GY

中华人民共和国广播电影电视行业标准

GY/T 257.1—2012

广播电视先进音视频编解码
第1部分:视频

Advanced coding of video and audio for broadcasting—
Part 1: video

2012-07-10 发布　　2012-07-10 实施

国家广播电影电视总局　发布

目　　次

前　言

本部分为GY/T 257《广播电视先进音视频编解码》的第1部分。

本部分按照GB/T 1.1-2009给出的规则起草。

本部分由全国广播电影电视标准化技术委员会（SAC/TC 239）归口。

本部分起草单位：中央电视台、北京大学、国家广播电影电视总局广播科学研究院、国家广播电影电视总局广播电视规划院、北京博雅华录视听技术研究院有限公司、广州柯维新数码科技有限公司、深圳市海思半导体有限公司、北京大学深圳研究生院、清华大学、中国科学院计算技术研究所、浙江大学、华为技术有限公司、上海国茂数字技术有限公司、中山大学、华中科技大学、北京工业大学。

本部分主要起草人：丁文华、高文、郭晓强、邓向冬、张伟民、黄铁军、虞露、何芸、马思伟、曾志华、梁凡、郑萧桢、张莉、郑建铧、潘晓菲、王强、董文辉、王荣刚、林永兵、张贤国、王振宇。

引　言

本部分遵循GB/T 20090.2-2006《信息技术 先进音视频编码 第2部分：视频》，增加了算术编码、帧级加权量化、同极性场跳过模式编码、增强场编码等技术。

本部分的发布机构提请注意如下事实，声明符合本部分时，可能涉及到8.2、8.4、9.2、9.3、9.4.2、9.4.5、9.4.6.2、9.4.6.3、9.4.9、9.4.9、9.5.1、9.5.2、9.6、9.7、9.9.2.2、9.9.2.3、9.11、附录A和附录D相关的专利的使用。

本部分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。

专利持有人已向本部分的发布机构保证，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部分的发布机构备案。

在本部分起草过程中，起草组织者AVS技术应用联合推进工作组根据会员签署同意的AVS工作组章程和AVS有关知识产权规定以及会员在提案、审阅等期间提出的专利披露与许可声明等对标准可能涉及的专利进行了识别。已经确知下表列出的专利权人持有本部分的专利：

专利持有人	地址
广州柯维新数码科技有限公司	广州大学城外环东路232号13栋B301（510006）
华为技术有限公司	广东省深圳市龙岗区坂田华为基地（518129）
北京大学深圳研究生院	广东省深圳大学城北京大学深圳研究生院（518055）
清华大学	北京市海淀区清华大学电子工程系（100084）
中国科学院计算技术研究所	北京市海淀区中关村科学院南路6号（100080）
浙江大学	浙江省杭州市浙江大学信息与通信工程研究所（310027）
联合信源数字音视频技术（北京）有限公司	北京市海淀区上地东路1号盈创动力大厦A座601室（100085）
上海国茂数字技术有限公司	上海市浦东张江高科技园区张衡路500弄1号楼5楼（201204）
华中科技大学	湖北省武汉市洪山区珞瑜路1037号电子与信息工程系（430074）
北京工业大学	北京市朝阳区平乐园100号计算机学院（100022）

上述专利权人同意对所持有的本部分的必要专利在合理和非歧视的条款和条件基础上，通过AVS专利池进行许可。由数字音视频编解码技术标准工作组推动成立的AVS专利池管理委员会是决定专利池具体许可条款的独立机构。对于专利池中的所有专利，标准实施者可通过专利池管理委员会认可的授权机构获得许可。有关资料可从数字音视频编解码技术标准工作组秘书处获得，联系方法如下：

联 系 人：黄铁军（数字音视频编解码技术标准工作组秘书长）

通讯地址：北京大学理科2号楼2641室

邮政编码：100871

电子邮件：tjhuang@pku.edu.cn

电　　话：+10-62756172

传　　真：+10-62751638

网　　址：http://www.avs.org.cn

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

广播电视先进音视频编解码
第1部分:视频

1 范围

本部分规定了多种比特率、分辨率和质量的视频压缩方法，并且规定了解码过程。

本部分适用于地面电视广播、有线电视广播、卫星电视广播等应用。

2 规范性引用文件

下列文件对于本部分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部分。凡是不注日期的引用文件，其最新版本（包括所有的修改单）适用于本部分。

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

3 术语和定义

下列术语和定义适用于本部分。

3.1

保留 reserved

定义了一些特定语法元素值，这些值用于将来对本部分的扩展。

注：这些值不出现在符合本部分的位流中。

3.2

变长编码 variable length coding

一个可逆的熵编码过程，它将短的码字分配给出现频率较高的符号，将长的码字分配给出现频率较低的符号。

3.3

变换系数 transform coefficient

变换域上的一个标量。

3.4

编码表示 encoding presentation

数据编码后的形式。

3.5

编码过程 encoding process

产生符合本部分位流的过程。

注：本部分不规定该过程。

3.6

编码器 encoder

完成编码过程的实体。

3.7

编码图像 coded picture

一帧图像的编码表示。

3.8

标志 flag

一个二值变量。

3.9

补偿 compensation

求由语法元素解码得到的样本残差与其对应的预测值之和。

3.10

残差 residual

样本或数据元素的重建值与其预测值之差。

3.11

参考索引 reference index

解码图像缓冲区中参考图像或其中场的编号。

3.12

参考图像 reference picture

解码过程中用于后续图像帧间预测的图像。

3.13

层 layer

位流中的分级结构，高层包含低层。编码层由高到低依次为：序列、图像、条带、宏块和块。

3.14

场 field

由构成帧的三个样本矩阵中相间的行构成。

3.15

二元符号 bin

组成二元符号串的符号，包括“0”和“1”。

3.16

二元符号串 bin string

有限位二元符号组成的有序序列，最左边符号是最高有效位(MSB)，最右边符号是最低有效位(LSB)。

3.17

分量 component

图像的三个样值矩阵（亮度和两个色度）中的一个矩阵或矩阵中的单个样值。

3.18

反变换 inverse transform

将变换系数矩阵转换成空域样值矩阵的过程。

3.19

反量化 dequantization

对量化系数缩放后得到变换系数的过程。

3.20

光栅扫描 raster scan

将二维矩形光栅映射到一维光栅，一维光栅的入口从二维光栅的第一行开始，然后扫描第二行、第三行，依次类推。光栅中的行从左到右扫描。

3.21

宏块 macroblock

包括一个16×16的亮度样值块和对应的色度样值块。

3.22

宏块地址 macroblock address

从图像左上角的宏块开始，沿光栅扫描的顺序编号，起始号为0。

3.23

宏块行 macroblock line

在编码的图像相同的垂直位置，从左边界到右边界连续的宏块，其高度是16个样本。

3.24

宏块位置 macroblock position

图像中一个宏块的二维坐标，表示为（x,y）。

示例：如果当前图像的两场的编码数据合并成帧，图像左上角的宏块（x,y）=（0,0），对每个宏块列，从左到右 x 依次加 1，对每个宏块行，从上到下 y 依次加 1。如果当前图像的两场的编码数据依次出现，第一场左上角的宏块（x,y）=（0,0），对第一场的每个宏块列，从左到右 x 依次加 1，对第一场的每个宏块行，从上到下 y 依次加 1；第二场左上角的宏块（x,y）=（0,(H+31）>>5），H 是图像垂直方向扫描行数，对第二场的每个宏块列，从左到右 x 依次加 1，对第二场的每个宏块行，从上到下 y 依次加 1。

