进行处理，然后滤波横向的边界，从编码单元的上部边界开始，按照从上到下的顺序进行处理。

图 15 表示图像的树形编码单元尺寸为 64×64，滤波块尺寸为 8×8。如果需要滤波，则根据 5.3.7.2 计算边界滤波参数，根据 5.3.7.3 采用对应的方式进行滤波。

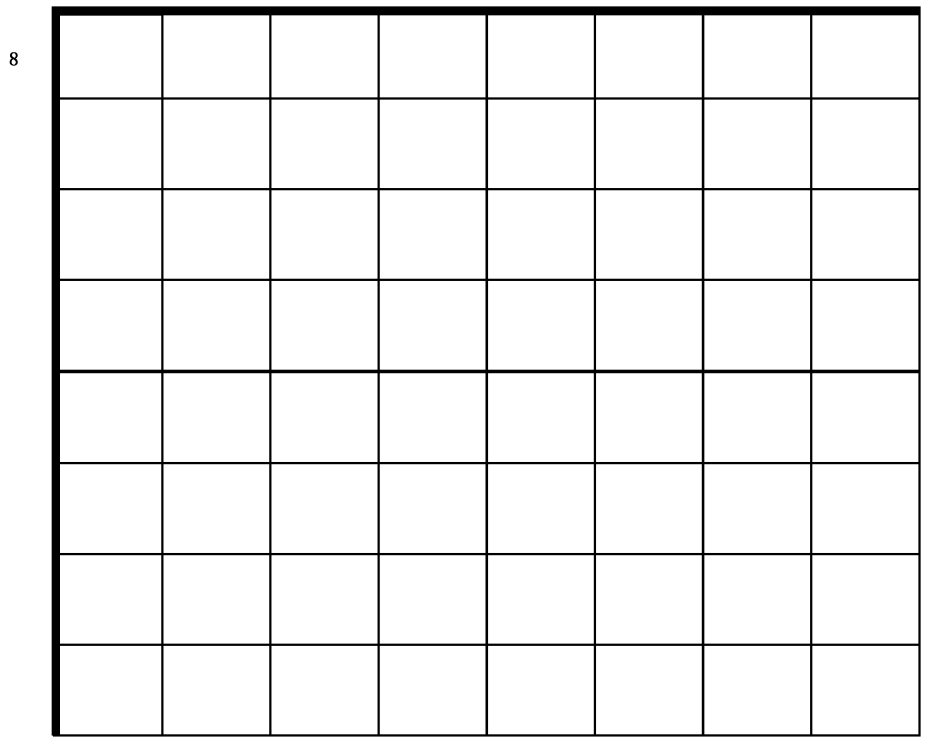


图 15 块边界去块滤波示意图

注 1：因为在解码当前编码单元时，下边与右边的编码单元数据没有解码出来，因此先做上一行编码单元的滤波。

注 2：帧内预测使用去块效应滤波前的重建图像样点值。

注 3：当前滤波块的垂直边界的滤波过程中修改的样值作为水平边界滤波过程的输入。

满足以下条件之一的边界不需要滤波：

a) 待滤波边界是图像边界或 Tile 边界；

b） 待滤波边界是 skip_flag 等于 1 的预测单元的内部边界；

c） 待滤波边界既不是编码单元的边界，又不是预测单元的边界也不是变换块的边界。

5.3.7.2 块边界滤波参数的推导过程

块边界滤波参数的推导过程如下：

a） 根据图像参数集中的 filter_level 计算得到每个编码块单元在不同情况下的 cu_filter_level 值：

首先计算每个 segment 对应的 delta，不同的 segment_id、不同的参考帧、不同的块模式对应不同的 delta 值。通过在图像参数集中 filter_level 的基础上，增加 delta 最终获得数组 lvl_lookup [8][6][2]的值。具体实现如下：

```
scale=1 << (filter_level >> 5)
for (seg_id=0; seg_id<8; seg_id++){
  lvl_seg=filter_level
  if(feature_enable[seg_id][1]){
    data=features_data[seg_id][1]
    lvl_seg=clip3(0,63,seg_abs_delta==1 ? data:filter_level+data)
  }
  if(! lf_delta_enable)
    lvl_lookup[seg_id][6][2]={lvl_seg,lvl_seg,...}
  else{
    intra_lvl=lvl_seg+ref_deltas[0]×scale
    lvl_lookup[seg_id][0][0]=clip3(0,63,intra_lvl)
    for (ref=1; ref<6; ++ref) {
      for(mode=0; mode<2; ++mode) {
        inter_lvl=lvl_seg+ref_deltas[ref]×scale+mode_deltas[mode]×scale
        lvl_lookup[seg_id][ref][mode]=clip3(0,63,inter_lvl)
      }
    }
  }
}
```

根据当前编码块的对应的 segment_id、参考帧类型 ref_frame、预测模式 mode 作为索引即可以计算得到当前编码块的 cu_filter_level。当块的预测模式为帧内预测时，mode 等于 0；当预测模式为帧间预测时，如果当前编码块的 mv_mode 为 ZERO_MV，则 mode 等于 0，否则 mode 等于 1。cu_filter_level 等于 lvl_lookup[segment_id][ref_frame][mode]。

b） 根据 sharpness_level 计算不同的 cu_filter_level 对应的阈值参数 limit、blimit、thresh 的值：

根据 sharpness_level 的取值阈值，参数数组 limit[64]、blimit[64]、thresh[64]的计算过程如下所示：

```
for(lvl=0; lvl<=63; lvl++){
  block_inside_limit=lvl >>((sharpness_level > 0)+(sharpness_level > 4))
  if((sharpness_lvl>0) && (block_inside_limit > (9-sharpness_level)))
    block_inside_limit=(9-sharpness_level);
  if(block_inside_limit < 1)
    block_inside_limit=1
```

limit[lvl]=block_inside_limit
blimit[lvl] = 2×(lvl+2)+block_inside_limit
thresh[lvl] = (lvl 〉〉 4)
}

根据当前编码块的 cu_filter_level 作为索引即可得到当前编码块的滤波阈值参数 limit、blimit 和 thresh。

c) 计算 filterSize：

如果该边界所属滤波块的 tx_size 为 TX_4x4，并且该边界不位于按 32×32 块划分的边界，则 filterSize 等于 0；否则 filterSize 等于 1。

5.3.7.3 块边界滤波过程

图 16 表示在水平或垂直边界两侧的 8 个样点，分别属于块 p 和块 q，边界用黑色粗线表示。

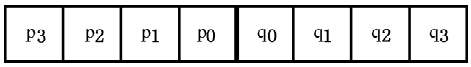


图 16 滤波边界(垂直或水平)两侧的样点

hev 的计算过程如下：

hev=0
threshBd=thresh 〈〈(BitDepth −8)
hev |=(abs(p1−p0)>threshBd)×−1
hev |=(abs(q1−q0)>threshBd)×−1

mask 的计算过程如下：

limitBd=limit 〈〈(BitDepth −8)
blimitBd=blimit〈〈(BitDepth −8)
mask=0
mask |=(abs(p3−p2)>blimitBd)×−1
mask |=(abs(p2−p1)>blimitBd)×−1
mask |=(abs(p1−p0)>blimitBd)×−1
mask |=(abs(q1−q0)>blimitBd)×−1
mask |=(abs(q2−q1)>blimitBd)×−1
mask |=(abs(q3−q2)>blimitBd)×−1
mask |=((abs(p0−q0)〈〈1)+(abs(p1−q1)〉〉1)>blimitBd)×−1
mask=~mask

flat 的计算过程如下：

如果 filtersize 等于 0，则 flat 等于 0，否则按如下过程计算 flat：

thresholdBd=1 〈〈 (BitDepth−8)
flat |=(abs(p1−p0)>thresholdBd)×−1
flat |=(abs(q1−q0)>thresholdBd)×−1
flat |=(abs(p2−p0)>thresholdBd)×−1
flat |=(abs(q2−q0)>thresholdBd)×−1
flat |=(abs(p3−p0)>thresholdBd)×−1
flat |=(abs(q3−q0)>thresholdBd)×−1
flat=~flat

根据 flat 和 mask 的取值按如下方法执行滤波：

——如果 flat 与 mask 都不为 0，则：

op2＝ROUND_POWER_OF_TWO(p3＋p3＋p3＋2×p2＋p1＋p0＋q0,3)

op1＝ROUND_POWER_OF_TWO(p3＋p3＋p2＋2×p1＋p0＋q0＋q1,3)

op0＝ROUND_POWER_OF_TWO(p3＋p2＋p1＋2×p0＋q0＋q1＋q2,3)

oq0＝ROUND_POWER_OF_TWO(p2＋p1＋p0＋2×q0＋q1＋q2＋q3,3)

oq1＝ROUND_POWER_OF_TWO(p1＋p0＋q0＋2×q1＋q2＋q3＋q3,3)

oq2＝ROUND_POWER_OF_TWO(p0＋q0＋q1＋2×q2＋q3＋q3＋q3,3)

——否则，令 shift＝BitDepth－8，有：

ps1＝p1－(0x80 ‹‹ shift)

ps0＝p0－(0x80 ‹‹ shift)

qs0＝q0－(0x80 ‹‹ shift)

qs1＝q1－(0x80 ‹‹ shift)

filter＝Clip3((－128)‹‹shift,(128‹‹shift)－1,(ps1－qs1)) & hev

filter＝Clip3((－128)‹‹shift,(128‹‹shift)－1,(filter＋3×(qs0－ps0))) & mask

filter1＝Clip3((－128)‹‹shift,(128‹‹shift)－1,(filter＋4)) ›› 3

filter2＝Clip3((－128)‹‹shift,(128‹‹shift)－1,(filter＋3))›› 3

filter＝ROUND_POWER_OF_TWO(filter1,1) & (～hev)

oq0＝Clip3((－128)‹‹shift,(128‹‹shift)－1,qs0－filter1)＋(0x80 ‹‹ shift)

op0＝Clip3((－128)‹‹shift,(128‹‹shift)－1,ps0＋filter2)＋(0x80 ‹‹ shift)

oq1＝Clip3((－128)‹‹shift,(128‹‹shift)－1,qs1－filter)＋(0x80 ‹‹ shift)

op1＝Clip3((－128)‹‹shift,(128‹‹shift)－1,ps1＋filter)＋(0x80 ‹‹ shift)

其中 BitDepth 为图像的样点比特精度，op2～op0、oq0～oq2 对应 p2～p0、q0～q2 滤波后的样点值，ROUND_POWER_OF_TWO(value,n)定义为 (((value)＋(1 ‹‹ ((n)－1))) ›› (n))。

5.3.8 样点偏移补偿

5.3.8.1 概述

如果 sao_enable 等于 1，调用样点偏移补偿(SAO)过程。

如果 sao_component_on[Y]，sao_component_on[U]，sao_component_on[V]的取值为 1，则对对应分量进行 SAO 操作。如果三个参数全为 0，则本帧数据跳过 SAO 操作。

根据 5.3.8.2 导出样点偏移补偿单元，根据 5.3.8.3 导出与当前样点偏移补偿单元对应的样点偏移补偿信息，然后按照 5.3.8.4 对当前样点偏移补偿单元内各个样点的各分量进行操作，得到偏移后样点值。

5.3.8.2 样点偏移补偿单元导出

SAO 操作以树形编码单元为单位，根据当前树形编码单元的区域按下列步骤得到当前样点偏移补偿单元的区域：

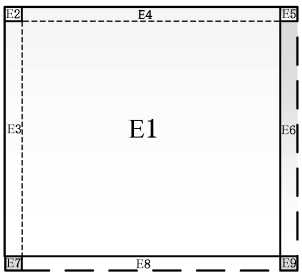


图 17　SAO 单元区域的导出

——将当前树形编码单元所在样点区域的亮度区域与色度区域均(图 17 中黑色虚线所包含的矩形区域)向左移四个样点单位,再向上移四个样点单位,得到区域 S1(由图 17 中的 E1,E2,E3,E4 四个部分组成的矩形区域);

——如果区域 S1 超出当前图像边界,则将超出边界部分去除(例:如当前树形编码单元为左边界的树形编码单元,则去除图 17 中的 E2,E3 部分;如当前树形编码单元为上边界的树形编码单元,则去除图 17 中的 E2,E4 部分,),得到区域 S2;否则令 S2 等于 S1;

——如果当前树形编码单元 C 包含图像最右列的样点且不包含图像最下行的样点,则将区域 S2 的右边界向右扩展至图像的右边界(即将图 17 中的 E5(若存在),则 E6 纳入当前 SAO 区域),得到区域 S3;

——否则,如果当前树形编码单元 C 包含图像最下行的样点且不包含图像最右列的样点,则将区域 S2 的下边界向下扩展至图像的边界(即将图 17 中的 E7(若存在),则 E8 纳入当前 SAO 区域),得到区域 S3;

——否则,如果当前树形编码单元 C 同时包含图像最右列的样点和最下行的样点,则将区域 S2 的右边界向右扩展至图像的右边界后,再将新区域的下边界向下至图像的下边界,(将 E5(若存在),E6,E7(若存在),E8,E9 均纳入当前 SAO 区域)得到区域 S3;

——否则,令 S3 等于 S2;

——将区域 S3 作为当前样点偏移补偿单元的区域。

5.3.8.3　样点偏移补偿信息导出

下述信息用于样点偏移补偿过程:参数融合标志 sao_merge_flag、融合类型 sao_merge_type、样点偏移补偿合并模式 sao_merge_mode、样点偏移补偿模式 sao_mode[compIdx]、样点偏移补偿值 sao_offset[compIdx][j]、样点偏移补偿边缘模式类型 sao_edge_type[compIdx]、样点偏移补偿区间模式类型的起始偏移子区间 sao_start_band[compIdx]等。

如果当前树形编码单元不是图像或者 Tile 的左边界的树形编码单元,MergeLeftAvail 等于 1,否则 MergeLeftAvail 等于 0。如果当前树形编码单元不是图片的上边界树形编码单元,MergeUpAvail 等于 1,否则 MergeUpAvail 等于 0。

根据 MergeLeftAvail、MergeUpAvail、sao_merge_flag 及 sao_merge_type 的值查表 56 得到样点偏移补偿合并模式 sao_merge_mode。

表 56　样点偏移补偿合并模式

MergeLeftAvail	MergeUpAvail	sao_merge_flag	sao_merge_type	sao_merge_mode
0	0	—	—	SAO_NON_MERGE
1	0	0	—	SAO_NON_MERGE
		1	—	SAO_MERGE_LEFT
0	1	0	—	SAO_NON_MERGE
		1	—	SAO_MERGE_UP
1	1	0	—	SAO_NON_MERGE
		1	1	SAO_MERGE_LEFT
			0	SAO_MERGE_UP

如果 sao_merge_mode 的值为‘SAO_MERGE_LEFT’，sao_mode[compIdx](compIdx＝0，1，2)的值等于左侧相邻 CTU 的样点偏移补偿单元的 sao_mode[compIdx]的值，当前 CTU 的 SAO 参数使用左侧相邻 CTU 的 SAO 参数；等于 0 表示当前 CTU 的 SAO 参数使用上方相邻 CTU 的 SAO 参数。

如果 sao_merge_mode 值为‘SAO_MERGE_UP’，sao_mode[compIdx] (compIdx＝0，1，2)的值等于上方相邻 CTU 样点偏移补偿单元的 sao_mode[compIdx]的值，当前 CTU 的 SAO 参数使用上方相邻 CTU 的 SAO 参数。

如果 sao_merge_mode 的值为‘SAO_NON_MERGE’，则首先从码流中解析得到 sao_mode[compIdx]的值，再根据 sao_mode[compIdx]的值从码流中获取相应的其他信息：

——如果 sao_mode[compIdx]的值为‘SAO_BO’，则从码流中解析得到 sao_start_band[compIdx]，并有：
sao_offset[compIdx][j]＝sao_offset_abs[compIdx][j]×Sign(0－sao_offset_sign[compIdx][j])(j＝0～3)；

——如果 sao_mode[compIdx]的值为‘SAO_EO’，则从码流中解析得到 sao_edge_offset[compIdx][j](j＝0～3)和 sao_edge_type[compIdx]，并有：
sao_offset[compIdx][j]＝sao_edge_offset[compIdx][j]；

——如果 sao_mode[compIdx]的值为‘SAO_OFF’，则不需要从码流继续读取该分量的 SAO 信息。

5.3.8.4　SAO 操作步骤

5.3.8.4.1　SAO_BO 操作

如果样点偏移补偿单元的 sao_mode[i]为‘SAO_BO’，则将样点取值划分为 32 个相等的区间，区间依次为 0～31，只对落入其中 4 个连续区间的样点值进行偏移补偿操作，每个区间对应一个偏移值，如图 18 所示。

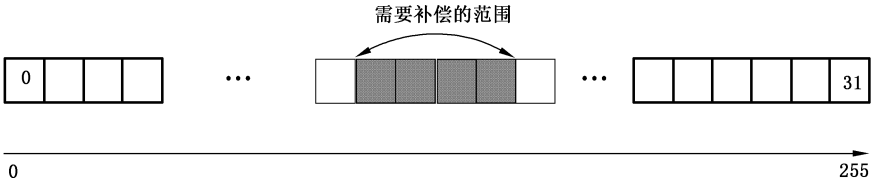


图 18　BO 示意图

其具体操作如下：

a)　第一步，根据样点的 i 分量值大小所处的范围查表 57 得到该分量对应的 SaoOffset[i]。

表 57　样点取值范围对应的偏移值

样点取值范围	偏移值(SaoOffset)
SaoBandPos[i][0]〈〈 shift ～ SaoBandPos[i][0]〈〈 shift+(2^ shift−1)	sao_offset[i][0]
SaoBandPos[i][1]〈〈 shift ～ SaoBandPos[i][1]〈〈 shift+(2^ shift−1)	sao_offset[i][1]
SaoBandPos[i][2]〈〈 shift ～ SaoBandPos[i][2]〈〈 shift+(2^ shift−1)	sao_offset[i][2]
SaoBandPos[i][3]〈〈 shift ～ SaoBandPos[i][3]〈〈 shift+(2^ shift−1)	sao_offset[i][3]

表中：

shift＝BitDepth−5

SaoBandPos [i][0]＝sao_start_band[i]

SaoBandPos [i][1]＝(sao_start_band[i]+1)％ 32

SaoBandPos [i][2]＝(sao_start_band[i]+2)％ 32

SaoBandPos [i][3]＝(sao_start_band[i]+3)％ 32

b)　第二步，当前样点的 i 分量偏移后取值 y[i]计算如下：

y[i]＝Clip3(0,(1〈〈BitDepth)−1,x[i]+SaoOffset[i])

其中，x[i]是该样点的 i 分量在去块滤波后的值。

5.3.8.4.2　SAO_EO 操作

如果样点偏移补偿单元的 sao_mode[i]为‘SAO_EO’则进行以下操作：

a)　第一步，根据 sao_edge_type[i]的值确定与当前样点 c 相邻的样点 a 和 b，如图 19 所示；如果 a 或 b 不在当前 Tile 内或者 a 或 b 不在当前图像内，则不改变样点 c 的值。

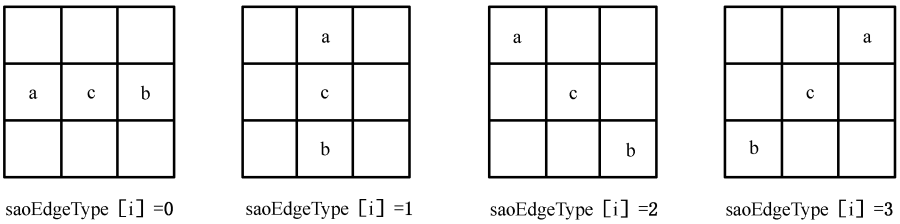


图 19　当前样点和相邻样点的位置关系

b)　第二步，根据表 58 及图 20，利用当前样点 c 的在去块效应滤波后的 i 分量值 xc 与相邻样点 a 和 b 的滤波后样点 i 分量值 xa 和 xb 的关系确定当前样点 i 分量的 SaoOffset[i]。

表 58　相邻样点取值与偏移补偿值的对应关系

样点分量值的关系		偏移补偿值(SaoOffset[i])
Valley	xc＜xa && xc＜xb	sao_offset[i][0]
Half valley	(xc＜xa && xc＝＝xb) || (xc＝＝xa && xc＞xb)	sao_offset[i][1]
Half peak	(xc＞xa && xc＝＝xb) || (xc＝＝xa && xc＞xb)	sao_offset[i][2]
peak	xc＞xa && xc＞xb	sao_offset[i][3]
flat	其他	0

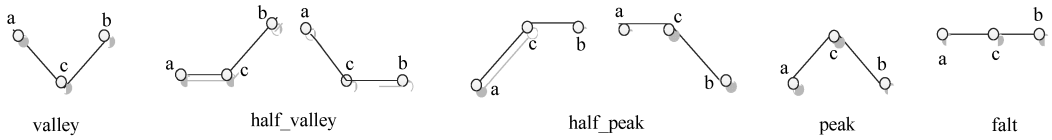


图 20　样点 c 与其相邻样点 a 和 b 的关系

c)　第三步，当前样点的 i 分量在偏移补偿后的取值 y[i]计算如下：

$$y[i]=Clip3(0,(1\ll BitDepth)-1,x[i]+SaoOffset[i])$$

其中，x[i]是该样点的 i 分量在去块效应滤波后的取值。

5.3.8.4.3　SAO_OFF 操作

如果 sao_mode[compIdx](compIdx 可以为 0，1，2 代表三个不同的分量)为 SAO_OFF，或当前样点是图像的边界，则将去块滤波后样点对应分量的值直接作为偏移后该样点分量的值。

5.3.9　样点滤波补偿

5.3.9.1　概述

如果 picture_alf_enable [comp_idx]的值为 0，则将偏移后样点分量的值直接作为解码图像中样点分量的值，否则，对相应的偏移后样点分量进行样点滤波补偿。其中 comp_idx 等于 0 表示亮度分量；等于 1 表示 Cb 分量；等于 2 表示 Cr 分量。

样点滤波补偿的单位是由树形编码单元导出的样点滤波补偿单元，按照光栅扫描顺序依次处理。首先根据 5.3.9.2 解码各分量的样点滤波补偿系数，然后根据 5.3.9.3 导出样点滤波补偿单元，根据 5.3.9.4确定当前样点滤波补偿单元亮度分量的样点滤波补偿系数索引，最后根据 5.3.9.5 对样点补偿滤波单元的亮度和色度分量进行样点滤波补偿，得到解码图像的样点。

5.3.9.2　样点滤波补偿系数解码

样点滤波补偿系数的解码过程如下：

a)　从比特流中解析得到亮度样点的第 i 组补偿系数 alf_coeff_luma[i][j](i＝0～alf_filter_num_minus1，j＝0～8)。对系数 alf_coeff_luma [i][9](即图 21 的系数 C9)做以下处理：

$$alf\_coeff\_luma[i]\ [9]+=64-\sum_{j=0}^{8}2\times alf\_coeff\_luma\ [i]\ [j]$$

其中 alf_coeff_luma [i][j](j＝0～8)的位宽是 7 位，取值应为－64～63。经上述处理后 alf_coeff_luma[i][9]的取值应为 0～127。

b) 根据 alf_region_distance[i](i>1)得到亮度分量样点滤波补偿系数索引数组(记作 alf_coeff_idx_tab [16]):

```
count=0
alf_coeff_idx_tab[0]=0
for(i=1; i<alf_filter_num_minus1+1; i++) {
    for(j=0; j<alf_region_distance[i]-1; j++) {
        alf_coeff_idx_tab [count+1]=alf_coeff_idx_tab [count]
        count=count+1
    }
    alf_coeff_idx_tab [count+1]=alf_coeff_idx_tab [count]+1
    count=count+1
}
for(i=count; i<16; i++)
    alf_coeff_idx_tab [i]=alf_coeff_idx_tab [count]
```

c) 从比特流中解析得到色度样点的补偿系数 alf_coeff_chroma[0][j]和 alf_coeff_chroma [1][j](j=0～8)。对系数 alf_coeff_chroma [0][9]和 alf_coeff_chroma [1][9](即图 21 的系数 C9),分别做以下处理:

$$\text{alf\_coeff\_chroma}[i][9]+=64-\sum_{j=0}^{8}2\times\text{alf\_coeff\_chroma}[i][j],\ i=0,1$$

其中 alf_coeff_chroma [i][j](j=0～7)的位宽是 7 位,取值应为－64～63。经上述处理后 alf_coeff_chroma [i][9]的取值应为 0～127。

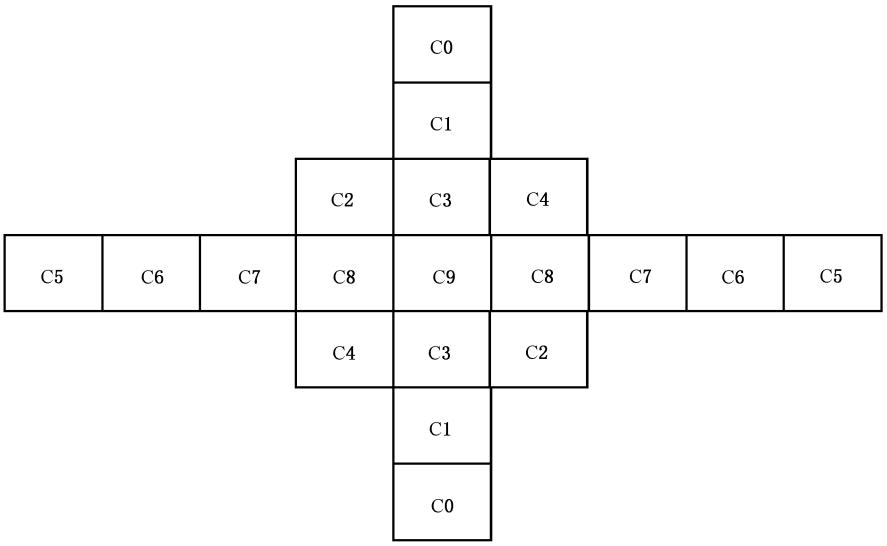


图 21 样点滤波补偿系数

5.3.9.3 导出样点滤波补偿单元

根据当前树形编码单元按下列步骤导出样点滤波补偿单元:

——将当前树形编码单元 C 所在样点区域超出图像边界的部分删除,得到样点区域 D;

——如果区域 D 的下边界所在样点不属于图像的下边界,将亮度分量和色度分量样点区域 D 的下边界向上收缩四行,得到区域 E1;否则,令 E1 等于 D;

——如果区域 E1 的上边界所在样点属于图像的上边界令 E2 等于 E1；否则，将亮度分量和色度分量样点区域 E1 的上边界向上扩展四行，得到区域 E2；

——将区域 E2 作为当前样点滤波补偿单元。图像的第一行样点为图像的上边界，最后一行样点为图像的下边界。亮度分量和色度分量样点区域(D,E1,E2)的第一行样点为区域的上边界，最后一行样点为区域的下边界。

5.3.9.4 确定亮度分量样点滤波补偿单元样点滤波补偿系数索引

根据以下方法计算当前亮度分量样点滤波补偿单元的样点滤波补偿系数索引，记作 filter_idx：

horizontal_size= width_minus_1+1

vertical_size=height_minus_1+1

如果 extended_sb_size_flag 等于 1，ctu_size_in_bit 等于 6，否则 ctu_size_in_bit 等于 5

x_interval=((((horizontal_size+(1 〈〈 ctu_size_in_bit)−1) 〉〉 ctu_size_in_bit)+1) 〉〉 2) 〈〈 ctu_size_in_bit

y_interval=((((vertical_size+(1 〈〈 ctu_size_in_bit)−1) 〉〉 ctu_size_in_bit)+1) 〉〉 2) 〈〈 ctu_size_in_bit

```
if (x_interval==0 && y_interval==0)
    index=15
else if (x_interval==0)
    index=Min(3,y/ y_interval)×4+3
else if (y_interval==0)
    index=Min(3,x/ x_interval)+12
else
    index=Min(3,y/ y_interval)×4+Min(3,x/ x_interval)
```

filter_idx=alf_coeff_idx_tab [region_table[index]]

其中 region_table[16]={0,1,4,5,15,2,3,6,14,11,10,7,13,12,9,8}，(x,y)是导出当前样点滤波补偿单元的树形编码单元左上角样点在图像中的坐标。

5.3.9.5 样点滤波操作

如果 **alf_ctu_enable**[comp_idx][ctu_index]等于 1，则对 comp_idx 分量进行样点滤波补偿，否则不进行样点补偿滤波。如果样点滤波补偿过程中用到了样点滤波补偿单元外的样点，则按照如下方式处理：

——如果当前样点滤波补偿单元左边界或右边界为图像边界，则样点滤波补偿单元左边界或右边界外的样点分别用样点滤波单元内距离该样点最近的样点代替；

——样点滤波补偿单元上边界和下边界外的样点分别用样点滤波单元内距离该样点最近的样点代替；

样点滤波补偿单元亮度分量的样点滤波补偿操作如下：

$$\text{ptmp}=\text{alf\_coeff\_luma}[\text{filter\_idx}][9]\times p(x,y)+\sum_{j=0}^{8}2\times \text{alf\_coeff\_luma}[\text{filter\_idx}][j]\times(p(x-hor[j],y-ver[j])+p(x+hor[j],y+ver[j]))$$

ptmp=(ptmp+32) 〉〉 6

p′(x,y)=Clip3(0,(1〈〈BitDepth)−1,ptmp)

其中，p(x,y)为偏移后样点，p′(x,y)为重建样点，hor[j]和 ver[j](j=0～7)见表 59。

表 59　样点补偿滤波坐标偏移值

j 的值	hor[j]的值	ver[j]的值
0	0	3
1	0	2
2	1	1
3	0	1
4	1	−1
5	3	0
6	2	0
7	1	0

5.3.10　空域可伸缩性视频编码(SVC)增强层编码片中 CTU 的解码过程

5.3.10.1　概述

解码 nal_unit_type 值为 3 和 4 的增强层编码片 NAL 单元中的树形编码单元时，调用本过程。空域 SVC 的增强层与基本层之间的图像宽度比与高度比支持 4:3，2:1，4:1，6:1 和 8:1，由 svc_ratio 决定。

当 svc_mode 等于 0 时，增强层的解码不使用跨层预测，只使用增强层内的帧间或帧内预测，解码过程与基本层相同。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。

当 svc_mode 等于 1 时，增强层的解码过程中使用层内预测和跨层预测，解码过程见 5.3.10.3。

当 svc_roi_flag 等于 1 时，增强层的解码过程见 5.3.10.5。

5.3.10.2　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图像样点的插值计算，通过以下数组实现：

```
svac2_sub_pel_filters_8[]
{
        {  0,  0,   0, 128,   0,   0,  0,  0 },
        {  0,  1,  −5, 126,   8,  −3,  1,  0 },
        { −1,  3, −10, 122,  18,  −6,  2,  0 },
        { −1,  4, −13, 118,  27,  −9,  3, −1 },
        { −1,  4, −16, 112,  37, −11,  4, −1 },
        { −1,  5, −18, 105,  48, −14,  4, −1 },
        { −1,  5, −19,  97,  58, −16,  5, −1 },
        { −1,  6, −19,  88,  68, −18,  5, −1 },
        { −1,  6, −19,  78,  78, −19,  6, −1 },
        { −1,  5, −18,  68,  88, −19,  6, −1 },
        { −1,  5, −16,  58,  97, −19,  5, −1 },
        { −1,  4, −14,  48, 105, −18,  5, −1 },
        { −1,  4, −11,  37, 112, −16,  4, −1 },
        { −1,  3,  −9,  27, 118, −13,  4, −1 },
        {  0,  2,  −6,  18, 122, −10,  3, −1 },
        {  0,  1,  −3,   8, 126,  −5,  1,  0 }
```

};

设低分辨率图像的宽高分别为 src_w 和 src_h，高分辨率图像的宽高分别为 dst_w 和 dst_h。factor_x 和 factor_y 为根据色度采样格式和块类型确定的缩放因子。计算出每个样点对应的缩放比例：

factor=(Y ? 1 : 2);

x_q4=x×(16/factor)×src_w/dst_w;

y_q4=y×(16/factor)×src_h/dst_h;

根据缩放比例 x_q4 和 y_q4，分别在插值数组中选择相应的插值系数，进行水平和垂直的插值计算，得到高分辨率图像样点。

5.3.10.3 svc_mode 等于 1 时增强层的预测及图像重建过程

本过程输入为增强层码流，对应基本层解码图像及各 8×8 块的 MV，输出为增强层的重构图像。

增强层的残差样点由码流解析生成的残差系数矩阵经逆扫描，反量化及反变换过程得到，具体过程见 5.3.6.2～5.3.6.5。

增强层预测矩阵的计算方式如下：

增强层的参考图像集应与基本层的参考图像集对应，即相同的参考帧类型对应同一图像的增强层和基本层。另外，根据增强层图像的参考帧类型来判断是跨层预测还是层内预测：

——如果参考帧为 static_ref，则增强层采用跨层预测。使用基本层解码图像经 5.3.10.2 插值放大后的图像作为参考帧，并根据运动矢量计算得到预测矩阵，其中运动矢量的计算过程见5.3.10.4；

——如果参考帧类型不是 static_ref，则增强层采用层内预测，帧间预测同 5.3.5，得到预测矩阵。

增强层的重建过程同 5.3.6.6。

5.3.10.4 跨层预测时运动矢量的计算

跨层预测且当在增强层与基本层的宽度比和高度比为 2：1 时，基本层放大后图像中每个 16×16 块中各 8×8 块的亮度运动矢量由基本层图像中对应的 8×8 块的亮度运动矢量进行如下扩展获得：

EL_MV=BL_MV ×2

EL_REF=BL_REF

其中 EL_MV 和 EL_REF 分别为增强层的 MV 和参考帧类型，BL_MV 和 BL_REF 分别为基本层的 MV 和参考帧类型。

增强层在进行候选运动矢量集导出且当增强层与基本层的宽度比和高度比等于 2：1 时，在5.3.5.2.1 节中第一步与第二步之间插入一个步骤：如果基本层放大后的当前块相同位置的块使用的参考帧与当前块的参考帧相同，则该块对应的基本层扩展放大后的 MV 加入候选运动矢量集。当增强层与基本层的宽度比和高度比为 2：1 时，候选运动矢量集的导出过程同 5.3.5.2.1。

5.3.10.5 svc_roi_flag 等于 1 时增强层的预测及重建过程

当 svc_roi_flag 等于 1 时，增强层支持 ROI 解码。ROI 区域的预测及重建过程同 5.3.10.3，非 ROI 区域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 5.3.10.2 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，输出为语法元素值。