3.25

后向预测 backward prediction

用显示顺序上将来的参考图像对当前图像进行预测。

3.26

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。

3.27

级 level

在某一类下对语法元素和语法元素参数值的限定集合。

3.28

交流系数 AC coefficient

AC系数

二维变换域上索引号不全为0的变换系数。

3.29

解码处理 decode processing

解析过程和解码过程。

3.30

解码过程 decoding process

由语法元素产生解码图像的过程。

3.31

解码器 decoder

完成解码处理的实体。

3.32

解码顺序 decoding order

解码过程根据图像之间的预测关系，对每帧图像解码的顺序。

3.33

解码图像 decoded picture

解码器根据位流重建的图像。

3.34

解码图像缓冲区 decoded picture buffer

保存解码图像并用于预测、输出重排序和输出定时的缓冲区。

3.35

解析过程 parse

由位流获得语法元素的过程。

3.36

禁止 forbidden

定义了一些特定语法元素值，这些值不应出现在符合本部分的位流中。

注：禁止某些值的目的通常是为了避免在位流中出现伪起始码。

3.37

块 block

一个M×N的样值矩阵或者变换系数矩阵（M列N行），大小为8×8的块又称为子块。

3.38

块扫描 block scan

量化系数的特定串行排序方式。

3.39

类 profile

本部分规定的语法、语义及算法的子集。

3.40

亮度 luma

表示亮度信号的样值矩阵或单个样值，符号为Y。

3.41

量化参数 quantization parameter

在解码过程对量化系数进行反量化的参数。

3.42

量化系数 quantization coefficient

反量化前变换系数的值。

3.43

X类解码器 x-profile decoder

能够解码符合某类规定的位流的解码器。

3.44

起始码 start code

长度为32位的二进制码字，其形式在整个位流中是唯一的。

注：起始码有多种用途，其中之一是用来标识位流语法结构的开始。

3.45

前向预测 forward prediction

用显示顺序上过去的参考图像对当前图像进行预测。

3.46

前向帧间解码图像 forward inter decoded picture

P帧

帧间预测中只使用前向预测解码的图像。

3.47

色度 chroma

两种色差信号中任一种的样值矩阵或单个样值，符号为Cr和Cb。

3.48

视频序列 sequence

编码位流的最高层语法结构，包括一个或多个连续的编码图像。

3.49

输出重排序延迟 output reorder delay

解码位流中一帧图像到输出该解码图像之间的延迟。这是由图像显示顺序和解码顺序不同造成的。

3.50

输出处理过程 output processing

由解码图像得到输出帧或场的过程。

3.51

输出顺序 output order

输出解码图像的顺序，与显示顺序相同。

3.52

双向预测 bidirectional prediction

用显示顺序上过去和将来的参考图像对当前图像进行预测。

3.53

双向帧间解码图像 bidirectional inter decoded picture

B帧

帧间预测中使用双向预测解码的图像。

3.54

随机访问 random access

从某一点而非位流起始点开始对位流解码并恢复出解码图像的能力。

3.55

随机访问点 random access point

位流中能进行随机访问的点。

3.56

填充位 stuffing bits

编码时插入位流中的位串，在解码时被丢弃。

3.57

条带 slice

按光栅扫描顺序排列的若干连续宏块。

3.58

条带头 slice header

编码的条带的一部分，是条带中宏块公用数据元素的编码表示。

3.59

跳过的宏块 skipped macroblock

除“跳过”指示外，无其他编码数据的宏块。

3.60

图像重排序 picture reordering

若解码顺序和输出顺序不同，对解码图像进行重排序的过程。

3.61

位串 bit string

有限个二进制位的有序序列，其最左边位是最高有效位（MSB），最右边位是最低有效位（LSB）。

3.62

位流 bitstream

编码图像所形成的二进制数据流。

3.63

位流缓冲区 bitstream buffer

存储位流的缓冲区。

3.64

位流顺序 bitstream order

编码图像在位流中的排列顺序，与图像解码的顺序相同。

3.65

显示顺序 display order

显示解码图像的顺序。

3.66

样本 sample

构成图像的基本元素。

3.67

样本宽高比 width height ratio

一帧图像中亮度样本列间的水平距离与行间的垂直距离之比。

表示为h:v，其中h为水平方向样本个数，v为垂直方向样本个数。

3.68

样值 sample value

样本的幅值。

3.69

游程 run

在解码过程中若干连续的相同数据元素个数。一方面指在块扫描中一个非0系数前（沿块扫描顺序）值为0的系数的个数；另一方面指跳过的宏块的数目。

3.70

预测 prediction

预测过程的具体实现。

3.71

预测过程 prediction process

使用预测器对当前解码样值或者数据元素进行估计。

3.72

预测值 prediction value

在样值或数据元素的解码过程中，用到的先前已解码的样值或数据元素的组合。

3.73

语法元素 syntax element

位流中的数据单元解析后的结果。

3.74

源 source

编码前视频素材或其某些属性。

3.75

运动矢量 motion vector

用于帧间预测的二维矢量，由当前图像指向参考图像，其值为当前块和参考块之间的坐标偏移量。

3.76

帧 frame

视频信号空间信息的表示，由一个亮度样本矩阵（Y）和两个色度样本矩阵（Cb和Cr）构成。

3.77

帧间编码 inter coding

使用帧间预测对宏块或图像进行编码。

3.78

帧间预测 inter prediction

使用先前解码图像（或场）生成当前图像（或场）样本预测值的过程。

3.79

帧内编码 intra coding

使用帧内预测对宏块或图像进行编码。

3.80

帧内解码图像 intra decoded picture

I帧

只使用帧内预测解码的图像。

注：如果I帧采用场编码，则第一场只使用帧内预测编码。

3.81

帧内预测 intra prediction

在相同解码图像（或场）中使用先前解码的样值生成当前样本预测值的过程。

3.82

直流系数 DC coefficient

DC系数

二维变换域上索引号全为0的变换系数。

3.83

字节 byte

8位的位串。

3.84

字节对齐 byte alignment

从位流的第一个二进制位开始，某二进制位的位置是8的整数倍。

4 缩略语

下列缩略语适用于本部分。

BBV 位流参考解码器（Bitstream Buffer Verifier）

CBR 恒定比特率（Constant Bit Rate）

LSB 最低有效位（Least Significant Bit）

MB 宏块（Macroblock）

MSB 最高有效位（Most Significant Bit）

VLC 变长编码（Variable Length Coding）

5 约定

本部分中使用的数学运算符和优先级与C语言使用的类似。但对整型除法和算术移位操作进行了特定定义。除特别说明外，约定编号和计数从0开始。

5.1 算术运算符

算术运算符定义见表1。

表1 算术运算符定义

算术运算符	定义
+	加法运算
–	减法运算（二元运算符）或取反（一元前缀运算符）
×	乘法运算
a^b	幂运算，表示a的b次幂。也可表示上标
/	整除运算，沿向0的取值方向截断。例如，7/4和-7/-4截断至1，-7/4和7/-4截断至-1
÷	除法运算，不做截断或四舍五入
$\frac{a}{b}$	除法运算，不做截断或四舍五入
$\sum_{i=a}^{b} f(i)$	自变量i取由a到b（含b）的所有整数值时，函数 $f(i)$ 的累加和
a % b	模运算，a除以b的余数，其中a与b都是正整数

5.2 逻辑运算符

逻辑运算符定义见表2。

表2 逻辑运算符定义

逻辑运算符	定义
a && b	a和b之间的与逻辑运算
a || b	a和b之间的或逻辑运算
!	逻辑非运算

5.3 关系运算符

关系运算符定义见表3。

表3 关系运算符定义

关系运算符	定义
>	大于
>=	大于或等于
<	小于
<=	小于或等于
==	等于
!=	不等于

5.4 位运算符

位运算符定义见表4。

表4　位运算符定义

位运算符	定义
&	与运算
|	或运算
~	取反运算
a >> *b*	将*a*以2的补码整数表示的形式向右移*b*位。仅当*b*取正数时定义此运算
a << *b*	将*a*以2的补码整数表示的形式向左移*b*位。仅当*b*取正数时定义此运算

5.5　赋值

赋值运算定义见表5。

表5　赋值运算定义

赋值运算	定义
=	赋值运算符
++	递增，*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
--	递减，*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
+=	自加指定值，例如*x* += 3相当于*x* = x + 3，*x* += (-3)相当于*x* = *x* + (-3)
-=	自减指定值，例如*x* -= 3相当于*x* = *x* - 3,，*x* -= (-3)相当于*x* = *x* - (-3)

5.6　数学函数

数学函数定义见式（1）～式（8）。

$$\mathrm{Abs}(x)=\begin{cases}x & ; \quad x>=0\\ -x & ; \quad x<0\end{cases} \qquad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots (1)$$

式中：

x ——自变量x。

$$\mathrm{Ceil}(x)=\lceil x\rceil \qquad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots (2)$$

式中：

x ——自变量x。

$$\mathrm{Clip1}(x)=\mathrm{Clip3}(0,2^{n}-1,x) \qquad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots (3)$$

式中：

x ——自变量x；

n ——样本点精度。

$$\mathrm{Clip3}(i,j,x)=\begin{cases}i & ; \quad x<i\\ j & ; \quad x>j\\ x & ; \quad \text{其他}\end{cases} \qquad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots (4)$$

式中：

x ——自变量x；

i ——下界；

j ——上界。

$$\mathrm{Median}(x,y,z)=x+y+z-\mathrm{Min}(x,\mathrm{Min}(y,z))-\mathrm{Max}(x,\mathrm{Max}(y,z)) \quad \cdots\cdots\cdots\cdots (5)$$

式中：

x ——自变量x；

y ——自变量y；

z ——自变量z。

$$\mathrm{Min}(x,y)=\begin{cases} x & ; \quad x<=y \\ y & ; \quad x>y \end{cases} \quad \cdots\cdots\cdots\cdots (6)$$

式中：

x ——自变量x；

y ——自变量y。

$$\mathrm{Max}(x,y)=\begin{cases} x & ; \quad x>=y \\ y & ; \quad x<y \end{cases} \quad \cdots\cdots\cdots\cdots (7)$$

式中：

x ——自变量x；

y ——自变量y。

$$\mathrm{Sign}(x)=\begin{cases} 1 & ; \quad x>=0 \\ -1 & ; \quad x<0 \end{cases} \quad \cdots\cdots\cdots\cdots (8)$$

式中：

x ——自变量x。

5.7 结构关系

结构关系符定义见表6。

表6 结构关系符定义

结构关系符	定义
->	例如：a->b表示a是一个结构，b是a的一个成员变量

5.8 位流语法、解析过程和解码过程的描述方法

5.8.1 位流语法的描述方法

位流语法描述方法类似C语言。位流的语法元素使用粗体字表示，每个语法元素通过名字（用下划线分割的英文字母组，所有字母都是小写）、语法和语义来描述。语法表和正文中语法元素的值用常规字体表示。

某些情况下，可在语法表中应用从语法元素导出的其他变量值，这样的变量在语法表或正文中用不带下划线的小写字母和大写字母混合命名。大写字母开头的变量用于解码当前以及相关的语法结构，也可用于解码后续的语法结构。小写字母开头的变量只在它们所在的小节内使用。

语法元素值的助记符和变量值的助记符与它们的值之间的关系在正文中说明。在某些情况下，二者等同使用。助记符由一个或多个使用下划线分隔的字母组表示，每个字母组以大写字母开始，也可包括多个大写字母。

位串的长度是4的整数倍时，可使用十六进制符号表示。十六进制的前缀是“0x”，例如“0x1a”表示位串“0001 1010”。

条件语句中0表示FALSE，非0表示TRUE。

语法表描述了所有符合本部分的位流语法的超集，附加的语法限制在相关条中说明。

表7给出了描述语法的伪代码例子。当语法元素出现时，表示从位流中读一个数据单元。

表7　语法描述的伪代码

伪代码	描述符
/*语句是一个语法元素的描述符，或者说明语法元素的存在、类型和数值，下面给出两个例子。*/	
syntax_element	ue(v)
conditioning statement	
/*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。*/	
{	
statement	
…	
}	
/*“while”语句测试condition是否为TRUE，如果为TRUE，则重复执行循环体，直到condition不为TRUE。*/	
while (condition)	
statement	
/*“do … while”语句先执行循环体一次，然后测试condition是否为TRUE，如果为TRUE，则重复执行循环体，直到condition不为TRUE。*/	
do	
statement	
while (condition)	
/*“if … else”语句首先测试condition，如果为TRUE，则执行primary语句，否则执行alternative语句。如果alternative语句不需要执行，结构的“else”部分和相关的alternative语句可忽略。*/	
if (condition)	
primary statement	
else	

表7（续）

伪代码	描述符
alternative statement	
/*“for”语句首先执行initial语句，然后测试condition，如果conditon为TRUE，则重复执行primary语句和subsequent语句直到condition不为TRUE。*/	
for (initial statement; condition; subsequent statement)	
primary statement	

解析过程和解码过程用文字和类似C语言的伪代码描述。

5.8.2 函数

以下函数用于语法描述。假定解码器中存在一个位流指针，这个指针指向位流中要读取的下一个二进制位的位置。函数由函数名及左右圆括号内的参数构成。函数也可没有参数。

5.8.2.1 byte_aligned( )

如果位流的当前位置是字节对齐的，返回TRUE，否则返回FALSE。

5.8.2.2 next_bits(n)

返回位流的随后*n*个二进制位，MSB在前，不改变位流指针。如果剩余的二进制位少于*n*，则返回0。

5.8.2.3 byte_aligned_next_bits(n)

如果位流当前位置不是字节对齐的,返回位流当前位置的下一个字节开始的*n*个二进制位,MSB在前，不改变位流指针；如果位流当前位置是字节对齐的，返回位流随后的*n*个二进制位，MSB在前，不改变位流指针。如果剩余的二进制位少于*n*，则返回0。

5.8.2.4 next_start_code( )

在位流中寻找下一个起始码，将位流指针指向起始码前缀的第一个二进制位。函数定义见表8。

表8 next_start_code 函数的定义

函数定义	描述符
next_start_code( ) {	
stuffing_bit	’1’
while (! byte_aligned( ))	
stuffing_bit	’0’
while (next_bits(24) != ’0000 0000 0000 0000 0000 0001’)	
stuffing_byte	’00000000’
}	

stuffing_byte应出现图像头之后和第一个条带起始码之前。

5.8.2.5 is_end_of_slice( )

在位流中检测是否已达到条带的结尾，如果已到达条带的结尾，返回TRUE，否则返回FALSE。此函数不修改位流指针。函数定义见表9。

表9 is_end_of_slice 函数的定义

函数定义	描述符
is_end_of_slice ( ) {	
if (byte_aligned ( )) {	
if (next_bits(32) == 0x80000001)	
return TRUE; // 条带结束	
}	
else {	
if ((byte_aligned_next_bits(24) == 0x000001) && is_stuffing_pattern( ))	
return TRUE; // 条带结束	
}	
return FALSE;	
}	

5.8.2.6 is_stuffing_pattern( )

在位流中检测当前字节中剩下的位或在字节对齐时下一个字节是否是条带结尾填充的二进制位，如果是，则返回TRUE，否则返回FALSE。此函数不修改位流指针。函数定义见表10。

表10 is_stuffing_pattern 函数的定义

函数定义	描述符
is_stuffing_pattern ( ) {	
if (next_bits(8-n) == (1<< (7-n))) // n：0～7，为位流指针在当前字节的位置偏移，n为0时位流指针指向当前字节最高位	
return TRUE;	
else	
return FALSE;	
}	

5.8.2.7 read_bits(n)

返回位流的随后*n*个二进制位，MSB在前，同时位流指针前移*n*个二进制位。如果*n*等于“0”，则返回0，位流指针不前移。

函数也用于解析过程和解码过程的描述。

5.8.3 描述符

描述符表示不同语法元素的解析过程，见表11。

表11 描述符

描述符	说明
ae(v)	高级熵编码的语法元素。解析过程在8.4中定义
b(8)	一个任意取值的字节。解析过程由函数read_bits(8)的返回值规定
ce(v)	变长编码的语法元素。解析过程在8.3中定义
f(*n*)	取特定值的连续*n*个二进制位。解析过程由函数read_bits(*n*)的返回值规定

表 11（续）

描述符	说明
i(*n*)	*n*位整数。在语法表中，如果*n*是“v”，其位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，该返回值用高位在前的2的补码表示
me(v)	用指数哥伦布码编码的语法元素。解析过程在8.2中定义
r(*n*)	连续*n*个“0”。解析过程由函数read_bits(*n*)的返回值规定
se(v)	有符号整数语法元素，用指数哥伦布码编码。解析过程在8.2中定义
u(*n*)	*n*位无符号整数。在语法表中，如果*n*是“v”，其位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，该返回值用高位在前的二进制表示
ue(v)	无符号整数语法元素，用指数哥伦布码编码。解析过程在8.2中定义

5.8.4 保留、禁止和标记

本部分定义的位流语法中，某些语法元素的值被标注为“保留”(reserved)或“禁止”(forbidden)。

“保留”定义了一些特定语法元素值用于将来对本部分的扩展。这些值不应出现在符合本部分的位流中。

“禁止”定义了一些特定语法元素值，这些值不应出现在符合本部分的位流中。

“标记”（marker_bit）指该位的值应为“1”。

位流中的“保留位”（reserved_bits）表明保留了一些语法单元用于将来对本部分的扩展，解码处理应忽略这些位。“保留位”不应出现从任意字节对齐位置开始的21个以上连续的“0”。

6 编码位流的结构

6.1 视频序列

6.1.1 概述

视频序列是位流的最高层语法结构。视频序列由序列头开始，后面跟着一个或多个编码图像，每帧图像之前应有图像头。编码图像在位流中按位流顺序排列，位流顺序应与解码顺序相同。解码顺序可与显示顺序不相同。序列结束码表明了一个视频序列的结束。

6.1.2 逐行和隔行视频序列

本部分支持两种序列：逐行序列和隔行序列。

帧由三个样本矩阵构成，包括一个亮度样本矩阵（Y）和两个色度样本矩阵（Cb和Cr）。样本矩阵元素的值为整数。Y、Cb和Cr三个分量与原始的（模拟）红、绿和蓝色信号之间的关系，包括原始信号的色度和转移特性等可在位流中定义，这些信息不影响解码过程。

场由构成帧的三个样本矩阵中相间的行构成，即帧样本矩阵的第一行、第三行、第五行等奇数行构成一个场，称为顶场；第二行、第四行、第六行等偶数行构成另一个场，称为底场。

解码器的输出是一系列帧或场，两帧之间存在着一个帧时间间隔。对隔行序列而言，每帧图像的两场之间存在着一个场时间间隔。对逐行序列而言，每帧图像的两场之间时间间隔为0。

6.1.3 序列头

视频序列头由视频序列起始码开始，后面跟着一串编码图像数据。

序列头可在位流中重复出现，称为重复序列头。使用重复序列头的主要目的是支持对视频序列的随机访问。

序列头后的第一个编码图像应是I帧。序列头后出现的P帧只能参考该序列头后出现的图像。在对位流进行编辑或随机访问的情况下，重复序列头之前的全部数据可被丢弃，这样得到的一个新的位流仍应符合本部分。

6.2 图像

6.2.1 概述

一幅图像是一帧，其编码数据由图像起始码开始，到序列起始码、序列结束码或下一个图像起始码结束。

在位流中，隔行扫描图像的两场的编码数据可依次出现，也可合并成帧。两场数据的解码和显示顺序在图像头中规定。

图像的解码处理包括解析过程和解码过程。

6.2.2 图像格式

6.2.2.1 4:2:0 格式

对于4:2:0格式，Cb和Cr矩阵水平和垂直方向的尺寸都只有Y矩阵的一半。Y矩阵的行数和每行样本数都应是偶数。另外，如果图像两场的编码数据依次出现，则Y矩阵的行数还应能被4整除。

亮度和色度样本位置见图1。

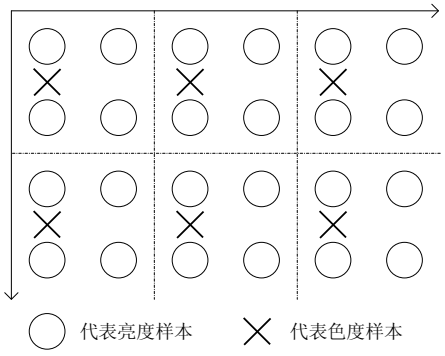


图1 4:2:0 格式下亮度和色度样本位置

6.2.2.2 4:2:2 格式

对于4:2:2格式，Cb和Cr矩阵在水平方向的尺寸只有Y矩阵的一半，在垂直方向的尺寸和Y相同。Y矩阵的每行样本数应是偶数。如果图像两场的编码数据依次出现，则Y矩阵的行数也应是偶数。

亮度和色度样本位置见图2。

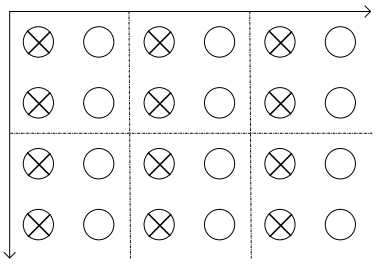


图2 4:2:2 格式下亮度和色度样本位置

6.2.3 图像类型

本部分定义了三种解码图像：

——帧内解码图像（I 帧）；

——前向帧间解码图像（P 帧）；

——双向帧间解码图像（B 帧）。

6.2.4 图像间的顺序

如果视频序列中没有B帧，解码顺序与显示顺序相同。如果视频序列中包含B帧，解码顺序与显示顺序不同，解码图像输出显示前应进行图像重排序。图像重排序规则如下：

——当前解码图像是 B 帧，输出由此 B 帧解码的图像；

——当前解码图像是 I 帧或 P 帧，如果存在前一个 I 帧或 P 帧的解码图像，输出前一个 I 帧或 P 帧的解码图像。如果不存在前一个 I 帧或 P 帧的解码图像，不输出任何解码图像；

——位流解码完成后，如果缓冲区中还有未输出的解码图像，则输出该图像。

下面举例说明图像重排序：I帧和P帧之间有两个B帧，两个连续的P帧之间也有两个B帧。用图像1I预测图像4P，用图像4P和1I预测图像2B和3B。解码顺序是1I，4P，2B，3B；显示顺序是1I，2B，3B，4P。

示例：

编码器输入顺序：

1	2	3	4	5	6	7	8	9	10	11	12	13
I	B	B	P	B	B	P	B	B	I	B	B	P

解码顺序：

1	4	2	3	7	5	6	10	8	9	13	11	12
I	P	B	B	P	B	B	I	B	B	P	B	B

解码器输出，即显示顺序：

1	2	3	4	5	6	7	8	9	10	11	12	13
I	B	B	P	B	B	P	B	B	I	B	B	P

6.2.5 参考图像

P帧或B帧最多可有两帧参考图像。P帧可参考前向的两帧。除B帧外，在一帧中，后解码的场还可参考当前帧的另外一场。B帧可参考一前一后的两帧。

运动矢量所指的参考像素可超出参考图像的边界，在这种情况下对超出参考图像边界的整数样本应使用距离该整数参考样本所指位置最近的图像内的整数样本进行边界扩展。对亮度样本矩阵，参考块的像素在水平和垂直方向均不应超出参考图像边界外16个像素。对色度样本矩阵：