对于 5.2.3 中的语法元素，ae(v)解析过程见 5.4.2，ue(v)、se(v)的解析过程见 5.4.3。

5.4.2 算术码解析过程

5.4.2.1 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。

ae(v)描述的语法元素解析过程如下：

a) 对 Tile 进行解析前，首先进行初始化，还会对一些语法元素的概率表进行更新，见 5.4.2.2；如果 wpp_enable 等于 1，在每 CTU 行解析开始前，需要重新初始化算术解码器，见 5.4.2.2.3；

b) 对二进制位串进行解析，见 5.4.2.4：

1) 二进制位串中每个二进制位的索引号为 binIdx，对应概率的获取见 5.4.2.4.4；

2) 根据 ctxIdx 解析二进制位，见 5.4.2.4.5；

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 ( 当前语法元素为编码片的第一个 ae(v)语法元素 ) {
    初始化算术码解码器
}
binIdx=－1
do {
    binIdx＋＋
    得到与 binIdx 对应的 ctxIdx
    得到与 ctxIdx 对应的概率模型 prob
    根据 prob 解析二进制位
} while ( ($b_0$,…,$b_{binIdx}$)不是语法元素的二进制位串 )
输出语法元素值
```

5.4.2.2 初始化

5.4.2.2.1 初始化概率模型

本过程的输出是初始化后算术码上下文变量所对应的 prob，即二进制位解析所需的概率值。该变量通过 binIdx、treeIdx 及 ctxIdx 索引。

comp 用来表示 MV 的概率表索引。comp 为 0 表示垂直坐标，comp 为 1 表示水平坐标。

对于每个上下文变量，应初始化每个 ctxIdx 索引所对应的概率状态。

初始化过程所涉及的语法元素概率表见表 60。

表 60 初始化过程所涉及的语法元素概率表汇总

语法元素	概率表
alf_ctu_enable	表 61
partition	表 62～表 63
skip_flag	表 64

表 60（续）

语法元素	概率表
inter_block	表 65
tx_size	表 66
prev_intra_luma_pred_flag	表 67
uv_fllow_y_flag	表 68
block_reference_mode	表 69
ref_frame	表 70～表 71
mv_mode	表 72～表 74
mv_joint	表 75
mvd_sign_0 和 mvd_sign_1	表 76
mvd_value_0 和 mvd_value_1	表 77～表 83
interp_filter_mode	表 84
dqp_abs	表 85
coef_value	表 86～表 96

语法元素 alf_ctu_enable 的初始概率表见表 61。

表 61　alf_ctu_enable_prob

ctxIdx	binIdx
	0
0	229
1	115
2	25
3	9

frame_type 等于 1 时，语法元素 partition 的初始概率表见表 62。frame_type 等于 0 时，语法元素 partition 的初始概率表见表 63。

表 62　inter_partition_prob

ctxIdx	treeIdx		
	0	1	2
	8×8 → 4×4		
0(a/l both not split)	199	122	141
1(a split,l not split)	147	63	159
2(l split,a not split)	148	133	118
3(a/l both split)	121	104	114

表 62（续）

ctxIdx	treeIdx		
	0	1	2
	16×16 → 8×8		
4(a/l both not split)	174	73	87
5(a split,l not split)	92	41	83
6(l split,a not split)	82	99	50
7(a/l both split)	53	39	39
	32×32 → 16×16		
8(a/l both not split)	177	58	59
9(a split,l not split)	68	26	63
10(l split,a not split)	52	79	25
11(a/l both split)	17	14	12
	64×64 → 32×32		
12(a/l both not split)	222	34	30
13(a split,l not split)	72	16	44
14(l split,a not split)	58	32	12
15(a/l both split)	10	7	6
	128×128 → 64×64		
16(a/l both not split)	220	33	28
17(a split,l not split)	70	15	43
18(l split,a not split)	57	31	11
19(a/l both split)	9	6	6

表 63　intra_partition_prob

ctxIdx	treeIdx		
	0	1	2
	8×8 → 4×4		
0(a/l both not split)	158	97	94
1(a split,l not split)	93	24	99
2(l split,a not split)	85	119	44
3(a/l both split)	62	59	67
	16×16 → 8×8		
4(a/l both not split)	149	53	53
5(a split,l not split)	94	20	48
6(l split,a not split)	83	53	24
7(a/l both split)	52	18	18

表 63（续）

ctxIdx	treeIdx		
	0	1	2
	32×32 → 16×16		
8(a/l both not split)	150	40	39
9(a split,l not split)	78	12	26
10(l split,a not split)	67	33	11
11(a/l both split)	24	7	5
	64×64 —〉32×32		
12(a/l both not split)	174	35	49
13(a split,l not split)	68	11	27
14(l split,a not split)	57	15	9
15(a/l both split)	12	3	3
	128×128 → 64×64		
16(a/l both not split)	160	32	45
17(a split,l not split)	65	10	23
18(l split,a not split)	54	12	8
19(a/l both split)	11	2	2

语法元素 skip_flag 使用的初始概率表见表 64。

表 64　skip_prob

ctxIdx	binIdx
	0
0	192
1	128
2	64

语法元素 inter_block 的初始概率表见表 65。

表 65　intra_inter_prob

ctxIdx	binIdx
	0
0	9
1	102
2	187
3	225

MAX_TX_SIZE 取值不同时，语法元素 tx_size 的初始概率表见表 66。

表 66　tx_probs

ctxIdx	binIdx		
	0	1	2
	TX_32×32		
0	3	136	37
1	5	52	13
	TX_16×16		
0	20	152	—
1	15	101	—
	TX_8×8		
0	100	—	—
1	66	—	—

语法元素 prev_intra_luma_pred_flag 的初始概率表见表 67。

表 67　mpm_flag_probs

ctxIdx	binIdx
	0
0	80
1	84
2	110
3	128
4	128
5	110
6	80

语法元素 uv_fllow_y_flag 的初始概率表见表 68。

表 68　uv_fllow_y_prob

ctxIdx	binIdx
	0
0	175

语法元素 block_reference_mode 的初始概率表见表 69。

表 69　comp_inter_prob

ctxIdx	binIdx
	0
0	239
1	183
2	119
3	96

语法元素 ref_frame 在 COMPOUND_REFERENCE 模式下的初始概率表见表 70，在 SINGLE_REFERENCE 模式下的初始概率表见表 71。

表 70　comp_ref_prob

ctxIdx	binIdx		
	0	1	2
0	50	50	50
1	126	126	126
2	123	123	123
3	221	221	221
4	226	226	226

表 71　single_ref_prob

ctxIdx	binIdx			
	0	1	2	3
0	33	16	16	16
1	77	74	74	74
2	142	142	142	142
3	172	170	170	170
4	238	247	247	247

语法元素 mv_mode 用到三个初始概率表，分别见表 72～表 74。

表 72　newmv_mode_prob

ctxIdx	binIdx
	0
0	200
1	180
2	150
3	150
4	110
5	70
6	60

表 73　zeromv_mode_prob

ctxIdx	binIdx
	0
0	192
1	64

表 74　refmv_mode_prob

ctxIdx	binIdx
	0
0	220
1	220
2	200
3	200
4	180
5	128
6	1
7	250

运动矢量解析相关的初始概率表见表 75～表 83。

表 75　mv_joint_probs

ctxIdx	treeIdx		
	0	1	2
0	190	155	212

表 76　mv_sign_probs

comp	binIdx
	0
0	128
1	128

表 77　mv_bits_probs

comp	binIdx									
	0	1	2	3	4	5	6	7	8	9
0	136	140	148	160	176	192	224	234	234	240
1	136	140	148	160	176	192	224	234	234	240

表 78　mv_class0_bit_probs

comp	binIdx
	0
0	216
1	208

表 79　mv_class_probs

comp	treeIdx									
	0	1	2	3	4	5	6	7	8	9
0	224	144	192	168	192	176	192	198	198	245
1	216	128	176	160	176	176	192	198	198	208

表 80　mv_class0_fr_probs

comp	mv_class0_bit	treeIdx		
		0	1	2
0	0	128	128	64
0	1	96	112	64
1	0	128	128	64
1	1	96	112	64

表 81　mv_class0_hp_probs

comp	binIdx
	0
0	160
1	160

表 82　mv_fr_probs

comp	treeIdx		
	0	1	2
0	64	96	64
1	64	96	64

表 83　mv_hp_probs

comp	binIdx
	0
0	128
1	128

语法元素 interp_filter_mode 的初始化概率见表 84。

表 84　switchable_interp_prob

ctxIdx	treeIdx		
	0	1	2
0	235	192	128
1	36	243	208
2	34	16	128
3	36	243	48
4	149	160	128

语法元素 dqp_abs 的初始概率表见表 85。

表 85　dqp_abs_prob

ctxIdx	treeIdx						
	0	1	2	3	4	5	6
0	180	190	161	144	138	124	79
1	121	133	161	107	80	100	50

语法元素 coef_value 的解析，包含 token 和 extra 的计算。token 的计算需要用到 coef_probs 概率表和 svac2_pareto8_full 概率表，见表 86～表 90；extra 的计算需要用到 svac2_cat1_prob、svac2_cat2_prob、svac2_cat3_prob、svac2_cat4_prob、svac2_cat5_prob 和 svac2_cat6_prob 概率表。coef_probs 的上下文模型共有 6 个维度的信息：TX_SIZES、PLANE_TYPES、REF_TYPES、COEF_BANDS、COEFF_CONTEXTS、binIdx。

表 86　coef_probs 4×4

PLANE_TYPES	REF_TYPES	COEF_BANDS	ctxIdx	binIdx		
				0	1	2
Y plane	Intra	0	0	195	29	183
			1	84	49	136
			2	8	42	71
		1	0	31	107	169
			1	35	99	159
			2	17	82	140
			3	8	66	114
			4	2	44	76
			5	1	19	32
		2	0	40	132	201
			1	29	114	187
			2	13	91	157

表 86（续）

PLANE_TYPES	REF_TYPES	COEF_BANDS	ctxIdx	binIdx		
				0	1	2
Y plane	Intra	2	3	7	75	127
			4	3	58	95
			5	1	28	47
		3	0	69	142	221
			1	42	122	201
			2	15	91	159
			3	6	67	121
			4	1	42	77
			5	1	17	31
		4	0	102	148	228
			1	67	117	204
			2	17	82	154
			3	6	59	114
			4	2	39	75
			5	1	15	29
		5	0	156	57	233
			1	119	57	212
			2	58	48	163
			3	29	40	124
			4	12	30	81
			5	3	12	31
	Inter	0	0	191	107	226
			1	124	117	204
			2	25	99	155
		1	0	29	148	210
			1	37	126	194
			2	8	93	157
			3	2	68	118
			4	1	39	69
			5	1	17	33
		2	0	41	151	213
			1	27	123	193
			2	3	82	144

表 86（续）

PLANE_TYPES	REF_TYPES	COEF_BANDS	ctxIdx	binIdx		
				0	1	2
Y plane	Inter	2	3	1	58	105
			4	1	32	60
			5	1	13	26
		3	0	59	159	220
			1	23	126	198
			2	4	88	151
			3	1	66	114
			4	1	38	71
			5	1	18	34
		4	0	114	136	232
			1	51	114	207
			2	11	83	155
			3	3	56	105
			4	1	33	65
			5	1	17	34
		5	0	149	65	234
			1	121	57	215
			2	61	49	166
			3	28	36	114
			4	12	25	76
			5	3	16	42
UV plane	Intra	0	0	214	49	220
			1	132	63	188
			2	42	65	137
		1	0	85	137	221
			1	104	131	216
			2	49	111	192
			3	21	87	155
			4	2	49	87
			5	1	16	28
		2	0	89	163	230
			1	90	137	220
			2	29	100	183

表 86（续）

PLANE_TYPES	REF_TYPES	COEF_BANDS	ctxIdx	binIdx		
				0	1	2
UV plane	Intra	2	3	10	70	135
			4	2	42	81
			5	1	17	33
		3	0	108	167	237
			1	55	133	222
			2	15	97	179
			3	4	72	135
			4	1	45	85
			5	1	19	38
		4	0	124	146	240
			1	66	124	224
			2	17	88	175
			3	4	58	122
			4	1	36	75
			5	1	18	37
		5	0	141	79	241
			1	126	70	227
			2	66	58	182
			3	30	44	136
			4	12	34	96
			5	2	20	47
	Inter	0	0	229	99	249
			1	143	111	235
			2	46	109	192
		1	0	82	158	236
			1	94	146	224
			2	25	117	191
			3	9	87	149
			4	3	56	99
			5	1	33	57
		2	0	83	167	237
			1	68	145	222
			2	10	103	177

表 86（续）

PLANE_TYPES	REF_TYPES	COEF_BANDS	ctxIdx	binIdx		
				0	1	2
UV plane	Inter	2	3	2	72	131
			4	1	41	79
			5	1	20	39
		3	0	99	167	239
			1	47	141	224
			2	10	104	178
			3	2	73	133
			4	1	44	85
			5	1	22	47
		4	0	127	145	243
			1	71	129	228
			2	17	93	177
			3	3	61	124
			4	1	41	84
			5	1	21	52
		5	0	157	78	244
			1	140	72	231
			2	69	58	184
			3	31	44	137
			4	14	38	105
			5	8	23	61

表 87　coef_probs 8×8

PLANE_TYPES	REF_TYPES	COEF_BANDS	ctxIdx	binIdx		
				0	1	2
Y plane	Intra	0	0	125	34	187
			1	52	41	133
			2	6	31	56
		1	0	37	109	153
			1	51	102	147
			2	23	87	128
			3	8	67	101
			4	1	41	63

表 87（续）

PLANE_TYPES	REF_TYPES	COEF_BANDS	ctxIdx	binIdx		
				0	1	2
Y plane	Intra	1	5	1	19	29
		2	0	31	154	185
			1	17	127	175
			2	6	96	145
			3	2	73	114
			4	1	51	82
			5	1	28	45
		3	0	23	163	200
			1	10	131	185
			2	2	93	148
			3	1	67	111
			4	1	41	69
			5	1	14	24
		4	0	29	176	217
			1	12	145	201
			2	3	101	156
			3	1	69	111
			4	1	39	63
			5	1	14	23
		5	0	57	192	233
			1	25	154	215
			2	6	109	167
			3	3	78	118
			4	1	48	69
			5	1	21	29
	Inter	0	0	202	105	245
			1	108	106	216
			2	18	90	144
		1	0	33	172	219
			1	64	149	206
			2	14	117	177
			3	5	90	141
			4	2	61	95

表 87（续）

PLANE_TYPES	REF_TYPES	COEF_BANDS	ctxIdx	binIdx		
				0	1	2
Y plane	Inter	1	5	1	37	57
		2	0	33	179	220
			1	11	140	198
			2	1	89	148
			3	1	60	104
			4	1	33	57
			5	1	12	21
		3	0	30	181	221
			1	8	141	198
			2	1	87	145
			3	1	58	100
			4	1	31	55
			5	1	12	20
		4	0	32	186	224
			1	7	142	198
			2	1	86	143
			3	1	58	100
			4	1	31	55
			5	1	12	22
		5	0	57	192	227
			1	20	143	204
			2	3	96	154
			3	1	68	112
			4	1	42	69
			5	1	19	32
UV plane	Intra	0	0	212	35	215
			1	113	47	169
			2	29	48	105
		1	0	74	129	203
			1	106	120	203
			2	49	107	178
			3	19	84	144
			4	4	50	84
			5	1	15	25

表 87（续）

PLANE_TYPES	REF_TYPES	COEF_BANDS	ctxIdx	binIdx		
				0	1	2
UV plane	Intra	2	0	71	172	217
			1	44	141	209
			2	15	102	173
			3	6	76	133
			4	2	51	89
			5	1	24	42
		3	0	64	185	231
			1	31	148	216
			2	8	103	175
			3	3	74	131
			4	1	46	81
			5	1	18	30
		4	0	65	196	235
			1	25	157	221
			2	5	105	174
			3	1	67	120
			4	1	38	69
			5	1	15	30
		5	0	65	204	238
			1	30	156	224
			2	7	107	177
			3	2	70	124
			4	1	42	73
			5	1	18	34
	Inter	0	0	225	86	251
			1	144	104	235
			2	42	99	181
		1	0	85	175	239
			1	112	165	229
			2	29	136	200
			3	12	103	162
			4	6	77	123
			5	2	53	84

表 87（续）

PLANE_TYPES	REF_TYPES	COEF_BANDS	ctxIdx	binIdx		
				0	1	2
UV plane	Inter	2	0	75	183	239
			1	30	155	221
			2	3	106	171
			3	1	74	128
			4	1	44	76
			5	1	17	28
		3	0	73	185	240
			1	27	159	222
			2	2	107	172
			3	1	75	127
			4	1	42	73
			5	1	17	29
		4	0	62	190	238
			1	21	159	222
			2	2	107	172
			3	1	72	122
			4	1	40	71
			5	1	18	32
		5	0	61	199	240
			1	27	161	226
			2	4	113	180
			3	1	76	129
			4	1	46	80
			5	1	23	41

表 88　**coef_probs** 16×16

PLANE_TYPES	REF_TYPES	COEF_BANDS	ctxIdx	binIdx		
				0	1	2
Y plane	Intra	0	0	7	27	153
			1	5	30	95
			2	1	16	30
		1	0	50	75	127

表 88　coef_probs 16×16

PLANE_TYPES	REF_TYPES	COEF_BANDS	ctxIdx	binIdx		
				0	1	2
Y plane	Intra	1	1	57	75	124
			2	27	67	108
			3	10	54	86
			4	1	33	52
			5	1	12	18
		2	0	43	125	151
			1	26	108	148
			2	7	83	122
			3	2	59	89
			4	1	38	60
			5	1	17	27
		3	0	23	144	163
			1	13	112	154
			2	2	75	117
			3	1	50	81
			4	1	31	51
			5	1	14	23
		4	0	18	162	185
			1	6	123	171
			2	1	78	125
			3	1	51	86
			4	1	31	54
			5	1	14	23
		5	0	15	199	227
			1	3	150	204
			2	1	91	146
			3	1	55	95
			4	1	30	53
			5	1	11	20
	Inter	0	0	19	55	240
			1	19	59	196
			2	3	52	105

表 88（续）

PLANE_TYPES	REF_TYPES	COEF_BANDS	ctxIdx	binIdx		
				0	1	2
Y plane	Inter	1	0	41	166	207
			1	104	153	199
			2	31	123	181
			3	14	101	152
			4	5	72	106
			5	1	36	52
		2	0	35	176	211
			1	12	131	190
			2	2	88	144
			3	1	60	101
			4	1	36	60
			5	1	16	28
		3	0	28	183	213
			1	8	134	191
			2	1	86	142
			3	1	56	96
			4	1	30	53
			5	1	12	20
		4	0	20	190	215
			1	4	135	192
			2	1	84	139
			3	1	53	91
			4	1	28	49
			5	1	11	20
		5	0	13	196	216
			1	2	137	192
			2	1	86	143
			3	1	57	99
			4	1	32	56
			5	1	13	24
UV plane	Intra	0	0	211	29	217
			1	96	47	156
			2	22	43	87

表 88（续）

PLANE_TYPES	REF_TYPES	COEF_BANDS	ctxIdx	binIdx		
				0	1	2
UV plane	Intra	1	0	78	120	193
			1	111	116	186
			2	46	102	164
			3	15	80	128
			4	2	49	76
			5	1	18	28
		2	0	71	161	203
			1	42	132	192
			2	10	98	150
			3	3	69	109
			4	1	44	70
			5	1	18	29
		3	0	57	186	211
			1	30	140	196
			2	4	93	146
			3	1	62	102
			4	1	38	65
			5	1	16	27
		4	0	47	199	217
			1	14	145	196
			2	1	88	142
			3	1	57	98
			4	1	36	62
			5	1	15	26
		5	0	26	219	229
			1	5	155	207
			2	1	94	151
			3	1	60	104
			4	1	36	62
			5	1	16	28
	Inter	0	0	233	29	248
			1	146	47	220
			2	43	52	140

表 88（续）

PLANE_TYPES	REF_TYPES	COEF_BANDS	ctxIdx	binIdx		
				0	1	2
UV plane	Inter	1	0	100	163	232
			1	179	161	222
			2	63	142	204
			3	37	113	174
			4	26	89	137
			5	18	68	97
		2	0	85	181	230
			1	32	146	209
			2	7	100	164
			3	3	71	121
			4	1	45	77
			5	1	18	30
		3	0	65	187	230
			1	20	148	207
			2	2	97	159
			3	1	68	116
			4	1	40	70
			5	1	14	29
		4	0	40	194	227
			1	8	147	204
			2	1	94	155
			3	1	65	112
			4	1	39	66
			5	1	14	26
		5	0	16	208	228
			1	3	151	207
			2	1	98	160
			3	1	67	117
			4	1	41	74
			5	1	17	31

表 89　coef_probs 32×32

PLANE_TYPES	REF_TYPES	COEF_BANDS	ctxIdx	binIdx		
				0	1	2
Y plane	Intra	0	0	17	38	140
			1	7	34	80
			2	1	17	29
		1	0	37	75	128
			1	41	76	128
			2	26	66	116
			3	12	52	94
			4	2	32	55
			5	1	10	16
		2	0	50	127	154
			1	37	109	152
			2	16	82	121
			3	5	59	85
			4	1	35	54
			5	1	13	20
		3	0	40	142	167
			1	17	110	157
			2	2	71	112
			3	1	44	72
			4	1	27	45
			5	1	11	17
		4	0	30	175	188
			1	9	124	169
			2	1	74	116
			3	1	48	78
			4	1	30	49
			5	1	11	18
		5	0	10	222	223
			1	2	150	194
			2	1	83	128
			3	1	48	79
			4	1	27	45
			5	1	11	17

表 89（续）

PLANE_TYPES	REF_TYPES	COEF_BANDS	ctxIdx	binIdx		
				0	1	2
Y plane	Inter	0	0	36	41	235
			1	29	36	193
			2	10	27	111
		1	0	85	165	222
			1	177	162	215
			2	110	135	195
			3	57	113	168
			4	23	83	120
			5	10	49	61
		2	0	85	190	223
			1	36	139	200
			2	5	90	146
			3	1	60	103
			4	1	38	65
			5	1	18	30
		3	0	72	202	223
			1	23	141	199
			2	2	86	140
			3	1	56	97
			4	1	36	61
			5	1	16	27
		4	0	55	218	225
			1	13	145	200
			2	1	86	141
			3	1	57	99
			4	1	35	61
			5	1	13	22
		5	0	15	235	212
			1	1	132	184
			2	1	84	139
			3	1	57	97
			4	1	34	56
			5	1	14	23

表 89（续）

PLANE_TYPES	REF_TYPES	COEF_BANDS	ctxIdx	binIdx		
				0	1	2
UV plane	Intra	0	0	181	21	201
			1	61	37	123
			2	10	38	71
		1	0	47	106	172
			1	95	104	173
			2	42	93	159
			3	18	77	131
			4	4	50	81
			5	1	17	23
		2	0	62	147	199
			1	44	130	189
			2	28	102	154
			3	18	75	115
			4	2	44	65
			5	1	12	19
		3	0	55	153	210
			1	24	130	194
			2	3	93	146
			3	1	61	97
			4	1	31	50
			5	1	10	16
		4	0	49	186	223
			1	17	148	204
			2	1	96	142
			3	1	53	83
			4	1	26	44
			5	1	11	17
		5	0	13	217	212
			1	2	136	180
			2	1	78	124
			3	1	50	83
			4	1	29	49
			5	1	14	23

表 89（续）

PLANE_TYPES	REF_TYPES	COEF_BANDS	ctxIdx	binIdx		
				0	1	2
UV plane	Inter	0	0	197	13	247
			1	82	17	222
			2	25	17	162
		1	0	126	186	247
			1	234	191	243
			2	176	177	234
			3	104	158	220
			4	66	128	186
			5	55	90	137
		2	0	111	197	242
			1	46	158	219
			2	9	104	171
			3	2	65	125
			4	1	44	80
			5	1	17	91
		3	0	104	208	245
			1	39	168	224
			2	3	109	162
			3	1	79	124
			4	1	50	102
			5	1	43	102
		4	0	84	220	246
			1	31	177	231
			2	2	115	180
			3	1	79	134
			4	1	55	77
			5	1	60	79
		5	0	43	243	240
			1	8	180	217
			2	1	115	166
			3	1	84	121
			4	1	51	67
			5	1	16	6

表 90　svac2_pareto8_full

ctxIdx	treeIdx							
	0	1	2	3	4	5	6	7
0	3	86	128	6	86	23	88	29
1	6	86	128	11	87	42	91	52
2	9	86	129	17	88	61	94	76
3	12	86	129	22	88	77	97	93
4	15	87	129	28	89	93	100	110
5	17	87	129	33	90	105	103	123
6	20	88	130	38	91	118	106	136
7	23	88	130	43	91	128	108	146
8	26	89	131	48	92	139	111	156
9	28	89	131	53	93	147	114	163
10	31	90	131	58	94	156	117	171
11	34	90	131	62	94	163	119	177
12	37	90	132	66	95	171	122	184
13	39	90	132	70	96	177	124	189
14	42	91	132	75	97	183	127	194
15	44	91	132	79	97	188	129	198
16	47	92	133	83	98	193	132	202
17	49	92	133	86	99	197	134	205
18	52	93	133	90	100	201	137	208
19	54	93	133	94	100	204	139	211
20	57	94	134	98	101	208	142	214
21	59	94	134	101	102	211	144	216
22	62	94	135	105	103	214	146	218
23	64	94	135	108	103	216	148	220
24	66	95	135	111	104	219	151	222
25	68	95	135	114	105	221	153	223
26	71	96	136	117	106	224	155	225
27	73	96	136	120	106	225	157	226
28	76	97	136	123	107	227	159	228
29	78	97	136	126	108	229	160	229
30	80	98	137	129	109	231	162	231
31	82	98	137	131	109	232	164	232
32	84	98	138	134	110	234	166	233

表 90（续）

ctxIdx	treeIdx							
	0	1	2	3	4	5	6	7
33	86	98	138	137	111	235	168	234
34	89	99	138	140	112	236	170	235
35	91	99	138	142	112	237	171	235
36	93	100	139	145	113	238	173	236
37	95	100	139	147	114	239	174	237
38	97	101	140	149	115	240	176	238
39	99	101	140	151	115	241	177	238
40	101	102	140	154	116	242	179	239
41	103	102	140	156	117	242	180	239
42	105	103	141	158	118	243	182	240
43	107	103	141	160	118	243	183	240
44	109	104	141	162	119	244	185	241
45	111	104	141	164	119	244	186	241
46	113	104	142	166	120	245	187	242
47	114	104	142	168	121	245	188	242
48	116	105	143	170	122	246	190	243
49	118	105	143	171	122	246	191	243
50	120	106	143	173	123	247	192	244
51	121	106	143	175	124	247	193	244
52	123	107	144	177	125	248	195	244
53	125	107	144	178	125	248	196	244
54	127	108	145	180	126	249	197	245
55	128	108	145	181	127	249	198	245
56	130	109	145	183	128	249	199	245
57	132	109	145	184	128	249	200	245
58	134	110	146	186	129	250	201	246
59	135	110	146	187	130	250	202	246
60	137	111	147	189	131	251	203	246
61	138	111	147	190	131	251	204	246
62	140	112	147	192	132	251	205	247
63	141	112	147	193	132	251	206	247
64	143	113	148	194	133	251	207	247
65	144	113	148	195	134	251	207	247

表 90（续）

ctxIdx	treeIdx							
	0	1	2	3	4	5	6	7
66	146	114	149	197	135	252	208	248
67	147	114	149	198	135	252	209	248
68	149	115	149	199	136	252	210	248
69	150	115	149	200	137	252	210	248
70	152	115	150	201	138	252	211	248
71	153	115	150	202	138	252	212	248
72	155	116	151	204	139	253	213	249
73	156	116	151	205	139	253	213	249
74	158	117	151	206	140	253	214	249
75	159	117	151	207	141	253	215	249
76	161	118	152	208	142	253	216	249
77	162	118	152	209	142	253	216	249
78	163	119	153	210	143	253	217	249
79	164	119	153	211	143	253	217	249
80	166	120	153	212	144	254	218	250
81	167	120	153	212	145	254	219	250
82	168	121	154	213	146	254	220	250
83	169	121	154	214	146	254	220	250
84	171	122	155	215	147	254	221	250
85	172	122	155	216	147	254	221	250
86	173	123	155	217	148	254	222	250
87	174	123	155	217	149	254	222	250
88	176	124	156	218	150	254	223	250
89	177	124	156	219	150	254	223	250
90	178	125	157	220	151	254	224	251
91	179	125	157	220	151	254	224	251
92	180	126	157	221	152	254	225	251
93	181	126	157	221	152	254	225	251
94	183	127	158	222	153	254	226	251
95	184	127	158	223	154	254	226	251
96	185	128	159	224	155	255	227	251
97	186	128	159	224	155	255	227	251
98	187	129	160	225	156	255	228	251

表 90（续）

ctxIdx	treeIdx							
	0	1	2	3	4	5	6	7
99	188	130	160	225	156	255	228	251
100	189	131	160	226	157	255	228	251
101	190	131	160	226	158	255	228	251
102	191	132	161	227	159	255	229	251
103	192	132	161	227	159	255	229	251
104	193	133	162	228	160	255	230	252
105	194	133	162	229	160	255	230	252
106	195	134	163	230	161	255	231	252
107	196	134	163	230	161	255	231	252
108	197	135	163	231	162	255	231	252
109	198	135	163	231	162	255	231	252
110	199	136	164	232	163	255	232	252
111	200	136	164	232	164	255	232	252
112	201	137	165	233	165	255	233	252
113	201	137	165	233	165	255	233	252
114	202	138	166	233	166	255	233	252
115	203	138	166	233	166	255	233	252
116	204	139	166	234	167	255	234	252
117	205	139	166	234	167	255	234	252
118	206	140	167	235	168	255	235	252
119	206	140	167	235	168	255	235	252
120	207	141	168	236	169	255	235	252
121	208	141	168	236	170	255	235	252
122	209	142	169	237	171	255	236	252
123	209	143	169	237	171	255	236	252
124	210	144	169	237	172	255	236	252
125	211	144	169	237	172	255	236	252
126	212	145	170	238	173	255	237	252
127	213	145	170	238	173	255	237	252
128	214	146	171	239	174	255	237	253
129	214	146	171	239	174	255	237	253
130	215	147	172	240	175	255	238	253
131	215	147	172	240	175	255	238	253

表 90（续）

ctxIdx	treeIdx							
	0	1	2	3	4	5	6	7
132	216	148	173	240	176	255	238	253
133	217	148	173	240	176	255	238	253
134	218	149	173	241	177	255	239	253
135	218	149	173	241	178	255	239	253
136	219	150	174	241	179	255	239	253
137	219	151	174	241	179	255	239	253
138	220	152	175	242	180	255	240	253
139	221	152	175	242	180	255	240	253
140	222	153	176	242	181	255	240	253
141	222	153	176	242	181	255	240	253
142	223	154	177	243	182	255	240	253
143	223	154	177	243	182	255	240	253
144	224	155	178	244	183	255	241	253
145	224	155	178	244	183	255	241	253
146	225	156	178	244	184	255	241	253
147	225	157	178	244	184	255	241	253
148	226	158	179	244	185	255	242	253
149	227	158	179	244	185	255	242	253
150	228	159	180	245	186	255	242	253
151	228	159	180	245	186	255	242	253
152	229	160	181	245	187	255	242	253
153	229	160	181	245	187	255	242	253
154	230	161	182	246	188	255	243	253
155	230	162	182	246	188	255	243	253
156	231	163	183	246	189	255	243	253
157	231	163	183	246	189	255	243	253
158	232	164	184	247	190	255	243	253
159	232	164	184	247	190	255	243	253
160	233	165	185	247	191	255	244	253
161	233	165	185	247	191	255	244	253
162	234	166	185	247	192	255	244	253
163	234	167	185	247	192	255	244	253
164	235	168	186	248	193	255	244	253

表 90（续）

ctxIdx	treeIdx							
	0	1	2	3	4	5	6	7
165	235	168	186	248	193	255	244	253
166	236	169	187	248	194	255	244	253
167	236	169	187	248	194	255	244	253
168	236	170	188	248	195	255	245	253
169	236	170	188	248	195	255	245	253
170	237	171	189	249	196	255	245	254
171	237	172	189	249	196	255	245	254
172	238	173	190	249	197	255	245	254
173	238	173	190	249	197	255	245	254
174	239	174	191	249	198	255	245	254
175	239	174	191	249	198	255	245	254
176	240	175	192	249	199	255	246	254
177	240	176	192	249	199	255	246	254
178	240	177	193	250	200	255	246	254
179	240	177	193	250	200	255	246	254
180	241	178	194	250	201	255	246	254
181	241	178	194	250	201	255	246	254
182	242	179	195	250	202	255	246	254
183	242	180	195	250	202	255	246	254
184	242	181	196	250	203	255	247	254
185	242	181	196	250	203	255	247	254
186	243	182	197	251	204	255	247	254
187	243	183	197	251	204	255	247	254
188	244	184	198	251	205	255	247	254
189	244	184	198	251	205	255	247	254
190	244	185	199	251	206	255	247	254
191	244	185	199	251	206	255	247	254
192	245	186	200	251	207	255	247	254
193	245	187	200	251	207	255	247	254
194	246	188	201	252	207	255	248	254
195	246	188	201	252	207	255	248	254
196	246	189	202	252	208	255	248	254
197	246	190	202	252	208	255	248	254

表 90（续）

ctxIdx	treeIdx							
	0	1	2	3	4	5	6	7
198	247	191	203	252	209	255	248	254
199	247	191	203	252	209	255	248	254
200	247	192	204	252	210	255	248	254
201	247	193	204	252	210	255	248	254
202	248	194	205	252	211	255	248	254
203	248	194	205	252	211	255	248	254
204	248	195	206	252	212	255	249	254
205	248	196	206	252	212	255	249	254
206	249	197	207	253	213	255	249	254
207	249	197	207	253	213	255	249	254
208	249	198	208	253	214	255	249	254
209	249	199	209	253	214	255	249	254
210	250	200	210	253	215	255	249	254
211	250	200	210	253	215	255	249	254
212	250	201	211	253	215	255	249	254
213	250	202	211	253	215	255	249	254
214	250	203	212	253	216	255	249	254
215	250	203	212	253	216	255	249	254
216	251	204	213	253	217	255	250	254
217	251	205	213	253	217	255	250	254
218	251	206	214	254	218	255	250	254
219	251	206	215	254	218	255	250	254
220	252	207	216	254	219	255	250	254
221	252	208	216	254	219	255	250	254
222	252	209	217	254	220	255	250	254
223	252	210	217	254	220	255	250	254
224	252	211	218	254	221	255	250	254
225	252	212	218	254	221	255	250	254
226	253	213	219	254	222	255	250	254
227	253	213	220	254	222	255	250	254
228	253	214	221	254	223	255	250	254
229	253	215	221	254	223	255	250	254
230	253	216	222	254	224	255	251	254

表 90（续）

ctxIdx	treeIdx							
	0	1	2	3	4	5	6	7
231	253	217	223	254	224	255	251	254
232	253	218	224	254	225	255	251	254
233	253	219	224	254	225	255	251	254
234	254	220	225	254	225	255	251	254
235	254	221	226	254	225	255	251	254
236	254	222	227	255	226	255	251	254
237	254	223	227	255	226	255	251	254
238	254	224	228	255	227	255	251	254
239	254	225	229	255	227	255	251	254
240	254	226	230	255	228	255	251	254
241	254	227	230	255	229	255	251	254
242	255	228	231	255	230	255	251	254
243	255	229	232	255	230	255	251	254
244	255	230	233	255	231	255	252	254
245	255	231	234	255	231	255	252	254
246	255	232	235	255	232	255	252	254
247	255	233	236	255	232	255	252	254
248	255	235	237	255	233	255	252	254
249	255	236	238	255	234	255	252	254
250	255	238	240	255	235	255	252	255
251	255	239	241	255	235	255	252	254
252	255	241	243	255	236	255	252	254
253	255	243	245	255	237	255	252	254
254	255	246	247	255	239	255	253	255
255	255	246	247	255	239	255	253	255

在 token 大于等于 5 时，需要计算 extra。

token 等于 5 时，extra 使用 svac2_cat1_prob 概率表，见表 91。

表 91　svac2_cat1_prob

ctx	binIdx
	0
0	159

token 等于 6 时，extra 使用 svac2_cat2_prob 概率表，见表 92。

表 92　svac2_cat2_prob

ctx	binIdx	
	0	1
0	165	145

token 等于 7 时，extra 使用 svac2_cat3_prob 概率表，见表 93。

表 93　svac2_cat3_prob

ctx	binIdx		
	0	1	2
0	173	148	140

token 等于 8 时，extra 使用 svac2_cat4_prob 概率表，见表 94。

表 94　svac2_cat4_prob

ctx	binIdx			
	0	1	2	3
0	176	155	140	135

token 等于 9 时，extra 使用 svac2_cat5_prob 概率表，见表 95。

表 95　svac2_cat5_prob

ctx	binIdx				
	0	1	2	3	4
0	180	157	141	134	130

token 等于 10 时，extra 使用 svac2_cat6_prob 概率表，见表 96。

表 96　svac2_cat6_prob

ctx	binIdx													
	0	1	2	3	4	5	6	7	8	9	10	11	12	13
0	254	254	254	252	249	243	230	196	177	153	140	133	130	129

下述语法元素使用固定概率值 128 进行解析：tx_mode，tx_mode_delta，coeff_update_prob_flag，not_single_ref，not_compound_ref，tile_idx，sao_merge_flag，sao_merge_type，sao_mode，sao_type，sao_start_band，sao_offset_abs，sao_offset_sign，sao_edge_type，sao_edge_offset[compIdx][0]，sao_edge_offset[compIdx][1]，sao_edge_offset[compIdx][2]，sao_edge_offset[compIdx][3]，coeff_sign，mpm_idx0，mpm_idx1，rem_pred_intra_mode，chroma_intra_mode。

5.4.2.2.2 概率表的更新

5.4.2.2.2.1 svac2_diff_update_prob

每帧图像的编码片开始解码前，会根据条件选择是否对语法元素的概率表进行更新。svac2_diff_update_prob 部分用以更新 tx_probs、switchable_interp_prob、newmv_prob、zeromv_prob、refmv_prob、comp_inter_prob、single_ref_prob、comp_ref_prob、coef_probs、skip_probs、alf_ctu_enable_prob、intra_inter_prob、partition_prob 和 abs_delta_q_prob 的概率表。

此过程的输入为原始的概率值 prob，输出为更新后的概率值 prob。

过程如下：

```
svac2_diff_update_prob(prob) {
  update_prob = ae(252)
  if(update_prob == 1){
    deltaProb = decode_term_subexp();
    prob = inv_remap_prob(deltaProb, prob);
  }
}
```

首先使用固定概率 252 通过 ae(v)解码出 1bit。若为 0 则无概率更新，采用默认初始化的概率值；若为 1 则表示概率做了更新，继续解码并计算出所采用的新的概率。

概率更新分为两步，第一步解码得到概率数组的索引 deltaProb：

```
decode_term_subexp() {
  bit = ae(128)
  if ( bit == 0 ) {
    sub_exp_val = u(4)
    return sub_exp_val
  }
  bit = ae(128)
  if ( bit == 0 ) {
    sub_exp_val_minus_16 = u(4)
    return (sub_exp_val_minus_16 + 16)
  }
  bit = ae(128)
  if ( bit == 0 ) {
    sub_exp_val_minus_32 = u(5)
    return (sub_exp_val_minus_32 + 32)
  }
  sub_exp_val = u(7)
  if (sub_exp_val <65) {
    return (sub_exp_val + 64)
  }
  bit = ae(128)
  return (sub_exp_val << 1) - 1 + bit
}
```

第二步根据索引 deltaProb，得到最终的概率：

```
inv_remap_prob(deltaProb, prob) {
  m = p
  v = delp
  v = inv_map_table[v]
  m--
  if ( (m << 1) <= 255 )
    m = 1 + inv_recenter_nonneg( v, m )
  else
    m = 255 - inv_recenter_nonneg( v, 255 - 1 - m )
  return m
}
```

inv_map_table 的定义如下：

```
inv_map_table[254] =
{
6, 19, 32, 45, 58, 71, 84, 97, 110, 123, 136, 149, 162, 175, 188,
201 , 214 , 227 , 240 , 253 , 0 , 1 , 2 , 3 , 4 , 5 , 7 , 8 , 9 , 10,
11 , 12 , 13 , 14 , 15 , 16 , 17 , 18 , 20 , 21 , 22 , 23 , 24 , 25 , 26,
27 , 28 , 29 , 30 , 31 , 33 , 34 , 35 , 36 , 37 , 38 , 39 , 40 , 41 , 42,
43 , 44 , 46 , 47 , 48 , 49 , 50 , 51 , 52 , 53 , 54 , 55 , 56 , 57 , 59,
60 , 61 , 62 , 63 , 64 , 65 , 66 , 67 , 68 , 69 , 70 , 72 , 73 , 74 , 75,
76 , 77 , 78 , 79 , 80 , 81 , 82 , 83 , 85 , 86 , 87 , 88 , 89 , 90 , 91,
92 , 93 , 94 , 95 , 96 , 98 , 99 , 100 , 101 , 102 , 103 , 104 , 105 , 106 , 107,
108, 109, 111, 112, 113, 114, 115, 116, 117, 118, 119, 120, 121, 122, 124,
125, 126, 127, 128, 129, 130, 131, 132, 133, 134, 135, 137, 138, 139, 140,
141, 142, 143, 144, 145, 146, 147, 148, 150, 151, 152, 153, 154, 155, 156,
157, 158, 159, 160, 161, 163, 164, 165, 166, 167, 168, 169, 170, 171, 172,
173, 174, 176, 177, 178, 179, 180, 181, 182, 183, 184, 185, 186, 187, 189,
190, 191, 192, 193, 194, 195, 196, 197, 198, 199, 200, 202, 203, 204, 205,
206, 207, 208, 209, 210, 211, 212, 213, 215, 216, 217, 218, 219, 220, 221,
222, 223, 224, 225, 226, 228, 229, 230, 231, 232, 233, 234, 235, 236, 237,
238, 239, 241, 242, 243, 244, 245, 246, 247, 248, 249, 250, 251, 252
};
```

inv_recenter_nonneg(v,m)计算如下：

```
if ( v >2 * m )
  return v
if ( v & 1 )
  return m - ((v + 1) >> 1)
return m + (v >> 1)
```

5.4.2.2.2.2 update_mv_probs

每帧图像的编码片开始解码前，会根据条件选择是否对语法元素的概率表进行更新。update_mv_probs 部分用以更新 mv_joint_probs、mv_sign_probs、mv_bits_probs、mv_class0_bit_probs、mv_class_

probs、mv_class0_fr_probs、mv_class0_hp_probs、mv_fr_probs 和 mv_hp_probs 的概率表。

此过程的输入为原始的概率值 prob，输出为更新后的概率值 prob。

过程如下：

```
update_mv_probs(prob) {
  for(i = 0;i<n;i + + ) {
    update_prob = ae(252)
    if(update_prob == 1){
      prob[i] = (u(7) << 1) | 1
    }
  }
}
```

n 为概率表的元素个数。

首先使用固定概率 252 通过 ae(v)解码出 1bit。若为 0 则无概率更新，采用默认初始化的概率值；若为 1 则表示概率做了更新，继续解码并计算出所采用的新的概率。

5.4.2.2.3 初始化算术码解码器

range、count、value、buffer、buffer_end 为算术解码器的变量。range 的位宽为 8 比特，value 位宽为 32 比特。算术解码器的初始化用如下：

第一步，变量初始化：

```
buffer = 码流地址
buffer_end = 码流地址 + 码流字节长度
range = 255
value = 0
count = -8
```

第二步，执行算术解码器的重整化过程：

```
buffer_t = buffer
buffer_start = buffer
value_t = value
count_t = count
loop_end = 0
shift = 16 - count
bytes_left = buffer_end - buffer
bits_left = bytes_left × 8
x = shift + 8 - bits_left
if(x >= 0){
    count_t + = 0x40000000
    loop_end = x
}
if(x <0 || bits_left){
    while (shift >= loop_end){
        count_t + = 8
        value_t | = *buffer_t + + << shift
        shift - = 8
```