——如果图像格式是 4:2:0，参考块的像素在水平和垂直方向均不应超出参考图像边界外 8 个像素。

——如果图像格式是 4:2:2，参考块的像素在水平方向不应超出参考图像边界外 8 个像素，在垂直方向不应超出参考图像边界外 16 个像素。

场边界扩展方法和参考图像边界扩展方法相同。

6.3 条带

条带是按光栅扫描顺序连续排列的若干宏块行，条带内的宏块不应重叠，条带之间也不应重叠。

如果当前图像的两场数据合并成帧，条带内宏块的解码处理不应使用本图像其他条带的数据。

如果当前图像的两场数据依次出现，这两场数据应属于不同的条带，条带内宏块的解码处理不应使用同一场其他条带的数据。

条带结构见图3。条带边界扩展方法和参考图像边界扩展方法相同。

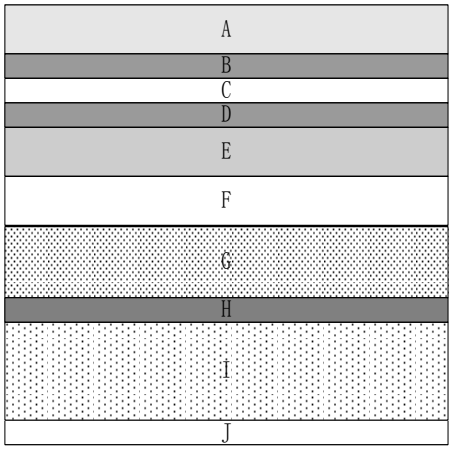


图3　条带结构

6.4　宏块

图像划分为宏块，宏块左上角的点不应超出图像边界。在位流中，当隔行扫描图像的两场编码数据依次出现时，任一宏块的像素应来自同一场。

宏块的划分见图4，这种划分用于运动补偿。图4中矩形里的数字表示宏块划分后运动矢量和参考索引在位流中的顺序。

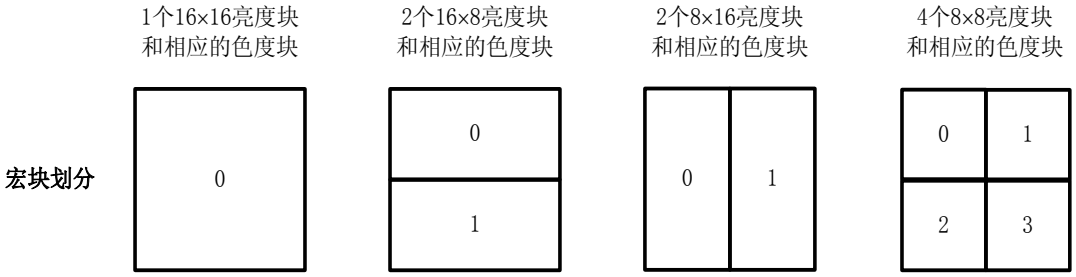


图4　宏块的划分

6.5　8×8 块

在4:2:0格式下，一个宏块包括4个8×8亮度块（Y）和2个8×8色度块（1个Cb，1个Cr），见图5。图中数字为宏块中8×8块的顺序号。

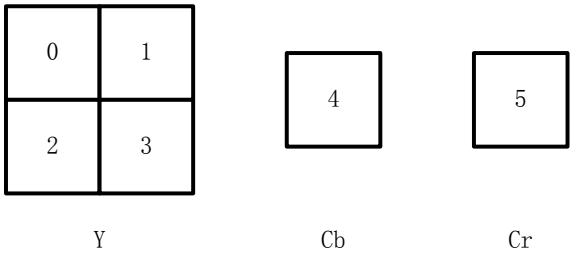


图5　宏块划分为 8×8 块（4:2:0 格式）

在4:2:2格式下，一个宏块包括4个8×8亮度块（Y）和4个8×8色度块（2个Cb，2个Cr），见图6。图中数字为宏块中8×8块的顺序号。

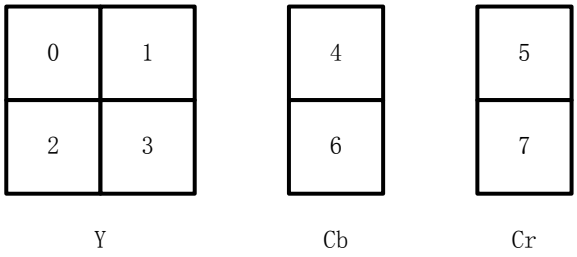


图6　宏块划分为 8×8 块（4:2:2 格式）

宏块中的各个8×8块在位流中出现的顺序由图5到图6中的数字规定。

7　位流的语法和语义

7.1　语法描述

7.1.1　起始码

起始码是一组特定的位串。在符合本部分的位流中，除起始码外的任何情况下都不应出现这些位串。

起始码由起始码前缀和起始码值构成。起始码前缀是位串“0000 0000 0000 0000 0000 0001”。所有的起始码都应字节对齐。

起始码值是一个8位整数，用来表示起始码的类型，见表12。

表12　起始码值

起始码类型	起始码值（十六进制）
条带起始码（slice_start_code）	00～AF
视频序列起始码（video_sequence_start_code）	B0
视频序列结束码（video_sequence_end_code）	B1
用户数据起始码（user_data_start_code）	B2
I图像起始码（i_picture_start_code）	B3
保留	B4
视频扩展起始码（extension_start_code）	B5
PB图像起始码（pb_picture_start_code）	B6
视频编辑码（video_edit_code）	B7
保留	B8
系统起始码	B9～FF

部分语法元素取特定值时可得到与起始码前缀相同的位串，称为伪起始码。符合本部分的编码器和解码器应使用附录A定义的方法处理伪起始码问题。

7.1.2　视频序列语法

7.1.2.1　视频序列定义

视频序列定义见表13。

表13　视频序列定义

视频序列定义	描述符
video_sequence() {	
do {	
sequence_header()	
extension_and_user_data(0)	
do {	
if (next_bits(32) == i_picture_start_code)	
i_picture_header()	
else	
pb_picture_header()	
extension_and_user_data(1)	
picture_data()	
} while ((next_bits(32) pb_picture_start_code) || (next_bits(32) i_picture_start_code))	
} while (next_bits(32) ! video_sequence_end_code && next_bits(32) != video_edit_code)	
if (next_bits(32) == video_sequence_end_code)	
video_sequence_end_code	f(32)
if (next_bits(32) == video_edit_code)	
video_edit_code	f(32)
}	

7.1.2.2　序列头定义

序列头定义见表14。

表14　序列头定义

序列头定义	描述符
sequence_header() {	
video_sequence_start_code	f(32)
profile_id	u(8)
level_id	u(8)
progressive_sequence	u(1)
horizontal_size	u(14)
vertical_size	u(14)
chroma_format	u(2)
sample_precision	u(3)
aspect_ratio	u(4)
frame_rate_code	u(4)
bit_rate_lower	u(18)
marker_bit	f(1)
bit_rate_upper	u(12)

表 14（续）

序列头定义	描述符
low_delay	u(1)
marker_bit	f(1)
bbv_buffer_size	u(18)
reserved_bits	r(3)
next_start_code()	
}	

7.1.2.3 扩展和用户数据定义

扩展和用户数据定义见表15。

表15 扩展和用户数据定义

扩展和用户数据定义	描述符
extension_and_user_data(i) {	
while ((next_bits(32) extension_start_code) || (next_bits(32) user_data_start_code)) {	
if (next_bits(32) extension_start_code)	
extension_data(i)	
if (next_bits(32) user_data_start_code)	
user_data()	
}	
}	

7.1.2.3.1 扩展数据定义

扩展数据定义见表16。

表16 扩展数据定义

扩展数据定义	描述符
extension_data(i) {	
while (next_bits(32) == extension_start_code) {	
extension_start_code	f(32)
if (i == 0) { /* 序列头之后 */	

表 16（续）

扩展数据定义	描述符
if (next_bits(4) == '0010') /* 序列显示扩展 */	
sequence_display_extension()	
else if (next_bits(4) == '0100') /* 版权扩展 */	
copyright_extension()	
else if (next_bits(4) == '1011') /* 摄像机参数扩展 */	
camera_parameters_extension()	
else	
while (next_bits(24) != '0000 0000 0000 0000 0000 0001')	
reserved_extension_data_byte	u(8)
}	
}	
else { /* 图像头之后 */	
if (next_bits(4) == '0100') /* 版权扩展 */	
copyright_extension()	
else if (next_bits(4) == '0111') /* 图像显示扩展 */	
picture_display_extension()	
else if (next_bits(4) == '1011') /* 摄像机参数扩展 */	
camera_parameters_extension()	
else {	
while (next_bits(24) != '0000 0000 0000 0000 0000 0001')	
reserved_extension_data_byte	u(8)
}	
}	
}	
}	

7.1.2.3.2 用户数据定义

用户数据定义见表17。

表17 用户数据定义

用户数据定义	描述符
user_data() {	
user_data_start_code	f(32)
while (next_bits(24) ! '0000 0000 0000 0000 0000 0001') {	
user_data	b(8)
}	
}	

7.1.2.4 序列显示扩展定义

序列显示扩展定义见表18。

表18　序列显示扩展定义

序列显示扩展定义	描述符
sequence_display_extension() {	
extension_id	f(4)
video_format	u(3)
sample_range	u(1)
colour_description	u(1)
if (colour_description) {	
colour_primaries	u(8)
transfer_characteristics	u(8)
matrix_coefficients	u(8)
}	
display_horizontal_size	u(14)
marker_bit	f(1)
display_vertical_size	u(14)
stereo_packing_mode	u(2)
next_start_code()	
}	

7.1.2.5　版权扩展定义

版权扩展定义见表19。

表19　版权扩展定义

版权扩展定义	描述符
copyright_extension() {	
extension_id	f(4)
copyright_flag	u(1)
copyright_id	u(8)
original_or_copy	u(1)
reserved_bits	r(7)
marker_bit	f(1)
copyright_number_1	u(20)
marker_bit	f(1)
copyright_number_2	u(22)
marker_bit	f(1)
copyright_number_3	u(22)
next_start_code()	
}	

7.1.2.6　摄像机参数扩展定义

摄像机参数扩展定义见表20。

表20　摄像机参数扩展定义

摄像机参数扩展定义	描述符
camera_parameters_extension() {	
extension_id	f(4)
reserved_bits	r(1)
camera_id	u(7)
marker_bit	f(1)
height_of_image_device	u(22)
marker_bit	f(1)
focal_length	u(22)
marker_bit	f(1)
f_number	u(22)
marker_bit	f(1)
vertical_angle_of_view	u(22)
marker_bit	f(1)
camera_position_x_upper	i(16)
marker_bit	f(1)
camera_position_x_lower	i(16)
marker_bit	f(1)
camera_position_y_upper	i(16)
marker_bit	f(1)
camera_position_y_lower	i(16)
marker_bit	f(1)
camera_position_z_upper	i(16)
marker_bit	f(1)
camera_position_z_lower	i(16)
marker_bit	f(1)
camera_direction_x	i(22)
marker_bit	f(1)
camera_direction_y	i(22)
marker_bit	f(1)
camera_direction_z	i(22)
marker_bit	f(1)
image_plane_vertical_x	i(22)
marker_bit	f(1)
image_plane_vertical_y	i(22)
marker_bit	f(1)
image_plane_vertical_z	i(22)
marker_bit	f(1)

表 20（续）

摄像机参数扩展定义	描述符
reserved_bits	r(32)
next_start_code()	
}	

7.1.3 图像定义

7.1.3.1 I 图像头定义

I图像头定义见表21。

表21 I 图像头定义

I图像头定义	描述符
i_picture_header() {	
i_picture_start_code	f(32)
bbv_delay	u(16)
if (profile_id == 0x48) {	
marker_bit	f(1)
bbv_delay_extension	u(7)
}	
time_code_flag	u(1)
if (time_code_flag == '1')	
time_code	u(24)
marker_bit	f(1)
picture_distance	u(8)
if (low_delay == '1')	
bbv_check_times	ue(v)
progressive_frame	u(1)
if (progressive_frame == '0')	
picture_structure	u(1)
top_field_first	u(1)
repeat_first_field	u(1)
fixed_picture_qp	u(1)
picture_qp	u(6)
if (progressive_frame == '0' && picture_structure == '0')	
skip_mode_flag	u(1)
}	
reserved_bits	r(4)

表 21（续）

I图像头定义	描述符
loop_filter_disable	u(1)
if (! loop_filter_disable) {	
loop_filter_parameter_flag	u(1)
if (loop_filter_parameter_flag) {	
alpha_c_offset	se(v)
beta_offset	se(v)
}	
}	
if (profile_id == 0x48) {	
weighting_quant_flag	u(1)
if (weighting_quant_flag == '1') {	
reserved_bits	r(1)
chroma_quant_param_disable	u(1)
if (chroma_quant_param_disable == '0') {	
chroma_quant_param_delta_cb	se(v)
chroma_quant_param_delta_cr	se(v)
}	
weighting_quant_param_index	u(2)
weighting_quant_model	u(2)
if (weighting_quant_param_index == '01') {	
for (i=0; i<6; i++)	
weighting_quant_param_delta1[i]	se(v)
}	
if (weighting_quant_param_index == '10') {	
for (i=0; i<6; i++)	
weighting_quant_param_delta2[i]	se(v)
}	
}	
}	
if (profile_id == 0x48) {	
aec_enable	u(1)
}	
next_start_code()	
}	

7.1.3.2 PB 图像头定义

PB图像头定义见表22。

表22 PB图像头定义

PB图像头定义	描述符
pb_picture_header() {	
pb_picture_start_code	f(32)
bbv_delay	u(16)
if (profile_id == 0x48) {	
marker_bit	f(1)
bbv_delay_extension	u(7)
}	
picture_coding_type	u(2)
picture_distance	u(8)
if (low_delay == '1')	
bbv_check_times	ue(v)
progressive_frame	u(1)
if (progressive_frame == '0') {	
picture_structure	u(1)
if (picture_structure == '0')	
advanced_pred_mode_disable	u(1)
}	
top_field_first	u(1)
repeat_first_field	u(1)
fixed_picture_qp	u(1)
picture_qp	u(6)
if (! (picture_coding_type=='10' && PictureStructure==1))	
picture_reference_flag	u(1)
no_forward_reference_flag	u(1)
pb_field_enhanced_flag	u(1)
reserved_bits	r(2)
skip_mode_flag	u(1)
loop_filter_disable	u(1)
if (! loop_filter_disable) {	
loop_filter_parameter_flag	u(1)
if (loop_filter_parameter_flag) {	
alpha_c_offset	se(v)
beta_offset	se(v)
}	
}	
if (profile_id == 0x48) {	
weighting_quant_flag	u(1)

表 22（续）

PB图像头定义	描述符
if (weighting_quant_flag == '1') {	
reserved_bits	r(1)
chroma_quant_param_disable	u(1)
if (chroma_quant_param_disable == '0') {	
chroma_quant_param_delta_cb	se(v)
chroma_quant_param_delta_cr	se(v)
}	
weighting_quant_param_index	u(2)
weighting_quant_model	u(2)
if (weighting_quant_param_index == '01') {	
for (i=0; i<6; i++)	
weighting_quant_param_delta1[i]	se(v)
}	
if (weighting_quant_param_index == '10') {	
for (i=0; i<6; i++)	
weighting_quant_param_delta2[i]	se(v)
}	
}	
}	
if (profile_id == 0x48) {	
aec_enable	u(1)
}	
next_start_code()	
}	

7.1.3.3 图像显示扩展定义

图像显示扩展定义见表23。

表23 图像显示扩展定义

图像显示扩展定义	描述符
picture_display_extension() {	
extension_id	f(4)
for (i 0; i NumberOfFrameCentreOffsets; i) {	
frame_centre_horizontal_offset	i(16)
marker_bit	f(1)
frame_centre_vertical_offset	i(16)
marker_bit	f(1)

表 23（续）

图像显示扩展定义	描述符
}	
next_start_code()	
}	

7.1.3.4 图像数据定义

图像数据定义见表24。

表24 图像数据定义

图像数据定义	描述符
picture_data() {	
do {	
slice()	
} while (next_bits(32) == slice_start_code)	
}	

7.1.3.5 条带定义

条带定义见表25。

表25 条带定义

条带定义	描述符
slice() {	
slice_start_code	f(32)
if (vertical_size > 2800)	
slice_vertical_position_extension	u(3)
if (fixed_picture_qp == '0') {	
fixed_slice_qp	u(1)
slice_qp	u(6)
}	
if (PictureType != 0 || (PictureStructure == 0 && MbIndex >= MbWidth × MbHeight / 2)) {	
slice_weighting_flag	u(1)
if (**slice_weighting_flag** == '1') {	
for (i=0; i<NumberOfReference; i++) {	
luma_scale	u(8)
luma_shift	i(8)
marker_bit	f(1)
chroma_scale	u(8)

表 25（续）

条带定义	描述符
chroma_shift	i(8)
marker_bit	f(1)
}	
mb_weighting_flag	u(1)
}	
}	
if (aec_enable == '1') {	
while (! byte_aligned())	
aec_byte_alignment_bit	f(1)
}	
do {	
if (PictureType != 0 || (PictureStructure == 0 && MbIndex >= MbWidth × MbHeight / 2)) {	
if (skip_mode_flag == '1')	
mb_skip_run	ue(v)|ae(v)
if (aec_enable == '1' && SkipMbCount != 0)	
aec_mb_stuffing_bit	ae(v)
}	
if (! is_end_of_slice()) {	
macroblock()	
if (aec_enable == '1')	
aec_mb_stuffing_bit	ae(v)
}	
} while (! is_end_of_slice())	
next_start_code()	
}	

7.1.3.6 宏块定义

宏块定义见表26。

表26 宏块定义

宏块定义	描述符
macroblock() {	
if (PictureType!=0 ||(PictureStructure==0 && MbIndex>=MbWidth×MbHeight/2))	
mb_type	ue(v)|ae(v)
if (MbType != 'P_Skip' && MbType != 'B_Skip') {	
if (MbType == 'B_8x8') {	
for (i=0; i<4; i++)	

表 26（续）

宏块定义	描述符
mb_part_type	u(2)|ae(v)
}	
if (MbType == 'I_8x8') {	
for (i=0; i<4; i++) {	
if (aec_enable == '0')	
pred_mode_flag	u(1)
if (! pred_mode_flag || aec_enable == '1')	
intra_luma_pred_mode	u(2) |ae(v)
}	
if (chroma_format != '00')	
intra_chroma_pred_mode	ue(v) |ae(v)
if (chroma_format == '10 ')	
intra_chroma_pred_mode_422	ue(v) |ae(v)
}	
if ((PictureType==1 || (PictureType==2 && PictureStructure==0) &&picture_reference_flag=='0') {	
for (i = 0; i < MvNum; i++)	
mb_reference_index	u(1)/u(2) |ae(v)
}	
for (i = 0; i < MvNum; i++) {	
mv_diff_x	se(v)|ae(v)
mv_diff_y	se(v)|ae(v)
}	
if (MbWeightingFlag == 1 && MbType != 'I_8x8')	
weighting_prediction	u(1)|ae(v)
if (aec_enable == '1')	
cbp	ae(v)
else if (! ((MbTypeIndex>=24 && PictureType==2) || (MbTypeIndex>=5 && (PictureType == 0 || PictureType == 1))))	
cbp	me(v)
if (chroma_format == '10')	
cbp_422	me(v)|ae(v)
if ((MbCBP > 0 || (MbCBP422 > 0 && chroma_format == '10')) && ! FixedQP)	
mb_qp_delta	se(v)|ae(v)
for (i = 0; i < 6; i++)	
block(i)	
if (chroma_format == '10 ') {	
for (i = 6; i < 8; i++)	

表 26（续）

宏块定义	描述符
block(i)	
}	
}	
}	

7.1.3.7 块定义

块定义见表27。

表27 块定义

块定义	描述符
block(i) {	
if ((i < 6 && (MbCBP & (1 << i))) || (i>=6 && (MbCBP422 & (1 << (i-6))))) {	
do {	
trans_coefficient	ce(v)|ae(v)
if (trans_coefficient >= 59 && aec_enable == '0')	
escape_level_diff	ce(v)
} while (trans_coefficient != 'EOB')	
}	
}	

7.2 语义描述

7.2.1 视频扩展

本部分定义了若干视频扩展，在语法的不同位置，可出现的视频扩展是不同的。每一种视频扩展都有一个唯一的视频扩展标号，见表28。

表28 视频扩展标号

视频扩展标号	含义
0000	保留
0001	保留
0010	序列显示扩展
0011	保留
0100	版权扩展
0101	保留
0110	保留
0111	图像显示扩展

表 28（续）

视频扩展标号	含义
1000 ～ 1010	保留
1011	摄像机参数扩展
1100 ～ 1111	保留

7.2.2 视频序列语义描述

7.2.2.1 视频序列

视频编辑码 video_edit_code

位串“0x000001B7”。说明紧跟video_edit_code的第一个I帧后续的B帧可能因缺少参考帧，不能正确解码。

视频序列结束码 video_sequence_end_code

位串“0x000001B1”。标识视频序列的结束。

7.2.2.2 序列头

视频序列起始码 video_sequence_start_code

位串“0x000001B0”。标识视频序列的开始。

类标号 profile_id

8位无符号整数。表示位流符合的类。

级标号 level_id

8位无符号整数。表示位流符合的级。

类和级见附录B。

逐行序列标志 progressive_sequence

标志。规定视频序列的扫描格式。值为“1”表示编码视频序列只包含逐行扫描的帧图像；值为“0”表示编码视频序列可包含逐行扫描和隔行扫描帧图像。

水平尺寸 horizontal_size

14位无符号整数。规定图像亮度分量可显示区域（该区域与图像的左侧边缘对齐）的宽度，即水平方向样本数。

以宏块为单位的MbWidth计算如下

$$MbWidth = (horizontal\_size + 15) / 16$$

horizontal_size不应为“0”。horizontal_size的单位应是图像每行样本数。可显示区域的左上角样本应与解码图像左上角样本对齐。

垂直尺寸 vertical_size

14位无符号整数。规定图像亮度分量可显示区域（该区域与图像的顶部边缘对齐）的高度，即垂直方向扫描行数。

在视频序列位流中，可能存在隔行扫描图像的两场编码数据依次出现时（即progressive_sequence的值为“0”时），以宏块为单位计算的MbHeight计算如下

$$MbHeight = 2 \times ((vertical\_size + 31) / 32)$$

在其他情况下，以宏块为单位计算的MbHeight计算如下

$$MbHeight = (vertical\_size + 15) / 16$$

vertical_size不应为0。vertical_size的单位应是图像样本的行数。

horizontal_size、vertical_size与图像边界的关系见图7。图7中，实线表示图像可显示区域边界，其宽度和高度分别由horizontal_size和vertical_size决定；虚线表示图像边界，其宽度和高度分别由MbWidth和MbHeight决定。例如horizontal_size的值为1920，vertical_size的值为1080，则MbWidth × 16等于1920，MbHeight × 16等于1088。