```
        }
    }
    buffer += buffer_t - buffer_start
    value = value_t
    count = count_t
```

第三步，使用固定概率 128 通过 ae(v)解码 1bit，如等于 0，算术解码器初始化成功，否则算术解码器初始化失败。

5.4.2.3 二值化

各语法元素的二值化过程如下：

hasRows 与 hasCols 均等于 1 时，语法元素 partition 的取值与二进制位串的对应关系见表 97。hasRows 等于 0 且 hasCols 等于 1 时，语法元素 partition 的取值与二进制位串的对应关系见表 98。hasRows 等于 1 且 hasCols 等于 0 时，语法元素 partition 的取值与二进制位串的对应关系见表 99。

表 97 hasRows 和 hasCols 均等于 1 时 partition 与二进制位串的关系

partition	二进制位串		
PARTITION_NONE	0		
PARTITION_HORZ	1	0	
PARTITION_VERT	1	1	0
PARTITION_SPLIT	1	1	1
binIdx	0	1	2

表 98 hasCols 等于 1 时 partition 与二进制位串的关系

partition	二进制位串
PARTITION_HORZ=1	1
PARTITION_SPLIT=3	0
binIdx	0

表 99 hasRows 等于 1 时 partition 与二进制位串的关系

partition	二进制位串
PARTITION_VERT=2	1
PARTITION_SPLIT=3	0
binIdx	0

语法元素 segment_id 的值与二进制位串的关系见表 100。

表 100　segment_id 与二进制位串的关系

segment_id	二进制位串		
0	0	0	0
1	0	0	1
2	0	1	0
3	0	1	1
4	1	0	0
5	1	0	1
6	1	1	0
7	1	1	1
binIdx	0	1	2

seg_id_predicted 与二进制位串的关系见表 101。

表 101　seg_id_predicted 与二进制位串的关系

seg_id_predicted 取值	二进制位串
0	0
1	1
binIdx	0

语法元素 tx_size 的值与二进制位串的关系见表 102。

表 102　变换块大小语法元素与二进制位串的关系

tx_size	二进制位串		
TX_4×4	0		
TX_8×8	1	0	
TX_16×16	1	1	0
TX_32×32	1	1	1
binIdx	0	1	2

skip_flag 与二进制位串的关系见表 103。

表 103　skip_flag 与二进制位串的关系

skip_flag 取值	二进制位串
0	0
1	1
binIdx	0

inter_block 与二进制位串的关系见表 104。

表 104　inter_block 与二进制位串的关系

inter_block 取值	二进制位串
0	0
1	1
binIdx	0

prev_intra_luma_pred_flag 与二进制位串的关系见表 105。

表 105　prev_intra_luma_pred_flag 与二进制位串的关系

prev_intra_luma_pred_flag 取值	二进制位串
0	0
1	1
binIdx	0

uv_fllow_y_flag 与二进制位串的关系见表 106。

表 106　uv_fllow_y_flag 与二进制位串的关系

uv_fllow_y_flag 取值	二进制位串
0	0
1	1
binIdx	0

帧间参考帧模式 REFERENCE_MODE 语法元素与二进制位串的关系见表 107。

表 107　block_reference_mode 与二进制位串的关系

block_reference_mode	二进制位串	
SINGLE_REFERENCE = 0	0	
COMPOUND_REFERENCE = 1	1	0
REFERENCE_MODE_SELECT = 2	1	1
binIdx	0	1

ref_frame 值与二进制位串的关系见表 108 和表 109。

如果 block_reference_mode 等于 SINGLE_REFERENCE，那么参考帧类型 ref_frame 二值化见表 108；如果 block_reference_mode 等于 COMPOUND_REFERENCE，那么参考帧类型 ref_frame 二值化见表 109。

表 108 SINGLE_REFERENCE 模式下 ref_frame 与二进制位串的关系

ref_per_frame	ref_frame	二进制位串			
2	DYNAMIC_REF	0			
	STATIC_REF	1			
3	DYNAMIC_REF	0			
	STATIC_REF	1	0		
	OPTIONAL_REF	1	1		
4	DYNAMIC_REF	0			
	STATIC_REF	1	0		
	OPTIONAL_REF	1	1	0	
	DYNAMIC_REF_1	1	1	1	
5	DYNAMIC_REF	0			
	STATIC_REF	1	0		
	OPTIONAL_REF	1	1	0	
	DYNAMIC_REF_1	1	1	1	0
	DYNAMIC_REF_2	1	1	1	1
binIdx		0	1	2	3

表 109 COMPOUND_REFERENCE 模式下 ref_frame 与二进制位串的关系

ref_per_frame	ref_frame	二进制位串		
3	DYNAMIC_REF	0		
	STATIC_REF	1		
4	DYNAMIC_REF	0		
	STATIC_REF	1	0	
	DYNAMIC_REF_1	1	1	
5	DYNAMIC_REF	0		
	STATIC_REF	1	0	
	DYNAMIC_REF_1	1	1	0
	DYNAMIC_REF_2	1	1	1
binIdx		0	1	2

mv_mode 的值与二进制位串的关系见表 110。

表 110 mv_mode 与二进制位串的关系

mv_mode	二进制位串		
NEARESTMV	1	1	0

表 110（续）

mv_mode	二进制位串		
NEARMV	1	1	1
ZEROMV	1	0	
NEWMV	0		
binIdx	0	1	2

interp_filter_mode 的值与二进制位串的关系见表 111。

表 111　interp_filter_mode 与二进制位串的关系

interp_filter_mode	二进制位串		
0（Regular）	0		
1（Smooth-1）	1	0	
2（Sharp）	1	1	
3（Smooth-2）	1	1	1
binIdx	0	1	2

编码块尺寸小于 8×8 或 is_compound 等于 1 时，mv_joint 的值与二进制位串的关系见表 112。

表 112　mv_joint 与二进制位串的关系-1

mv_joint	二进制位串		
MV_JOINT_HNZVNZ = 0 Both components nonzero	0		
MV_JOINT_HNZVZ = 1， Vert zero，hor nonzero	1	0	
MV_JOINT_HZVNZ = 2， Hor zero，vert nonzero	1	1	0
MV_JOINT_ZERO = 3 Zero vector	1	1	1
binIdx	0	1	2

如果编码块尺寸不小于 8×8 且 is_compound 等于 0 时，mv_joint 的值与二进制位串的关系见表 113。

表 113　mv_joint 与二进制位串的关系-2

mv_joint	二进制位串	
MV_JOINT_HNZVNZ = 0 Both components nonzero	0	

表 113（续）

mv_joint	二进制位串	
MV_JOINT_HNZVZ = 1，Vert zero，hor nonzero	1	0
MV_JOINT_HZVNZ = 2，Hor zero，vert nonzero	1	1
binIdx	0	1

mvd_sign_0 和 mvd_sign_1 与二进制位串的关系见表 114。

表 114　mvd_sign_0 和 mvd_sign_1 与二进制位串的关系

mvd_sign_0	mvd_sign_1	二进制位串
0	0	0
1	1	1
binIdx		0

mvd_value_0 和 mvd_value_1 的二值化，使用 ae(v)解析非零部分的整数样点部分所对应的 mv_class，其取值与二进制位串的关系见表 115。

表 115　mv_class 与二进制位串的关系

mv_class	二进制位串						
MV_CLASS_0 = 0	0						
MV_CLASS_1 = 1	1	0					
MV_CLASS_2 = 2	1	1	0	0			
MV_CLASS_3 = 3	1	1	0	1			
MV_CLASS_4 = 4	1	1	1	0	0		
MV_CLASS_5 = 5	1	1	1	0	1		
MV_CLASS_6 = 6	1	1	1	1	0		
MV_CLASS_7 = 7	1	1	1	1	1	0	0
MV_CLASS_8 = 8	1	1	1	1	1	0	1
MV_CLASS_9 = 9	1	1	1	1	1	1	0
MV_CLASS_10 = 10	1	1	1	1	1	1	1
binIdx	0	1	2	3	4	5	6

而后根据 mv_class 的值进行 ae(v)解析其整数样点位置偏移 d，d 二值化后的二进制串的长度见表 116，取值为 d 的二进制补码值，高位在前。

表 116　d 的解码长度与 mv_class 的关系

mv_class	d 的解码长度
MV_CLASS_0 = 0	1
MV_CLASS_1 = 1	1
MV_CLASS_2 = 2	2
MV_CLASS_3 = 3	3
MV_CLASS_4 = 4	4
MV_CLASS_5 = 5	5
MV_CLASS_6 = 6	6
MV_CLASS_7 = 7	7
MV_CLASS_8 = 8	8
MV_CLASS_9 = 9	9
MV_CLASS_10 = 10	10

最后 ae(v)解析非零部分的分数样点的位置偏移 fr，fr 的值与二进制位串的关系见表 117。

表 117　fr 与二进制位串的关系

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

hp 与二进制位串的关系见表 118。

表 118　hp 与二进制位串的关系

usehp	hp	二进制位串
0	1	无
0	1	无
1	0	0
1	1	1

usehp 等于 1 表示 MV 使用高精度部分。

由 ae(v)解析出的 mv_class 计算 MV 偏移的基准：

mv_class_base = mv_class ? 2 << (mv_class + 2) : 0

由 ae(v)解析出的 d 和 fr 计算 MV 在基准上的偏移，其中 hp 高精度部分，默认为 1：

offset = (d << 3) | (fr << 1) | hp

mvd_value_0 或 mvd_value_1 的值即等于(mv_class_base+offset+1)。

dqp_abs 与二进制位串的关系见表 119。当 dqp_abs 大于 7 时，dqp_abs 减 7 后的部分用 0 阶指数哥伦布码。

表 119　dqp_abs 与二进制位串的关系

dqp_abs	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	
6	1	1	1	1	1	1	0
7	1	1	1	1	1	1	1
binIdx	0	1	2	3	4	5	6

dqp_sign 与二进制位串的关系见表 120。

表 120　dqp_sign 与二进制位串的关系

dqp_sign	二进制位串
0	0
1	1
binIdx	0

alf_ctu_enable 与二进制位串的关系见表 121。

表 121　alf_ctu_enable 与二进制位串的关系

alf_ctu_enable	二进制位串
0	0
1	1
binIdx	0

块系数 coeff_value 的二值化，包括 token 和 extra 两个部分。当 coeff_value 等于 EOB 或者小于 5 时，coeff_value 等于 token。当 coffe_value 大于 5 时，对应关系如下：

coeff_value＝3＋2^(token－4)＋extra

连续 N(N＞＝0)个等于 0 的系数与一个非零系数可以用一组 token 和 extra 表示。

token 的第一个二进制位为 EOB 指示位，如果该位为 0，表示对应的 token 为 EOB，见表 122。EOB 解析使用 binIdx 等于 0 的 coef_probs 概率表。

EOB 指示位后为 0 系数指示位，长度为 N，对应的 token 等于 0，见表 123。判断 token 是否为 0 的解析，使用 binIdx 等于 1 的 coef_probs 概率表。

表 122　EOB 与二进制位串的关系

token	token EOB 指示二进制位
EOB	0
非 EOB	1
binIdx	0

表 123　0 系数指示位与二进制位串的关系

token	token 0 系数指示二进制位
0	0
非 0	1
binIdx	0

token 与二进制位串的对应关系见表 124，其中，binIdx 等于 0 的位为 0 系数指示位。解析 binIdx 等于 1 的位使用 binIdx 等于 2 的 coef_probs 概率表，之后的位解析使用 svac2_pareto8_full 概率表。

表 124　token 与二进制位串的关系

token	二进制位串					
0	0					
1	1	0				
2	1	1	0	0		
3	1	1	0	1	0	
4	1	1	0	1	1	
5	1	1	1	0	0	
6	1	1	1	0	1	
7	1	1	1	1	0	0
8	1	1	1	1	0	1
9	1	1	1	1	1	0
10	1	1	1	1	1	1
binIdx	0	1	2	3	4	5

extra 二值化的位串长度与 token 取值的对应关系见表 125，取值为 extra 的二进制补码值，高位在前。

表 125　extra 二值化的位串长度与 token 的对应关系

token	extra 二进制位串长度
0	无
1	无
2	无

表 125（续）

token	extra 二进制位串长度
3	无
4	无
5	1
6	2
7	3
8	4
9	5
10	14

tx_mode_delta，coeff_update_prob_flag，not_single_ref，not_compound_ref，sao_merge_flag，sao_merge_type，sao_mode，sao_type，sao_offset_sign， coeff_sign，mpm_idx0，rem_pred_intra_mode，sao_edge_offset[compIdx][1]，sao_edge_offset[compIdx][2]的取值等于 0 时，二进制位串为'0'；等于 1 时，二进制位串为'1'。

tx_mode， sao_edge_type，mpm_idx1，chroma_intra_mode 的二进制位串的长度为 2，取值为语法元素值的二进制补码值，高位在前。

sao_start_band 的二进制位串的长度为 5，取值为语法元素值的二进制补码值，高位在前。

tile_idx，sao_offset_abs，sao_edge_offset[compIdx][0]， sao_edge_offset[compIdx][3]的二进制位串等于其值对应的 0 阶无符号指数哥伦布码。

5.4.2.4 二进制位串解析

5.4.2.4.1 概述

本过程的输入是 5.4.2.2 的初始化 ctx 索引所对应的概率状态以及算术解码引擎中的内部状态变量。

本过程的输出是二进制码值。

5.4.2.4.2 相关变量与含义

相关变量与含义如下：

——block_size：编码块的尺寸；

——AvailU：为 1，上边块可用；为 0，上边块不可用；

——AvailL：为 1，左边块可用；为 0，左边块不可用；

——AboveIntra：为 1，上边块为 Intra；为 0，上边块非 Intra；

——LeftIntra：为 1，左边块为 Intra；为 0，左边块非 Intra；

——AboveSegIdPredicted：上边块的 seg_id_predicted；

——LeftSegIdPredicted：左边块的 seg_id_predicted；

——AboveSkip：为 1，上边块为 skip；为 0，上边块非 skip；

——LeftSkip：为 1，左边块为 skip；为 0，左边块非 skip；

——maxTxSize：最大变换尺寸；

——AboveTxSize：上边块的变换尺寸；

——LeftTxSize：左边块的变换尺寸；
——idy：子块的纵向坐标；
——idx：子块的横向坐标；
——AbovePredMode：上块的预测模式；
——LeftPredMode：左块的预测模式；
——AboveSingle：为 1，上块为 single reference 模式；为 0，上边块为 compound reference 模式；
——LeftSingle：为 1，左块为 single reference 模式；为 0，左块为 compound reference 模式；
——CompFixedRef：参考帧预测时根据上下文得到的参考帧类型；
——CurSelectRef：single reference 模式下，对于 binIdx 等于 3 和 4 的情况，值分别为 OPTIONAL_REF 和 DYANMIC_REF_1；
——AboveInter：为 1，上边块为 inter；为 0，上边块非 inter；
——LeftInter：为 1，左边块为 inter；为 0，左边块非 inter；
——AboveInterpFilter：上边块的 interp_filter_mode；
——LeftInterpFilter：左边块的 interp_filter_mode；
——tx_size：变换块尺寸；
——c：coefs 系数的计数值；
——plane：0 表示亮度，1 表示色度；
——is_plane_y：为 1 表示为亮度；为 0 表示色度；
——tx_type：变换类型；
——AboveAlfEnable：为 1，上块自适应滤波有效；为 0，则无效；
——LeftAlfEnable：为 0，左块自适应滤波有效；为 0，则无效；
——above_seg_context[]：上行块的 ctx；
——left_seg_context[]：左列块的 ctx；
——AboveMode[]：上块的预测模式；
——LeftMode[]：左块的预测模式；
——CurMode[]：当前块的预测模式；
——AboveRefFrame[]：上块的参考帧；
——LeftRefFrame[]：左块的参考帧；
——mode_context[]：参考帧模式的上下文索引；
——CurRefFrame[]：当前块的参考帧；
——AboveNonzeroContext[]：上行的非零系数上下文索引；
——LeftNonzeroContext[]：左列的非零系数上下文索引。

5.4.2.4.3 treeIdx 的计算过程

一些语法元素在查找概率表时，需要使用 treeIdx 和对应数组计算出对应的 binIdx。

这些语法元素包括：partition、segment_id、mv_joint、mvd_value_0 和 mvd_value_1、interp_filter_mode、dqp_abs 及 token。

partition：

在 hasRows 和 hasCols 均等于 1 时，使用数组 partition_tree，计算 binIdx 所对应的 treeIdx。

partition_tree[6] = {−0, 2, −1, 4, −2, −3}

segment_id：使用 segment_tree：

segment_tree[14] = {2, 4, 6, 8, 10, 12, 0, −1, −2, −3, −4, −5, −6, −7}

interp_filter_mode：使用 interp_filter_tree：

interp_filter_tree[6] = {-0, 2,4, -2, -1, -3}

mv_joint:树的选择依赖于 is_compound:

——如果 is_compound 等于 1,选择 mv_joint_tree。

——否则如果 is_compound 等于 0,选择 mv_joint_tree2。

mv_joint_tree[6] = {-0, 2,-1, 4,-2, -3}

mv_joint_tree2[4] = {-0, 2,-1, -2}

mvd_value_0 和 **mvd_value_1** 主要有 mv_class、mv_class0_fp 和 mv_fp 需要通过 treeIdx 计算 binIdx。

mv_class:使用 mv_class_tree:

mv_class_tree[20] = {-0, 2,-1, 4,6, 8,-2, -3,10, 12,-4, -5,-6, 14,16, 18,-7, -8,-9, -10}

mv_class0_fp 和 **mv_fp**:均使用 mv_fp_tree:

mv_fp_tree[6] = {-0, 2,-1, 4,-2, -3}

token:

在使用 svac2_pareto8_full[ctx][treeIdx]概率表计算时,需要使用 coeff_subtree_high:

coeff_subtree_high[16] = {2, 6, -2, 4,-3, -4, 8, 10, -5, -6, 12, 14, -7, -8, -9, -10}

上述变量解析时,treeIdx 通过如下计算得到:

设 T 为整型数组组成的树,P 为变量所对应的概率表。初始令 treeIdx 等于 0,n 等于 0。

以 treeIdx 索引获得的概率按照 5.4.2.4.5 解析出一个二进制位 B。如果语法元素解析完成,则本过程结束;否则令 n 等于 T[n + B], treeIdx 等于 n>>1,继续获取下一个二进制位的概率进行解析。

5.4.2.4.4 概率选择过程

此过程的输入为语法元素的 binIdx 及 treeIdx,输出为解析该语法元素二进制位所用的概率值 prob。

以下为各个语法元素的概率计算:

alf_ctu_enable:

概率等于 alf_ctu_enable_prob[ctx],其中 ctx 的计算如下:

```
if(AvailL && AvailU)
  ctx = AboveAlfEnable && LeftAlfEnable ? 3: AboveAlfEnable || LeftAlfEnable
else if(AvaiL || AvailU)
  ctx = 2 * (AvailL ? LeftAlfEnable : AboveAlfEnable)
else
  ctx = 1
```

partition:

如果 frame_type 等于 0,概率值等于 intra_partition_probs[ctx][treeIdx];否则,概率值等于 inter_partition_probs[ctx][treeIdx]。

在 hasRows 和 hasCols 均等于 1 的情况下,索引二进制位对应的概率的 treeIdx 的计算见 5.4.2.4.3 节;当 hasCols 等于 1 且 hasRows 等于 0 时,treeIdx 等于 1;当 hasRows 等于 1 且 hasCols 等于 0 时,treeIdx 等于 2。

索引 ctx 的值根据如下方式计算得到:

```
above = 0
left = 0
```

```
bsl = mi_width_log2_lookup[bsize]
bs = 1 << bsl
for (i = 0; i <bs; i++){
  above |= above_seg_context[i]
  left |= left_seg_context[i]
}
above = (above & bs) >0
left  = (left & bs)>0
ctx = (left * 2 + above) + bsl * 4
```

mi_width_log2_lookup[BLOCK_SIZES] ={0, 0, 0, 0, 0, 1, 1, 1, 2, 2, 2, 3, 3, 3, 4, 4}

segment_id：

概率等于 seg_tree_probs[treeIdx]，其中 treeIdx 的计算见 5.4.2.4.3。

seg_id_predicted：

概率等于 seg_pred_probs[ctx]，其中 ctx 的计算如下：

```
ctx = 0
if(AvailU)
  ctx += AboveSegIdPredicted
if(AvailL)
  ctx += LeftSegIdPredicted
```

skip_flag：

概率等于 skip_prob[ctx]，其中 ctx 的计算如下：

```
ctx = 0
if(AvailU)
  ctx += AboveSkip
if(AvailL)
  ctx += LeftSkip
```

inter_block：

概率等于 intra_inter_prob[ctx]，其中 ctx 的计算如下：

```
if(AvailU && AvailL)
  ctx = (LeftIntra && AboveIntra) ? 3 : LeftIntra || AboveIntra
else if ( AvailU || AvailL )
  ctx = 2 * (AvailU ? AboveIntra : LeftIntra)
else
  ctx = 0
```

tx_size：

概率值等于 tx_probs[MAX_TX_SIZE][ctx][binIdx]得到，其中 ctx 的计算如下：

```
above = maxTxSize
left = maxTxSize
if ( AvailU && ! AboveSkip)
    above = AboveTxSize
if ( AvailL && ! LeftSkip )
    left = LeftTxSize
if ( ! AvailL )
```

```
        left = above
    if ( ! AvailU )
        above = left
    ctx = (above + left)>maxTxSize
```

maxTxSize 由编码树划分的预测块尺寸决定，见表 126。

表 126　maxTxSize 与预测块尺寸的关系

maxTxSize	partitiond block size
TX_4×4	BLOCK_4×4
TX_4×4	BLOCK_4×8
TX_4×4	BLOCK_8×4
TX_8×8	BLOCK_8×8
TX_8×8	BLOCK_8×16
TX_8×8	BLOCK_16×8
TX_16×16	BLOCK_16×16
TX_16×16	BLOCK_16×32
TX_16×16	BLOCK_32×16
TX_32×32	BLOCK_32×32
TX_32×32	BLOCK_32×64
TX_32×32	BLOCK_64×32
TX_32×32	BLOCK_64×64
TX_32×32	BLOCK_64×128
TX_32×32	BLOCK_128×64
TX_32×32	BLOCK_128×128

prev_intra_luma_pred_flag：

概率等于 mpm_flag_probs[ctx]，其中 ctx 的计算如下：

```
if(AbovePredMode == LeftPredMode) {
  if (LeftPredMode>2)
    ctx = 0
  else
    ctx = 1
} else {
  mode0 = LeftPredMode>AbovePredMode ? AbovePredMode : LeftPredMode
  mode1 = LeftPredMode>AbovePredMode ? LeftPredMode : AbovePredMode
  if (mode0>2) {
    if(mode1-mode0<4)
      ctx = 2
    else if(mode1-mode0<12)
      ctx = 3
```

```
        else
          ctx = 4
      }else {
        if(mode1 == 3)
          ctx = 5
        else
          ctx = 6
      }
    }
```

uv_fllow_y_flag：

概率固定为 175。

block_reference_mode：

概率等于 comp_inter_prob[ctx]，其中 ctx 的计算如下：

```
    if ( AvailU && AvailL ) {
      if ( AboveSingle && LeftSingle )
        ctx = (AboveRefFrame[0] == CompFixedRef) ^ (LeftRefFrame[0] == CompFixedRef)
      else if ( AboveSingle )
        ctx = 2 + (AboveRefFrame[0] == CompFixedRef || AboveIntra)
      else if ( LeftSingle )
        ctx = 2 + (LeftRefFrame[0] == CompFixedRef || LeftIntra)
      else
        ctx = 4
    } else if ( AvailU ) {
      if ( AboveSingle )
        ctx = AboveRefFrame[0] == CompFixedRef
      else
        ctx = 3
    } else if ( AvailL ) {
      if ( LeftSingle )
        ctx = LeftRefFrame[0] == CompFixedRef
      else
        ctx = 3
    } else {
      ctx = 1
    }
```

ref_frame

根据 block_reference_mode 分为 compound reference 模式和 single reference 模式。

compound reference 模式一共有 3 个 binIdx 的 ctxIdx 要计算；single reference 模式一共有 4 个 binIdx 的 ctxIdx 要计算。

compound reference 模式下 binIdx 等于 0 的情况：

概率等于 comp_ref_prob[ctx][0]，ctx 的计算如下：

```
    VarRefIdx = 0
    if ( AvailU && AvailL ) {
```

```
          if ( AboveIntra && LeftIntra ) {
            ctx = 2
          } else if ( LeftIntra ) {
            if ( AboveSingle )
              ctx = 1 + 2 * ((AboveRefFrame[0] == STATIC_REF) ||
                (AboveRefFrame[0] == DYNAMIC_REF_1) || (AboveRefFrame[0] == DYNAMIC_REF_2))
            else
              ctx = 1 + 2 * ((AboveRefFrame[VarRefIdx] == STATIC_REF) ||
                (AboveRefFrame[VarRefIdx] == DYNAMIC_REF_1) ||
                (AboveRefFrame[VarRefIdx] == DYNAMIC_REF_2))
          } else if ( AboveIntra ) {
            if ( LeftSingle )
        ctx = 1 + 2 * ((LeftRefFrame[0] == STATIC_REF) ||
                (LeftRefFrame[0] == DYNAMIC_REF_1) || (LeftRefFrame[0] == DYNAMIC_REF_2))
            else
        ctx = 1 + 2 * ((LeftRefFrame[VarRefIdx] == STATIC_REF) ||
        (LeftRefFrame[VarRefIdx] == DYNAMIC_REF_1) ||
        (LeftRefFrame[VarRefIdx] == DYNAMIC_REF_2))
          } else {
            vrfa = AboveSingle ? AboveRefFrame[0] : AboveRefFrame[VarRefIdx]
            vrfl = LeftSingle ? LeftRefFrame[0] : LeftRefFrame[VarRefIdx]
            if ( vrfa == vrfl && ((vrfa == STATIC_REF) || (vrfa == DYNAMIC_REF_1) || (vrfa
== DYNAMIC_REF_2)) ) {
              ctx = 0
            } else if ( LeftSingle && AboveSingle ) {
              if ((vrfa == OPTIONAL_REF && vrfl == DYNAMIC_REF ||
                   (vrfl == OPTIONAL_REF && vrfa == DYNAMIC_REF )
                ctx = 4
              else if ( vrfa == vrfl || (vrfl == DYNAMIC_REF && vrfa == DYNAMIC_REF))
                ctx = 3
              else
                ctx = 1
            } else if ( LeftSingle || AboveSingle ) {
              vrfc = LeftSingle ? vrfa : vrfl
              rfs = AboveSingle ? vrfa : vrfl
             if (((vrfc == STATIC_REF) || (vrfc == DYNAMIC_REF_1) || (vrfc == DYNAMIC_REF_2)) &&
! ((vrfs == STATIC_REF) || (vrfs == DYNAMIC_REF_1) || (vrfs == DYNAMIC_REF_2)) )
                ctx = 1
              else if (((rfs == STATIC_REF) || (rfs == DYNAMIC_REF_1) || (rfs == DYNAMIC_REF_
2)) && ! ((vrfc == STATIC_REF) || (vrfc == DYNAMIC_REF_1) || (vrfc == DYNAMIC_REF_2)) )
                ctx = 2
              else
                ctx = 4
```

```
            } else if (((vrfa == STATIC_REF) || (vrfa == DYNAMIC_REF_1) || (vrfa == DYNAMIC
_REF_2)) && ((vrfl == STATIC_REF) || (vrfl == DYNAMIC_REF_1) || (vrfl == DYNAMIC_REF_2))) {
              ctx = 4
            } else {
              ctx = 2
            }
          }
        } else if ( AvailU ) {
          if ( AboveIntra ) {
            ctx = 2
          } else {
            if ( AboveSingle )
              ctx = 3 * (! ((AboveRefFrame[0] == STATIC_REF) || (AboveRefFrame[0] == DY-
NAMIC_REF_1) || (AboveRefFrame[0] == DYNAMIC_REF_2)))
            else
              ctx = 4 * (! ((AboveRefFrame[VarRefIdx] == STATIC_REF) || (AboveRefFrame[VarRe-
fIdx] == DYNAMIC_REF_1) || (AboveRefFrame[VarRefIdx] == DYNAMIC_REF_2)))
          }
        } else if ( AvailL ) {
          if ( LeftIntra ) {
            ctx = 2
          } else {
            if ( LeftSingle )
              ctx = 3 * (! ((LeftRefFrame[0] == STATIC_REF) || (LeftRefFrame[0] ==
DYNAMIC_REF_1) || (LeftRefFrame[0] == DYNAMIC_REF_2)))
            else
              ctx = 4 * (! ((LeftRefFrame[VarRefIdx] == STATIC_REF) || (LeftRefFrame
[VarRefIdx] == DYNAMIC_REF_1) || (LeftRefFrame[VarRefIdx] == DYNAMIC_REF_2)))
          }
        } else {
          ctx = 2
        }
```

compound reference 模式下 binIdx 等于 1 的情况：

概率等于 comp_ref_prob[ctx][1]，ctx 的计算如下：

```
        VarRefIdx = 0
        if ( AvailU && AvailL ) {
          if ( AboveIntra && LeftIntra ) {
            ctx = 2
          } else if ( LeftIntra ) {
            if ( AboveSingle )
              ctx = 1 + 2 * (AboveRefFrame[0] == STATIC_REF)
            else
              ctx = 1 + 2 * (AboveRefFrame[VarRefIdx] == STATIC_REF)
```

```
} else if ( AboveIntra ) {
  if ( LeftSingle )
    ctx = 1 + 2 * (LeftRefFrame[0] == STATIC_REF)
  else
    ctx = 1 + 2 * (LeftRefFrame[VarRefIdx] == STATIC_REF)
} else {
  vrfa = AboveSingle ? AboveRefFrame[0] : AboveRefFrame[VarRefIdx]
  vrfl = LeftSingle ? LeftRefFrame[0] : LeftRefFrame[VarRefIdx]
  if ( vrfa == vrfl && STATIC_REF == vrfa ) {
    ctx = 0
  } else if ( LeftSingle && AboveSingle ) {
    if ( (vrfa == STATIC_REF && vrfl == STATIC_REF) )
      ctx = 4
    else if ( vrfa == STATIC_REF || vrfl == STATIC_REF )
      ctx = 3
    else if (vrfa == DYNAMIC_REF || vrfl == DYNAMIC_REF)
      ctx = 1 + (vrfa == vrfl)
    else
      ctx = 1
  } else if ( LeftSingle || AboveSingle ) {
    vrfc = LeftSingle ? vrfa : vrfl
    rfs = AboveSingle ? vrfa : vrfl
    if ( vrfc == STATIC_REF && rfs == STATIC_REF )
      ctx = 4
    else if ( vrfc == STATIC_REF && rfs ! = STATIC_REF )
      ctx = 3
    else if ( vrfc ! = STATIC_REF && rfs == STATIC_REF )
      ctx = 2 + (! vrfc == DYNAMIC_REF)
    else {
      if (vrfc == DYNAMIC_REF || rfs == DYNAMIC_REF)
        ctx = 2
      else
        ctx = 1
    }
  } else {
    if (vrfa == STATIC_REF && vrfl == STATIC_REF)
      ctx = 4
    else if (vrfa == STATIC_REF || vrfl == STATIC_REF)
      ctx = 3
    else if (vrfa ! = STATIC_REF && vrfl ! = STATIC_REF)
      ctx = 1
    else
      ctx = 2
```

```
        }
      }
    } else if ( AvailU ) {
      if ( AboveIntra ) {
        ctx = 2
      } else {
        if ( ! AboveSingle )
          ctx = 4 * (AboveRefFrame[VarRefIdx] ! = STATIC_REF)
        else {
          if(AboveRefFrame[0] == STATIC_REF)
            ctx = 3
          else if(AboveRefFrame[0] == DYNAMIC_REF)
            ctx = 2
          else
            ctx = 1
        }
      }
    } else if ( AvailL ) {
      if ( LeftIntra ) {
        ctx = 2
      } else {
        if ( ! LeftSingle)
          ctx = 4 * (LeftRefFrame [VarRefIdx] ! = STATIC_REF)
        else {
          if(LeftRefFrame [0] == STATIC_REF)
            ctx = 3
          else if(LeftRefFrame [0] == DYNAMIC_REF)
            ctx = 2
          else
            ctx = 1
        }
      }
    } else {
      ctx = 2
    }
```

compound reference 模式下 binIdx 等于 2 的情况：

概率等于 comp_ref_prob[ctx][2]，ctx 的计算如下：

```
    VarRefIdx = 0
    if ( AvailU && AvailL ) {
      if ( AboveIntra && LeftIntra ) {
        ctx = 2
      } else if ( LeftIntra ) {
        if ( AboveSingle )
```

```
            ctx = 1 + 2 * (AboveRefFrame[0] == DYNAMIC_REF_1)
          else
            ctx = 1 + 2 * (AboveRefFrame[VarRefIdx] == DYNAMIC_REF_1)
        } else if ( AboveIntra ) {
          if ( LeftSingle )
            ctx = 1 + 2 * (LeftRefFrame[0] == DYNAMIC_REF_1)
          else
            ctx = 1 + 2 * (LeftRefFrame[VarRefIdx] == DYNAMIC_REF_1)
        } else {
          vrfa = AboveSingle ? AboveRefFrame[0] : AboveRefFrame[VarRefIdx]
          vrfl = LeftSingle ? LeftRefFrame[0] : LeftRefFrame[VarRefIdx]
          if ( vrfa == vrfl && DYNAMIC_REF_1 == vrfa ) {
            ctx = 0
          } else if ( LeftSingle && AboveSingle ) {
            if ( (vrfa == DYNAMIC_REF_1 && vrfl == DYNAMIC_REF_1) )
              ctx = 4
            else if ( vrfa == DYNAMIC_REF_1 || vrfl == DYNAMIC_REF_1 )
              ctx = 3
            else if (((vrfa == DYNAMIC_REF) || (vrfa == STATIC_REF)) || ((vrfl == DYNAMIC
_REF) || (vrfl == STATIC_REF)))
              ctx = 1 + (vrfa == vrfl)
            else
              ctx = 1
          } else if ( LeftSingle || AboveSingle ) {
            vrfc = LeftSingle ? vrfa : vrfl
            rfs = AboveSingle ? vrfa : vrfl
            if ( vrfc == DYNAMIC_REF_1 && rfs == DYNAMIC_REF_1 )
             ctx = 4
            else if ( vrfc == DYNAMIC_REF_1 && rfs ! = DYNAMIC_REF_1 )
              ctx = 3
            else if ( vrfc ! = DYNAMIC_REF_1 && rfs == DYNAMIC_REF_1 )
              ctx = 2 + (! ((vrfc == DYNAMIC_REF) || (vrfc == STATIC_REF)))
            else if(vrfc ! = DYNAMIC_REF_1 && rfs ! = DYNAMIC_REF_1){
              if (((vrfc == DYNAMIC_REF) || (vrfc == STATIC_REF)) || ((rfs == DYNAMIC_
REF) || (rfs == STATIC_REF)))
                ctx = 2
              else
                ctx = 1
            }
          } else {
            if (vrfa == DYNAMIC_REF_1 && vrfl == DYNAMIC_REF_1)
              ctx = 4
            else if (vrfa == DYNAMIC_REF_1 || vrfl == DYNAMIC_REF_1)
```

```
                ctx = 3
              else if (vrfa ! = DYNAMIC_REF_1 && vrfl ! = DYNAMIC_REF_1)
                ctx = 1
              else
                ctx = 2
            }
          }
        } else if ( AvailU ) {
          if ( AboveIntra ) {
            ctx = 2
          } else {
            if ( ! AboveSingle )
              ctx = 4 * (AboveRefFrame[VarRefIdx] ! = DYNAMIC_REF_1)
            else {
              if(AboveRefFrame[0] == DYNAMIC_REF_1)
                ctx = 3
              else if(((AboveRefFrame[0] == DYNAMIC_REF) || (AboveRefFrame[0] == STATIC_
REF)))
                ctx = 2
              else
                ctx = 1
            }
          }
        } else if ( AvailL ) {
          if ( LeftIntra ) {
            ctx = 2
          } else {
            if ( ! LeftSingle)
              ctx = 4 * (LeftRefFrame [VarRefIdx] ! = DYNAMIC_REF_1)
            else {
              if(LeftRefFrame [0] == DYNAMIC_REF_1)
                ctx = 3
              else if(((LeftRefFrame[0] == DYNAMIC_REF) || (LeftRefFrame[0] == STATIC_REF)))
                ctx = 2
              else
                ctx = 1
            }
          }
        } else {
          ctx = 2
        }
```

single reference 模式下 binIdx 等于 0 的情况：

概率等于 single_ref_prob[ctx][0],ctx 的计算如下：

```
if ( AvailU && AvailL ) {
  if ( AboveIntra && LeftIntra ) {
    ctx = 2
  } else if ( LeftIntra ) {
    if ( AboveSingle )
      ctx = 4 * (AboveRefFrame[0] = = DYNAMIC_REF)
    else
      ctx = 1 + (AboveRefFrame[0] = = DYNAMIC_REF || AboveRefFrame[1] = = DYNAMIC_REF)
  } else if ( AboveIntra ) {
    if ( LeftSingle )
      ctx = 4 * (LeftRefFrame[0] = = DYNAMIC_REF)
    else
      ctx = 1 + (LeftRefFrame[0] = = DYNAMIC_REF || LeftRefFrame[1] = = DYNAMIC_REF)
  } else {
    if ( ! AboveSingle && ! LeftSingle ) {
      ctx = 1 + (AboveRefFrame[0] = = DYNAMIC_REF || AboveRefFrame[1] = = DYNAMIC_REF
         ||  LeftRefFrame[0] = = DYNAMIC_REF || LeftRefFrame[1] = = DYNAMIC_REF)
    } else if ( ! AboveSingle || ! LeftSingle ) {
      rfs = AboveSingle ? AboveRefFrame[0] : LeftRefFrame[0]
      crf1 = AboveSingle ? LeftRefFrame[0] : AboveRefFrame[0]
      crf2 = AboveSingle ? LeftRefFrame[1] : AboveRefFrame[1]
      if ( rfs = = DYNAMIC_REF )
        ctx = 3 + (crf1 = = DYNAMIC_REF || crf2 = = DYNAMIC_REF)
      else
        ctx = crf1 = = DYNAMIC_REF || crf2 = = DYNAMIC_REF
    } else {
      ctx = 2 * (AboveRefFrame[0] = = DYNAMIC_REF) + 2 * (LeftRefFrame[0] = = DYNAMIC_REF)
    }
  }
} else if ( AvailU ) {
  if ( AboveIntra ) {
    ctx = 2
  } else { // inter
    if ( AboveSingle )
      ctx = 4 * (AboveRefFrame[0] = = DYNAMIC_REF)
    else
      ctx = 1 + (AboveRefFrame[0] = = DYNAMIC_REF || AboveRefFrame[1] = = DYNAMIC_REF)
  }
} else if ( AvailL ) {
  if ( LeftIntra ) {
    ctx = 2
```

```
    } else {
      if ( LeftSingle )
        ctx = 4 * (LeftRefFrame[0] = = DYNAMIC_REF)
      else
        ctx = 1 + (LeftRefFrame[0] = = DYNAMIC_REF || LeftRefFrame[1] = = DYNAMIC_REF)
    }
  } else {
    ctx = 2
  }
```

single reference 模式下 binIdx 等于 1 的情况：

概率等于 single_ref_prob[ctx][1],ctx 的计算如下：

```
  if ( AvailU && AvailL ) {
    if ( AboveIntra && LeftIntra ) {
      ctx = 2
    } else if ( LeftIntra ) {
      if ( AboveSingle ) {
        if ( AboveRefFrame[0] = = DYNAMIC_REF )
          ctx = 3
        else
          ctx = 4 * (AboveRefFrame[0] = = STATIC_REF)
      } else {
        ctx = 1 + 2 * (AboveRefFrame[0] = = STATIC_REF || AboveRefFrame[1] = = STATIC_REF)
      }
    } else if ( AboveIntra ) {
      if ( LeftSingle ) {
        if ( LeftRefFrame[0] = = DYNAMIC_REF )
          ctx = 3
        else
          ctx = 4 * (LeftRefFrame[0] = = STATIC_REF)
      } else {
        ctx = 1 + 2 * (LeftRefFrame[0] = = STATIC_REF || LeftRefFrame[1] = = STATIC_REF)
      }
    } else {
      if ( ! AboveSingle && ! LeftSingle ) {
        if(AboveRefFrame[0] = = LeftRefFrame[0] && AboveRefFrame[1] = = LeftRefFrame[1])
          ctx = 3 * (AboveRefFrame[0] = = STATIC_REF || AboveRefFrame[1] = = STATIC_REF)
        else
         ctx = 2
      } else if ( ! AboveSingle || ! LeftSingle ) {
        rfs = AboveSingle ? AboveRefFrame[0] : LeftRefFrame[0]
        crf1 = AboveSingle ? LeftRefFrame[0] : AboveRefFrame[0]
        crf2 = AboveSingle ? LeftRefFrame[1] : AboveRefFrame[1]
        if ( rfs = = STATIC_REF )
```

```
        ctx = 3 + (crf1 == STATIC_REF || crf2 == STATIC_REF)
      else if ( rfs == OPTIONAL_REF )
        ctx = crf1 == STATIC_REF || crf2 == STATIC_REF
      else
        ctx = 1 + 2 * (crf1 == STATIC_REF || crf2 == STATIC_REF)
    }
  } else {
    if ( AboveRefFrame[0] == DYNAMIC_REF && LeftRefFrame[0] == DYNAMIC_REF ) {
      ctx = 3
    } else if ( AboveRefFrame[0] == DYNAMIC_REF ) {
      ctx = 4 * (LeftRefFrame[0] == STATIC_REF)
    } else if ( LeftRefFrame[0] == DYNAMIC_REF ) {
      ctx = 4 * (AboveRefFrame[0] == STATIC_REF)
    } else {
      ctx = 2 * (AboveRefFrame[0] == STATIC_REF) + 2 * (LeftRefFrame[0] == STATIC_REF)
    }
  }
} else if ( AvailU ) {
  if ( AboveIntra || (AboveRefFrame[0] == DYNAMIC_REF && AboveSingle) )
    ctx = 2
  else if ( AboveSingle )
    ctx = 4 * (AboveRefFrame[0] == STATIC_REF)
  else
    ctx = 3 * (AboveRefFrame[0] == STATIC_REF || AboveRefFrame[1] == STATIC_REF)
} else if ( AvailL ) {
  if ( LeftIntra || (LeftRefFrame[0] == DYNAMIC_REF && LeftSingle) )
    ctx = 2
  else if ( LeftSingle )
    ctx = 4 * (LeftRefFrame[0] == STATIC_REF)
  else
    ctx = 3 * (LeftRefFrame[0] == STATIC_REF || LeftRefFrame[1] == STATIC_REF)
} else {
  ctx = 2
}
```

single reference 模式下 binIdx 分别等于 2 和 3 的情况：

概率分别等于 single_ref_prob[ctx][2]和 single_ref_prob[ctx][3]，ctx 的计算如下：

```
if(CurSelectRef == OPTIONAL_REF) {
  is_opt_ref = 1
}
if ( AvailU && AvailL ) {
  if ( AboveIntra && LeftIntra ) {
    ctx = 2
  } else if (LeftIntra) {
```

```
        if ( AboveSingle ) {
          if (AboveRefFrame[0] = = CurSelectRef) {
            ctx = 4
          } else {
            if (is_opt_ref) {
              if (AboveRefFrame[0] = = DYANMIC_REF_1)
                ctx = 0
              else
                ctx = 3
            } else {
              if (AboveRefFrame[0] = = DYANMIC_REF_2)
                ctx = 0
              else
                ctx = 2
            }
          }
        } else {
          ctx = 1 + 2 * (AboveRefFrame[0] = = CurSelectRef|| AboveRefFrame[1] = = CurSelectRef)
        }
      } else if (AboveIntra) {
        if ( LeftSingle ) {
          if (LeftRefFrame[0] = = CurSelectRef) {
            ctx = 4
          } else {
            if (is_opt_ref) {
              if (LeftRefFrame[0] = = DYANMIC_REF_1)
                ctx = 0
              else
                ctx = 3
            } else {
              if (LeftRefFrame[0] = = DYANMIC_REF_2)
                ctx = 0
              else
                ctx = 2
            }
          }
        } else {
          ctx = 1 + 2 * (LeftRefFrame[0] = = CurSelectRef || LeftRefFrame[1] = = CurSelectRef)
        }
      } else {
        if ( ! AboveSingle && ! LeftSingle ) {
          if(AboveRefFrame[0] = = LeftRefFrame[0]&& AboveRefFrame[1] = = LeftRefFrame[1]){
            ctx = 3 * (AboveRefFrame[0] = = CurSelectRef || AboveRefFrame[1] = = CurSelec-
```

```
tRef || LeftRefFrame[0] = = CurSelectRef|| LeftRefFrame[1] = = CurSelectRef)
          } else {
            ctx = 2
          }
        } else if( ! AboveSingle || ! LeftSingle ) {
          rfs = AboveSingle ? AboveRefFrame[0] : LeftRefFrame[0]
          crf1 = AboveSingle ? LeftRefFrame[0] : AboveRefFrame[0]
          crf2 = AboveSingle ? LeftRefFrame[1] : AboveRefFrame[1]
          if ( rfs = = CurSelectRef )
            ctx = 3 + (crf1 = = CurSelectRef || crf2 = = CurSelectRef)
          else {
            if (is_opt_ref)
              ctx = 2
            else {
              if (rfs = = DYNAMIC_REF_2)
                ctx = 2 * (crf1 = = CurSelectRef || crf2 = = CurSelectRef)
              else
                ctx = 2
            }
          }
        } else {
          if (is_opt_ref)
             ctx = 2 * (AboveRefFrame[0] = = OPTIONAL_REF) + 2 * (LeftRefFrame[0] = =
OPTIONAL_REF)
          else {
            if (AboveRefFrame[0] = = DYNAMIC_REF_1 && LeftRefFrame[0] = = DYNAMIC_REF_1)
              ctx = 4
            else if(AboveRefFrame[0] = = DYNAMIC_REF_1&&LeftRefFrame[0]! = DYNAMIC_REF_1)
              ctx = 3
            else if(AboveRefFrame[0]! = DYNAMIC_REF_1&&LeftRefFrame[0] = = DYNAMIC_REF_1)
              ctx = 3
            else
              ctx = 2
          }
        }
      }
    } else if ( AvailU ) {
      if ( AboveIntra || (AboveRefFrame[0] ! = CurSelectRef && AboveSingle)) {
        ctx = 2
      } else if (AboveSingle) {
        ctx = 4 * (AboveRefFrame[0] = = CurSelectRef)
      } else {
        if (is_opt_ref) {
```

```
          if( AboveRefFrame[0] = = DYNAMIC_REF_1 || AboveRefFrame[1] = = DYNAMIC_REF_1 )
            ctx = 1
          else
            ctx = 2
        } else {
          if( AboveRefFrame[0] = = DYNAMIC_REF_1 || AboveRefFrame[1] = = DYNAMIC_REF_1 )
            ctx = 3
          else if(AboveRefFrame[0] = = DYNAMIC_REF_2|| AboveRefFrame[1] = = DYNAMIC_REF_2 )
            ctx = 1
          else
            ctx = 2
        }
    }
    } else if ( AvailL ) {
      if ( LeftIntra || (LeftRefFrame[0] ! = CurSelectRef && LeftSingle)) {
        ctx = 2
      } else if (LeftSingle) {
        ctx = 4 * (LeftRefFrame[0] = = CurSelectRef)
      } else {
        if (is_opt_ref) {
          if(LeftRefFrame[0] = = DYNAMIC_REF_1 || LeftRefFrame[1] = = DYNAMIC_REF_1 )
            ctx = 1
          else
            ctx = 2
        } else {
          if(LeftRefFrame[0] = = DYNAMIC_REF_1 || LeftRefFrame[1] = = DYNAMIC_REF_1 )
            ctx = 3
          else if(LeftRefFrame[0] = = DYNAMIC_REF_2 || LeftRefFrame[1] = = DYNAMIC_REF_2 )
            ctx = 1
          else
            ctx = 2
        }
      }
    } else {
      ctx = 2
    }
```

mv_mode

解析 binIdx 等于 0 的二进制位时，概率等于 newmv_mode_prob[ctx]，ctx 的计算如下：

```
    mode_ctx = mode_context[CurRefFrame[0]]
    if(ref_frame[1]>NONE)
      mode_ctx &= (mode_context[CurRefFrame[1]] | 0x00FF)
    if(block_size<BLOCK_8X8)
      mode_ctx &= 0x00FF;
```

ctx = mode_ctx & ((1 〈〈 ZEROMV_OFFSET)-1)

解析 binIdx 等于 1 的二进制位时，概率等于 zeromv_mode_prob[ctx]，ctx 的计算如下：

```
mode_ctx = mode_context[CurRefFrame[0]]
if(ref_frame[1]>NONE)
  mode_ctx &= (mode_context[CurRefFrame[1]] | 0x00FF)
if(block_size<BLOCK_8X8)
  mode_ctx &= 0x00FF;
ctx = (mode_ctx 〉〉 ZEROMV_OFFSET) & ((1 〈〈 (REFMV_OFFSET-ZEROMV_OFFSET))-1)
```

解析 binIdx 等于 2 的二进制位时，概率等于 refmv_mode_prob[ctx]，ctx 的计算如下：

```
mode_ctx = mode_context[CurRefFrame[0]]
if(ref_frame[1]>NONE)
  mode_ctx &= (mode_context[CurRefFrame[1]] | 0x00FF)
if(block_size<BLOCK_8×8)
  mode_ctx &= 0x00FF;
ctx = (mode_ctx 〉〉 REFMV_OFFSET) & ((1 〈〈 (8-REFMV_OFFSET))-1)
```

mv_joint：

概率等于 mv_joint_probs[treeIdx]，其中 treeIdx 的计算过程见 5.4.2.4.3。

mvd_sign_0：

概率等于 mv_sign_probs[0]。

mvd_sign_1：

概率等于 mv_sign_probs[1]。

mvd_value_0 和 mvd_value_1：通过以下信息计算得到，分别对应 comp 为 0 和 1 的情况。

comp 用来索引以下概率表，为 0 表示垂直坐标，为 1 表示水平坐标。

使用的概率表包括 mv_bits_probs、mv_class_probs、mv_class0_fr_probs、mv_class0_hp_prob、mv_fr_probs 和 mv_hp_prob。

mv_class：

概率等于 mv_class_probs[comp] [treeIdx]。

d：

在 mv_class 等于 0 时，概率通过 mv_class0_bit_probs[comp]得到；

在 mv_class 不等于 0 时，概率通过 mv_bits_probs[comp][bit_offset]得到。

fr：

在 mv_class 等于 0 时，概率通过 mv_class0_fr[comp][mv_class0_bit][treeIdx]得到；

在 mv_class 不等于 0 时，概率通过 mv_fr_probs[comp][treeIdx]得到。

treeIdx 的计算过程见 5.4.2.4.3。

hp：

在 mv_class 等于 0 时，概率通过 mv_class0_hp_prob[comp]得到；

在 mv_class 不等于 0 时，概率通过 mv_hp_prob[comp]得到。

interp_filter_mode：

概率等于 switchable_interp_prob[ctx][treeIdx]，treeIdx 的计算过程见 5.4.2.4.3，ctx 的计算如下：

```
leftInterp = ( AvailL && LeftInter ) ? LeftInterpFilter : 3
aboveInterp = ( AvailU && AboveInter ) ? AboveInterpFilter : 3
if ( leftInterp = = aboveInterp )
```

```
  ctx = leftInterp
else if ( leftInterp = = 3 && aboveInterp ! = 3 )
  ctx = aboveInterp
else if ( leftInterp ! = 3 && aboveInterp = = 3 )
  ctx = leftInterp
else
  ctx = 3
```

dqp_abs

概率等于 dqp_abs_prob[ctx][treeIdx],treeIdx 的计算过程见 5.4.2.4.3,ctx 的计算如下：

```
ctx = last_dqindex ! = 0
```

last_dqindex 为上一个解码出来的 dqp_abs。

coeff_value：

coeff_value 通过 token 和 extra 计算得到。

亮度系数的 token 的概率根据 coef_probs[tx_size][0][inter_block][band][ctx][binIdx]或者 svac2_pareto8_full[ctx][treeIdx]确定。

band 根据 tx_size 查表 127 及表 128 获取：

```
if(tx_size = = TX_4×4)
  band = coefband_trans_4×4[c]
else
  band = coefband_trans_8×8plus[c]
```

表 127　coefband_trans_4×4

coefband_trans_4×4[16] = {
0,1,1,2,2,2,3,3,3,3,4,4,4,5,5,5,
}

表 128　coefband_trans_8×8plus

coefband_trans_8×8plus[1024] ={
0,1,1,2,2,2,3,3,3,3,4,4,4,4,4,4,
4,4,4,4,4,5,
// beyond MAXBAND_INDEX+1 all values are filled as 5
5,5,5,5,5,5,5,5,5,5,
5,5,5,5,5,5,5,5,5,5,5,5,5,5,5,5,
5,5,5,5,5,5,5,5,5,5,5,5,5,5,5,5,
5,5,5,5,5,5,5,5,5,5,5,5,5,5,5,5,
5,5,5,5,5,5,5,5,5,5,5,5,5,5,5,5,
5,5,5,5,5,5,5,5,5,5,5,5,5,5,5,5,
5,5,5,5,5,5,5,5,5,5,5,5,5,5,5,5,
5,5,5,5,5,5,5,5,5,5,5,5,5,5,5,5,

表 128（续）

5,5,5,5,5,5,5,5,5,5,5,5,5,5,5,5,
5,5,5,5,5,5,5,5,5,5,5,5,5,5,5,5,
5,5,5,5,5,5,5,5,5,5,5,5,5,5,5,5,
5,5,5,5,5,5,5,5,5,5,5,5,5,5,5,5,
5,5,5,5,5,5,5,5,5,5,5,5,5,5,5,5,
5,5,5,5,5,5,5,5,5,5,5,5,5,5,5,5,
5,5,5,5,5,5,5,5,5,5,5,5,5,5,5,5,
5,5,5,5,5,5,5,5,5,5,5,5,5,5,5,5,
5,5,5,5,5,5,5,5,5,5,5,5,5,5,5,5,
5,5,5,5,5,5,5,5,5,5,5,5,5,5,5,5,
5,5,5,5,5,5,5,5,5,5,5,5,5,5,5,5,
5,5,5,5,5,5,5,5,5,5,5,5,5,5,5,5,
5,5,5,5,5,5,5,5,5,5,5,5,5,5,5,5,
5,5,5,5,5,5,5,5,5,5,5,5,5,5,5,5,
5,5,5,5,5,5,5,5,5,5,5,5,5,5,5,5,
5,5,5,5,5,5,5,5,5,5,5,5,5,5,5,5,
5,5,5,5,5,5,5,5,5,5,5,5,5,5,5,5,
5,5,5,5,5,5,5,5,5,5,5,5,5,5,5,5,
5,5,5,5,5,5,5,5,5,5,5,5,5,5,5,5,
5,5,5,5,5,5,5,5,5,5,5,5,5,5,5,5,
5,5,5,5,5,5,5,5,5,5,5,5,5,5,5,5,
5,5,5,5,5,5,5,5,5,5,5,5,5,5,5,5,
5,5,5,5,5,5,5,5,5,5,5,5,5,5,5,5,
5,5,5,5,5,5,5,5,5,5,5,5,5,5,5,5,
5,5,5,5,5,5,5,5,5,5,5,5,5,5,5,5,
5,5,5,5,5,5,5,5,5,5,5,5,5,5,5,5,
5,5,5,5,5,5,5,5,5,5,5,5,5,5,5,5,
5,5,5,5,5,5,5,5,5,5,5,5,5,5,5,5,
5,5,5,5,5,5,5,5,5,5,5,5,5,5,5,5,
5,5,5,5,5,5,5,5,5,5,5,5,5,5,5,5,
5,5,5,5,5,5,5,5,5,5,5,5,5,5,5,5,
5,5,5,5,5,5,5,5,5,5,5,5,5,5,5,5,
5,5,5,5,5,5,5,5,5,5,5,5,5,5,5,5,
5,5,5,5,5,5,5,5,5,5,5,5,5,5,5,5,
5,5,5,5,5,5,5,5,5,5,5,5,5,5,5,5,

表 128（续）

5,5,5,5,5,5,5,5,5,5,5,5,5,5,5,5,
5,5,5,5,5,5,5,5,5,5,5,5,5,5,5,5,
5,5,5,5,5,5,5,5,5,5,5,5,5,5,5,5,
5,5,5,5,5,5,5,5,5,5,5,5,5,5,5,5,
5,5,5,5,5,5,5,5,5,5,5,5,5,5,5,5,
5,5,5,5,5,5,5,5,5,5,5,5,5,5,5,5,
5,5,5,5,5,5,5,5,5,5,5,5,5,5,5,5,
5,5,5,5,5,5,5,5,5,5,5,5,5,5,5,5,
5,5,5,5,5,5,5,5,5,5,5,5,5,5,5,5,
5,5,5,5,5,5,5,5,5,5,5,5,5,5,5,5,
5,5,5,5,5,5,5,5,5,5,5,5,5,5,5,5,
5,5,5,5,5,5,5,5,5,5,5,5,5,5,5,5,
5,5,5,5,5,5,5,5,5,5,5,5,5,5,5,5,
5,5,5,5,5,5,5,5,5,5,5,5,5,5,5,5,
5,5,5,5,5,5,5,5,5,5,5,5,5,5,5,5,
5,5,5,5,5,5,5,5,5,5,5,5,5,5,5,5,
5,5,5,5,5,5,5,5,5,5,5,5,5,5,5,5,
5,5,5,5,5,5,5,5,5,5,5,5,5,5,5,5,
5,5,5,5,5,5,5,5,5,5,5,5,5,5,5,5,
5,5,5,5,5,5,5,5,5,5,5,5,5,5,5,5,
}

ctx 通过如下计算得到：

```
if(c = 0) {
  numpts = 1 〈〈 tx_size
  above = 0
  left = 0
  for(i = 0;i<numpts;i + + ) {
    above | = AboveNonzeroContext[i]
    left | = LeftNonzeroContext[i]
  }
  ctx = above + left
} else {
  ctx = (1 + TokenCache[nb[MAX_NEIGHBORS * c]] + TokenCache[nb[MAX_NEIGHBORS * c + 1]]) 〉〉1
}
```

nb 通过如下计算查表 129～138 得到：

```
if(inter_block || ! is_plane_y) {
  if(tx_size = = TX_4×4) {
    nb = default_scan_4×4_neighbors
```

```
    } else if(tx_size = = TX_8 × 8) {
      nb = default_scan_8 × 8_neighbors
    } else if(tx_size = = TX_16 × 16) {
      nb = default_scan_16 × 16_neighbors
    } else {
      nb = default_scan_32 × 32_neighbors
    }
  } else {
    if(tx_size = = TX_4 × 4) {
      if (tx_type = = ADST_DCT) {
        nb = row_scan_4 × 4_neighbors
      } else if(tx_type = = DCT_ADST) {
        nb = col_scan_4 × 4_neighbors
      } else {
        nb = default_scan_4 × 4_neighbors
      }
    } else if(tx_size = = TX_8 × 8) {
      if (tx_type = = ADST_DCT) {
        nb = row_scan_8 × 8_neighbors
      } else if(tx_type = = DCT_ADST) {
        nb = col_scan_8 × 8_neighbors
      } else{
        nb = default_scan_8 × 8_neighbors
      }
    } else if(tx_size = = TX_16 × 16) {
      if (tx_type = = ADST_DCT) {
        nb = row_scan_16 × 16_neighbors
      } else if(tx_type = = DCT_ADST) {
        nb = col_scan_16 × 16_neighbors
      } else {
        nb = default_scan_16 × 16_neighbors
      }
    } else {
      nb = default_scan_32 × 32_neighbors
    }
  }
```

表 129　default_scan_4×4_neighbors

default_scan_4×4_neighbors[17 * MAX_NEIGHBORS] = {
0,0,0,0,0,0,1,4,4,4,1,1,8,8,5,8,2,2,2,5,9,12,6,9,
3,6,10,13,7,10,11,14,0,0,
};

表 130　col_scan_4×4_neighbors

col_scan_4×4_neighbors[17 * MAX_NEIGHBORS] = {
0,0,0,0,4,4,0,0,8,8,1,1,5,5,1,1,9,9,2,2,6,6,2,2,3,
3,10,10,7,7,11,11,0,0,
};

表 131　row_scan_4×4_neighbors

row_scan_4×4_neighbors[17 * MAX_NEIGHBORS] = {
0,0,0,0,0,0,1,1,4,4,2,2,5,5,4,4,8,8,6,6,8,8,9,9,12,
12,10,10,13,13,14,14,0,0,
};

表 132　col_scan_8×8_neighbors

col_scan_8×8_neighbors[65 * MAX_NEIGHBORS] = {
0,0,0,0,8,8,0,0,16,16,1,1,24,24,9,9,1,1,32,32,17,17,2,
2,25,25,10,10,40,40,2,2,18,18,33,33,3,3,48,48,11,11,26,
26,3,3,41,41,19,19,34,34,4,4,27,27,12,12,49,49,42,42,20,
20,4,4,35,35,5,5,28,28,50,50,43,43,13,13,36,36,5,5,21,21,
51,51,29,29,6,6,44,44,14,14,6,6,37,37,52,52,22,22,7,7,30,
30,45,45,15,15,38,38,23,23,53,53,31,31,46,46,39,39,54,54,
47,47,55,55,0,0,
};

表 133　row_scan_8×8_neighbors

row_scan_8×8_neighbors[65 * MAX_NEIGHBORS] = {
0,0,0,0,1,1,0,0,8,8,2,2,8,8,9,9,3,3,16,16,10,10,16,16,
4,4,17,17,24,24,11,11,18,18,25,25,24,24,5,5,12,12,19,19,
32,32,26,26,6,6,33,33,32,32,20,20,27,27,40,40,13,13,34,34,
40,40,41,41,28,28,35,35,48,48,21,21,42,42,14,14,48,48,36,
36,49,49,43,43,29,29,56,56,22,22,50,50,57,57,44,44,37,37,
51,51,30,30,58,58,52,52,45,45,59,59,38,38,60,60,46,46,53,
53,54,54,61,61,62,62,0,0,
};

表 134　default_scan_8×8_neighbors

default_scan_8×8_neighbors[65 * MAX_NEIGHBORS] = {
0,0,0,0,0,0,8,8,1,8,1,1,9,16,16,16,2,9,2,2,10,17,17,
24,24,24,3,10,3,3,18,25,25,32,11,18,32,32,4,11,26,33,19,
26,4,4,33,40,12,19,40,40,5,12,27,34,34,41,20,27,13,20,5,
5,41,48,48,48,28,35,35,42,21,28,6,6,6,13,42,49,49,56,36,
43,14,21,29,36,7,14,43,50,50,57,22,29,37,44,15,22,44,51,
51,58,30,37,23,30,52,59,45,52,38,45,31,38,53,60,46,53,39,
46,54,61,47,54,55,62,0,0,
};

表 135　col_scan_16×16_neighbors

col_scan_16×16_neighbors[257 * MAX_NEIGHBORS] =
{
0,0,0,0,16,16,32,32,0,0,48,48,1,1,64,64,
17,17,80,80,33,33,1,1,49,49,96,96,2,2,65,65,
18,18,112,112,34,34,81,81,2,2,50,50,128,128,3,3,
97,97,19,19,66,66,144,144,82,82,35,35,113,113,3,3,
51,51,160,160,4,4,98,98,129,129,67,67,20,20,83,83,
114,114,36,36,176,176,4,4,145,145,52,52,99,99,5,5,
130,130,68,68,192,192,161,161,21,21,115,115,84,84,37,37,
146,146,208,208,53,53,5,5,100,100,177,177,131,131,69,69,
6,6,224,224,116,116,22,22,162,162,85,85,147,147,38,38,
193,193,101,101,54,54,6,6,132,132,178,178,70,70,163,163,
209,209,7,7,117,117,23,23,148,148,7,7,86,86,194,194,
225,225,39,39,179,179,102,102,133,133,55,55,164,164,8,8,
71,71,210,210,118,118,149,149,195,195,24,24,87,87,40,40,
56,56,134,134,180,180,226,226,103,103,8,8,165,165,211,211,
72,72,150,150,9,9,119,119,25,25,88,88,196,196,41,41,
135,135,181,181,104,104,57,57,227,227,166,166,120,120,151,151,
197,197,73,73,9,9,212,212,89,89,136,136,182,182,10,10,
26,26,105,105,167,167,228,228,152,152,42,42,121,121,213,213,
58,58,198,198,74,74,137,137,183,183,168,168,10,10,90,90,
229,229,11,11,106,106,214,214,153,153,27,27,199,199,43,43,
184,184,122,122,169,169,230,230,59,59,11,11,75,75,138,138,

表 135（续）

200,200,215,215,91,91,12,12,28,28,185,185,107,107,154,154,
44,44,231,231,216,216,60,60,123,123,12,12,76,76,201,201,
170,170,232,232,139,139,92,92,13,13,108,108,29,29,186,186,
217,217,155,155,45,45,13,13,61,61,124,124,14,14,233,233,
77,77,14,14,171,171,140,140,202,202,30,30,93,93,109,109,
46,46,156,156,62,62,187,187,15,15,125,125,218,218,78,78,
31,31,172,172,47,47,141,141,94,94,234,234,203,203,63,63,
110,110,188,188,157,157,126,126,79,79,173,173,95,95,219,219,
142,142,204,204,235,235,111,111,158,158,127,127,189,189,220,
220,143,143,174,174,205,205,236,236,159,159,190,190,221,221,
175,175,237,237,206,206,222,222,191,191,238,238,207,207,223,
223,239,239,0,0,
};

表 136　row_scan_16×16_neighbors

row_scan_16×16_neighbors[257 * MAX_NEIGHBORS] =
{
0,0,0,0,1,1,0,0,2,2,16,16,3,3,17,17,
16,16,4,4,32,32,18,18,5,5,33,33,32,32,19,19,
48,48,6,6,34,34,20,20,49,49,48,48,7,7,35,35,
64,64,21,21,50,50,36,36,64,64,8,8,65,65,51,51,
22,22,37,37,80,80,66,66,9,9,52,52,23,23,81,81,
67,67,80,80,38,38,10,10,53,53,82,82,96,96,68,68,
24,24,97,97,83,83,39,39,96,96,54,54,11,11,69,69,
98,98,112,112,84,84,25,25,40,40,55,55,113,113,99,99,
12,12,70,70,112,112,85,85,26,26,114,114,100,100,128,128,
41,41,56,56,71,71,115,115,13,13,86,86,129,129,101,101,
128,128,72,72,130,130,116,116,27,27,57,57,14,14,87,87,
42,42,144,144,102,102,131,131,145,145,117,117,73,73,144,144,
88,88,132,132,103,103,28,28,58,58,146,146,118,118,43,43,
160,160,147,147,89,89,104,104,133,133,161,161,119,119,160,160,
74,74,134,134,148,148,29,29,59,59,162,162,176,176,44,44,
120,120,90,90,105,105,163,163,177,177,149,149,176,176,135,135,
164,164,178,178,30,30,150,150,192,192,75,75,121,121,60,60,

表 136（续）

136,136,193,193,106,106,151,151,179,179,192,192,45,45,165,165,
166,166,194,194,91,91,180,180,137,137,208,208,122,122,152,152,
208,208,195,195,76,76,167,167,209,209,181,181,224,224,107,107,
196,196,61,61,153,153,224,224,182,182,168,168,210,210,46,46,
138,138,92,92,183,183,225,225,211,211,240,240,197,197,169,169,
123,123,154,154,198,198,77,77,212,212,184,184,108,108,226,226,
199,199,62,62,227,227,241,241,139,139,213,213,170,170,185,185,
155,155,228,228,242,242,124,124,93,93,200,200,243,243,214,214,
215,215,229,229,140,140,186,186,201,201,78,78,171,171,109,109,
156,156,244,244,216,216,230,230,94,94,245,245,231,231,125,125,
202,202,246,246,232,232,172,172,217,217,141,141,110,110,157,
157,187,187,247,247,126,126,233,233,218,218,248,248,188,188,
203,203,142,142,173,173,158,158,249,249,234,234,204,204,219,
219,174,174,189,189,250,250,220,220,190,190,205,205,235,235,
206,206,236,236,251,251,221,221,252,252,222,222,237,237,238,
238,253,253,254,254,0,0,
};

表 137　**default_scan_16×16_neighbors**

default_scan_16×16_neighbors[257 * MAX_NEIGHBORS] =
{
0,0,0,0,0,0,16,16,1,16,1,1,32,32,17,32,
2,17,2,2,48,48,18,33,33,48,3,18,49,64,64,64,
34,49,3,3,19,34,50,65,4,19,65,80,80,80,35,50,
4,4,20,35,66,81,81,96,51,66,96,96,5,20,36,51,
82,97,21,36,67,82,97,112,5,5,52,67,112,112,37,52,
6,21,83,98,98,113,68,83,6,6,113,128,22,37,53,68,
84,99,99,114,128,128,114,129,69,84,38,53,7,22,7,7,
129,144,23,38,54,69,100,115,85,100,115,130,144,144,130,145,
39,54,70,85,8,23,55,70,116,131,101,116,145,160,24,39,
8,8,86,101,131,146,160,160,146,161,71,86,40,55,9,24,
117,132,102,117,161,176,132,147,56,71,87,102,25,40,147,162,
9,9,176,176,162,177,72,87,41,56,118,133,133,148,103,118,
10,25,148,163,57,72,88,103,177,192,26,41,163,178,192,192,

表 137（续）

10,10,119,134,73,88,149,164,104,119,134,149,42,57,178,193,
164,179,11,26,58,73,193,208,89,104,135,150,120,135,27,42,
74,89,208,208,150,165,179,194,165,180,105,120,194,209,43,58,
11,11,136,151,90,105,151,166,180,195,59,74,121,136,209,224,
195,210,224,224,166,181,106,121,75,90,12,27,181,196,12,12,
210,225,152,167,167,182,137,152,28,43,196,211,122,137,91,106,
225,240,44,59,13,28,107,122,182,197,168,183,211,226,153,168,
226,241,60,75,197,212,138,153,29,44,76,91,13,13,183,198,
123,138,45,60,212,227,198,213,154,169,169,184,227,242,92,107,
61,76,139,154,14,29,14,14,184,199,213,228,108,123,199,214,
228,243,77,92,30,45,170,185,155,170,185,200,93,108,124,139,
214,229,46,61,200,215,229,244,15,30,109,124,62,77,140,155,
215,230,31,46,171,186,186,201,201,216,78,93,230,245,125,140,
47,62,216,231,156,171,94,109,231,246,141,156,63,78,202,217,
187,202,110,125,217,232,172,187,232,247,79,94,157,172,126,141,
203,218,95,110,233,248,218,233,142,157,111,126,173,188,188,203,
234,249,219,234,127,142,158,173,204,219,189,204,143,158,235,
250,174,189,205,220,159,174,220,235,221,236,175,190,190,205,
236,251,206,221,237,252,191,206,222,237,207,222,238,253,223,
238,239,254,0,0,
};

表 138　default_scan_32×32_neighbors

default_scan_32×32_neighbors[1025 * MAX_NEIGHBORS] =
{
0,0,0,0,0,0,32,32,1,32,1,1,64,64,33,64,
2,33,96,96,2,2,65,96,34,65,128,128,97,128,3,34,
66,97,3,3,35,66,98,129,129,160,160,160,4,35,67,98,
192,192,4,4,130,161,161,192,36,67,99,130,5,36,68,99,
193,224,162,193,224,224,131,162,37,68,100,131,5,5,194,225,
225,256,256,256,163,194,69,100,132,163,6,37,226,257,6,6,
195,226,257,288,101,132,288,288,38,69,164,195,133,164,258,289,
227,258,196,227,7,38,289,320,70,101,320,320,7,7,165,196,
39,70,102,133,290,321,259,290,228,259,321,352,352,352,197,228,

表 138（续）

134,165,71,102,8,39,322,353,291,322,260,291,103,134,353,384,
166,197,229,260,40,71,8,8,384,384,135,166,354,385,323,354,
198,229,292,323,72,103,261,292,9,40,385,416,167,198,104,135,
230,261,355,386,416,416,293,324,324,355,9,9,41,72,386,417,
199,230,136,167,417,448,262,293,356,387,73,104,387,418,231,262,
10,41,168,199,325,356,418,449,105,136,448,448,42,73,294,325,
200,231,10,10,357,388,137,168,263,294,388,419,74,105,419,450,
449,480,326,357,232,263,295,326,169,200,11,42,106,137,480,480,
450,481,358,389,264,295,201,232,138,169,389,420,43,74,420,451,
327,358,11,11,481,512,233,264,451,482,296,327,75,106,170,201,
482,513,512,512,390,421,359,390,421,452,107,138,12,43,202,233,
452,483,265,296,328,359,139,170,44,75,483,514,513,544,234,265,
297,328,422,453,12,12,391,422,171,202,76,107,514,545,453,484,
544,544,266,297,203,234,108,139,329,360,298,329,140,171,515,
546,13,44,423,454,235,266,545,576,454,485,45,76,172,203,330,
361,576,576,13,13,267,298,546,577,77,108,204,235,455,486,577,
608,299,330,109,140,547,578,14,45,14,14,141,172,578,609,331,
362,46,77,173,204,15,15,78,109,205,236,579,610,110,141,15,46,
142,173,47,78,174,205,16,16,79,110,206,237,16,47,111,142,
48,79,143,174,80,111,175,206,17,48,17,17,207,238,49,80,
81,112,18,18,18,49,50,81,82,113,19,50,51,82,83,114,608,608,
484,515,360,391,236,267,112,143,19,19,640,640,609,640,516,547,
485,516,392,423,361,392,268,299,237,268,144,175,113,144,20,51,
20,20,672,672,641,672,610,641,548,579,517,548,486,517,424,455,
393,424,362,393,300,331,269,300,238,269,176,207,145,176,114,
145,52,83,21,52,21,21,704,704,673,704,642,673,611,642,580,
611,549,580,518,549,487,518,456,487,425,456,394,425,363,394,
332,363,301,332,270,301,239,270,208,239,177,208,146,177,115,
146,84,115,53,84,22,53,22,22,705,736,674,705,643,674,581,612,
550,581,519,550,457,488,426,457,395,426,333,364,302,333,271,
302,209,240,178,209,147,178,85,116,54,85,23,54,706,737,675,
706,582,613,551,582,458,489,427,458,334,365,303,334,210,241,
179,210,86,117,55,86,707,738,583,614,459,490,335,366,211,242,
87,118,736,736,612,643,488,519,364,395,240,271,116,147,23,23,
768,768,737,768,644,675,613,644,520,551,489,520,396,427,365,

表 138（续）

396,272,303,241,272,148,179,117,148,24,55,24,24,800,800,769,
800,738,769,676,707,645,676,614,645,552,583,521,552,490,521,
428,459,397,428,366,397,304,335,273,304,242,273,180,211,149,
180,118,149,56,87,25,56,25,25,832,832,801,832,770,801,739,
770,708,739,677,708,646,677,615,646,584,615,553,584,522,553,
491,522,460,491,429,460,398,429,367,398,336,367,305,336,274,
305,243,274,212,243,181,212,150,181,119,150,88,119,57,88,26,
57,26,26,833,864,802,833,771,802,709,740,678,709,647,678,585,
616,554,585,523,554,461,492,430,461,399,430,337,368,306,337,
275,306,213,244,182,213,151,182,89,120,58,89,27,58,834,865,
803,834,710,741,679,710,586,617,555,586,462,493,431,462,338,
369,307,338,214,245,183,214,90,121,59,90,835,866,711,742,587,
618,463,494,339,370,215,246,91,122,864,864,740,771,616,647,
492,523,368,399,244,275,120,151,27,27,896,896,865,896,772,803,
741,772,648,679,617,648,524,555,493,524,400,431,369,400,276,
307,245,276,152,183,121,152,28,59,28,28,928,928,897,928,866,
897,804,835,773,804,742,773,680,711,649,680,618,649,556,587,
525,556,494,525,432,463,401,432,370,401,308,339,277,308,246,
277,184,215,153,184,122,153,60,91,29,60,29,29,960,960,929,
960,898,929,867,898,836,867,805,836,774,805,743,774,712,743,
681,712,650,681,619,650,588,619,557,588,526,557,495,526,464,
495,433,464,402,433,371,402,340,371,309,340,278,309,247,278,
216,247,185,216,154,185,123,154,92,123,61,92,30,61,30,30,
961,992,930,961,899,930,837,868,806,837,775,806,713,744,682,
713,651,682,589,620,558,589,527,558,465,496,434,465,403,434,
341,372,310,341,279,310,217,248,186,217,155,186,93,124,62,93,
31,62,962,993,931,962,838,869,807,838,714,745,683,714,590,621,
559,590,466,497,435,466,342,373,311,342,218,249,187,218,94,
125,63,94,963,994,839,870,715,746,591,622,467,498,343,374,219,
250,95,126,868,899,744,775,620,651,496,527,372,403,248,279,
124,155,900,931,869,900,776,807,745,776,652,683,621,652,528,
559,497,528,404,435,373,404,280,311,249,280,156,187,125,156,
932,963,901,932,870,901,808,839,777,808,746,777,684,715,653,
684,622,653,560,591,529,560,498,529,436,467,405,436,374,405,
312,343,281,312,250,281,188,219,157,188,126,157,964,995,933,

表 138（续）

964,902,933,871,902,840,871,809,840,778,809,747,778,716,747,
685,716,654,685,623,654,592,623,561,592,530,561,499,530,468,
499,437,468,406,437,375,406,344,375,313,344,282,313,251,282,
220,251,189,220,158,189,127,158,965,996,934,965,903,934,841,
872,810,841,779,810,717,748,686,717,655,686,593,624,562,593,
531,562,469,500,438,469,407,438,345,376,314,345,283,314,221,
252,190,221,159,190,966,997,935,966,842,873,811,842,718,749,
687,718,594,625,563,594,470,501,439,470,346,377,315,346,222,
253,191,222,967,998,843,874,719,750,595,626,471,502,347,378,
223,254,872,903,748,779,624,655,500,531,376,407,252,283,904,
935,873,904,780,811,749,780,656,687,625,656,532,563,501,532,
408,439,377,408,284,315,253,284,936,967,905,936,874,905,812,
843,781,812,750,781,688,719,657,688,626,657,564,595,533,564,
502,533,440,471,409,440,378,409,316,347,285,316,254,285,968,
999,937,968,906,937,875,906,844,875,813,844,782,813,751,782,
720,751,689,720,658,689,627,658,596,627,565,596,534,565,503,
534,472,503,441,472,410,441,379,410,348,379,317,348,286,317,
255,286,969,1000,938,969,907,938,845,876,814,845,783,814,721,
752,690,721,659,690,597,628,566,597,535,566,473,504,442,473,
411,442,349,380,318,349,287,318,970,1001,939,970,846,877,815,
846,722,753,691,722,598,629,567,598,474,505,443,474,350,381,
319,350,971,1002,847,878,723,754,599,630,475,506,351,382,876,
907,752,783,628,659,504,535,380,411,908,939,877,908,784,815,
753,784,660,691,629,660,536,567,505,536,412,443,381,412,940,
971,909,940,878,909,816,847,785,816,754,785,692,723,661,692,
630,661,568,599,537,568,506,537,444,475,413,444,382,413,972,
1003,941,972,910,941,879,910,848,879,817,848,786,817,755,786,
724,755,693,724,662,693,631,662,600,631,569,600,538,569,507,
538,476,507,445,476,414,445,383,414,973,1004,942,973,911,942,
849,880,818,849,787,818,725,756,694,725,663,694,601,632,570,
601,539,570,477,508,446,477,415,446,974,1005,943,974,850,881,
819,850,726,757,695,726,602,633,571,602,478,509,447,478,975,
1006,851,882,727,758,603,634,479,510,880,911,756,787,632,663,
508,539,912,943,881,912,788,819,757,788,664,695,633,664,540,
571,509,540,944,975,913,944,882,913,820,851,789,820,758,789,

表 138（续）

696,727,665,696,634,665,572,603,541,572,510,541,976,1007,945,
976,914,945,883,914,852,883,821,852,790,821,759,790,728,759,
697,728,666,697,635,666,604,635,573,604,542,573,511,542,977,
1008,946,977,915,946,853,884,822,853,791,822,729,760,698,729,
667,698,605,636,574,605,543,574,978,1009,947,978,854,885,823,
854,730,761,699,730,606,637,575,606,979,1010,855,886,731,762,
607,638,884,915,760,791,636,667,916,947,885,916,792,823,761,
792,668,699,637,668,948,979,917,948,886,917,824,855,793,824,
762,793,700,731,669,700,638,669,980,1011,949,980,918,949,887,
918,856,887,825,856,794,825,763,794,732,763,701,732,670,701,
639,670,981,1012,950,981,919,950,857,888,826,857,795,826,733,
764,702,733,671,702,982,1013,951,982,858,889,827,858,734,765,
703,734,983,1014,859,890,735,766,888,919,764,795,920,951,889,
920,796,827,765,796,952,983,921,952,890,921,828,859,797,828,
766,797,984,1015,953,984,922,953,891,922,860,891,829,860,798,
829,767,798,985,1016,954,985,923,954,861,892,830,861,799,830,
986,1017,955,986,862,893,831,862,987,1018,863,894,892,923,924,
955,893,924,956,987,925,956,894,925,988,1019,957,988,926,957,
895,926,989,1020,958,989,927,958,990,1021,959,990,991,1022,0,0,
};

Token_Cache 为 32×32 的表，其值的初始化由 token 决定，token[scan[c]]与 token 关系见表 139：

表 139　token[scan[c]]与 token 的关系

token	token[scan[c]]
0	0
1	1
2	2
3	3
4	3
5	4
6	4
7	5
8	5
9	5
10	5
11	5

svac2_pareto8_full[ctx] [treeIdx]中 ctx 的计算如下：

ctx = prob[2]−1

prob[2]是 binIdx 等于 2 的 coef_probs 的概率表。

treeIdx 的计算过程见 5.4.2.4.3。

extra 会使用到 svac2_cat1_prob、svac2_cat2_prob、svac2_cat3_prob、svac2_cat4_prob、svac2_cat5_prob 和 svac2_cat6_prob 概率表，均通过 binIdx 索引得到对应的概率值，具体对应概率表见 5.4.2.2.1。

5.4.2.4.5 二进制位解析

此过程的输入为 range、count、value、buffer、buffer_end 以及概率值 prob，输出为 bit 即解码出的二进制位。其中，range 的位宽为 8 比特，value 位宽为 32 比特。

二进制位的解析过程如下：

第一步，如果 count 的值小于 0，执行算术解码器的重整化过程，与 5.4.2.2 初始化中的重整化过程相同；

第二步，执行 decode 过程得到 bit，该过程用伪代码描述如下：