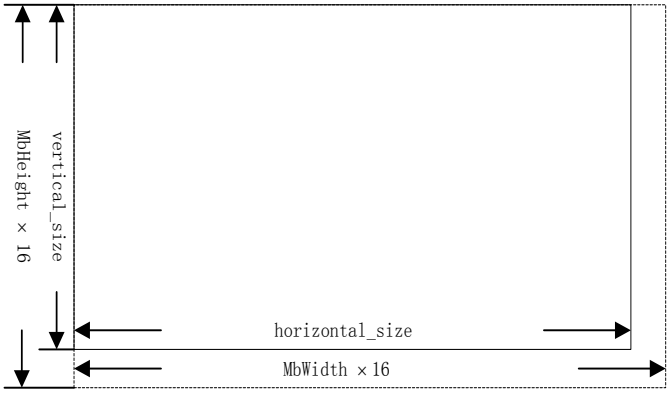


图7　图像边界示意图

色度格式　chroma_format

2位无符号整数。规定色度分量的格式，见表29。

表29　色度格式

chroma_format的值	含义
00	保留
01	4:2:0
10	4:2:2
11	保留

样本精度　sample_precision

3位无符号整数。规定亮度和色度样本的精度，见表30。

表30　样本精度

sample_precision的值	含义
000	禁止
001	亮度和色度均为8 bit精度
010 ～ 111	保留

宽高比　aspect_ratio

4位无符号整数。规定重建图像的样本宽高比（SAR）或显示宽高比（DAR）。见表31。

表31　宽高比

aspect_ratio的值	样本宽高比（SAR）	显示宽高比（DAR）
0000	禁止	禁止
0001	1.0	-
0010	-	4 ： 3
0011	-	16 ： 9
0100	-	2.21 ： 1
0101 ～ 1111	-	保留

如果位流中没有序列显示扩展，那么整个重建图像将要映射到整个活动显示区域。样本宽高比按下式计算。

$$SAR = DAR \times vertical\_size \div horizontal\_size$$

注：在这种情况下，horizontal_size和vertical_size受源图像的样本宽高比和选定的显示宽高比限制。

如果位流中有序列显示扩展出现，样本宽高比按下式计算。

$$SAR = DAR \times display\_vertical\_size \div display\_horizontal\_size$$

帧率代码　frame_rate_code

4位无符号整数。规定帧率，见表32。

表32　帧率代码

frame_rate_code的值	帧率
0000	禁止
0001	24000 ÷ 1001 (23.976...)
0010	24
0011	25
0100	30000 ÷ 1001 (29.97...)
0101	30
0110	50
0111	60000 ÷ 1001 (59.94...)
1000	60
1001 ～ 1111	保留

连续两帧之间的时间间隔是帧率的倒数。隔行扫描帧中两场之间的时间间隔是帧率的倒数的1/2。

比特率低位　bit_rate_lower

BitRate的低18位。

比特率高位　bit_rate_upper

BitRate的高12位。

BitRate = (bit_rate_upper << 18) + bit_rate_lower

BitRate以400bit/s为单位计算视频位流的比特率，并向上取整。BitRate不应为0。

低延迟　low_delay

标志。值为“1”说明视频序列不包含B帧，不存在图像重排序延时，位流中可能包含所谓“大图像”（见附录C）；值为“0”说明视频序列可包含B帧，存在图像重排序延时，位流中不包含所谓“大图像”（见附录C）。

位流缓冲区尺寸　bbv_buffer_size

18位无符号整数。规定了位流参考解码器对视频序列解码的位流缓冲区尺寸（见附录C）。BBS是位流参考解码器对视频序列解码所需的位流缓冲区最小尺寸（按位计算），其定义如下：

$$BBS = 16 \times 1024 \times bbv\_buffer\_size$$

7.2.2.3 扩展和用户数据

7.2.2.3.1 扩展数据

视频扩展起始码 extension_start_code

位串“0x000001B5”。标识视频扩展数据的开始。

视频扩展数据保留字节 reserved_extension_data_byte

8位无符号整数。保留位。解码器应丢弃这些数据。视频扩展数据保留字节中不应出现从任意字节对齐位置开始的21个以上连续的“0”。

7.2.2.3.2 用户数据

用户数据起始码 user_data_start_code

位串“0x000001B2”。标识用户数据的开始。用户数据连续存放，直到下一个起始码。

用户数据字节 user_data

8位整数。用户数据的含义由用户自行定义。用户数据中不应出现从任意字节对齐位置开始的21个以上连续的“0”。

7.2.2.4 序列显示扩展

本部分不定义显示过程。这一扩展中的信息对解码过程没有影响，解码器可忽略这些信息。

视频扩展标号 extension_id

位串“0010”。标识序列显示扩展。

视频格式 video_format

3位无符号整数。说明视频在按本部分进行编码之前的格式，见表33。如果位流中没有出现序列显示扩展，可假设视频格式为“未作规定的视频格式”。

表33 视频格式

video_format的值	含义
000	分量信号
001	PAL
010	NTSC
011	SECAM
100	MAC
101	未作规定的视频格式
110	保留
111	保留

样值范围 sample_range

标志。说明亮度和色度信号样值的范围。如果位流中没有出现序列显示扩展，设sample_range为“0”。

彩色信息描述 colour_description

标志。值为“1”说明位流中有colour_primaries、transfer_characteristics和matrix_coefficients；值为“0”说明位流中没有colour_primaries、transfer_characteristics和matrix_coefficients。

彩色三基色 colour_primaries

8位无符号整数。说明源图像三基色的色度坐标，见表34。

表34 彩色三基色

colour_primaries的值	彩色三基色			备注
0	禁止			
1	基色 绿 蓝 红 白 D65	x 0.300 0.150 0.640 0.3127	y 0.600 0.060 0.330 0.3290。	参见GY/T 155－2000
2	未作规定的视频 图像特性未知			
3	保留			
4	基色 绿 蓝 红 白 C	x 0.21 0.14 0.67 0.310	y 0.71 0.08 0.33 0.316。	参见ITU-R BT.470 2 System M
5	基色 绿 蓝 红 白 D65	x 0.29 0.15 0.64 0.313	y 0.60 0.06 0.33 0.329。	参见ITU-R BT.470 2 System B, G
6	基色 绿 蓝 红 白 D65	x 0.310 0.155 0.630 0.3127	y 0.595 0.070 0.340 0.3290。	参见SMPTE 170M
7	基色 绿 蓝 红 白 D65	x 0.310 0.155 0.630 0.3127	y 0.595 0.070 0.340 0.3291。	参见SMPTE 240M (1987)
8	普通胶片（彩色滤光镜，C光源） 基色 绿 蓝 红	x 0.243 0.145 0.681	y 0.692 (Wratten 58) 0.049 (Wratten 47) 0.319 (Wratten 25)。	
9 ～ 255	保留			

如果位流中没有序列显示扩展，或者colour_description的值是“0”，假设色度已由应用本身隐含定义。

光电转移特性 transfer_characteristics

8位无符号整数。说明源图像的光电转移特性，见表35。

表35 光电转移特性

transfer_characteristics的值	光电转移特性	备注
0	禁止	
1	$E' = 1.099\ L^{0.45} - 0.099$，$1.33 >= L >= 0.018$ $E' = 4.500\ L$，$0.018 > L >= -0.0045$ $E' = -\{1.099(-4L)^{0.45}-0.099\}/4$，$-0.0045 > L >= -0.25$。	参见GY/T 155—2000
2	未作规定的视频 图像特性未知	
3	保留	
4	假设显示伽玛值为2.2。	参见ITU-R BT.470 2 System M
5	假设显示伽玛值为2.8。	参见ITU-R BT.470-2 System B,G
6	$V = 1.099\ Lc^{0.45} - 0.099$，$1 >= Lc >= 0.018$ $V = 4.500\ Lc$，$0.018 > Lc >= 0$。	参见SMPTE 170M
7	$V = 1.1115\ Lc^{0.45} - 0.1115$，$Lc >= 0.0228$ $V = 4.0\ Lc$，$0.0228 > Lc$。	参见SMPTE 240M (1987)
8	线性转移特性 即 $V = L_c$	
9	对数转移特性 (范围100:1) $V = 1.0-(\log_{10}(Lc))/2$，$1 >= Lc >= 0.01$ $V = 0.0$，$0.01> Lc$	
10	对数转移特性 (范围316.22777:1) $V = 1.0-(\log_{10}(Lc))/2.5$，$1 >= Lc >= 0.0031622777$ $V = 0.0$，$0.0031622777> Lc$	
11 ～ 255	保留	

如果位流中没有序列显示扩展，或者colour_description的值是“0”，假设光电转移特性已由应用本身隐含定义。

彩色信号转换矩阵 matrix_coefficients

8位无符号整数。说明从红绿蓝三基色转换为亮度和色度信号时采用的转换矩阵，见表36。

表36　彩色信号转换矩阵

matrix_coefficients的值	彩色信号转换矩阵	备注
0	禁止	
1	$E_Y' = 0.2126E_R' + 0.7152\ E_G' + 0.0722E_B'$ $E_{CB}' = (E_B' - E_Y')/1.8556$ $E_{CR}' = (E_R' - E_Y')/1.5748$。	参见GY/T 155－2000
2	未作规定的视频 图像特性未知	
3	保留	
4	$E'_Y = 0.59E'_G + 0.11E'_B + 0.30E'_R$ $E'_{PB} = -0.331\ E'_G + 0.500\ E'_B - 0.169\ E'_R$ $E'_{PR} = -0.421E'_G - 0.079\ E'_B + 0.500\ E'_R$。	FCC
5	$E'_Y = 0.587\ E'_G + 0.114\ E'_B + 0.299\ E'_R$ $E'_{PB} = -0.331\ E'_G + 0.500\ E'_B - 0.169\ E'_R$ $E'_{PR} = -0.419\ E'_G - 0.081\ E'_B + 0.500\ E'_R$ 。	参见ITU-R BT.470-2 System B,G
6	$E'_Y = 0.587\ E'_G + 0.114\ E'_B + 0.299\ E'_R$ $E'_{PB} = -0.331\ E'_G + 0.500\ E'_B - 0.169\ E'_R$ $E'_{PR} = -0.419\ E'_G - 0.081\ E'_B + 0.500\ E'_R$ 。	参见SMPTE 170M
7	$E'_Y = 0.701\ E'_G + 0.087\ E'_B + 0.212\ E'_R$ $E'_{PB} = -0.384\ E'_G + 0.500\ E'_B - 0.116\ E'_R$ $E'_{PR} = -0.445\ E'_G - 0.055\ E'_B + 0.500\ E'_R$ 。	参见SMPTE 240M (1987)
8 ～ 255	保留	