```
bit = 0
split = (range ×prob + (256−prob)) 〉〉 8;
value_t = value
count_t = count
range_t = split
bigsplit = split 〈〈 24
if (value_t>= bigsplit) {
  range_t = range-split
  value_t = value_t-bigsplit
  bit = 1
}
shift = norm[range_t]
range_t 〈〈= shift
value_t 〈〈= shift
count_t−= shift
value = value_t
count = count_t
range = range_t
```

第三步，如果 bit 的值为 0，则二进制位为‘0’；如果 bit 的值为 1，则二进制位为‘1’。

5.4.3 ue(v)与 se(v)的解析过程

5.4.3.1 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 140 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 140 k 阶指数哥伦布码表

阶数	码字结构	CodeNum 取值范围
k=0	1	0
	0 1 x0	1…2
	0 0 1 x1 x0	3…6
	0 0 0 1 x2 x1 x0	7…14
	……	……
k=1	1 x0	0…1
	0 1 x1 x0	2…5
	0 0 1 x2 x1 x0	6…13
	0 0 0 1 x3 x2 x1 x0	14…29
	……	……
k=2	1 x1 x0	0…3
	0 1 x2 x1 x0	4…11
	0 0 1 x3 x2 x1 x0	12…27
	0 0 0 1 x4 x3 x2 x1 x0	28…59
	……	……
k=3	1 x2 x1 x0	0…7
	0 1 x3 x2 x1 x0	8…23
	0 0 1 x4 x3 x2 x1 x0	24…55
	0 0 0 1 x5 x4 x3 x2 x1 x0	56…119
	……	…

5.4.3.2 ue(v)

ue(v)描述的语法元素使用 0 阶无符号指数哥伦布码，其语法元素其语法元素的取值等于 CodeNum。

5.4.3.3 se(v)

se(v)描述的语法元素使用 0 阶指数哥伦布码，其语法元素的取值与 CodeNum 的映射关系见表 141。

表 141　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	$(-1)^{k+1}\times \mathrm{Ceil}(k \div 2)$

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 22 和图 23 给出了音频编码器和解码器框图。

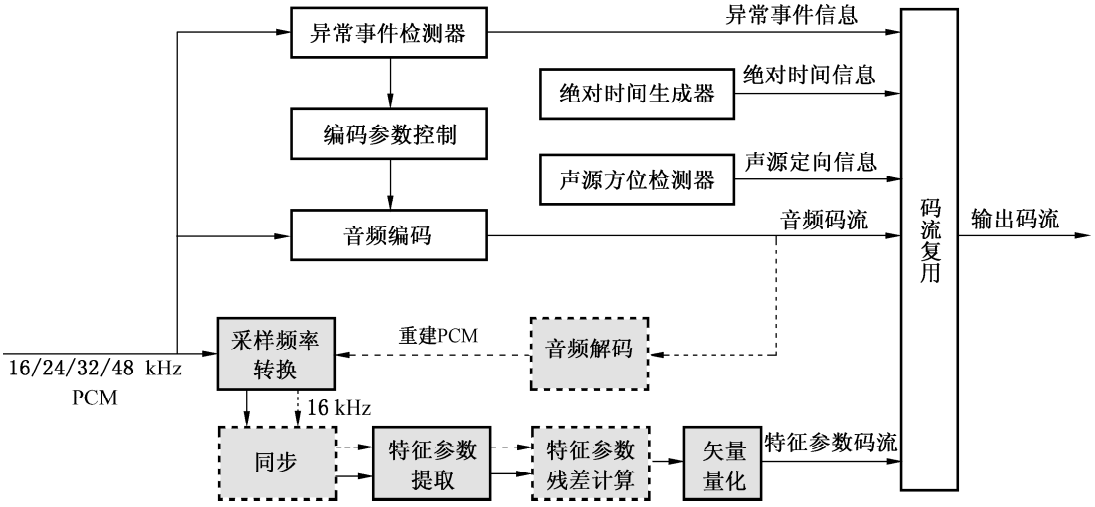


图 22　音频编码器框图

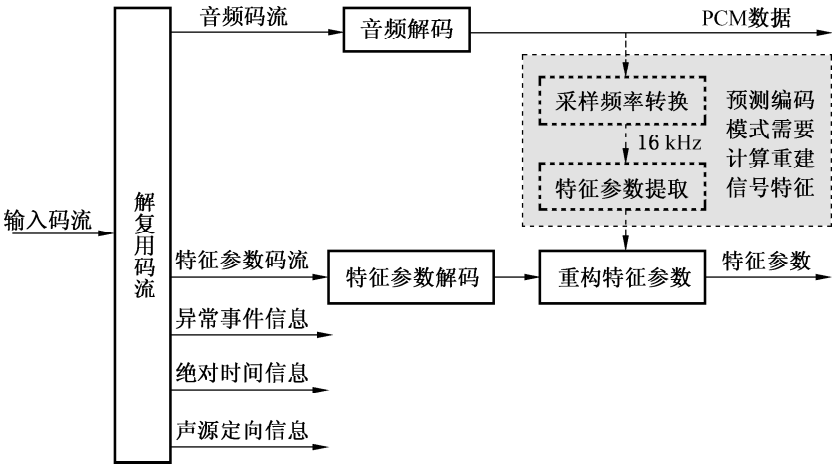


图 23　音频解码器框图

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 24 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 $F_s/2$。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 25 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。

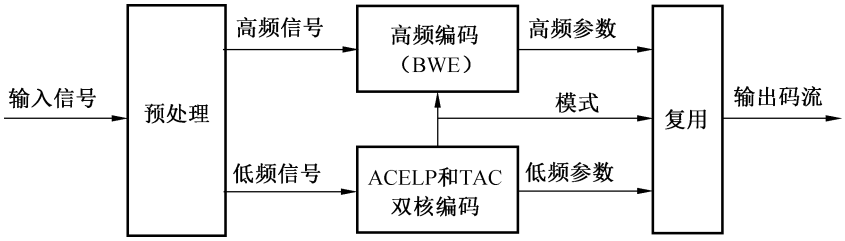


图 24 双核音频编码器流程图

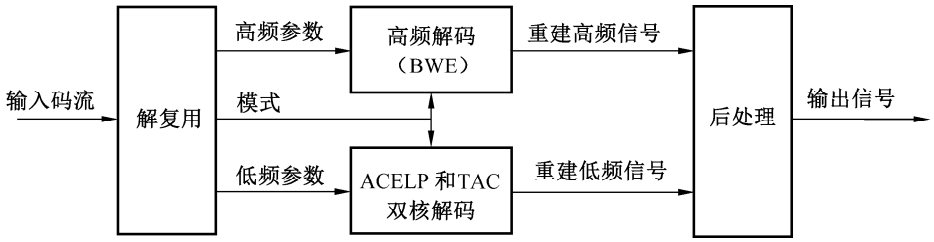


图 25 双核音频解码器流程图

6.1.4 识别特征参数编码描述

图 26 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。

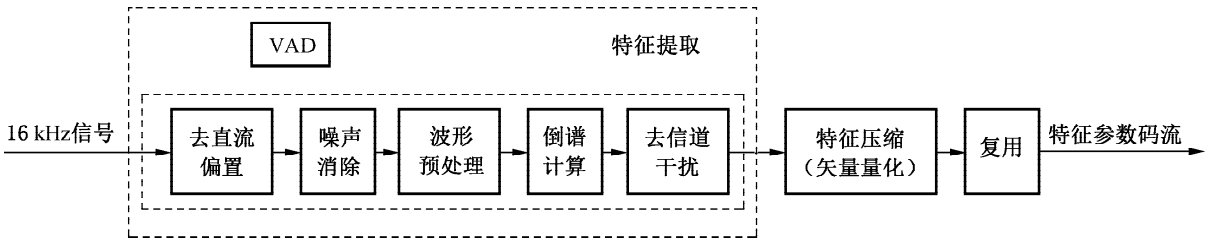


图 26　识别特征参数编码框图

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

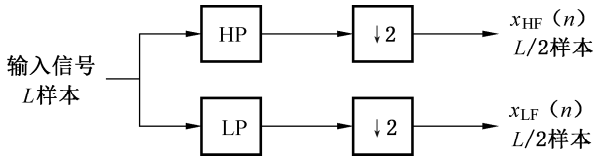
6.2.1　预处理

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 F_s。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 F_s 的输入信号通过截止频率在 $F_s/4$ 的低通滤波器，再作 2 倍临界下采样，得到 $F_s/2$ 采样低频信号 $x_{LF}(n)$；同样，经过截止频率在 $F_s/4$ 的高通滤波器，再作 2 倍临界下采样，得到 $F_s/2$ 采样高频信号 $x_{HF}(n)$，见图 27。



说明：

L——音频超帧长度。

图 27　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见式(1)：

$$H_{h1}(z)=\frac{b_0-b_1z^{-1}+b_2z^{-2}}{1-a_1z^{-1}+a_2z^{-2}} \qquad \cdots\cdots(1)$$

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

$F_s/2$ 频率采样的低频信号(0～$F_s/4$ 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 $F_s/2$ 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 28 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,需要加下一帧的前 32 个样本用于帧重叠。

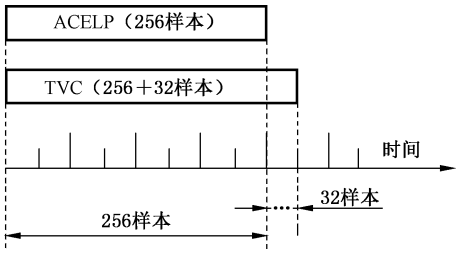


图 28 帧类型的时间图

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。子帧的分段信噪比见式(2):

$$segSNR_i = 20\log_{10}\left\{\frac{\sum_{n=0}^{N-1} x_w^2(n)}{\sum_{n=0}^{N-1}(x_w(n)-\hat{x}_w(n))^2}\right\} \quad \cdots\cdots(2)$$

式中:

N——子帧长度(64 个样点)。

每帧分段信噪比均值计算见式(3):

$$\overline{segSNR} = \frac{1}{N_{SF}}\sum_{i=0}^{N_{SF}-1} segSNR_i \quad \cdots\cdots(3)$$

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见式(4)：

$$H(z)=\frac{1}{\hat{A}(z)}=\frac{1}{1+\sum_{i=1}^{m}\hat{a}_i z^{-i}} \qquad \cdots\cdots(4)$$

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 29 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。

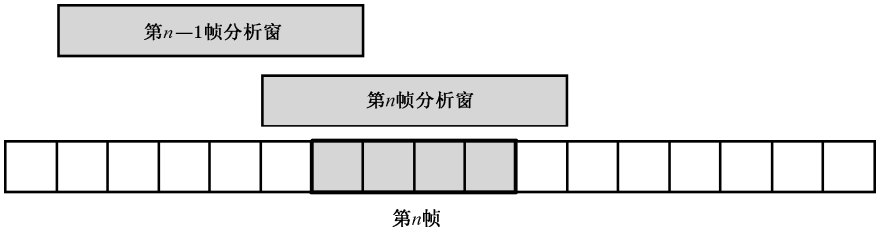


图 29 线性预测分析帧结构图

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，见式(5)：

$$w(n)=\begin{cases}0.54-0.46\cos\left(\dfrac{2n\pi}{2L_1-1}\right) & n=0,\cdots,L_1-1\\ \cos\left(\dfrac{2\pi(n-L_1)}{4L_2-1}\right) & n=L_1,\cdots,L_1+L_2-1\end{cases} \qquad \cdots\cdots(5)$$

式中：

$L_1=256$，$L_2=128$。

设加窗后信号为 $s'(n)$，则有式(6)：

$$s'(n)=w(n)s(n) \qquad n=0,\cdots,383 \qquad \cdots\cdots(6)$$

式中：

$w(n)$——加窗函数；

$s(n)$ ——预加重后的信号。

$s'(n)$ 对应的自相关函数见式(7)：

$$r(k)=\sum_{n=k}^{383}s'(n)s'(n-k) \qquad k=0,\cdots,16 \qquad \cdots\cdots(7)$$

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 $w_{lag}(i)$ 的表达式见式(8)：

$$w_{lag}(i)=\exp\left[-\frac{1}{2}\left(\frac{2\pi f_0 i}{f_s}\right)^2\right],\qquad i=1,\cdots,16 \quad \cdots\cdots(8)$$

式中：

f_0——扩展的带宽，$f_0=60$ Hz；

f_s——采样频率，$f_s=12.8$ kHz。

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 $r'(k)$ 如式(9)：

$$r'(k)=\begin{cases}1.000\ 1r(0), & k=0\\ r(k)w_{lag}(k), & k=1,\cdots,16\end{cases} \quad \cdots\cdots(9)$$

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，见式(10)：

$$\sum_{k=1}^{16}a_k r'(|i-k|)=-r'(i),\qquad i=1,\cdots,16 \quad \cdots\cdots(10)$$

该方程组可用莱文逊—杜宾算法求解，算法计算步骤如式(11)：

$$\begin{aligned}&E(0)=r'(0);\\&for(i=1;i<=16;++i)\\&\{\\&k_i=-\left[r'(i)+\sum_{j=1}^{i-1}a_j^{i-1}r'(i-j)\right]/E(i-1);\\&a_i^{(i)}=k_i;\\&for\ (j=1\ ;j<=i-1;++j)\\&\{\\&\qquad a_j^{(i)}=a_j^{(i-1)}+k_i a_{i-j}^{(i-1)};\\&\}\\&E(i)=(1-k_i^2)E(i-1);\\&\}\end{aligned} \quad \cdots\cdots(11)$$

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 系数就是下面多项式式(12)和式(13)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 $F_2(z)$，见式(14)和式(15)：

$$F_1(z)=F'_1(z) \quad \cdots\cdots(14)$$

$$F_2(z)=F'_2(z)/(1-z^{-2}) \quad \cdots\cdots(15)$$

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（$e^{\pm j\omega_i}$），由式(16)和式(17)表示：

$$F_1(z)=(1+a_{16})\prod_{i=0,2,\cdots,14}(1-2q_i z^{-1}+z^{-2}) \quad \cdots\cdots(16)$$

$$F_2(z)=(1-a_{16})\prod_{i=1,3,\cdots,13}(1-2q_i z^{-1}+z^{-2}) \quad \cdots\cdots(17)$$

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 $f_2(i)$，见式(18)：

```
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 $z=e^{j\omega}$ 处可表示为式(19)和式(20)：

$$F_1(\omega)=2\mathrm{e}^{-\mathrm{j}8\omega}C_1(x) \text{ 和 } F_2(\omega)=2\mathrm{e}^{-\mathrm{j}7\omega}C_2(x) \quad \cdots\cdots(19)$$

$$C_1(x)=\sum_{i=0}^{7}(f_1(i)T_{8-i}(x)+f_1(8)/2) \text{ 和 } C_2(x)=\sum_{i=0}^{6}(f_2(i)T_{8-i}(x)+f_2(7)/2) \quad \cdots(20)$$

式中：

$T_m=\cos(m\omega)$ ——切比雪夫多项式第 m 个根。

多项式 $C(x)$ 在 $x=\cos(\omega)$ 处的递归计算如式(21)：

```
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，用式(16)、式(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 $f_1(i)$，见式(22)：

```
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 $f'_2(i)$ 如式(23)：

$$\begin{aligned} f'_2(i)&=f_2(i)-f_2(i-2), \quad i=2,\cdots,m/2-1 \\ f'_1(i)&=f_1(i), \qquad\qquad i=0,\cdots,m/2 \end{aligned} \qquad \cdots\cdots(23)$$

$F'_1(z)$ 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式(24)：

$$\begin{aligned} f'_2(i)&=(1-q_{m-1})f'_2(i), \qquad i=0,\cdots,m/2-1 \\ f'_1(i)&=(1+q_{m-1})f'_1(i), \qquad i=0,\cdots,m/2 \end{aligned} \qquad \cdots\cdots(24)$$

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，见式(25)：

$$a_i=\begin{cases} 0.5f'_1(i)+0.5f'_2(i), & i=1,\cdots,m/2-1 \\ 0.5f'_1(i)-0.5f'_2(i), & i=m/2+1,\cdots,m-1 \\ 0.5f'_1(m/2), & i=m/2 \\ q_{m-1}, & i=m \end{cases} \qquad \cdots\cdots(25)$$

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 系数表达式见式(26)：

$$f_i=\begin{cases} \dfrac{f_s}{2\pi}\arccos(q_i), & i=0,\cdots,14 \\ \dfrac{f_s}{4\pi}\arccos(q_i), & i=15 \end{cases} \qquad \cdots\cdots(26)$$

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 $f^t=[f_0,f_1,\cdots,f_{15}]$，$t$ 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 $r(n)$，其表达式见式(27)和式(28)：

$$z(n)=isf_n-mean\_isf \qquad \cdots\cdots(27)$$

$$r(n)=z(n)-p(n) \qquad \cdots\cdots(28)$$

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，见式(29)：

$$p(n)=\frac{1}{3}\hat{r}(n-1) \qquad \cdots\cdots(29)$$

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，见式(30)：

$$\mathrm{VQ2}=\{res\_isf_0,res\_isf_2,res\_isf_4\}\ 和\ \mathrm{VQ3}=\{res\_isf_6,res\_isf_8,res\_isf_{10}\} \quad \cdots\cdots(30)$$

量化采用的误差准则为均方量化误差准则，见式(31)：

$$E=\sum_i (res\_isf_i-res\_isf_i^q)^2,\ i=\{0,2,\cdots,10\} \quad \cdots\cdots(31)$$

量化得到对应的最佳量化矢量为 VQ′2 和 VQ′3，见式(32)：

$$\mathrm{VQ'}2=\{res\_isf_0^q,res\_isf_2^q,res\_isf_4^q\}\ 和\ \mathrm{VQ'}3=\{res\_isf_6^q,res\_isf_8^q,res\_isf_{10}^q\} \quad \cdots\cdots(32)$$

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 $res\_isf'_1$，见式(33)：

$$\begin{aligned} res\_isf_1^{\mathrm{predict}}&=\theta+\alpha_0\times res\_isf_0^q+\beta_2\times res\_isf_2^q \\ res\_isf'_1&=res\_isf_1-res\_isf_1^{\mathrm{predict}} \end{aligned} \quad \cdots\cdots(33)$$

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，见式(34)：

$$\begin{aligned} \mathrm{VQ4}&=\{res\_isf_{12},res\_isf_{14},res\_isf'_1\} \\ \mathrm{VQ'}4&=\{res\_isf_{12}^q,res\_isf_{14}^q,res\_isf_1^q\} \end{aligned} \quad \cdots\cdots(34)$$

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 7 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，见式(35)：

$$\begin{aligned} res\_isf_i^{\mathrm{predict}}&=\theta_i+\alpha_{i-1}\times res\_isf_{i-1}^q+\beta_{i+1}\times res\_isf_{i+1}^q,\ i=3,5,7,9,11,13 \\ res\_isf'_i&=res\_isf_i-res\_isf_i^{\mathrm{predict}},\quad i=3,5,7,9,11,13 \end{aligned} \quad \cdots\cdots(35)$$

式中：

$res\_isf_i^{\mathrm{predict}}$ ——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，见式(36)：

$$\begin{aligned} \mathrm{VQ7}&=\{res\_isf'_3,res\_isf'_5,res\_isf'_7\} \\ \mathrm{VQ8}&=\{res\_isf'_9,res\_isf'_{11},res\_isf'_{13},res\_isf_{15}\} \end{aligned} \quad \cdots\cdots(36)$$

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 $q_i^{(n)}$ 插值见式(37)：

$$\begin{aligned} q_1^{(n)} &= 0.55q^{(n-1)} + 0.45q^{(n)} \\ q_2^{(n)} &= 0.2q^{(n-1)} + 0.8q^{(n)} \\ q_3^{(n)} &= 0.04q^{(n-1)} + 0.96q^{(n)} \\ q_4^{(n)} &= q^{(n)} \end{aligned} \quad \cdots\cdots(37)$$

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见式(38)：

$$W(z) = \frac{A(z/\gamma_1)}{H_{\text{emph}}(z)} \quad \cdots\cdots(38)$$

式中：

$\gamma_1 = 0.92$；

$H_{\text{emph}}(z)$——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，预加重滤波器见式(39)：

$$H_{\text{emph\_hi}}(z) = 1 - \mu_1 z^{-1} \quad \cdots\cdots(39)$$

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，预加重滤波器见式(40)：

$$H_{\text{emph\_low}}(z) = 1 + \mu_2 z^{-1} \quad \cdots\cdots(40)$$

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，逆感知加权滤波器见式(41)：

$$\frac{1}{W(z)} = \frac{H_{\text{emph}}(z)}{A(z/\gamma_1)} \quad \cdots\cdots(41)$$

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 $H_{\text{decim2}}(z)$ 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，计算如式(42)：

$$corr' = \frac{\sum_{n=0}^{63} s_{wd}(n)s_{wd}(n-delay)}{\sqrt{\sum_{n=0}^{63} {s_{wd}}^2(n) \sum_{n=0}^{63} {s_{wd}}^2(n-delay)}} \quad \cdots\cdots(42)$$

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 30 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。

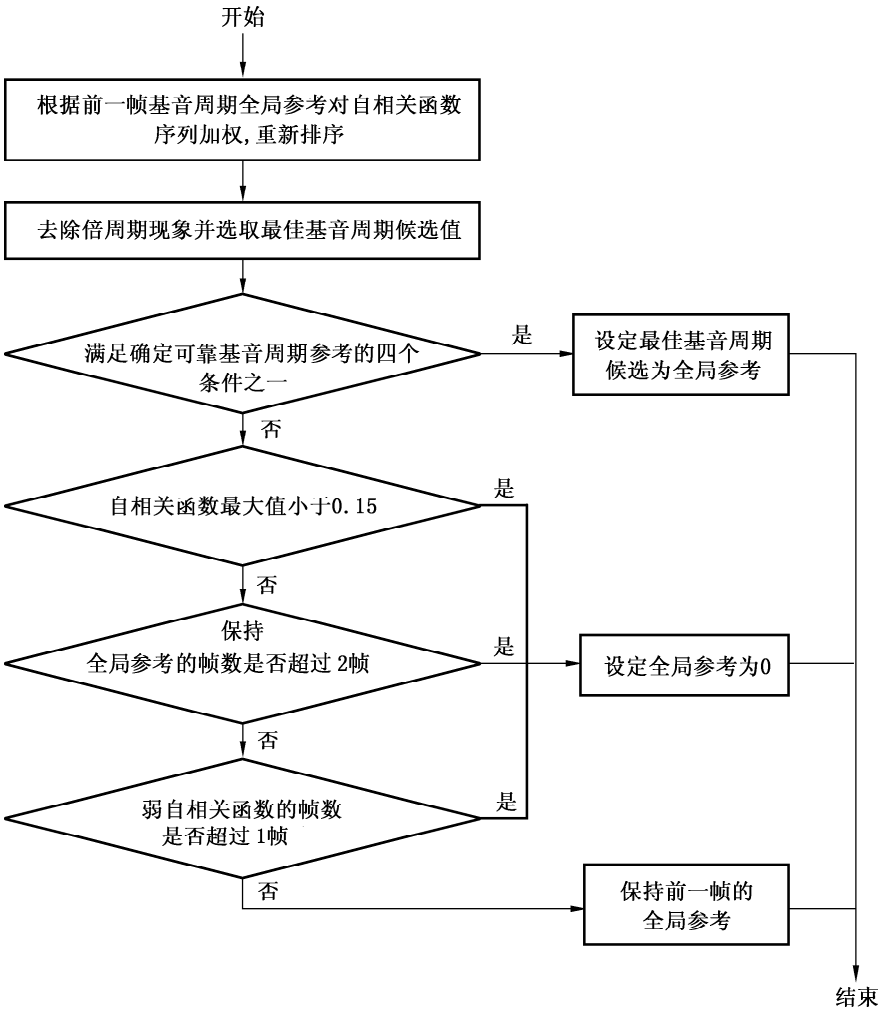


图 30 基音周期全局参考确定流程

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，具体方法如图 31 所示。

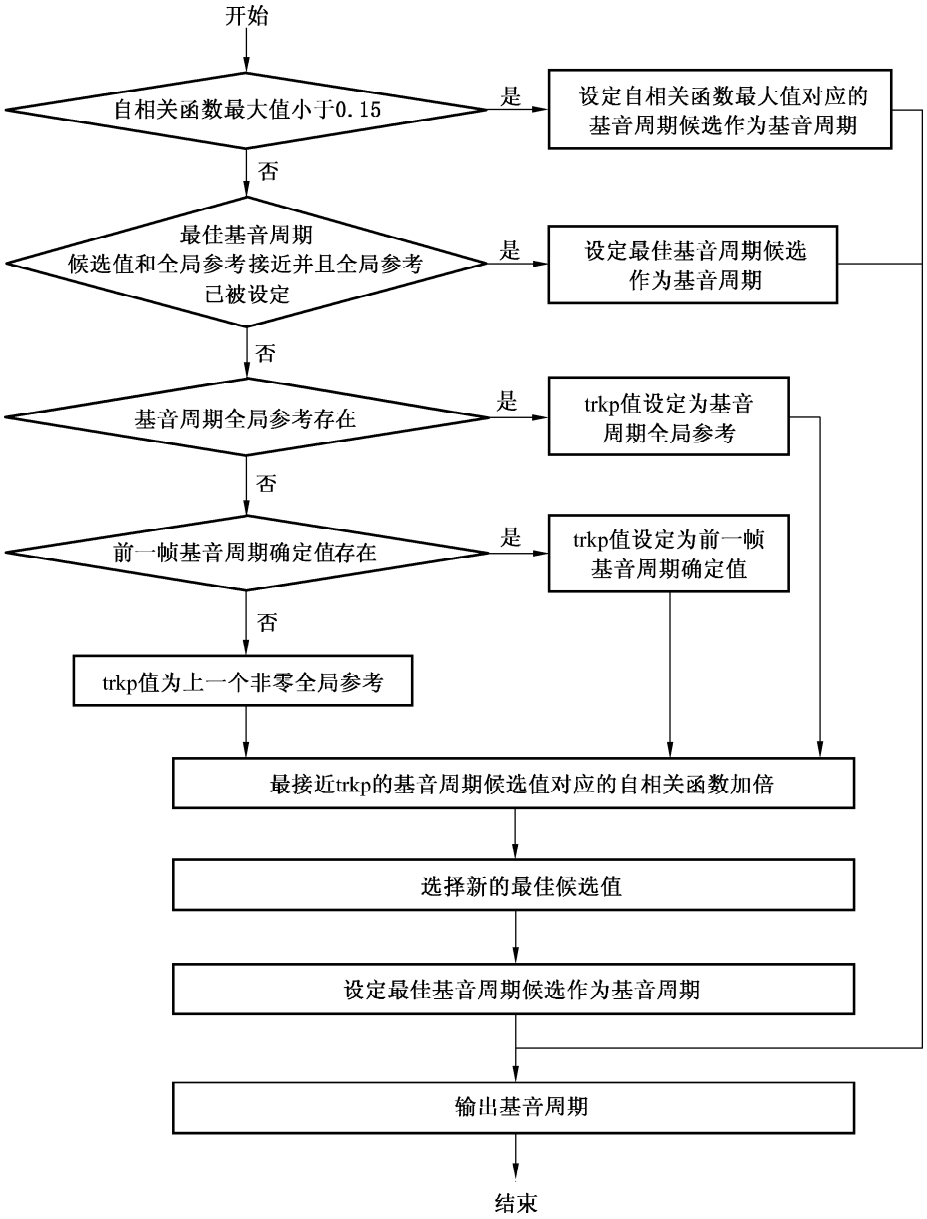


图 31　基音周期确定流程

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 $1/H_{\text{emph}}(z)$ 得到脉冲响应 $h(n)$。

$$H(z)W(z)=\frac{A(z/\gamma_1)}{\hat{A}(z)H_{\text{emph}}(z)} \qquad \cdots\cdots(43)$$

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，见式(44)：

$$x(n)=s_w(n)-\hat{s}_0 \qquad \cdots\cdots(44)$$

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，即使式(45)中的 $R(k)$ 最大。

$$R(k)=\frac{\sum_{n=0}^{63}x(n)y_k(n)}{\sqrt{\sum_{n=0}^{63}y_k^2(n)}} \qquad \cdots\cdots(45)$$

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：

$exc(k)$，　$k=-(231+17),\cdots,63$——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 $v(n)$，见式(47)：

$$v(n)=\sum_{i=0}^{15}(exc(n-k-i)b_{64}(t+4i)+exc(n-k+1+i)b_{64}(4(i+1)-t)),$$
$$n=0,\cdots,63 \quad t=0,\cdots,3 \qquad \cdots\cdots(47)$$

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 $v\_low(n)$，其计算过程如式(48)：

$$v\_low(n)=\sum_{k=-1}^{1}b(k)v(n-k), \quad n=0,\cdots,63 \qquad \cdots\cdots(48)$$

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 $v\_high(n)$，见式(49)：

$$v\_high(n)=v(n)-v\_low(n), \qquad n=0,\cdots,63 \qquad \cdots\cdots(49)$$

计算 LP 残差信号 $r(n)$，见式(50)：

$$r(n)=s(n)+\sum_{i=1}^{16}\hat{a}_i s(n-i), \qquad n=0,\cdots,63 \qquad \cdots\cdots(50)$$

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 $corr$，见式(51)：

$$corr=\frac{\sum_{n=0}^{63}r(n)\times v\_high(n)}{\sqrt{\sum_{n=0}^{63}r^2(n)\times\sum_{n=0}^{63}v\_high^2(n)}} \qquad \cdots\cdots(51)$$

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 见式(52)，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 见式(53)：

$$g_p=\frac{\sum_{n=0}^{63}x(n)\times synth'(n)}{\sum_{n=0}^{63}(synth'(n))^2} \qquad \cdots\cdots(53)$$

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 142 所示。

表 142 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 143 所示(脉冲 Pi 的序号 i 表示搜索顺序)。

表 143 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 144。

表 144 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 145。

表 145 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 146。

表 146 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 147。

表 147 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 148。

表 148 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 149。

表 149 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 kkbps 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 150。

表 150 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，见式(54)：

$$x_0'(n)=x_0(n)-g_p y_1(n) \qquad \cdots\cdots(54)$$

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，则使用式(55)计算平方误差：

$$e=\sum_{n=0}^{N-1}[x_0'(n)-g_c y_2(n)]^2 \qquad \cdots\cdots(55)$$

式中：

N——子帧长度。

最小的 g_c（通过 e 对 g_c 的偏导数为零得到）见式(56)：

$$g_c=\frac{\sum_{n=0}^{N-1}x_0'(n)y_2(n)}{\sum_{n=0}^{N-1}y_2^{\ 2}(n)} \qquad \cdots\cdots(56)$$

将得到的 g_c 代入式(55)得最小平方误差，见式(57)：

$$e_{\min}=\sum_{n=0}^{N-1}x'^{\ 2}_0(n)-\frac{\left[\sum_{n=0}^{N-1}x_0'(n)y_2(n)\right]^2}{\sum_{n=0}^{N-1}y_2^2(n)} \qquad \cdots\cdots(57)$$

选择使 $e_{\min}$ 最小的代数码书激励矢量，即选择使式(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，见式(58)：

$$Q_k=\frac{(x'^{\ t}_0Hc_k)^2}{c_k^tH^tHc_k}=\frac{(d^tc_k)^2}{c_k^t\Phi c_k}=\frac{(R_k)^2}{E_k} \qquad \cdots\cdots(58)$$

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，见式(59)和式(60)：

$$d(n)=\sum_{i=n}^{63}x'_0(i)h(i-n),\qquad n=0,\cdots,63 \qquad \cdots\cdots(59)$$

$$\phi(i,j)=\sum_{n=j}^{63}h(n-i)h(n-j),\quad i=0,\cdots,63,\quad j=i,\cdots,63 \qquad \cdots\cdots(60)$$

因为代数码矢量只有少量的非零脉冲，所以公式(58)中的分子表示如式(61)：

$$C=\sum_{i=0}^{N_p-1}a_i d(m_i) \qquad \cdots\cdots(61)$$

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)的分母表示如式(62)：

$$E=\sum_{i=0}^{N_p-1}\phi(m_i,m_i)+2\sum_{i=0}^{N_p-2}\sum_{j=i+1}^{N_p-1}a_i a_j\phi(m_i,m_j) \qquad \cdots\cdots(62)$$

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 $b(n)$ 的计算公式见式(63)和式(64)：

$$b(n)=\frac{res'_{\mathrm{LTP}}(n)}{\sqrt{\sum_{i=0}^{63}res'_{\mathrm{LTP}}(i)res'_{LTP}(i)}}+\frac{d(n)}{\sqrt{\sum_{i=0}^{63}d(i)d(i)}},\qquad n=0,\cdots,63 \qquad \cdots\cdots(63)$$

$$res'_{\mathrm{LTP}}(n)=res_{\mathrm{LTP}}(n)\otimes w(n) \qquad \cdots\cdots(64)$$

式中：

$w(n)$ ——谱倾斜滤波器 $(1-0.3z^{-1})$ 的脉冲响应；

$res_{\mathrm{LTP}}(n)$ ——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 $s_b(n)=sign\ [b(n)]$ 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)可分别转化为式(65)和式(66)：

$$C=\sum_{i=0}^{N_p-1}d'(m_i) \qquad \cdots\cdots(65)$$

$$E=\sum_{i=0}^{N_p-1}\phi'(m_i,m_i)+2\sum_{i=0}^{N_p-2}\sum_{j=i+1}^{N_p-1}\phi'(m_i,m_j) \qquad \cdots\cdots(66)$$

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 151。

表 151　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 $8\times16=128$(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 $8\times16\times4=512$(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：$512\times4=2\ 048$(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 32 给出了分类组合索引编码的处理流程。

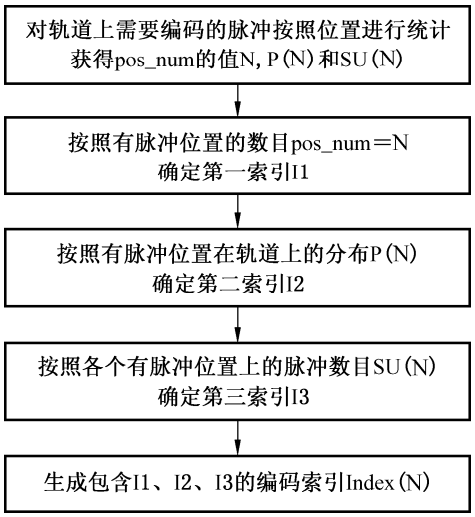


图 32　代数码书分类组合索引编码流程框图

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∈[1,ω]。有脉冲位置矢量 P(N)={p(0),p(1),…,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∈[0,N−1],p(n)∈[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),…,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)+…+su(N−1)=ω。

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),…,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 $P(N)$,见式(67):

$$P(N)=\{p(0),p(1),\cdots,p(N-1)\} \quad \cdots\cdots(67)$$

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,计算公式见式(68):

$$I2 = C_M^N - C_{M-p(0)}^N + \sum_{n=1}^{N-1}\left[C_{M-p(n-1)-1}^{N-n} - C_{M-p(n)}^{N-n}\right] \quad \cdots\cdots(68)$$

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∈[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)+…+su(N−1)=ω,且ω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∈[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，合并公式见式(69)：

$$I23 = I3 \times C_M^N + I2 \quad \cdots\cdots(69)$$

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 $Index(N)$的表示见式(70)：

$$Index(N) = I1 + I23 \times 2^N + s(0) \times 2^{N-1} + s(1) \times 2^{N-2} + \cdots + s(N-1) \quad \cdots\cdots(70)$$

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 152～表 157。

表 152 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	I3×4 368+I2							s(0)	s(1)	s(2)	s(3)	s(4)	
4	0x080000	0	0	1	I3×1 820 + I2							s(0)	s(1)	s(2)	s(3)	
3	0x1C0000	0	1	1	1	0	0	I3×560 + I2						s(0)	s(1)	s(2)
2	0x1D0000	0	1	1	1	0	1	I3×120 + I2							s(0)	s(1)
1	0x1E0000	0	1	1	1	1	0	I2								s(0)

表 153 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	I3×1 820 + I2							s(0)	s(1)	s(2)	s(3)
3	0x10000	0	0	0	1	I3×560 + I2							s(0)	s(1)	s(2)
2	0x08000	0	0	0	0	1	I3×120 + I2							s(0)	s(1)
1	0x00000	0	0	0	0	0	I2								s(0)

表 154 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 155　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 156　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 157　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 $\overline{E}_s$ 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 $E_s(n)$ 是第 n 个子帧代数码书激励矢量的能量，计算公式见式(71)：

$$E_s(n)=10\log_{10}\left(\frac{1}{N}g_c^2\sum_{i=0}^{N-1}c^2(i)\right)=20\log_{10}(g_c)+E_i \qquad \cdots\cdots(71)$$

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，见式(72)：

$$E_i=10\log_{10}\left(\frac{1}{N}\sum_{i=0}^{N-1}c^2(i)\right) \qquad \cdots\cdots(72)$$

计算代数码书平均能量 $\overline{E}_s$ 并量化，然后用于计算代数码书预测增益 g'_c，见式(73)：

$$g'_c=10^{0.05(\overline{E}_s-E_i)} \qquad \cdots\cdots(73)$$

上式是从公式 $\overline{E}_s=20\log_{10}(g'_c)+E_i$ 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 由式(74)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，计算公式见式(75)：

$$E=\frac{1}{N}\sum_{n=0}^{N-1}\left[x_0(n)-\hat{g}_p y_1(n)-\hat{g}_c y_2(n)\right]^2 \quad \cdots\cdots(75)$$

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 $\overline{E}_s$ 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，见式(76)：

$$E_{\mathrm{res}}(n)=10\log_{10}\left(\frac{1}{N}\sum_{i=0}^{N-1}r^2(i)\right) \quad \cdots\cdots(76)$$

然后计算每帧的平均残差能量，见式(77)：

$$\overline{E}_{\mathrm{res}}=\frac{1}{4}\sum_{n=0}^{3}E_{\mathrm{res}}(n) \quad \cdots\cdots(77)$$

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 $\overline{E}_s$。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，见式(78)：

$$\overline{E}_s=\overline{E}_{\mathrm{res}}-10\overline{R} \quad \cdots\cdots(78)$$

式中：

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 $\overline{E}_s$ 每帧用 2 比特量化，有 4 个量化级别：18，30，42，54。通过每次对 $\overline{E}_s$ 加上 12(量化索引加 1)迭代，来限制 $\overline{E}_s$ 量化范围 $(E_{\max}-27)<\overline{E}_s\leqslant 54$。其中 $E_{\max}$ 是 4 个子帧中 $E_{\mathrm{res}}(n)$ 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 33。

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。

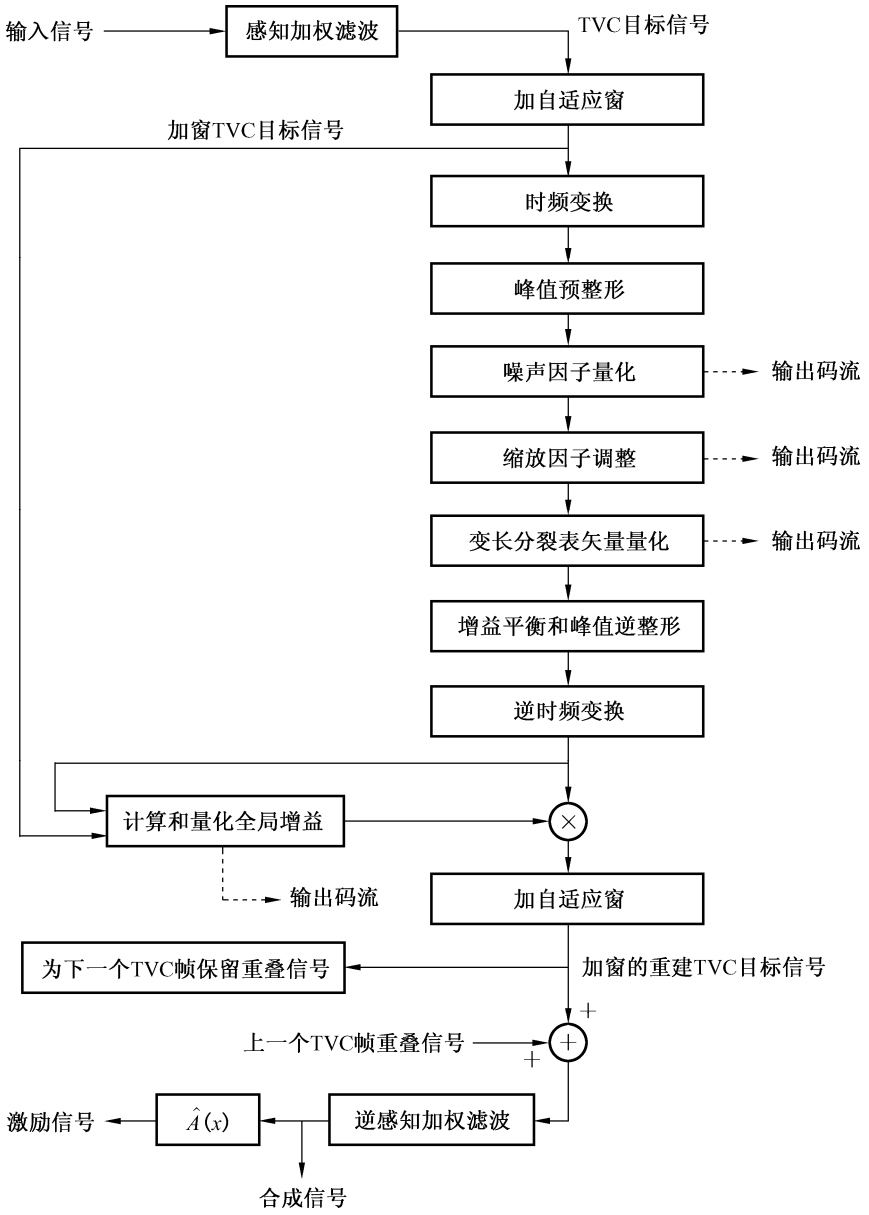


图 33　TVC 编码过程图

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见式(79)：

$$W(z)=\frac{\hat{A}(z/\gamma_1)}{H_{\text{emph}}(z)} \qquad \cdots\cdots(79)$$

式中：

$\gamma_1=0.92$；

$\hat{A}(z)$　——由量化系数构成的 LP 滤波器；

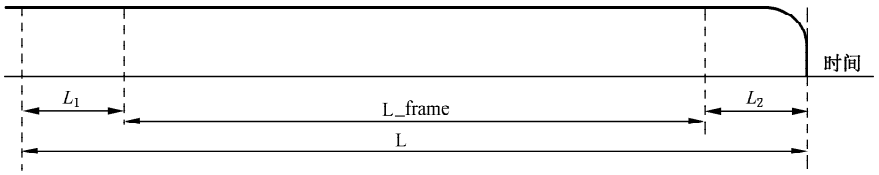
$H_{\text{emph}}(z)$——预加重滤波器。

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 34 所示。



说明：

$L_1=16$；

$L_2=16$；

L_frame=256；

$L=288$。

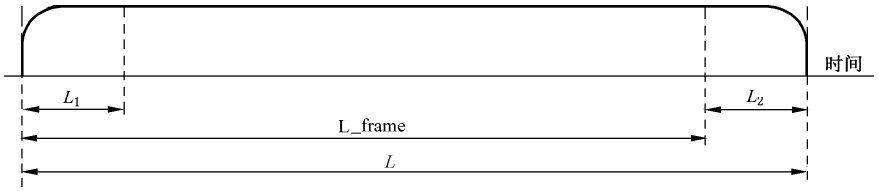
图 34 上一帧 ACELP 编码时自适应窗

图 34 中自适应窗的窗函数见式(80)：

$$\begin{aligned} w_1(n)&=1, \quad n=0,\cdots,L-L_2-1 \\ w_2(n)&=\cos(2\pi n/(4L_2)), \quad n=0,\cdots,L_2-1 \end{aligned} \qquad \cdots\cdots(80)$$

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 35 所示。



说明：

$L_1=16$(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 35 上一帧 TVC 编码时自适应窗

图 35 中对应自适应窗的窗函数见式(81)：

$$\begin{aligned} w_1(n)&=\sin(2\pi n/(4L_1)), && n=0,\cdots,L_1-1 \\ w_2(n)&=1, && n=0,\cdots,L-L_1-L_2-1 \\ w_3(n)&=\cos(2\pi n/(4L_2)), && n=0,\cdots,L_2-1 \end{aligned} \qquad \cdots\cdots(81)$$

6.2.4.4 时频变换

加窗信号通过 DFT 变换到频域，见式(82)。

$$X(k)=\sum_{n=0}^{L_{\mathrm{DFT}}-1}x(n)\mathrm{e}^{\frac{-2jnk\pi}{L_{\mathrm{DFT}}}} \qquad \cdots\cdots(82)$$

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 36。

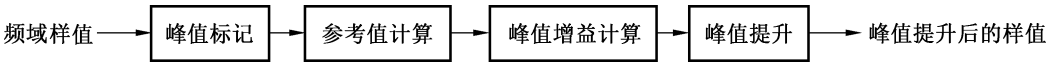


图 36 峰值预整形算法流程

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 见式(83)：

$$E_k=\max(2,\sum_{m=0}^{7}B_k[m]B_k[m]) \qquad \cdots\cdots(83)$$

根据 E_k 得到消耗比特数的初始估计，见式(84)：

$$R_k(1)=5\log_2(\frac{E_k}{2}) \qquad \cdots\cdots(84)$$

比特消耗估计迭代：

初始条件：设 fac=128，offset=0 和 nbits_max= $0.95\times(b_{\max}-K)$，$b_{\max}$ 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。

噪声因子计算见式(85)：

$$\delta_{\text{noise}} = 10^{\frac{\log 10(2)}{10}(\text{nbits}-5)} \quad \cdots\cdots(85)$$

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 $idx = floor((8-10\times\delta_{\text{noise}})+0.5)$。

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 37。

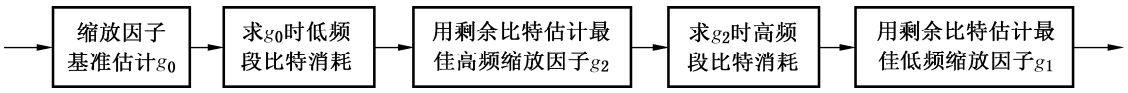


图 37 两个缩放因子的调整算法

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 $b_h<b_{\max}-b_l$ 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 $\frac{g_1}{g_2}$ 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 38，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。

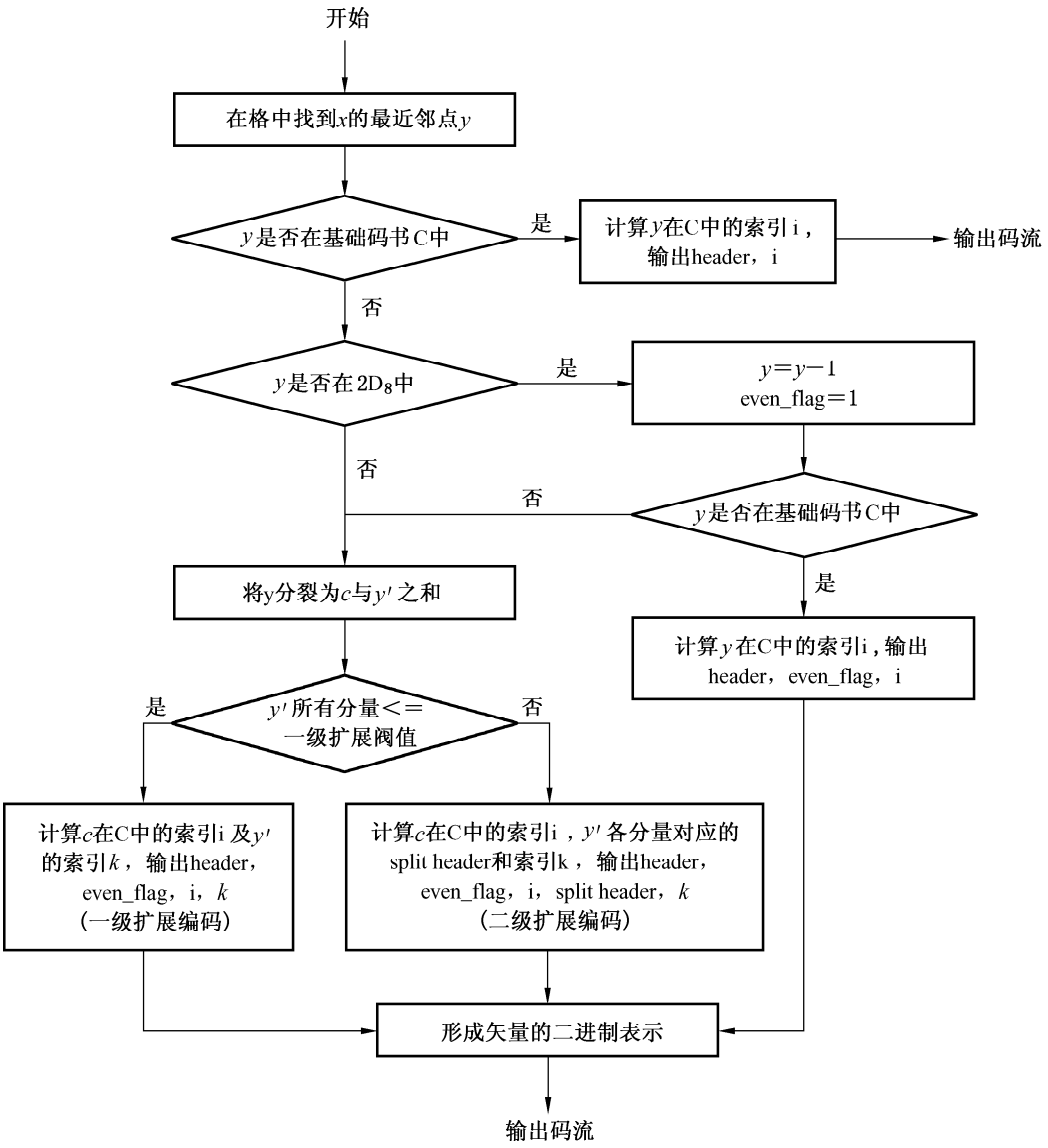


图 38　基于分裂表的矢量量化方法编码示意图

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 至 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 $D_8 = \{(x_1,x_2,\cdots,x_8) \in Z^8 \mid x_1 + \cdots + x_8$ 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 158。

表 158 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)				√	

表 158（续）

特征码字	Q_0	Q_2	Q_3	Q_4	inv_Q_4
(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 = 1100　　基础码书为 Q_3
header = 1110　　基础码书为 Q_4
header = 11111110　　基础码书为 inv_Q_4

6.2.4.8.4 基础码书编码

基础码书编码包括直接编码，特殊特征码字编码，$2D_8$ 数据的奇化编码，以及基于缓存的常见特征码字快速码书搜索，如下：

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 39。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。

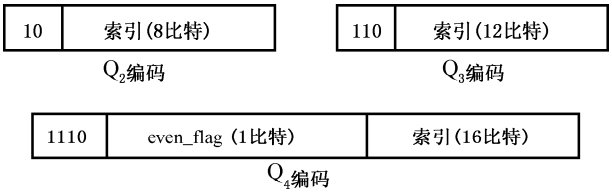


图 39　Q_2，Q_3，Q_4 输出编码格式

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 40。

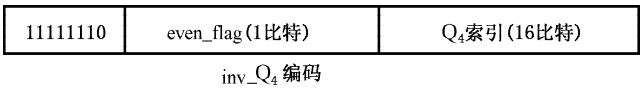


图 40　特殊特征码字的输出编码格式

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 41 所示。主要步骤为：

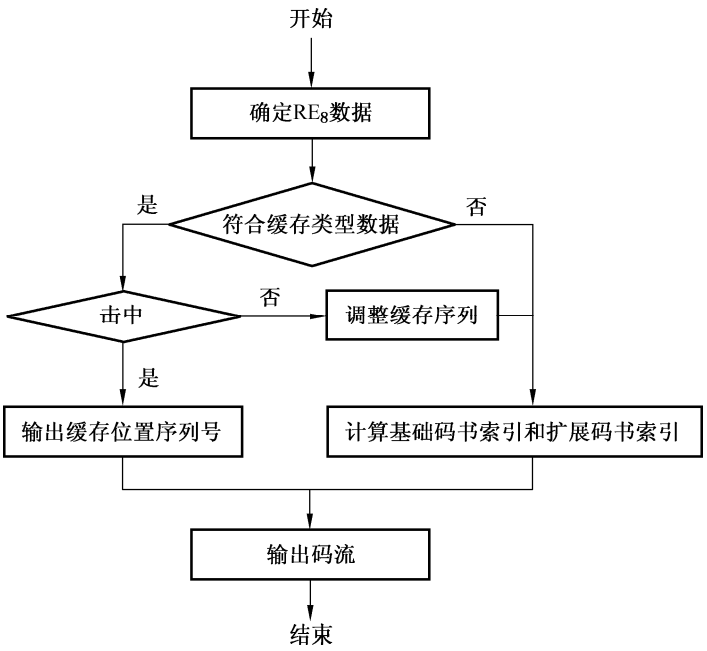


图 41　基于缓存的快速码书搜索编码流程图

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

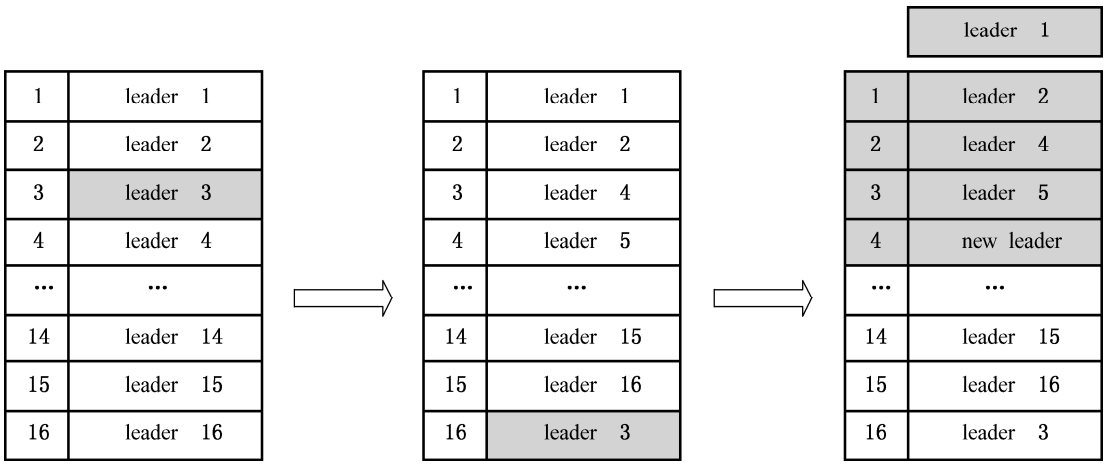
将码书中出现机率最高的特征码字选出来，遇到这类数据时便对其进行与缓存相关的操作，这些数据称之为缓存类型数据。选取特征码字为(1,1,1,1,1,1,1,1)和(2,2,0,0,0,0,0,0)的数据作为缓存类型数据。选取了 16 个空间作为缓存空间，即如果用二进制表示，需要 4 比特。

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 42。leader1 到 leader16 及 New leader 均为特征码字的标号。

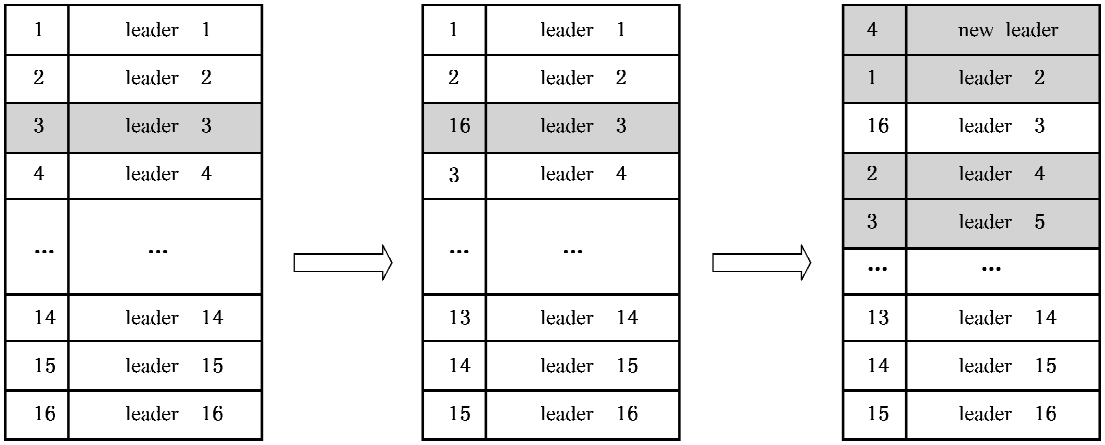


a) 某状态下缓存数据存放情况　b) 特征码字 leader 3 被击中后数据更新情况　c) 在 b)状态下新的数据进来后未被击中时缓存数据替换情况

图 42　缓存中的数据替换过程

图 42a)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 40b)。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 42c)。图 42c)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 43。图 43 与图 42 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 43 实际编码中缓存数据替换过程

图 43a)为某状态下缓存里的数据存放情况,图 43b)表示击中特征码字 leader 3 后缓存中的数据处理过程,图 43c)表示未击中情况下新存入 1 个数据的过程。与图 43 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、4 的特征码字编号都减 1,其余特征码字对应的编号不变。

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 43a)～b),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 38,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 159，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 44 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 160 所示。

表 159 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

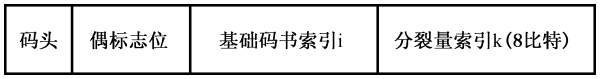


图 44 一级扩展编码的输出编码格式

表 160 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见表 161，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)，定义见表 162。

表 161 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

表 161（续）

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）
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 162　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见图 45，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见表 163。

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，所消耗的比特数见表 164。

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 45　二级扩展编码的输出编码格式

表 163　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 164　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见式(86)：

$$X_{\text{balance}}=\left[X_q(0,1,\cdots,N/2),\quad \frac{g_2}{g_1}X_q(N/2+1,N/2+2,\cdots,N)\right] \quad \cdots\cdots\cdots\cdots(86)$$

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 46。

图 46　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 变换定义见式(87)：

$$\hat{x}(n) = \frac{1}{L_{\mathrm{DFT}}} \sum_{k=0}^{L_{\mathrm{DFT}}-1} \hat{X}(k) e^{\frac{2jnk\pi}{L_{\mathrm{DFT}}}} \qquad \cdots\cdots(87)$$

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 $\hat{x}$ ，则最佳全局增益见式(88)：

$$g^* = \frac{\sum_{n=0}^{L_{\mathrm{DFT}}-1} x(n)\hat{x}(n)}{\sum_{n=0}^{L_{\mathrm{DFT}}-1} \hat{x}(n)\hat{x}(n)} \qquad \cdots\cdots(88)$$

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：$E = \sum_{n=0}^{L_{\mathrm{DFT}}-1} \hat{x}^2(n)$ ；

b) 计算 RMS 值：$rms = 4\sqrt{\frac{E}{L_{\mathrm{DFT}}}}$ ；

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 $\hat{g}^*$ 计算见式(89)：

$$\hat{g}^* = 10^{\frac{index}{28}} / rms \qquad \cdots\cdots(89)$$

6.2.5 高频信号编码(BWE)

高频信号编码框图见图 47 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 $F_s/4$ 的信号分量，高频信号的带宽取决于输入信号的采样频率。

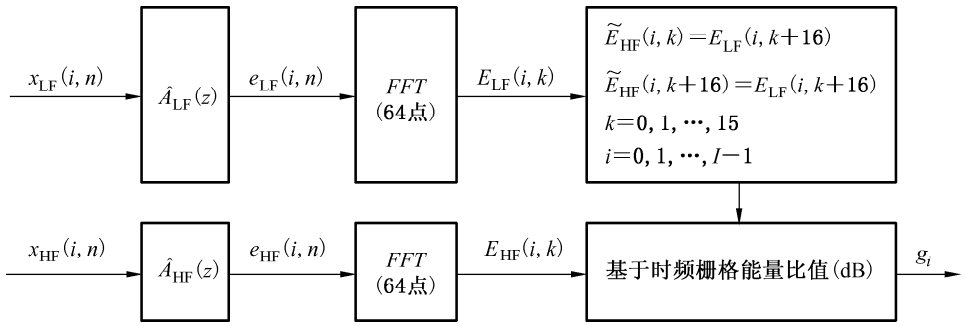


图 47 高频编码框图

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 插值公式(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 $\tilde{E}_{HF}(i,k)$。

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 48。

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 48 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。

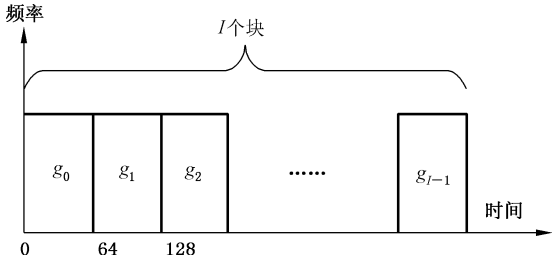


图 48　时频栅格划分图

每个超帧 BWE 参数的总比特消耗为：$16\times(I/4)$，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如式(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_{in} 为缓存中前 400 个样本。图 49 描述了波形预处理模块的原理。

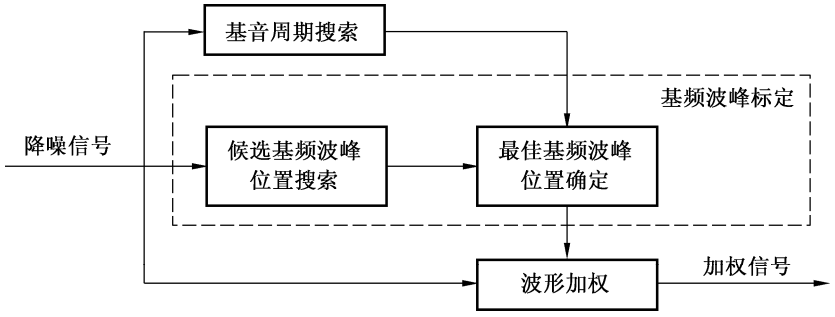


图 49 波形预处理原理框图

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):

$$corr=\frac{\sum_{n=0}^{239}s_{nr}(n)s_{nr}(n-delay)}{\sqrt{\sum_{n=0}^{239}s_{nr}^2(n)\sum_{n=0}^{239}s_{nr}^2(n-delay)}},\quad 40\leqslant delay\leqslant 160 \qquad \cdots\cdots(91)$$

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：$Pc[i]$，$i=1,2,\cdots,M$；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)：

$$Pc[1],\cdots,Pc[M]=\underset{n=0,\cdots,L-1}{\overset{\text{最大的}M\text{个值}}{\operatorname{argmax}}}(|s_{nr}(n)|) \quad \cdots\cdots(92)$$

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 T_p 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)：

$$Mp\_avg[i]=\frac{\left|\sum_{j}s(Pc[i]+j\times T_p)\right|}{D},\quad i=1,\cdots,M \quad \cdots\cdots(93)$$

式中：

D ——满足 $0\leqslant Pc[i]+j\times T_p\leqslant N_{\text{in}}-1$ 的 j 的个数；

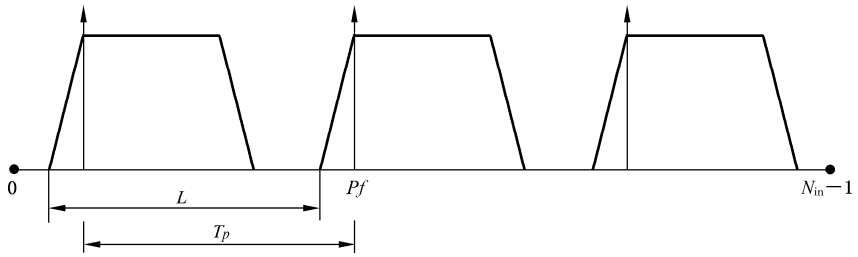
$Mp\_avg[i]$——根据第 i 个候选基频波峰位置所计算的平均幅度值，M 为候选基频波峰位置个数。

假设最佳的基频波峰位置为 Pf，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。如图 50 所示。



说明：

L ——窗函数的长度；

T_p ——基音周期；

Pf ——最佳的基频波峰位置。

图 50 波形加权处理框图

窗函数的定义见式(95)：

$$w_{swp}(n)=\begin{cases}0.8, & n=0\\ 1.0, & n=1\\ 1.2, & 2\leqslant n\leqslant 0.6T_p\\ 1.0, & n=0.6T_p+1\\ 0.8, & 0.6T_p+2\leqslant n\leqslant T_p-1\end{cases} \quad \cdots\cdots(95)$$

按照每个基音周期进行加权，见式(96)：

$$\begin{aligned}s_{swp}(n)&=w_{swp}(i)\times s_{nr}(n),\quad n=Pf\_n+i-2,\quad 0\leqslant i<T_p\\ Pf\_n&+=T_p\end{aligned} \quad \cdots\cdots(96)$$

第一个基音周期和最后一个基音周期边界处理，保证 $0\leqslant n<N_{in}$。Pf_n 表示按基音周期扩展的基频波峰位置，其初值通过式(97)获得：

$$\begin{aligned}Pf\_n&=Pf\\ While(Pf\_n>0)&\quad Pf\_n-=T_p\end{aligned} \quad \cdots\cdots(97)$$

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，见式(98)和式(99)：

$$\ln E=\begin{cases}\ln(E_{swp}), & E_{swp}\geqslant E_{THRESH}\\ \ln(E_{THRESH}), & E_{swp}<E_{THRESH}\end{cases} \quad \cdots\cdots(98)$$

$$E_{THRESH}=\exp(-1),E_{swp}=\sum_{n=0}^{N_{in}-1}S_{swp}(n)\times S_{swp}(n) \quad \cdots\cdots(99)$$

6.2.6.1.4.2 预加重

预加重滤波器处理后的信号 $s_{swp\_pe}(n)$ 见式(100)：

$$s_{swp\_pe}(n)=s_{swp}(n)-0.9\times s_{swp}(n-1) \quad \cdots\cdots(100)$$

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，见式(101)：

$$s_{swp\_w}(n)=\left[0.54-0.46\times\cos\left(\frac{2\pi\times(n+0.5)}{N_{in}}\right)\right]\times s_{swp\_pe}(n),\quad 0\leqslant n\leqslant N_{in}-1 \quad \cdots(101)$$

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 $X_{swp}(bin)$，见式(102)：

$$X_{swp}(bin)=FFT\{s_{swp\_w}(n)\},\quad 0\leqslant bin\leqslant N_{FFT}/2 \quad \cdots\cdots(102)$$

相应的功率谱 $P_{swp}(bin)$ 见式(103)：

$$P_{swp}(bin)=|X_{swp}(bin)|^2,\quad 0\leqslant bin\leqslant N_{FFT}/2 \quad \cdots\cdots(103)$$

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 索引公式见式(104)：

$$index\{f\} = round\left\{\frac{f}{f_{samp}} \times N_{FFT}\right\} \quad \cdots\cdots(104)$$

Mel 函数计算公式见式(105)：

$$Mel\{x\} = \Lambda \times \log_{10}\left(1+\frac{x}{\mu}\right) = \lambda \times \ln\left(1+\frac{x}{\mu}\right), \quad \lambda = \frac{\Lambda}{\ln(10)} \quad \cdots\cdots(105)$$

Mel 反函数计算公式见式(106)：

$$Mel^{-1}\{y\} = \mu \times \left(\exp\left(\frac{y}{\lambda}\right)-1\right) \quad \cdots\cdots(106)$$

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 51 给出了线性频率和 Mel 频率之间的映射关系。

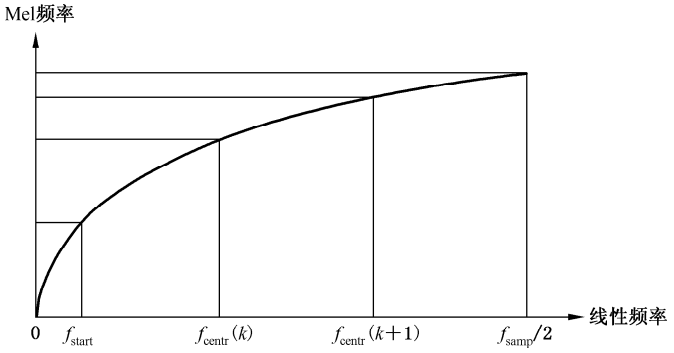


图 51　线性频率和 Mel 频率映射

中心频率的计算见式(107)：

$$f_{centr}(k) = Mel^{-1}\left\{Mel\{f_{start}\} + k \times \frac{Mel\{f_{samp}/2\} - Mel\{f_{start}\}}{K_{FB}+1}\right\}, \quad 1 \leqslant k \leqslant K_{FB} \cdots(107)$$

式中：

$f_{start} = 64$ Hz；

$f_{samp} = 16\ 000$ Hz；

$\mu = 700$；

$\Lambda = 2\ 595$；

$\lambda = 1\ 127$；

$K_{FB} = 32$。

将 Mel 滤波器的中心频率表示为 FFT 索引，见式(108)：

$$bin_{centr}(k) = index\{f_{centr}(k)\} = round\left\{\frac{f_{centr}(k)}{f_{samp}} \times N_{FFT}\right\}, \quad 1 \leqslant k \leqslant K_{FB} \quad \cdots\cdots(108)$$

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，见式(109)：

$$W_{left}(i,k) = \frac{i - bin_{centr}(k-1) + 1}{bin_{centr}(k) - bin_{centr}(k-1) + 1}, \quad 1 \leqslant k \leqslant K_{FB}, bin_{centr}(k-1) \leqslant i \leqslant bin_{centr}(k)$$

$$W_{right}(i,k) = 1 - \frac{i - bin_{centr}(k)}{bin_{centr}(k+1) - bin_{centr}(k) + 1}, \quad 1 \leqslant k \leqslant K_{FB}, bin_{centr}(k) < i \leqslant bin_{centr}(k+1)$$

$$\cdots\cdots(109)$$

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，见式(110)：

$$E_{FB}(k)=\sum_{i=bin_{centr}(k-1)}^{bin_{centr}(k)}W_{left}(i,k)\times P_{swp}(i)+\sum_{i=bin_{centr}(k)+1}^{bin_{centr}(k+1)}W_{right}(i,k)\times P_{swp}(i),\ 1\leqslant k\leqslant K_{FB}\cdots(110)$$

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，见式(111)～式(113)：

$$E_{FB}(24)=\frac{\alpha E_{FB}(24)+\beta E_{FB}(25)+\gamma E_{FB}(26)}{\alpha+\beta+\gamma}\quad\cdots\cdots(111)$$

$$E_{FB}(25)=\frac{\alpha E_{FB}(27)+\beta E_{FB}(28)+\gamma E_{FB}(29)}{\alpha+\beta+\gamma}\quad\cdots\cdots(112)$$

$$E_{FB}(26)=\frac{\alpha E_{FB}(30)+\beta E_{FB}(31)+\gamma E_{FB}(32)}{\alpha+\beta+\gamma}\quad\cdots\cdots(113)$$

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，见式(114)：

$$S_{FB}(k)=\ln(E_{FB}(k)),\quad 1\leqslant k\leqslant K_{FB}\cdots\cdots(114)$$

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，见式(115)：

$$c(i)=\sum_{k=1}^{K_{FB}}S_{FB}(k)\times\cos\left(\frac{i\times\pi}{K_{FB}}\times(k-0.5)\right),\ 0\leqslant i\leqslant 12\quad\cdots\cdots(115)$$

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，见式(116)：

$$c_{eq}(i)=c(i)-TranCep(i),i=1,\cdots,12\quad\cdots\cdots(116)$$

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，见式(117)：

$$TranCep(i)=TranCep(i)(1-\alpha)+(c(i)-RefCep(i))\beta_1+c(i)\beta_2 \quad \cdots\cdots(117)$$

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，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 165 设置。

表 165　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，见式(118)：

$$C_{eq}(t)=[c_{eq}(1,t),c_{eq}(2,t),\cdots,c_{eq}(12,t)]^T \quad \cdots\cdots(118)$$

式中：

t——帧索引。

特征矢量还包括 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见式(119)：

$$y(t)=\begin{bmatrix} C_{eq}(t) \\ VAD(t) \\ c(0,t) \\ \ln E(t) \end{bmatrix} \quad \cdots\cdots(119)$$

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 $\ln E$]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，见式(120)和式(121)：

$$d_j^{i,i+1}=\begin{bmatrix} y_i(t) \\ y_{i+1}(t) \end{bmatrix}-q_j^{i,i+1} \quad \cdots\cdots(120)$$

$$idx^{i,i+1}(t)=\underset{0\leqslant j\leqslant(N^{i,i+1}-1)}{\operatorname{argmin}}\{(d_j^{i,i+1})^T W^{i,i+1}(d_j^{i,i+1})\},\quad i=\{0,2,4,\cdots,12\} \quad \cdots\cdots(121)$$

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 $W^{12,13}$ 见(122)：

$$W^{12,13}=\begin{bmatrix}0.000\ 717\ 185 & 0\\ 0 & 1\end{bmatrix} \quad \cdots\cdots(122)$$

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 166),另一组是量化残差矢量(见表 167)。

表 166 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	$[c(0),\ln E]$

表 167 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	$[c(0),\ln E]$

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,如图 52 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。