在表36中：

——E'_Y是值在“0”和“1”之间的模拟量；

——E'_{PB}和E'_{PR}是值在“-0.5”和“0.5”之间的模拟量；

——E'_R、E'_G和E'_B是值在“0”和“1”之间的模拟量；

——Y、Cb和Cr与E'_Y、E'_{PB}和E'_{PR}的关系如下：

如果sample_range为“0”：

$$Y = (219 \times 2^{n-8} \times E'_Y) + 2^{n-4}$$

$$Cb = (224 \times 2^{n-8} \times E'_{PB}) + 2^{n-1}$$

$$Cr = (224 \times 2^{n-8} \times E'_{PR}) + 2^{n-1}$$

如果sample_range为“1”：

$$Y = ((2^n - 1) \times E'_Y)$$

$$Cb = ((2^n - 1) \times E'_{PB}) + 2^{n-1}$$

$$Cr = ((2^n - 1) \times E'_{PR}) + 2^{n-1}$$

其中n是样本点精度。例如：

n = 8，sample_range为“0”时：

$$Y = (219 \times E'_Y) + 16$$

$$Cb = (224 \times E'_{PB}) + 128$$

$$Cr = (224 \times E'_{PR}) + 128$$

Y的取值范围是16～235，Cb和Cr的取值范围是16～240。

n = 8，sample_range为“1”时：

$$Y = (255 \times E'_Y)$$

$$Cb = (255 \times E'_{PB}) + 128$$

$$Cr = (255 \times E'_{PR}) + 128$$

Y、Cb和Cr的取值范围都是0～255。

注1：本部分规定的解码过程将输出的Y、Cb和Cr的样值范围限制在0～2^n-1。如果位流中没有出现序列显示扩展，或者colour_description的值是“0”，假设转换矩阵已由应用本身隐含定义。

注2：某些应用可能有多个不同的视频信号，而不同的视频信号又可能具有不同的彩色三基色、转移特性和/或转换矩阵。在这种情况下建议应用首先将这些不同的参数集转换到一个统一的参数集。

水平显示尺寸　display_horizontal_size

垂直显示尺寸　display_vertical_size

display_horizontal_size和display_vertical_size都是14位无符号整数。它们共同定义了一个矩形，如果该矩形的尺寸比编码图像的尺寸小，宜只显示编码图像的一部分；如果该矩形的尺寸比编码图像的尺寸大，宜只在显示设备的一部分上显示重建图像。

display_horizontal_size的单位应是编码图像每行样本数。display_vertical_size的单位应是编码图像的行数。

display_horizontal_size和display_vertical_size对解码过程没有影响。它们可被显示过程使用。本部分不定义显示过程。

双视点拼接模式　stereo_packing_mode

2位无符号整数。规定双目立体视频的拼接方式，见表37。

表37　双视点拼接模式

stereo_packing_mode的值	含义
00	无拼接
01	左右拼接
10	上下拼接
11	保留

表37中无拼接表示当前视频即为通常的单目视频;左右拼接表示当前视频序列为左右拼接双目立体视频，在解码图像中，左视点的图像的像素均在右视点的图像的像素的左侧；上下拼接表示当前视频序列为上下拼接双目立体视频，在解码图像中，左视点的图像的像素均在右视点的图像的像素的上侧。

7.2.2.5　版权扩展

视频扩展标号　extension_id

位串“0100”。标识版权扩展。

版权标志　copyright_flag

标志。值为“1”说明该版权扩展定义的版权信息有效期直到下一个版权扩展或视频序列结束码；值为“0”说明该版权扩展没有定义版权信息。

版权信息由copyright_id和CopyrightNumber进一步说明。

版权标号　copyright_id

8位无符号整数。版权所有者的代码，由版权注册机构统一分配。如果为“0”，说明没有相关版权信息。

如果copyright_id的值为“0”，CopyrightNumber应为“0”。

如果copyright_flag的值为“0”，copyright_id应为“0”。

原创或拷贝　original_or_copy

标志。值为“1”说明源视频的内容是原创的；值为“0”说明源视频的内容是拷贝的。

版权号1 copyright_number_1

20位无符号整数。CopyrightNumber的第44到第63位。

版权号2 copyright_number_2

22位无符号整数。CopyrightNumber的第22到第43位。

版权号3 copyright_number_3

22位无符号整数。CopyrightNumber的第0到第21位。

CopyrightNumber是64位无符号整数，定义如下：

CopyrightNumber = (copyright_number_1 << 44)+
(copyright_number_2 << 22)+copyright_number_3

如果copyright_flag的值是“1”，CopyrightNumber和该版权扩展说明的源视频内容一一对应，CopyrightNumber为“0”说明没有相关信息；如果copyright_flag的值是“0”， CopyrightNumber也应为“0”。

7.2.2.6 摄像机参数扩展

视频扩展标号 extension_id

位串“1011”。标识摄像机参数扩展。

摄像机标号 camera_id

7位无符号整数。摄像机标识符。

图像设备高度 height_of_image_device

22位无符号整数。给出图像设备的高度，以0.001mm为单位，范围从0到4,194.303mm。

焦距 focal_length

22位无符号整数。给出摄像机的焦距，以0.001mm为单位，范围从0到4,194.303mm。

光圈 f_number

22位无符号整数。给出摄像机的光圈（光圈 = 焦距 ÷ 镜头的有效孔径），以0.001为单位，范围从0到4,194.303。

垂直视角 vertical_angle_of_view

22位无符号整数。给出由图像设备顶端和底端决定的垂直视角，以0.0001°为单位，范围从0°到180°。

摄像机坐标X高位，摄像机坐标Y高位，摄像机坐标Z高位 camera_position_x_upper, camera_position_y_upper, camera_position_z_upper

分别表示CameraPositionX，CameraPositionY和CameraPositionZ的高16位。

摄像机坐标X低位，摄像机坐标Y低位，摄像机坐标Z低位 camera_position_x_lower, camera_position_y_lower, camera_position_z_lower

分别表示CameraPositionX，CameraPositionY和CameraPositionZ的低16位。

CameraPositionX，CameraPositionY和CameraPositionZ是一组32位整数，用2的补码表示。说明摄像机光学原点在由用户定义的全局坐标系中的坐标值，每个坐标值都以0.001mm为单位，范围从-2,147,483.648mm到2,147,483.647mm。

摄像机方向矢量X，摄像机方向矢量Y，摄像机方向矢量Z camera_direction_x, camera_direction_y,camera_direction_z

一组22位整数，用2的补码表示。说明摄像机的方向，每个值范围从-2,097,152到2,097,151。摄像机的方向用从摄像机光学原点到摄像机前面位于摄像机光轴上的某点的矢量表示。

图像平面垂直矢量X，图像平面垂直矢量Y，图像平面垂直矢量Z image_plane_vertical_x, image_plane_vertical_y, image_plane_vertical_z

一组22位整数，用2的补码表示。说明摄像机向上的方向，每个值的范围从-2,097,152到2,097,151。摄像机向上的方向用平行于设备的边缘，方向从底到顶的矢量表示。

本条内容如图8和图9所示。

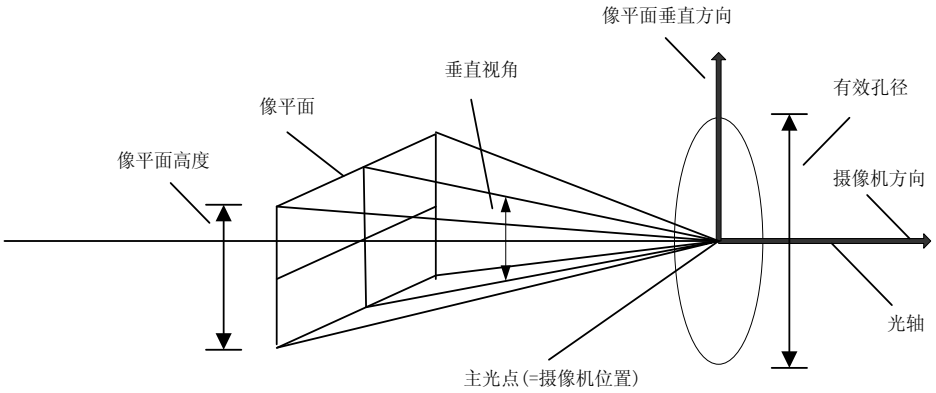


图8　摄像机原理示意图

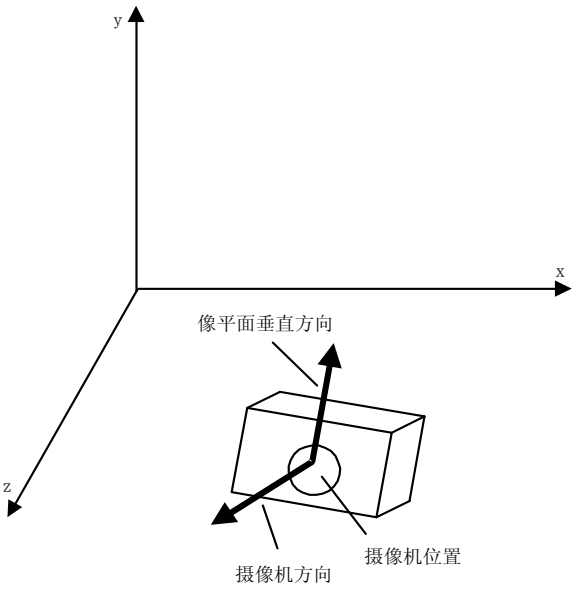


图9　摄像机坐标系统示意图

7.2.3　图像

7.2.3.1　I图像头

I图像起始码　i_picture_start_code

位串“0x000001B3”。标识I帧或G帧的开始。

BBV延时　bbv_delay

16位无符号整数。

BBV延迟扩展　bbv_delay_extension

7位无符号整数。

如果profile_id的值等于0x20，BbvDelay的值等于bbv_delay的值；否则，BbvDelay的值等于(bbv_delay << 7) + bbv_delay_extension。