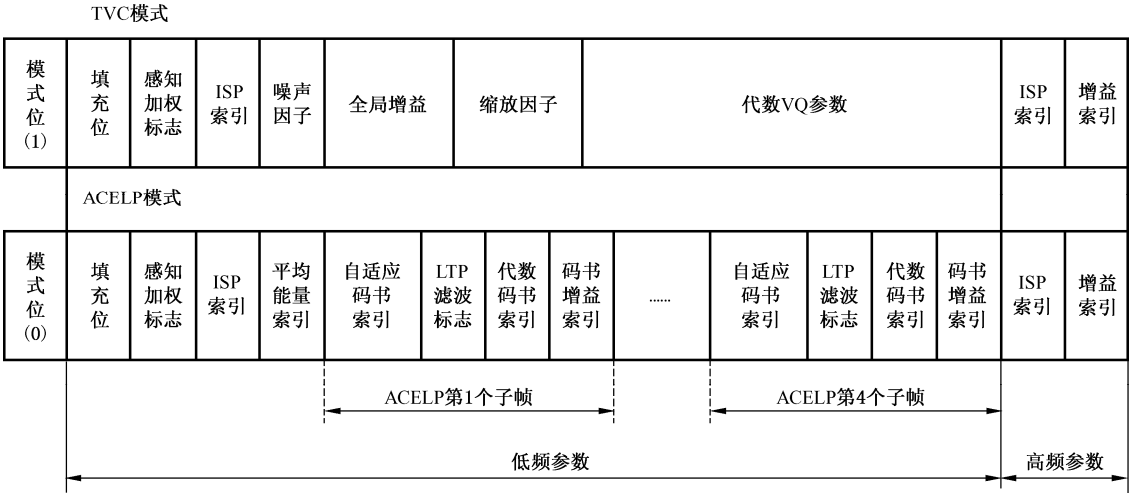


图 52　音频帧编码参数的数据格式

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 53 和图 54 所示。

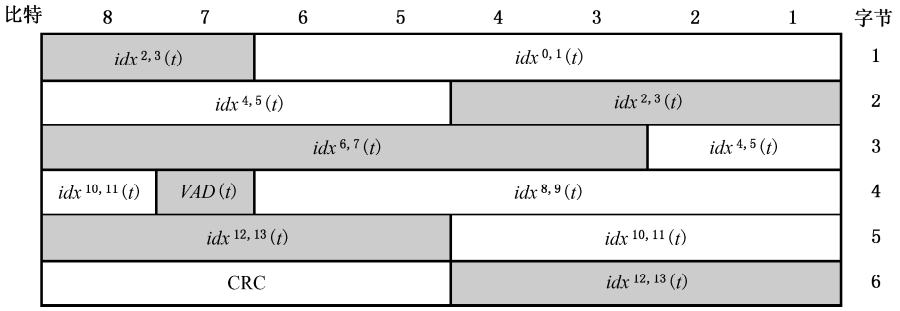


图 53　直接编码模式帧的数据格式（48 比特）

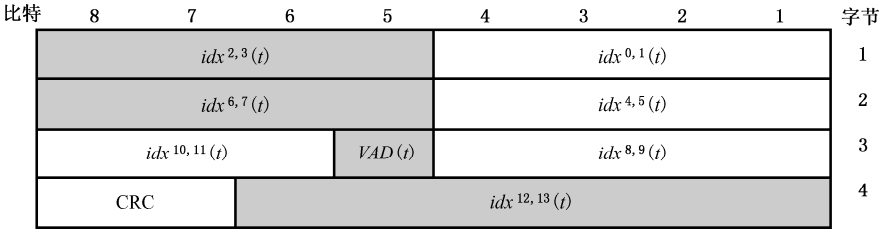


图 54　预测编码模式帧的数据格式（32 比特）

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 55 所示。

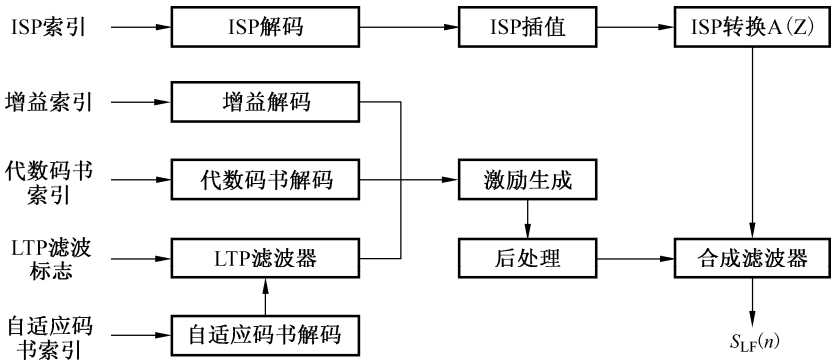


图 55 ACELP 解码合成框图

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，见式(123)：

$$isf\_diff = \frac{\sum_{i=0}^{14} (isf\_new_i - isf\_old_i)^2}{400\ 000} \quad \cdots\cdots (123)$$

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，见式(124)：

$$g_0 = \begin{cases} \max(g_{-1}, \hat{g}_c/1.06), & \hat{g}_c > g_{-1} \\ \min(g_{-1}, \hat{g}_c \times 1.06), & \hat{g}_c \leqslant g_{-1} \end{cases} \quad \cdots\cdots (124)$$

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，见式(125)：

$$\hat{g}_c = \begin{cases} 0.17 g_0 + 0.83 \hat{g}_c, & isf\_diff > 0.58 \\ 0.83 g_0 + 0.17 \hat{g}_c, & isf\_diff \leqslant 0.58 \end{cases} \quad \cdots\cdots (125)$$

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：

$$\hat{s}(n) = \hat{u}(n) - \sum_{i=1}^{16} \hat{\alpha}_i \hat{s}(n-i), n = 0, \cdots, 63 \quad \cdots\cdots (127)$$

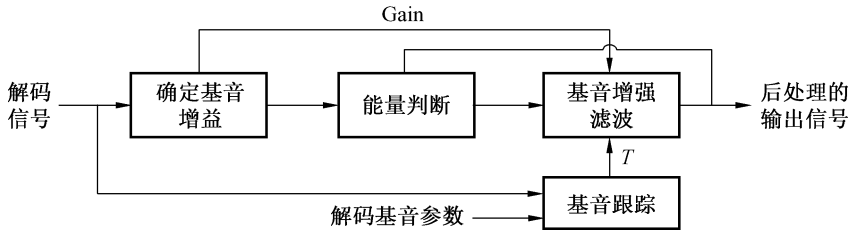
式中：

$\hat{\alpha}_i$——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如图 56 所示。



说明：

T ——解码信号的基音周期；

Gain ——解码信号的基音增益。

图 56 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)：

$$E_c = \sqrt{\frac{\sum_{n=0}^{63}\hat{s}^2(n)}{\sum_{n=0}^{63}\hat{s}^2(n-T)}} \qquad \cdots\cdots(128)$$

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，见式(129)：

$$s_E(n) = \begin{cases} \hat{s}(n), & E_c < E_t \\ \hat{s}(n) \otimes h(n), & E_c \geqslant E_t \end{cases} \qquad \cdots\cdots(129)$$

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见式(130)：

$$H(z) = G_1(1 + \lambda z^{-T}) \qquad \cdots\cdots(130)$$

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 的计算式见(131)：

$$G_1 = \frac{1}{1 + \lambda \times \text{Gain}} \qquad \cdots\cdots(131)$$

式中：

Gain——a)中计算的基音增益。

基音增强滤波的输出信号见式(132)：

$$s_E(n)=G_1(\hat{s}(n)+\lambda\times\hat{s}(n-T)) \quad \cdots\cdots(132)$$

6.3.1.2 TVC 解码

6.3.1.2.1 TVC 解码过程

TVC 解码过程如图 57 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。

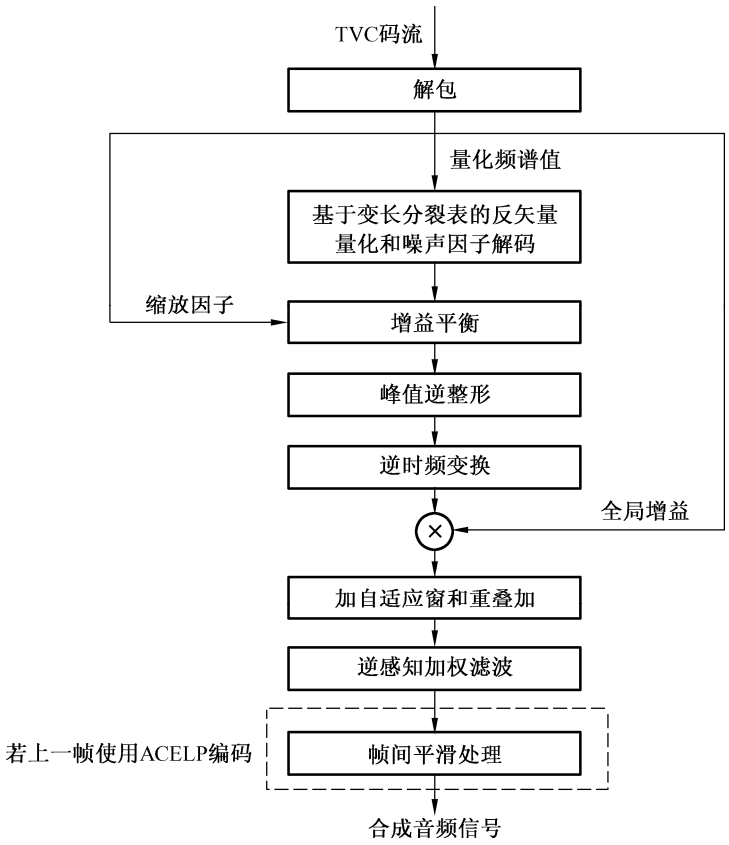


图 57 TVC 解码过程图

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 58 给出了量化频谱解码流程。

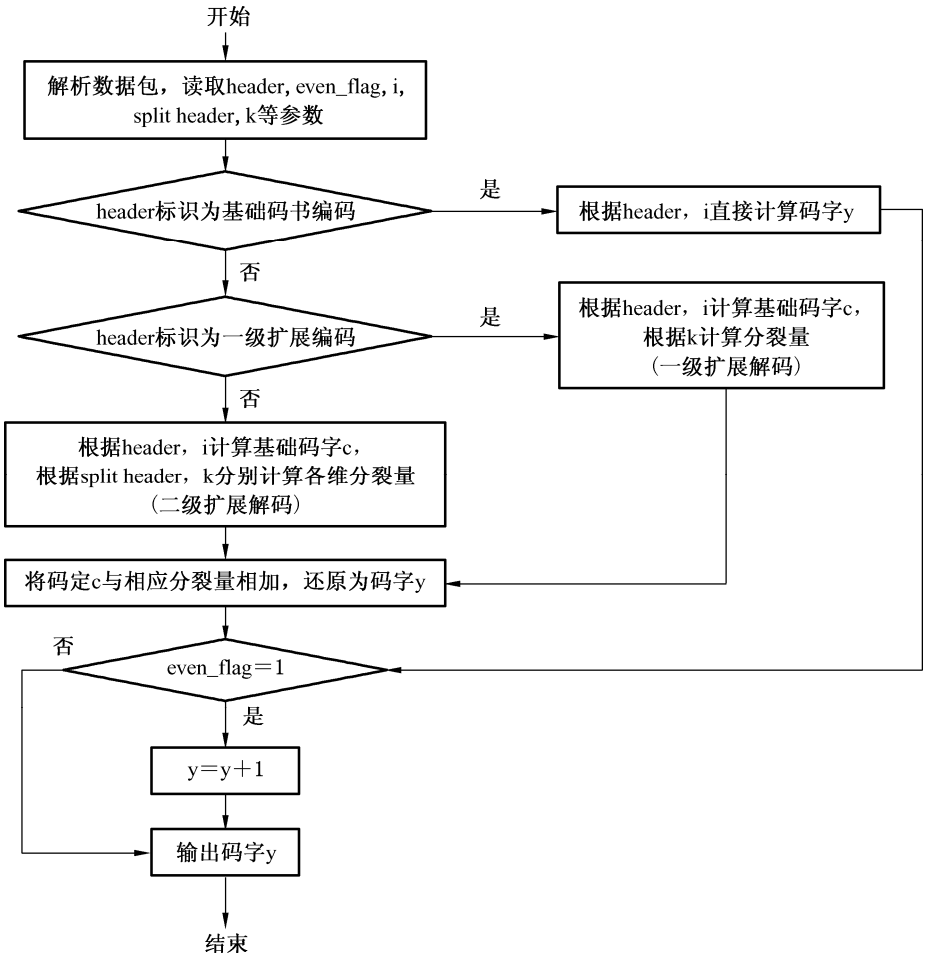


图 58　量化频谱解码过程图

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 $K/6 \leqslant k \leqslant K$ 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，见式(133)：

$$\delta_{noise}\times[\cos\theta_1 \quad \sin\theta_1 \quad \cos\theta_2 \quad \sin\theta_2 \quad \cos\theta_3 \quad \sin\theta_3 \quad \cos\theta_4 \quad \sin\theta_4] \cdots\cdots\cdots\cdots (133)$$

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 59 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。

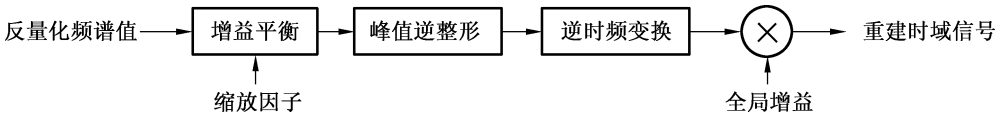


图 59　增益平衡和频谱逆整形的解码框图

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 60，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，其定义见式(134)：

$$w_1(n)=n/L_1,\qquad n=0,\cdots,L_1-1$$
$$w_2(n)=(L_1-n)/L_1,\qquad n=0,\cdots,L_1-1 \quad \cdots\cdots(134)$$

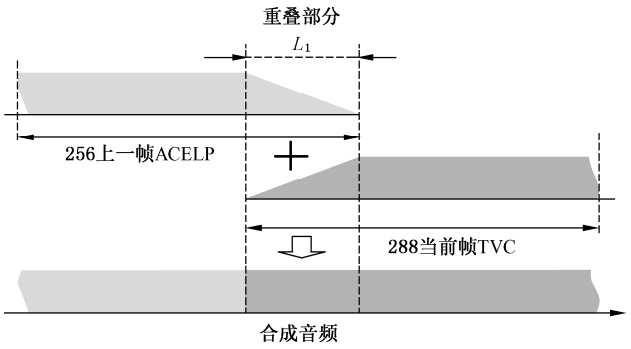


图 60　帧间平滑处理示意图

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 61 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。

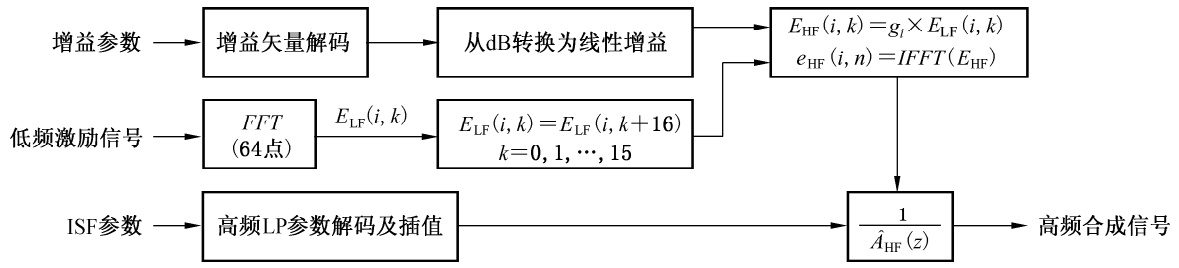


图 61　高频解码过程

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 isf_hf_q 的计算见式(135)：

$$isf\_hf\_q = cb1(i_1) + cb2(i_2) + mean\_isf\_hf + \mu_{isf\_hf} \times mem\_isf\_hf \quad \cdots\cdots(135)$$

式中：

$cb1(i_1)$ ——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 (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，然后将插值后的 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)：

$$(g_0, g_1, g_2, g_3) = cb\_gain\_hf(idx) \quad \cdots\cdots(136)$$

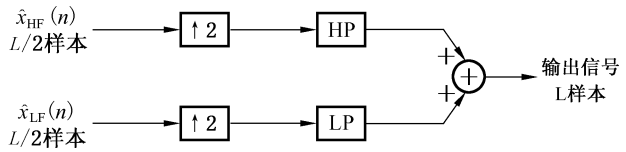
式中：

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 $F_s/2$ 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 F_s 采样频率，然后相加就得到全带输出信号，见图 62。



L——音频超帧长度。

图 62 低频和高频信号合成

如果输出信号的采样频率不同于 F_s，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，见式(137)：

$$\begin{bmatrix} \hat{y}_i(t) \\ \hat{y}_{i+1}(t) \end{bmatrix} = q^{i,i+1}_{idx^{i,i+1}(t)}, \quad i=\{0,2,4,\cdots,12\} \quad \cdots\cdots(137)$$

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 168 和表 169 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 168 ACELP 编码模式下的比特分配表 单位为比特每帧

描述	比特(MSB-LSB)							
	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

表 168（续）

单位为比特每帧

描述	比特(MSB-LSB)							
	488	424	392	344	312	280	248	216
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 169　TVC 编码模式下的比特分配表

单位为比特每帧

描述	比特(MSB-LSB)							
	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 170 规定了音频编码的编码模式和码率。

表 170 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 F_s，表 171 列出了音频编码器支持的内部采样频率。

表 171 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 171 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 172 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 172　音频编码的帧类型

帧类型	编码模式	25.6 kHz 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4.8 kbps，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：

$$N=Floor\left(\frac{\text{超帧的样本点数}}{\text{内部采样频率(Hz)}\times 0.01}\right)$$

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 173，一个音频数据单元打包一个音频超帧数据。

表 173　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 174。

表 174　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 175。

表 175　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
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)

表 175（续）

audio_frame_extension_header( ){	描述符	说明
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，具体类型定义见附录 H 中的表 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，角度定义如图 63 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°。

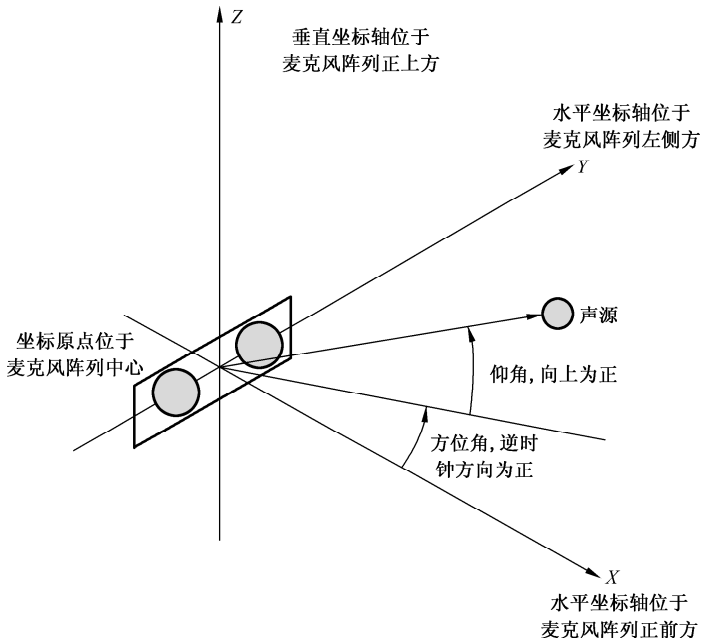


图 63 麦克风阵列的坐标系统

编码时角度量化公式为：

$$\hat{\alpha}=round(\alpha/1.406\ 25) \qquad \hat{\beta}=round(\beta/1.406\ 25)$$

解码时角度的反量化公式为：

$$\alpha' = round(\hat{\alpha} \times 1.406\ 25) \qquad \beta' = round(\hat{\beta} \times 1.406\ 25)$$

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 176。

表 176 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 5.2.3.5 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 177，其中绝对时间信息中未加说明的各语法元素定义同表 26。

表 177 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)	

表 177（续）

audio_time_extension( ){	描述符	说明
month_minus1_bits	u(4)	
date_minus1_bits	u(5)	
}		
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。

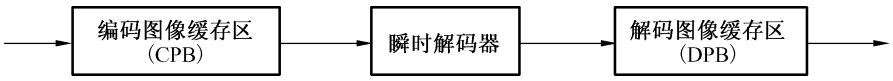


图 A.1 假设参考解码器（HRD）

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，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。

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，

$t_{ai}(n)=\mathrm{Max}(t_{af}(n-1),t_{ai,earliest}(n))$

$$t_{ai,earliest}(n)=t_c\times\sum_{m=0}^{n-1}\mathrm{cpb\_removal\_delay}[m]$$

其中cpb_removal_delay(n)见 D.2.2 定义。

$t_{af}(n)$的计算如下：

$t_{af}(0)=b(0)\div\mathrm{BitRate}[\mathrm{SchedSelIdx}]$

$t_{af}(n)=t_{ai}(n)+b(n)\div\mathrm{BitRate}[\mathrm{SchedSelIdx}]$

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，

$t_{r,n}(0)=\mathrm{initial\_cpb\_removal\_delay}[\mathrm{SchedSelIdx}]\div 90\,000$

对于图像 n，

$t_{r,n}(n)=t_{r,n}(n-1)+t_c\times\mathrm{cpb\_removal\_delay}(n)$

$t_{r,n}(n-1)$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 $t_{r,n}(n)<t_{af}(n)$)，

$t_r(n)=t_{r,n}(n)+t_c\times\mathrm{Ceil}((t_{af}(n)-t_{r,n}(n))\div t_c)$

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：

$t_{o,dpb}(n)=t_r(n)+t_c\times\mathrm{dpb\_output\_delay}(n)$

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；

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 $t_{o,dpb}(m)<=tr(n)$。

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，且根据序列参数集得到的 FrameWidth，FrameHeight，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。如果需要存储，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 语法元素只可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均应支持。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	基准档次
0x22	保留
0x33	高级档次

C.2.2 基准档次

比特流若与基准档次相一致，应遵循如下限制：

——profile_id 的值应为 0x11；

——NAL 单元流中不应包含 nal_unit_type 的取值为 3、4 和 15 的 NAL 单元；

——序列参数集中的参数 ldp_mode_flag 的取值应为 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 的取值应为 0；

——序列参数集中的参数 refs_per_frame 的取值应不大于 3；
——序列参数集中的参数 extended_sb_size_flag 的取值应为 0；
——序列参数集中的参数 alf_enable 的取值应为 0；
——序列参数集中的参数 spatial_svc_flag 的取值应为 0；
——解码所需的参考帧缓冲区个数应不大于 4。

profile_id 的取值为 0x11 时，比特流与基准档次相一致。基准档次的比特流支持的级别包括 6.0，7.0 和 8.0。

C.2.3 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；
——序列参数集中的参数 ldp_mode_flag 的取值应为 0 或 1；
——序列参数集中的参数 chroma_format_idc 的取值应为 1 或 2；
——序列参数集中的参数 bit_depth 的取值应在 0～2 之间，包括 0 和 2；
——序列参数集中的参数 refs_per_frame 的取值应不大于 5；
——序列参数集中的参数 extended_sb_size_flag 的取值应为 0 或 1；
——序列参数集中的参数 alf_enable 的取值应为 0 或 1；
——序列参数集中的参数 spatial_svc_flag 的取值应为 0 或 1；
——解码所需的参考帧缓冲区个数应不大于 8。

profile_id 的取值为 0x33 时，比特流与高级档次相一致。高级档次支持的级别有 6.0，6.2，7.0，7.2，8.0 和 8.2。

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
0x40	6.0
0x42	6.2
0x50	7.0
0x52	7.2
0x60	8.0
0x62	8.2
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.5。

表 C.3　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)	15 000 000	25 000 000
图像格式	4∶2∶0	4∶2∶0 或 4∶2∶2
DPB 最大容量 MaxDPB(kB)	12 240	24 480
CPB 最大容量 MaxCPB(kB)	20 000	40 000

表 C.4　级别的参数限制

参数	级别	
	7.0	7.2
每行最大样点数	2 592	2 592
每帧最大行数	1 944	1 944
每秒最大帧数	30	30
亮度样点速率	151 165 440	151 165 440
最大比特率 MaxBR(bps)	30 000 000	50 000 000
图像格式	4∶2∶0	4∶2∶0 或 4∶2∶2
DPB 最大容量 MaxDPB(kB)	29 525	59 049
CPB 最大容量 MaxCPB(kB)	40 000	80 000

表 C.5　级别的参数限制

参数	级别	
	8.0	8.2
每行最大样点数	4 096	4 096
每帧最大行数	2 304	2 304
每秒最大帧数	30	30
亮度样点速率	283 115 520	283 115 520
最大比特率 MaxBR(bps)	50 000 000	80 000 000
图像格式	4∶2∶0	4∶2∶0 或 4∶2∶2
DPB 最大容量 MaxDPB(kB)	56 624	113 248
CPB 最大容量 MaxCPB(kB)	80 000	160 000

注：表 C.3～表 C.5 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)
pb_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 27MHz 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 if(PayloadType == 2)	
spherical_pano_video_parameter_set(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
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.1.4 全景视频参数 SEI 消息语法

全景视频参数 SEI 消息语法见表 E.4。

表 E.4 全景视频参数 SEI 消息语法表

spherical_pano_video_parameter_set(PayloadSize){	描述符
spherical_pano_flag	u(1)
if(spherical_pano_flag) {	
projection_type	u(2)
if(projection_type == 0){	
arrangement_raster_mode	u(10)
arrangement_matrix_mode	u(2)
}	
fullpano_width_inpixel	u(16)
fullpano_height_inpixel	u(16)
croppedarea_left_inpixel	u(16)
croppedarea_top_inpixel	u(16)
stereo_mode	u(2)
horizontal_angle_range	se(v)
pitch_angle_up_limit	se(v)
pitch_angle_down_limit	se(v)
initial_viewazimuth_angle	se(v)
initial_view_pitch_angle	se(v)
initial_view_roll_angle	se(v)
display_width_inpixels	u(8)
display_height_inpixels	u(8)

表 E.4（续）

spherical_pano_video_parameter_set(PayloadSize){	描述符
initial_horizonal_view_angle	u(8)
horizontal_view_angle_max	u(8)
horizontal_view_angle_min	u(8)
name_len	u(8)
for(i=0;i<name_len;i++)	
stitching_software_name[i]	u(8)
camera_count	ue(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。

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，即从图像的第一个比特到达 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，本标准规定以 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。

E.2.3 全景视频参数 SEI 消息定义

spherical_pano_flag 球面全景视频标识位，0—其他，1—球面全景视频。

projection_type 投影方式标识位，0—立方体，1—圆柱，2—棱锥。

arrangement_raster_mode 当 projection_type = 0，即投影方式为立方体时，6 个正方形图像（前、后、左、右、上和下，索引值分别为：1、2、3、4、5 和 6）按照光栅排列的方式。取值范围：0～719，共 720 种排列方式。

arrangement_matrix_mode 当 projection_type = 0，即投影方式为立方体时，6 个正方形图像的排列方式。取值范围：0～3，共 4 种数组类型：0—1x6，1—6x1，2—2x3，3—3x2。

fullpano_width_inpixel 指示一帧全部图像（单目二维视频：包含一幅图像；双目三维视频：包含两幅图像）的宽度，单位为样点个数。

fullpano_height_inpixel 指示一帧全部图像（单目二维视频：包含一幅图像；双目三维视频：包含两

幅图像)的高度,单位为样点个数。

croppedarea_left_inpixel 指示一幅全景图像左侧被裁掉的列数,单位为样点个数。

croppedarea_top_inpixel 指示一幅全景图像顶部被裁掉的行数,单位为样点个数。

stereo_mode 立体影像模式及画面布局标识位,0—单目二维视频,1—双目三维左右排列视频,2—双目三维上下排列视频。

croppedarea_width_inpixel 指示一幅全景图像的宽度,单位为样点个数。

croppedarea_height_inpixel 指示一幅全景图像的高度,单位为样点个数。

croppedarea_width_inpixel 和 croppedarea_height_inpixel 的取值根据 stereo_mode 的值按照表 E.5 进行设置。

表 E.5 全景图像宽度与高度在不同 stereo_mode 取值时的计算方法

语法元素	单目二维视频 (stereo_mode = 0)	双目三维左右排列视频 (stereo_mode = 1)	双目三维上下排列视频 (stereo_mode = 2)
croppedarea_width_inpixel	fullpano_width_inpixel- croppedarea_left_inpixel	fullpano_width_inpixel/2- croppedarea_left_inpixel	fullpano_width_inpixel- croppedarea_left_inpixel
croppedarea_height_inpixel	fullpano_height_inpixel- croppedarea_top_inpixel	fullpano_height_inpixel- croppedarea_top_inpixel	fullpano_height_inpixel/2- croppedarea_top_inpixel

horizontal_angle_range 全景图像水平方向的环绕度数,以度为单位,取值范围是 0～360。

pitch_angle_up_limit 全景图像俯仰角的上限,其值必须大于等于 pitch_angle_down_limit,以度为单位,取值范围是－90～90。

pitch_angle_down_limit 全景图像俯仰角的下限,其值必须小于等于 pitch_angle_up_limit,以度为单位,取值范围是－90～90。

initial_viewazimuth_angle 初始显示画面中心的水平方位角度,取值范围是－ horizontal_angle_range /2～ horizontal_angle_range /2。当取值为 0 时,如果 projection_type = 0,则初始显示画面中心的水平方位角度设置为裁剪之后图像的中心列所对应的水平方位角度;如果 projection_type = 1,则初始显示画面中心的水平方位角度设置为裁剪之后向前图的中心列所对应的水平方位角度。

initial_view_pitch_angle 初始显示画面中心的俯仰角,以度为单位,取值范围是 pitch_angle_down_limit ～ pitch_angle_up_limit。

initial_view_roll_angle 绕显示画面中心旋转的角度,以度为单位,取值范围是－180～180。

display_width_inpixels 指示显示画面的图像宽度,单位为样点个数。

display_height_inpixels 指示显示画面的图像高度,单位为样点个数。

initial_horizonal_view_angle 指示画面显示的初始水平视角,以度为单位。

horizontal_view_angle_max 指示画面显示的最大水平视角,以度为单位,其值应小于或等于 180。

horizontal_view_angle_min 指示画面显示的最小水平视角,以度为单位,其值应大于 0。

name_len 指示拼接软件名称的长度。

stitching_software_name 表示拼接软件的名称。

camera_count 指示拍摄全景图像的摄像机个数。

附 录 F
（规范性附录）
智能分析数据描述

F.1 智能分析数据语法

F.1.1 图像分析规则

图像分析规则语法见表 F.1。

表 F.1 图像分析规则语法表

analysis_rule() {	描述符
object_min_width_minus1	u(16)
object_min_height_minus1	u(16)
min_dura_time	u(16)
max_dura_time	u(32)
line_num	u(8)
trigger_direction	u(2)
invade_action_type	u(4)
face_similarity	u(8)
density_unit	u(2)
}	

F.1.2 运动目标检测

运动目标检测语法见表 F.2。

表 F.2 运动目标检测语法表

Moving_object_detection() {	描述符
object_num	u(8)
analysis _level	u(1)
for(i=0;i< object_num;i++){	
object_id[i]	u(16)
object_width_minus1 [i]	u(16)
object_height_minus1 [i]	u(16)
position_top_left_x [i]	u(16)
position_top_left_y [i]	u(16)
object_color[i]	u(8)

表 F.2（续）

Moving_object_detection() {	描述符
object_sort[i]	u(3)
if(object_sort==0x03){	
vehicle_sort[i]	u(3)
vehicle_info_id[i]	u(8)
}	
if(analysis _level==0x01){	
object_speed_val[i]	u(16)
object_speed_rad[i]	u(9)
object_traipnt_x[i]	u(16)
object_traipnt_y[i]	u(16)
}	
}	
while(byte_aligned() == FALSE){	
reserved_bit	u(1)
}	
}	

F.1.3 人员属性分析

人员属性分析语法见表 F.3。

表 F.3 人员属性分析语法表

Human_property_analysis() {	描述符
human_num	u(8)
for(i=0;i< human_num;i++){	
human_id[i]	u(16)
human_property_num[i]	u(8)
for(j=0;j< property_num;i++)	
human_property[i,j]	u(8)
}	
}	

F.1.4 机动车特征分析

机动车特征分析语法见表 F.4。

表 F.4 机动车特征分析语法表

Vehicle_property_analysis() {	描述符
vehicle_num	u(8)
for(i=0;i< vehicle_num;i++){	
vehicle_id[i]	u(16)
vehicle_property_num[i]	u(8)
for(j=0;j< property_num;i++)	
vehicle_property[i,j]	u(8)
}	
}	

F.1.5 人脸比对

人脸比对语法见表 F.5。

表 F.5 人脸比对语法表

face_match() {	描述符
face_num	u(8)
for(i=0;i< face_num;i++){	
face_id[i]	u(16)
face_similarity[i]	u(8)
}	
}	

F.1.6 车牌识别

车牌识别语法见表 F.6。

表 F.6 车牌识别语法表

vehicle_licence_recognition() {	描述符
vehicleLicence_num	u(8)
analysis _level	u(1)
for(i=0;i< vehicleLicence_num;i++){	
vehicle_licence_type[i]	u(4)
vehicle_licence_color[i]	u(2)
vehicle_licence_no[i]	u(64)
if(analysis _level==0x01){	
vehicle_licence_modify_flag[i]	u(1)

表 F.6（续）

vehicle_licence_recognition() {	描述符
vehicle_licence_cover_flag[i]	u(1)
}	
}	
while(byte_aligned() == FALSE)	
reserved_bit	u(1)
}	

F.1.7 绊线检测

绊线检测语法见表 F.7。

表 F.7 绊线检测语法表

pass_extension() {	描述符
pass_num	u(8)
for(i=0; i<pass_num; i++) {	
object_category	u(2)
object_size	u(2)
moving_direction	u(4)
position_top_left_x[i]	u(16)
position_top_left_y[i]	u(16)
position_width_minus1[i]	u(16)
position_height_minus1[i]	u(16)
}	
}	

F.1.8 入侵检测

入侵检测语法见表 F.8。

表 F.8 入侵检测语法表

invade_extension() {	描述符
invade_num	u(8)
for(i=0; i<invade_num; i++) {	
object_category	u(2)
object_size	u(2)
moving_direction	u(4)

表 F.8（续）

invade_extension() {	描述符
position_top_left_x[i]	u(16)
position_top_left_y[i]	u(16)
position_width_minus1[i]	u(16)
position_height_minus1[i]	u(16)
}	
}	

F.1.9 逆行检测

逆行检测语法见表 F.9。

表 F.9 逆行检测语法表

retrograde_extension() {	描述符
retrograde_num	u(8)
for(i=0; i< retrograde_num; i++) {	
object_category	u(2)
object_size	u(2)
moving_direction	u(4)
position_top_left_x[i]	u(16)
position_top_left_y[i]	u(16)
position_width_minus1[i]	u(16)
position_height_minus1[i]	u(16)
}	
}	

F.1.10 徘徊检测

徘徊检测语法见表 F.10。

表 F.10 徘徊检测语法表

hover_extension() {	描述符
hover_num	u(8)
for(i=0; i< hover_num; i++) {	
object_category	u(2)
object_size	u(2)
moving_direction	u(4)

表 F.10（续）

hover_extension() {	描述符
position_top_left_x[i]	u(16)
position_top_left_y[i]	u(16)
position_width_minus1[i]	u(16)
position_height_minus1[i]	u(16)
}	
}	

F.1.11 遗留物检测

遗留物检测语法见表 F.11。

表 F.11 遗留物检测语法表

remnant_extension() {	描述符
remnant_num	u(8)
for(i=0; i< remnant_num; i++) {	
object_category	u(2)
object_size	u(2)
object_color	u(4)
position_top_left_x[i]	u(16)
position_top_left_y[i]	u(16)
position_width_minus1[i]	u(16)
position_height_minus1[i]	u(16)
}	
}	

F.1.12 目标移除检测

目标移除物检测语法见表 F.12。

表 F.12 目标移除物检测语法表

moveout_extension() {	描述符
moveout_num	u(8)
for(i=0; i< moveout_num; i++) {	
object_category	u(2)
object_size	u(2)
object_color	u(4)

表 F.12（续）

moveout_extension() {	描述符
position_top_left_x[i]	u(16)
position_top_left_y[i]	u(16)
position_width_minus1[i]	u(16)
position_height_minus1[i]	u(16)
}	
}	

F.1.13 目标数量统计

目标数量统计语法见表 F.13。

表 F.13 目标数量统计语法表

object_statistics() {	描述符
begin_time	u(32)
end_time	u(32)
object_num	u(16)
person_num	u(16)
face_num	u(16)
vehicle_num	u(16)
thing_num	u(16)
object_density_abs	u(2)
person_density_abs	u(2)
face_density_abs	u(2)
vehicle_density_abs	u(2)
thing_density_abs	u(2)
object_density_rlt	u(8)
person_density_rlt	u(8)
face_density_rlt	u(8)
vehicle_density_rlt	u(8)
thing_density_rlt	u(8)
total_object_flowrate	u(16)
person_object_flowrate	u(16)
vehicle_object_flowrate	u(16)
reserved_bits	u(6)
}	

F.2 智能分析信息扩展语义

F.2.1 图像分析规则

object_min_width_minus1 加 1 和 object_min_height_minus1 加 1 等于目标的最小宽度和最小高度。

min_dura_time 表示最小持续时间，单位为秒。

max_dura_time 表示最大持续时间，单位为秒。

line_num 表示包含绊线的条数。

trigger_direction 表示触发方向，0 为从左到右，1 为从右到左，2 位任意方向。

invade_action_type 表示入侵行为的类型，如表 F.14 所示。

表 F.14 invade_action_type 的取值说明

invade_action_type 取值	含义
0	进入区域
1	离开区域
2	区域内出现
3	区域内消失
4	在区域内
5～15	自定义

face_similarity 为人脸相似度，取值为百分比，取值不带百分号，大于该值认为是人脸。density_unit 为密度检测数值单位，0 为密度等级，1 为密度百分比，2 为个数，其余数值保留。

F.2.2 运动目标检测

object_num 为 8 位无符号整数，表示识别出的目标数量。

analysis _level 为 1 位无符号整数，表示分析级别。analysis _level 等于 0 表示基本级别，analysis _level 等于 1 表示高级。

object_id[i]为 16 位无符号整数，表示第 i 个目标的编号。

object_width_minus1[i]加 1 和 **object _height_minus1**[i]加 1 分别等于第 i 个目标的宽度和高度。以样点为单位计算的第 i 个目标的宽度、高度为：

以样点为单位计算的第 i 个目标的宽度、高度为：

objectWidthInSample[i] = object_width_minus1[i] + 1

objectHeightInSample[i] = object_height_minus1[i] + 1

position_top_left_x[i]和 **position_top_left_y[i]** 分别表示第 i 个目标的左上角的横坐标值、纵坐标值。以样点为单位计算的第 i 个目标的左上角位置的横坐标值、纵坐标值为：

objectTopLeftSamplePositionX[i] = position_top_left_x[i]

objectTopLeftSamplePositionY[i] = position_top_left_y[i]

object_color[i]为 8 位无符号整数，表示第 i 个目标的主体颜色编号，见表 F.15。

表 F.15　object_color 的取值说明

object_color 取值	含义
01	黑
02	白
03	灰
04	红
05	蓝
06	黄
07	橙
08	棕
09	绿
10	紫
11	青
12	粉
13	透明
…	…
99	其他

object_sort[**i**]为 3 位无符号整数，表示第 i 个目标的类别，见表 F.16。

表 F.16　object_sort 的取值说明

object_sort 取值	含义
0x01	人员
0x02	人脸
0x03	机动车
0x04	非机动车
0x05	物品
0x06	场景

vehicle_sort[**i**]为 3 位无符号整数，表示第 i 个目标如果为车辆时车辆分类信息，见表 F.17。

表 F.17　vehicle_sort 的取值说明

vehicle_sort 取值	含义
0x01	客车
0x02	货车
0x03	其他车辆

vehicle_info_id[i]为 11 位无符号整数，表示第 i 个目标如果为车辆时车辆详细信息编号。

object_speed_val[i]为 16 位无符号整数，表示第 i 个目标的运动速度，单位为样点数/秒。

object_speed_rad[i]为 9 位无符号整数，表示第 i 个目标运动方向，以角度为单位，取值范围[0，359]，水平向右为 0，逆时针转动时角度增加。

object_traipnt_x[i]为 16 位无符号整数，表示第 i 个目标的运动轨迹点 X 坐标，单位为样点值。

object_traipnt_y[i]为 16 位无符号整数，表示第 i 个目标的轨迹点 Y 坐标，单位为样点值。

object_traipnt_x[i]和 **object_traipnt_y**[i]分别表示第 i 个目标的运动轨迹点的横坐标值、纵坐标值。以样点为单位计算的第 i 个目标的运动轨迹点位置的横坐标值、纵坐标值为：

objectTraipntSamplePositionX[i] = object_traipnt_x[i]

objectTraipntSamplePositionY[i] = object_traipnt_y[i]

reserved_bit 应等于 0。

F.2.3 人员属性分析

human_num 为 8 位无符号整数，表示识别出的人员数量。

human_id[i]为 16 位无符号整数，表示识别出的第 i 个人员编号。

human_property_num[i]为 8 位无符号整数，表示识别出的第 i 个人员的属性数量。

human_Property[i,j]为 8 位无符号整数，表示识别出的第 i 个人员的第 j 个属性。

F.2.4 机动车特征分析

vehicle_num 为 8 位无符号整数，表示识别出的车辆数量。

vehicle_id[i]为 16 位无符号整数，表示识别出的车辆编号。

vehicle_property_num[i]为 8 位无符号整数，表示识别出的第 i 个车辆的属性数量。

vehicle_property[i,j]为 8 位无符号整数，表示识别出的第 i 个车辆的第 j 个属性。

F.2.5 人脸比对

face_num 为 8 位无符号整数，表示识别出的符合比对条件的人脸数量。

face_id[i]为 16 位无符号整数，表示识别出的第 i 个人脸编号。

face_similarity[i]为 8 位无符号整数，表示人脸相似度。

F.2.6 车牌识别

vehicle_licence_num 为 8 位无符号整数，表示识别出的车牌数量。

vehicle_licence_type[i]为 4 位无符号整数，表示识别出的第 i 个车牌种类，见表 F.18。

表 F.18 vehicle_licence_type 的取值说明

vehicle_licence_type 取值	含义
0x00	大型汽车号牌
0x01	小型汽车号牌
0x02	使、领馆汽车号牌
0x03	港澳入出境车号牌
0x04	教练汽车号牌
0x05	警用汽车号牌
0x06	武警汽车号牌
0x07	军用汽车号牌
reserved	其他

vehicle_licence_color[i]为 2 位无符号整数，表示识别出的第 i 个车牌颜色，见表 F.19。

表 F.19　vehicle_licence_color 的取值说明

vehicle_licence_color 取值	含义
0x00	蓝色
0x01	白色
0x02	黄色
0x03	黑色

vehicle_licence_no[i]为 8 位字符串，表示识别出的第 i 个车牌编号。

vehicle_licence_modify_flag[i]为 1 位无符号整数，等于 1 表示车牌有涂改；等于 0 表示车牌无涂改。

vehicle_licence_cover_flag[i]为 1 位无符号整数，等于 1 表示车牌有遮挡；等于 0 表示车牌无遮挡。

reserved_bit 应等于 0。

F.2.7　绊线检测

pass_num 指示通过警戒线目标的个数。

object_category 指示通过警戒线目标的类别，object_category 等于 0 表示人，object_category 等于 1 表示车，object_category 等于 2 表示其他物体。

object_size 指示通过警戒线目标的尺寸，object_size 等于 0 表示小尺寸，object_size 等于 1 表示中等尺寸，object_size 等于 2 表示大尺寸，object_size 等于 3 表示巨大尺寸。

moving_direction 指示通过警戒线目标的运动方向，moving_direction 等于 0 表示北，moving_direction 等于 1 表示东北，moving_direction 等于 2 表示东，moving_direction 等于 3 表示东南，moving_direction 等于 4 表示南，moving_direction 等于 5 表示西南，moving_direction 等于 6 表示西，moving_direction 等于 7 表示西北，moving_direction 等于 8 表示上，moving_direction 等于 9 表示右上，moving_direction 等于 10 表示右下，moving_direction 等于 11 表示下，moving_direction 等于 12 表示左下，moving_direction 等于 13 表示左上，moving_direction 等于 14 表示左，moving_direction 等于 15 表示右。

position_top_left_x[i]和 **position_top_left_y**[i] 分别表示第 i 个通过警戒线目标的左上角的横坐标值、纵坐标值。以样点为单位计算的第 i 个通过警戒线目标的左上角位置的横坐标值、纵坐标值为：

passTopLeftSamplePositionX[i] = position_top_left_x[i]

passTopLeftSamplePositionY[i] = position_top_left_y[i]

position_width_minus1[i]加 1 和 **position_height_minus1**[i]加 1 分别等于第 i 个通过警戒线目标的宽度和高度。以样点为单位计算的第 i 个通过警戒线目标的宽度、高度为：

passWidthInSample[i] = position_width_minus1[i] + 1

passHeightInSample[i] = position_height_minus1[i] + 1

F.2.8　入侵检测

invade_num 指示进入禁入区域目标的个数。

object_category 指示进入禁入区域目标的类别，object_category 等于 0 表示人，object_category 等于 1 表示车，object_category 等于 2 表示其他物体。

object_size 指示进入禁入区域目标的尺寸，object_size 等于 0 表示小尺寸，object_size 等于 1 表示

中等尺寸，object_size 等于 2 表示大尺寸，object_size 等于 3 表示巨大尺寸。

moving_direction 指示进入禁入区域目标的运动方向，moving_direction 等于 0 表示北，moving_direction 等于 1 表示东北，moving_direction 等于 2 表示东，moving_direction 等于 3 表示东南，moving_direction 等于 4 表示南，moving_direction 等于 5 表示西南，moving_direction 等于 6 表示西，moving_direction 等于 7 表示西北，moving_direction 等于 8 表示上，moving_direction 等于 9 表示右上，moving_direction 等于 10 表示右下，moving_direction 等于 11 表示下，moving_direction 等于 12 表示左下，moving_direction 等于 13 表示左上，moving_direction 等于 14 表示左，moving_direction 等于 15 表示右。

position_top_left_x[i]和 **position_top_left_y[i]** 分别表示第 i 个进入禁入区域目标的左上角的横坐标值、纵坐标值。以样点为单位计算的第 i 个进入禁入区域目标的左上角位置的横坐标值、纵坐标值为：

passTopLeftSamplePositionX[i] = position_top_left_x[i]

passTopLeftSamplePositionY[i] = position_top_left_y[i]

position_width_minus1[i]加 1 和 **position_height_minus1[i]**加 1 分别等于第 i 个进入禁入区域目标的宽度和高度。以样点为单位计算的第 i 个进入禁入区域目标的宽度、高度为：

passWidthInSample[i] = position_width_minus1[i] + 1

passHeightInSample[i] = position_height_minus1[i] + 1

F.2.9 逆行检测

retrograde_num 指示向反方向运动目标的个数。

object_category 指示向反方向运动目标的类别，object_category 等于 0 表示人，object_category 等于 1 表示车，object_category 等于 2 表示其他物体。

object_size 指示向反方向运动目标的尺寸，object_size 等于 0 表示小尺寸，object_size 等于 1 表示中等尺寸，object_size 等于 2 表示大尺寸，object_size 等于 3 表示巨大尺寸。

moving_direction 指示向反方向运动目标的运动方向，moving_direction 等于 0 表示北，moving_direction 等于 1 表示东北，moving_direction 等于 2 表示东，moving_direction 等于 3 表示东南，moving_direction 等于 4 表示南，moving_direction 等于 5 表示西南，moving_direction 等于 6 表示西，moving_direction 等于 7 表示西北，moving_direction 等于 8 表示上，moving_direction 等于 9 表示右上，moving_direction 等于 10 表示右下，moving_direction 等于 11 表示下，moving_direction 等于 12 表示左下，moving_direction 等于 13 表示左上，moving_direction 等于 14 表示左，moving_direction 等于 15 表示右。

position_top_left_x[i]和 **position_top_left_y**[i] 分别表示第 i 个向反方向运动目标的左上角的横坐标值、纵坐标值。以样点为单位计算的第 i 个向反方向运动目标的左上角位置的横坐标值、纵坐标值为：

passTopLeftSamplePositionX[i] = position_top_left_x[i]

passTopLeftSamplePositionY[i] = position_top_left_y[i]

position_width_minus1[i]加 1 和 **position_height_minus1**[i]加 1 分别等于第 i 个向反方向运动目标的宽度和高度。以样点为单位计算的第 i 个向反方向运动目标的宽度、高度为：

passWidthInSample[i] = position_width_minus1[i] + 1

passHeightInSample[i] = position_height_minus1[i] + 1

F.2.10 徘徊检测

hover_num 指示在警戒区域徘徊运动目标的个数。

object_category 指示在警戒区域徘徊运动目标的类别，object_category 等于 0 表示人，object_category 等于 1 表示车，object_category 等于 2 表示其他物体。

object_size 指示在警戒区域徘徊运动目标的尺寸，object_size 等于 0 表示小尺寸，object_size 等于 1 表示中等尺寸，object_size 等于 2 表示大尺寸，object_size 等于 3 表示巨大尺寸。

moving_direction 指示在警戒区域徘徊运动目标的运动方向，moving_direction 等于 0 表示北，moving_direction 等于 1 表示东北，moving_direction 等于 2 表示东，moving_direction 等于 3 表示东南，moving_direction 等于 4 表示南，moving_direction 等于 5 表示西南，moving_direction 等于 6 表示西，moving_direction 等于 7 表示西北，moving_direction 等于 8 表示上，moving_direction 等于 9 表示右上，moving_direction 等于 10 表示右下，moving_direction 等于 11 表示下，moving_direction 等于 12 表示左下，moving_direction 等于 13 表示左上，moving_direction 等于 14 表示左，moving_direction 等于 15 表示右。

position_top_left_x[i]和 **position_top_left_y**[i] 分别表示第 i 个在警戒区域徘徊运动目标的左上角的横坐标值、纵坐标值。以样点为单位计算的第 i 个在警戒区域徘徊运动目标的左上角位置的横坐标值、纵坐标值为：

passTopLeftSamplePositionX[i] = position_top_left_x[i]

passTopLeftSamplePositionY[i] = position_top_left_y[i]

position_width_minus1[i]加 1 和 **position_height_minus1**[i]加 1 分别等于第 i 个在警戒区域徘徊运动目标的宽度和高度。以样点为单位计算的第 i 个在警戒区域徘徊运动目标的宽度、高度为：

passWidthInSample[i] = position_width_minus1[i] + 1

passHeightInSample[i] = position_height_minus1[i] + 1

F.2.11 遗留物检测

remnant_num 指示警戒区域内遗留物的个数。

object_category 指示警戒区域内遗留物的类别，object_category 等于 0 表示人，object_category 等于 1 表示车，object_category 等于 2 表示其他物体。

object_size 指示警戒区域内遗留物的尺寸，object_size 等于 0 表示小尺寸，object_size 等于 1 表示中等尺寸，object_size 等于 2 表示大尺寸，object_size 等于 3 表示巨大尺寸。

object_color 指示警戒区域内遗留物的颜色，见表 F.20。

表 F.20 object_color 的取值说明

object_color 取值	含义
01	黑
02	白
03	灰
04	红
05	蓝
06	黄
07	橙
08	棕
09	绿
10	紫

表 F.20（续）

object_color 取值	含义
11	青
12	粉
13	透明
…	…
15	其他

position_top_left_x[i]和 **position_top_left_y**[i] 分别表示第 i 个警戒区域内遗留物的左上角的横坐标值、纵坐标值。以样点为单位计算的第 i 个警戒区域内遗留物的左上角位置的横坐标值、纵坐标值为：

passTopLeftSamplePositionX[i] = position_top_left_x[i]

passTopLeftSamplePositionY[i] = position_top_left_y[i]

position_width_minus1[i]加 1 和 **position_height_minus1**[i]加 1 分别等于第 i 个警戒区域内遗留物的宽度和高度。以样点为单位计算的第 i 个警戒区域内遗留物的宽度、高度为：

passWidthInSample[i] = position_width_minus1[i] + 1

passHeightInSample[i] = position_height_minus1[i] + 1

F.2.12 目标移除检测

moveout_num 指示警戒区域内移除目标的个数。

object_category 指示警戒区域内移除目标的类别，object_category 等于 0 表示人，object_category 等于 1 表示车，object_category 等于 2 表示其他物体。

object_size 指示警戒区域内移除目标的尺寸，object_size 等于 0 表示小尺寸，object_size 等于 1 表示中等尺寸，object_size 等于 2 表示大尺寸，object_size 等于 3 表示巨大尺寸。

object_color 指示警戒区域内移除目标的颜色，见表 F.21。

表 F.21 object_color 的取值说明

object_color 取值	含义
01	黑
02	白
03	灰
04	红
05	蓝
06	黄
07	橙
08	棕
09	绿
10	紫
11	青

表 F.21（续）

object_color 取值	含义
12	粉
13	透明
…	…
15	其他

position_top_left_x[i]和 **position_top_left_y**[i] 分别表示第 i 个警戒区域内移除目标的左上角的横坐标值、纵坐标值。以样点为单位计算的第 i 个警戒区域内移除目标的左上角位置的横坐标值、纵坐标值为：

passTopLeftSamplePositionX[i] = position_top_left_x[i]

passTopLeftSamplePositionY[i] = position_top_left_y[i]

position_width_minus1[i]加 1 和 **position_height_minus1**[i]加 1 分别等于第 i 个警戒区域内移除目标的宽度和高度。以样点为单位计算的第 i 个警戒区域内移除目标的宽度、高度为：

passWidthInSample[i] = position_width_minus1[i] + 1

passHeightInSample[i] = position_height_minus1[i] + 1

F.2.13 目标数量统计

begin_time 表示统计开始时间，对连续视频有效，取值等于从公元 1970 年 1 月 1 日 0 时整至该值所表示的实际时间的秒数。

end_time 表示统计结束开始时间，对连续视频有效，取值等于从公元 1970 年 1 月 1 日 0 时整至该值所表示的实际时间的秒数。

object_num 表示目标总数。

person_num 表示目标为人员的总数。

face_num 表示目标为人脸的总数。

vehicle_num 表示目标为车辆的总数。

thing_num 表示目标为物体的总数。

object_density_abs 表示区域内目标密度等级，取值如表 F.22 所示。

person_density_abs 表示区域内人员密度等级，取值如表 F.22 所示。

face_density_abs 表示区域内人脸密度等级，取值如表 F.22 所示。

vehicle_density_abs 表示区域内车辆密度等级，取值如表 F.22 所示。

thing_density_abs 表示区域内物体密度等级，取值如表 F.22 所示。

表 F.22 目标密度取值与含义对应关系表

取值	含义
0	很稀疏
1	稀疏
2	密集
3	很密集

object_density_rlt 表示区域内目标相对密度，取值为百分比，不含百分号。

person_density_rlt 表示区域内人员相对密度，取值为百分比，不含百分号。
face_density_rlt 表示区域内人脸相对密度，取值为百分比，不含百分号。
vehicle_density_rlt 表示区域内车辆相对密度，取值为百分比，不含百分号。
thing_density_rlt 表示区域内物体相对密度，取值为百分比，不含百分号。
total_object_flowrate 表示时间段内的目标总个数。
person_object_flowrate 表示时间段内的人员总个数。
vehicle_object_flowrate 表示时间段内的人员总个数。
reserved_bits 应等于 0。

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标准。

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：

a) 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：

```
if(frame<15&&Acceleration<2.5)
  tracker= MAX(tracker,input)
if(input<tracker×UpperBound&&input>tracker×LowerBound)
  tracker=a×tracker+(1−a) ×input
if(input<tracker×Floor)
  tracker=b×tracker+(1−b)×input
if(input>tracker×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。

b) 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：

$input=p\times currentInput+(1-p)\times PreviousInput$；

$if(Frame<15)\quad tracker=MAX(tracker,input)$；

$if(input<tracker\times UpperBound\quad \&\&\quad input>tracker\times LowerBound)$

$\quad tracker=a\times tracker+(1-a)\times input$；

$if(input<tracker\times Floor)$

$\quad tracker=b\times tracker+(1-b)\times input$；

$if(input>tracker\times threshold)$

$\quad return\quad true$；

$else$

$\quad return\quad false$；

式中：

$p=0.75$；

$threshold=3.25$，其他参数和方法 1 中相同。

c) 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：

$$\frac{1}{N_{SPEC}}\sum_{bin=0}^{N_{SPEC}-1}(H_2(bin))^2-\left(\sum_{bin=0}^{N_{SPEC}-1}H_2(bin)\right)^2/N_{SPEC}^2 \qquad \cdots\cdots(\text{I.1})$$

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，如图 I.1 所示。

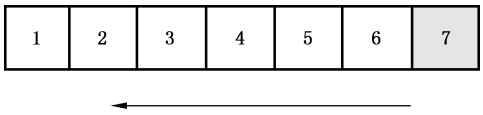


图 I.1　VAD 缓冲区示意图

VAD 决策算法处理步骤如下所示：

a)　$V_N = Measurement1 \quad or \quad Measurement2 \quad or \quad Measurement3$

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。

$$M = \mathrm{MAX}\left\{\begin{matrix} C++, & V_i = true \\ C = 0, & V_i = false \end{matrix} \quad , \quad 1 < i < N \right\}\Bigg|_C \qquad \cdots\cdots\cdots\cdots\cdots\cdots (\mathrm{I.2})$$

b)　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。

c)　$if(M \geqslant S_P \quad \&\& \quad T < L_s)$

$T = L_s$；

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）。

d)　$if(M \geqslant S_L \quad \&\& \quad F > F_s)$

$T = L_M$；

$else \quad if(M \geqslant S_L)$

$T = L_L$；

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。

e)　$if(M < S_P \quad \&\& \quad T > 0)$

$T--$；

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下才会减少。

f)　$if(T > 0)$

$return \quad true$；

$else$

$return \quad false$；

如果 T 大于 0 时，输出为 true（语音帧）；否则，输出为 false（非语音帧）。

g)　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。

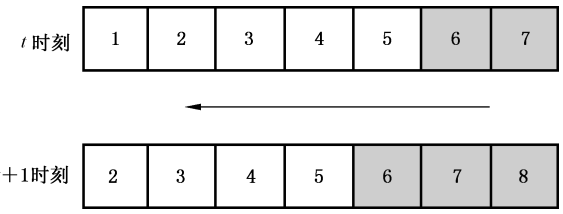


图 I.2　预测机制下缓冲区示意图

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 2～5 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。

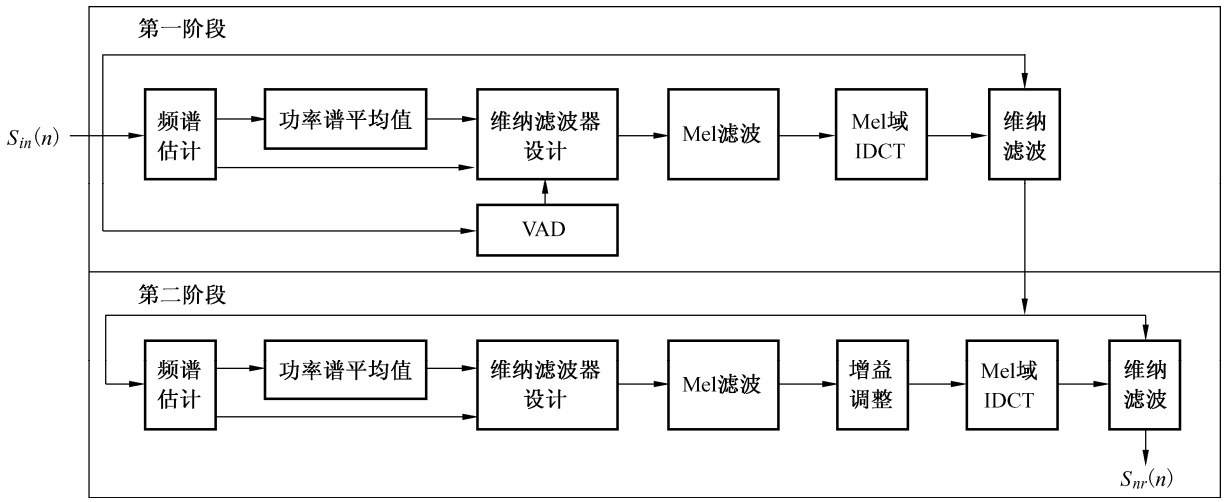


图 J.1 噪声消除流程图

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(20ms)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。

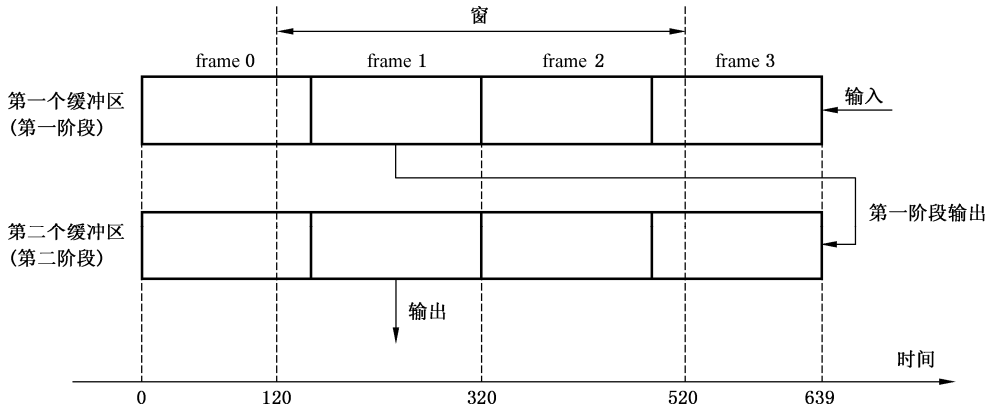


图 J.2　两阶段噪声消除过程的缓冲区示意图

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。

$$s_w(n)=s_{in}(n)\times w_{\text{Hann}}(n),\quad 0\leqslant n\leqslant N_{in}-1 \qquad \cdots\cdots(\text{J.1})$$

$$w_{\text{Hann}}(n)=0.5-0.5\times\cos\left(\frac{2\times\pi\times(n+0.5)}{N_{in}}\right) \qquad \cdots\cdots(\text{J.2})$$

N_{in} 和 $N_{\text{FFT}}-1$ 之间的样本补零，$N_{\text{FFT}}=512$ 是 FFT 长度：

$$s_{\text{FFT}}(n)=\begin{cases}s_w(n), & 0\leqslant n\leqslant N_{in}-1\\ 0, & N_{in}\leqslant n\leqslant N_{\text{FFT}}-1\end{cases} \qquad \cdots\cdots(\text{J.3})$$

对 $s_{\text{FFT}}(n)$ 进行 FFT，以求出频谱：

$$X(bin)=\text{FFT}\{s_{\text{FFT}}(n)\} \qquad \cdots\cdots(\text{J.4})$$

式中：

bin——FFT 频率索引。

计算功率谱 $P(bin)$：

$$P(bin)=|X(bin)|^2,\quad 0\leqslant bin\leqslant N_{\text{FFT}}/2 \qquad \cdots\cdots(\text{J.5})$$

再对功率谱 $P(bin)$ 进行平滑处理：

$$P_{in}(bin)=\frac{P(2\times bin)+P(2\times bin+1)}{2},\quad 0\leqslant bin\leqslant N_{\text{FFT}}/4 \qquad \cdots\cdots(\text{J.6})$$

$$P_{in}(N_{\text{FFT}}/4)=P(N_{\text{FFT}}/2)$$

平滑处理后，功率谱的长度缩短为 $N_{\text{SPEC}}=N_{\text{FFT}}/4+1$。

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。

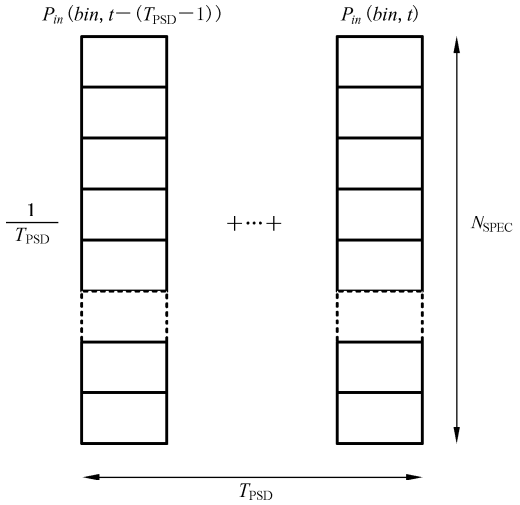


图 J.3　功率谱平均值

功率谱平均值为：

$$P_{in\_PSD}(bin,t)=\frac{1}{T_{PSD}}\sum_{i=0}^{T_{PSD}-1}P_{in}(bin,t-i),\qquad 0\leqslant bin\leqslant N_{SPEC}-1 \quad \cdots\cdots\cdots\cdots(J.7)$$

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 $lambdaLTE$：

$$\begin{aligned}&if(t<NB\_FRAME\_THRESHOLD\_LTE)\\&\quad lambdalTE=1-1/t;\\&else\\&\quad lambdaLTE=LAMBDA\_LTE;\end{aligned}\qquad \cdots\cdots\cdots\cdots\cdots(J.8)$$

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：

$$frameEn=0.5+\frac{16}{\ln 2}\times\ln\left(\frac{\left(64+\sum_{i=0}^{M-1}s_{in}(n)^2\right)}{64}\right) \quad \cdots\cdots\cdots\cdots\cdots\cdots(J.9)$$

用 $frameEn$ 更新 $meanEn$：

```
if((frameEn−meanEn)<SNR_THRESHOLD_UPD_LTE)||(t<MIN_FRAME))
{
  if((frameEn<meanEn)||(t<MIN_FRAME))
    meanEn=meanEn+(1−lambdaLTE)×(frameEn−meanEn);
  else                                                          ……(J.10)
    meanEn=meanEn+(1−lambdaLTEhigherE)×(frameEn−meanEn);
  if(meanEn<ENERGY_FLOOR)
    meanEn=ENERGY_FLOOR;
}
```

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 $lambdaNSE$：

$If(t<NB\_FRAME\_THRESHOLD\_NSE)$

$lambdaNSE=1-1/t$；

$else$

$lambdaNSE=LAMBDA\_NSE$；

其中：

$NB\_FRAME\_THRESHOLD\_NSE=100$；

$LAMBDA\_NSE=0.99$。

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：

$$\begin{cases} P_{noise}^{1/2}(bin,t_n)=\max(lambdaNSE\times P_{noise}^{1/2}(bin,t_n-1)+ \\ \qquad (1-lambdaNSE)\times P_{in\_PSD}^{1/2}(bin,t_n),EPS) \\ P_{noise}^{1/2}(bin,t)=P_{noise}^{1/2}(bin,t_n) \end{cases} \qquad \text{……(J.12)}$$

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：

$$P_{den}^{1/2}(bin,t)=BETA\times P_{den3}^{1/2}(bin,t-1)+(1-BETA)\times T[P_{in\_PSD}^{1/2}(bin,t)-P_{noise}^{1/2}(bin,t)] \qquad \text{……(J.13)}$$

式中，$P_{den}^{1/2}(bin,-1)$初始化为 0，$BETA$ 等于 0.98，阈值函数 T 为：

$$T[z(bin,t)]=\begin{cases} z(bin,t), & z(bin,t)>0 \\ 0, & z(bin,t)\leqslant 0 \end{cases} \qquad \text{……(J.14)}$$

因此，先验信噪比 $\eta(bin,t)$为：

$$\eta(bin,t)=\frac{P_{den}(bin,t)}{P_{noise}(bin,t)} \qquad \text{……(J.15)}$$

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：

$\eta_{TH}=0.079\ 432\ 823$（对应的 SNR 为 −22 dB）。

相应地，滤波器传递函数 $H_2(bin,t)$ 更新为：

$$H_2(bin,t)=\frac{\sqrt{\eta_2(bin,t)}}{1+\sqrt{\eta_2(bin,t)}},0\leqslant bin\leqslant N_{\mathrm{SPEC}}-1 \quad \cdots\cdots(J.19)$$

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 $H_2(bin)$，$0\leqslant bin\leqslant N_{\mathrm{SPEC}}-1$，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：

$$f_{mel}(k)=k\times\frac{MEL\{f_{lin\_samp}/2\}}{K_{FB}+1} \quad \cdots\cdots(J.23)$$

式中：

$f_{lin\_samp}=16$ kHz——采样频率。

两个边缘子带的中心频率 $f_{centr}(0)$ 和 $f_{centr}(K_{FB}+1)=f_{lin\_samp}/2$ 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：

$$bin_{centr}(k)=round\left(\frac{f_{centr}(k)}{f_{lin\_samp}}\times 2\times(N_{\mathrm{SPEC}}-1)\right) \quad \cdots\cdots(J.24)$$

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：

$$W(k,i)=\frac{i-bin_{centr}(k-1)}{bin_{centr}(k)-bin_{centr}(k-1)},bin_{centr}(k-1)+1\leqslant i\leqslant bin_{centr}(k) \quad \cdots\cdots(J.25)$$

$$W(k,i)=1-\frac{i-bin_{centr}(k)}{bin_{centr}(k+1)-bin_{centr}(k)},bin_{centr}(k)+1\leqslant i\leqslant bin_{centr}(k+1) \quad \cdots\cdots(J.26)$$

i 取其他值时 $W(k,i)=0$。

$k=0$ 的窗函数计算如下：

$$W(0,i)=1-\frac{i}{bin_{centr}(1)-bin_{centr}(0)},0\leqslant i\leqslant bin_{centr}(1)-bin_{centr}(0)-1 \quad \cdots\cdots(\text{J.27})$$

i 取其他值时 $W(0,i)=0$。

$k=K_{FB}+1$ 的窗函数计算如下：

$$W(K_{FB}+1,i)=\frac{i-bin_{centr}(K_{FB})}{bin_{centr}(K_{FB}+1)-bin_{centr}(K_{FB})},bin_{centr}(K_{FB})+1\leqslant i\leqslant bin_{centr}(K_{FB}+1) \quad \cdots(\text{J.28})$$

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：

$$H_{2\_mel}(k)=\frac{1}{\sum_{i=0}^{N_{\text{SPEC}}-1}W(k,i)}\sum_{i=0}^{N_{\text{SPEC}}-1}W(k,i)\times H_2(i) \quad \cdots\cdots\cdots\cdots\cdots(\text{J.29})$$

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：

$$E_{den}(t)=\sum_{bin=0}^{N_{\text{SPEC}}-1}P_{den3}^{1/2}(bin,t) \quad \cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots(\text{J.30})$$

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：

$$E_{noise}(t)=\sum_{bin=0}^{N_{\text{SPEC}}-1}P_{noise}^{1/2}(bin,t) \quad \cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots(\text{J.31})$$

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：

$$\begin{aligned}&Ratio=\frac{E_{den}(t-2)\times E_{den}(t-1)\times E_{den}(t)}{E_{noise}(t)\times E_{noise}(t)\times E_{noise}(t)};\\&if(Ratio>0.000\ 01)\\&\quad SNR_{over}(t)=20/3\times\log_{10}(Ratio);\\&else\\&\quad SNR_{over}(t)=-100/3;\end{aligned} \quad \cdots\cdots\cdots\cdots\cdots(\text{J.32})$$

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：

$$\begin{aligned}&if(((SNR_{over}(t)-SNR_{lower\_track}(t-1))<10\ ||\ (t<10))\\&\quad SNR_{low\_track(t)}=\lambda_{SNR}(t)\times SNR_{low\_track}(t-1)+(1-\lambda_{SNR}(t))\times SNR_{over}(t);\\&else\\&\quad SNR_{low\_track}(t)=SNR_{low\_track}(t-1);\end{aligned} \quad \cdots\cdots(\text{J.33})$$

式中：

SNR_{low_track}——初始化为 0；

$\lambda_{SNR}(t)$——遗忘因子，计算如下：

$$\begin{aligned}&if(t<10)\\&\quad \lambda_{SNR}(t)=1-1/t;\\&else\\&\quad if(SNR_{over}(t)<SNR_{low\_track}(t))\\&\quad\quad \lambda_{SNR}(t)=0.95;\\&\quad else\\&\quad\quad \lambda_{SNR}(t)=0.99;\end{aligned} \quad \cdots\cdots\cdots\cdots\cdots\cdots\cdots(\text{J.34})$$

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 $\alpha_{GF}(t)$：

$$H_{2\_mel\_GF}(k,t)=(1-\alpha_{GF}(t))+\alpha_{GF}(t)\times H_{2\_mel}(k,t),\quad 0\leqslant k\leqslant K_{FB}+1 \quad \cdots\cdots(J.36)$$

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：

$$h_{WF}(n)=\sum_{k=0}^{K_{FB}+1}H_{2\_mel}(k)\times IDCT_{mel}(k,n),\quad 0\leqslant n\leqslant K_{FB}+1 \quad \cdots\cdots(J.37)$$

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：

$$f_{centr}(k)=\frac{1}{\sum_{i=0}^{N_{\mathrm{SPEC}}-1}W(k,i)}\sum_{i=0}^{N_{\mathrm{SPEC}}-1}W(k,i)\times i\times\frac{f_{samp}}{2\times(N_{\mathrm{SPEC}}-1)} \quad \cdots\cdots(J.38)$$

式中：

f_{samp}——采样频率，$f_{samp}=16$ kHz；

$f_{centr}(0)=0$ kHz；

$f_{centr}(K_{FB}+1)=f_{samp}/2$。

则，$IDCT_{mel}(k,n)$ 为：

$$IDCT_{mel}(k,n)=\cos\left(\frac{2\times\pi\times n\times f_{centr}(k)}{f_{samp}}\right)\times df(k),0\leqslant k\leqslant K_{FB}+1,\quad 0\leqslant n\leqslant K_{FB}+1 \qquad \text{(J.39)}$$

式中：

$f_{centr}(k)$——Mel 子带 k 所对应的中心频率。

$df(k)$为：

$$\begin{aligned} df(k)&=\frac{f_{centr}(k+1)-f_{centr}(k-1)}{f_{samp}},1\leqslant k\leqslant K_{FB}\\ df(0)&=\frac{f_{centr}(1)-f_{centr}(0)}{f_{samp}}\\ df(K_{FB}+1)&=\frac{f_{centr}(K_{FB}+1)-f_{centr}(K_{FB})}{f_{samp}} \end{aligned} \qquad \text{(J.40)}$$

维纳滤波器的脉冲响应扩展到 $0\leqslant k\leqslant 2\times(K_{FB}+1)$：

$$h_{WF\_mirr}(n)=\begin{cases} h_{WF}(n), & 0\leqslant n\leqslant K_{FB}+1\\ h_{WF}(2\times(K_{FB}+1)+1-n), & k_{FB}+2\leqslant n\leqslant 2\times(K_{FB}+1) \end{cases} \qquad \text{(J.41)}$$

J.11 维纳滤波

根据 $h_{WF_mirr}(n)$得出因果的脉冲响应 $h_{WF\_caus}(n,t)$：

$$\begin{cases} h_{WF\_caus}(n,t)=h_{WF\_mirr}(n+K_{FB}+1,t), & n=0,\cdots,K_{FB}\\ h_{WF\_caus}(n,t)=h_{WF\_mirr}(n-K_{FB}-1,t), & n=K_{FB}+1,\cdots,2\times(K_{FB}+1) \end{cases} \qquad \text{(J.42)}$$

截断后脉冲响应 $F_{WF_trunc(n,t)}$为：

$$h_{WF\_trunc}(n,t)=h_{WF\_caus}(n+K_{FB}+1-(FL-1)/2,t),\quad n=0,\cdots,FL-1 \qquad \text{(J.43)}$$

滤波器长度 FL 等于 17。截断后脉冲响应加汉宁窗处理：

$$h_{WF\_w}(n,t)=\left\{0.5-0.5\times\cos\left(\frac{2\times\pi\times(n+0.5)}{FL}\right)\right\}\times h_{WF\_trunc}(n,t),\quad 0\leqslant n\leqslant FL-1 \qquad \text{(J.44)}$$

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}：

$$s_{nr}(n)=\sum_{i=-(FL-1)/2}^{(FL-1)/2}h_{WF\_w}(i+(FL-1)/2)\times s_{in}(n-i),\quad 0\leqslant n\leqslant M-1 \qquad \text{(J.45)}$$

式中：

$FL=17$——滤波器长度；

$M=160$——帧移样本数。

参 考 文 献

[1] 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

[2] 3GPP TS 26.290, Version 7.0.0,"Extended Adaptive Multi-Rate—Wideband (AMR-WB+) codec; Transcoding functions", Mar. 2007.

[3] 3GPP TS 26.190, Version 7.0.0, " AMR Wideband speech codec; Transcoding functions ", Jun. 2007.

[4] ETSI ES 202 050, Version 1.1.5, "Distributed Speech Recognition; Advanced Front-end Feature Extraction Algorithm; Compression Algorithm", Jan. 2007.

[5] 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.

[6] ISO/IEC IS 14496-1.Information Technology—Generic coding of audio-visual objects.Part 1: Systems. Nov.1998.

[7] ISO/IEC IS 14496-2.Information Technology—Generic coding of audio-visual objects.Part 2: Visual. Nov.1998.

[8] ISO/IEC JCT1/SC29 WG11 N3342. Overview of MPEG-7 standard. Maui,1999.

[9] 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.

[10] 姚天任, 孙洪. 现代数字信号处理[M]. 武汉: 华中理工大学出版社,1999.

[11] A.V.奥本海姆, R.W. 谢弗. 离散时间信号处理[M]. 2版. 刘树棠, 黄建国, 译. 西安: 西安交通大学出版社,2001.

[12] 鲍长春. 低比特率数字语音通信编码基础[M]. 北京: 北京工业大学出版社,2001.

[13] 张贤达, 现代信号处理[M]. 2版. 北京: 清华大学出版社,2002.

[14] 赵力. 语音信号处理[M]. 北京: 机械工业出版社,2003.

[15] 夸特尔瑞. 离散时间语音信号处理——原理与应用[M]. 赵胜辉等, 译. 北京: 电子工业出版社,2004.

[16] 王炳锡, 王洪. 变速率语音编码[M]. 西安:西安电子科技大学出版社,2004.

[17] 胡航. 语音信号处理[M]. 3版. 哈尔滨: 哈尔滨工业大学出版社,2005.

[18] 程佩清. 数字信号处理教程[M]. 3版. 北京: 清华大学出版社,2007.

[19] 吴家安, 现代语音编码技术[M]. 北京: 科学出版社,2007.

[20] 万帅,杨付正. 新一代高效视频编码 H.265/HEVC[M]. 北京:电子工业出版社,2014.

[21] Kenneth.R.Castleman. Digital Image Processing. 北京.清华大学出版社. 1998.

[22] ITU-T Draft Recommendation and Final Draft International Standard of JointVideo Specification (ITU-T Rec. H. 264 | ISO/IEC 14496-10 AVC). 7th Meeting: Pattaya, Thailand, 7-14 Mar.2003.

[23] Abdul H. Sadka. Compressed Video Communications. Hohn Wiley & Sons, Ltd. England. 2002.

[24] Iain E.G.Richardson. H.264 and MPEG-4 Video Compression. Hohn Wiley & Sons,Ltd. England. 2002.

[25] 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.

[26] 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.

[27] 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.

[28] 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.

[29] 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.

[30] Adrian Grange, Peter de Rivaz,and Jonathan Hunt,"VP9 Bitstream & Decoding Process Specification",2016.