如果profile_id的值等于0x20，则BbvDelayMax的值等于0xFFFF；否则，BbvDelayMax的值等于0x7FFFFF。

如果BbvDelay不等于BbvDelayMax,它规定了BBV从收到图像起始码的最后一个字节到开始解码图像之间要等待的时间。这个时间用从27 MHz系统时钟导出的90 kHz时钟周期数来表示。如果视频序列中某一帧图像的BbvDelay等于BbvDelayMax，那么整个视频序列中所有图像的BbvDelay均应等于BbvDelayMax。见附录C。

时间编码标志 time_code_flag

标志。值为“1”表示位流中包含time_code；值为“0”表示位流中没有time_code。

时间编码 time_code

24位位串，包括以下字段：DropFrameFlag，TimeCodeHours，TimeCodeMinutes，TimeCodeSeconds和TimeCodePictures，见表38。表中TimeCodeHours，TimeCodeMinutes，TimeCodeSeconds和TimeCodePictures字段是无符号整数。这些参数与IEC 60461标准《视频磁带录像机的时间和控制编码》中定义的参数相对应。time_code描述从当前帧开始（含当前帧）的位流中第一帧（显示顺序）的显示时间。

表38 时间编码（time_code）

time_code 的字段	取值	描述符
DropFrameFlag		u(1)
TimeCodeHours	0～23	u(5)
TimeCodeMinutes	0～59	u(6)
TimeCodeSeconds	0～59	u(6)
TimeCodePictures	0～59	u(6)

图像间距 picture_distance

8位无符号整数。picture_distance等于前一编码帧（显示顺序）的picture_distance加1，再加上当前帧和前一编码帧之间被跳过的图像帧数(视频序列起始码与随后第一个视频序列结束码或视频序列起始码与随后第一个视频编辑码之间，按显示顺序每两个连续编码帧之间被跳过的帧数均应小于32，按显示顺序每两个相邻的非双向帧间解码图像之间被跳过的帧数和B帧数之和均应小于127)，最后模256。

BBV检测次数 bbv_check_times

如果low_delay的值为“0”， 位流中不应出现bbv_check_times，此时BbvCheckTimes等于“0”。如果位流中出现bbv_check_times，由bbv_check_times解析得到BbvCheckTimes，解析过程见8.2。bbv_check_times的值应小于2^{16}-1。

检测BBV缓冲区的次数为BbvCheckTimes加1次，BbvCheckTimes大于“0”说明当前图像是一个大图像（见附录C）。

逐行帧标志 progressive_frame

标志。值为“0”表示该帧的两场是隔行场，这两场之间存在一个场时间间隔，在这种情况下，repeat_first_field的值应是“0”。

如果progressive_frame的值是“1”，表示该帧的两场实际上来自同一时刻，在这种情况下，PictureStructure的值应是“1”。

图像编码结构标志 picture_structure

标志。picture_structure的值为“0”表示当前图像的两场的编码数据依次出现；值为“1”表示当前图像的两场的编码数据合并成帧。如果progressive_sequence的值为“1”，则picture_structure的值也应为“1”。PictureStructure的值等于picture_structure。

顶场在先 top_field_first

标志。其含义由progressive_sequence、progressive_frame、picture_structure和repeat_first_field决定。

a）如果progressive_sequence的值是“0”，top_field_first说明解码场的输出显示顺序。

1） 如果PictureStructure的值是“0”，top_field_first的值是“1”,说明顶场的编码数据首先出现在位流中，顶场在底场之前输出显示；top_field_first的值是“0”，说明底场的编码数据首先出现在位流中，底场在顶场之前输出显示。

2） 如果PictureStructure的值是“1”，解码处理首先解码整帧。如果top_field_first的值是“1”则顶场在底场之前输出，top_field_first的值是“0”则底场在顶场之前输出。

b）如果progressive_sequence的值是“1”，top_field_first和repeat_first_field一起说明解码处理输出帧的次数（1次、2次或3次）。

1） 如果repeat_first_field的值是“0”，top_field_first的值也应是“0”，解码处理输出一个帧。

2） 如果top_field_first的值是“0”，repeat_first_field的值是“1”，解码处理输出两个完全一样的帧。

3） 如果top_field_first和repeat_first_field的值都是“1”，解码处理输出三个完全一样的帧。

如果profile_id的值是0x20，progressive_sequence值是“0”，并且PictureStructure的值是“0”，则top_field_first的值应是“1”。

如果profile_id的值不是0x20，progressive_sequence的值是“0”，则视频序列中所有图像的top_field_first的值应相同。

重复首场 repeat_first_field

标志。只在progressive_frame的值为“1”时起作用；否则repeat_first_field的值应为“0”。

a） 如果progressive_sequence和progressive_frame的值都是“0”，repeat_first_field的值也应是“0”，解码处理输出两个场，第一场（由top_field_first决定是顶场还是底场），后面跟着第二场。

b） 如果progressive_sequence的值是“0”，progressive_frame的值是“1”，那么：

1） 如果repeat_first_field的值是“0”,解码处理输出两个场,第一场(由top_field_first决定是顶场还是底场），后面跟着第二场。

2） 如果repeat_first_field的值是“1”,解码处理输出三个场,第一场(由top_field_first决定是顶场还是底场），后面跟着第二场，最后重复输出第一场。

c） 如果progressive_sequence的值是“1”，那么：

1） 如果repeat_first_field的值是“0”，解码处理输出一帧。

2） 如果repeat_first_field的值是“1”，解码处理输出两或三帧，由top_field_first决定。

固定图像量化因子 fixed_picture_qp

标志。值为“1”说明在该帧图像内量化因子不变；“0”说明在该帧图像内量化因子可变。

图像量化因子 picture_qp

6位无符号整数。给出图像的量化因子。量化因子取值范围是0～63。

宏块跳过模式标志　skip_mode_flag

标志。值为“1”表示宏块跳过模式使用游程编码；值为“0”表示宏块跳过模式的编码由mb_type决定，见9.4.1。

环路滤波禁用标志　loop_filter_disable

标志。值为“1”表示不应使用环路滤波；值为“0”表示应使用环路滤波。

环路滤波参数标志　loop_filter_parameter_flag

标志。值为“1”表示位流中包含alpha_c_offset和beta_offset；值为“0”表示位流中没有alpha_c_offset和beta_offset。

α和C索引的偏移　alpha_c_offset

当前图像环路滤波α和C索引的偏移，alpha_c_offset取值范围是-8～8，环路滤波参数AlphaCOffset等于alpha_c_offset。如果位流中没有alpha_c_offset，AlphaCOffset等于“0”。

β索引的偏移　beta_offset

当前图像环路滤波β索引的偏移，beta_offset取值范围为是-8～8，环路滤波参数BetaOffset等于beta_offset。如果位流中没有beta_offset，BetaOffset等于“0”。

加权量化标志　weighting_quant_flag

标志。值为“1”表示应使用加权量化；值为“0”表示不应使用加权量化。如果当前图像的图像头不存在weighting_quant_flag，则weighting_quant_flag的值为“0”。

色度量化参数禁用标志　chroma_quant_param_disable

标志。值为“1”表示当前图像的图像头中不存在chroma_quant_param_delta_cb和chroma_quant_param_delta_cr；值为“0”表示当前图像的图像头中存在chroma_quant_param_delta_cb和chroma_quant_param_delta_cr。

色度量化参数增量Cb　chroma_quant_param_delta_cb

色度量化参数增量Cr　chroma_quant_param_delta_cr

色度块量化参数相对于CurrentQP的增量，取值范围-16～16。解析过程见8.2，解码过程见9.6.1。如果当前图像的图像头中不存在chroma_quant_param_delta_cb和chroma_quant_param_delta_cr，则chroma_delta_param_u和chroma_delta_param_v的值均为“0”。

加权量化参数索引　weighting_quant_param_index

2位无符号整数。当前图像的加权量化参数索引。值为“11”保留。

加权量化矩阵模型　weighting_quant_model

2位无符号整数，规定加权量化参数的分布模型。值为“11”保留。

加权量化参数增量1　weighting_quant_param_delta1[i]

加权量化参数增量2　weighting_quant_param_delta2[i]

当前图像的加权量化参数的增量，取值范围是-128～127。解析过程见8.2。解码过程见9.2。如果当前图像的图像头中不存在weighting_quant_param_delta1[i]或weighting_quant_param_delta2[i]，则weighting_quant_param_delta1[i]或weighting_quant_param_delta2[i]的值为0。

高级熵编码允许标志　aec_enable

标志。值为“0”表示语法元素使用基本熵编码；值为“1”表示语法元素使用高级熵编码。如果当前图像的图像头中没有aec_enable，则aec_enable的值为“0”。

7.2.3.2　PB图像头

PB图像起始码　pb_picture_start_code

位串“0x000001B6”。标识P帧或B帧的开始。

图像编码方式　picture_coding_type

2位无符号整数。规定图像的编码方式，见表39。

表39　图像编码方式

picture_coding_type的值	编码方式
00	禁止
01	前向预测编码（P）
10	双向预测编码（B）
11	保留

高级预测模式禁用标志　advanced_pred_mode_disable

标志。值应为“1”，说明禁止使用高级预测模式。值为“0”保留。

图像参考标志　picture_reference_flag

标志。值为“1”表示所有宏块都使用缺省参考图像；值为“0”表示由每个宏块在宏块层自行确定参考图像。确定参考图像的方法见9.4.5。如果当前图像的图像头中没有picture_reference_flag，则picture_reference_flag的值为“1”。

无前向参考标志　no_forward_reference_flag

标志。值为“1”表示当前图像不应参考前向参考图像；值为“0”表示当前图像可参考前向参考图像。

PB帧场编码预测标志　pb_field_enhanced_flag

标志。如果profile_id的值等于0x48，则PFieldSkip和BFieldEnhanced的值等于pb_field_enhanced_flag的值；否则，PFieldSkip和BFieldEnhanced的值均应为0。

7.2.3.3　图像显示扩展

本部分不定义显示过程。这一扩展中的信息对解码处理没有影响，解码器可忽略这些信息。

图像显示扩展允许显示矩形（其尺寸由序列显示扩展定义）按图像移动。其中一项应用是实现全景扫描。

视频扩展标号　extension_id

位串“0111”。标识图像显示扩展。

帧中心水平偏移　frame_centre_horizontal_offset

16位整数。以1/16样本为单位给出水平偏移。正值表示重建图像的中心位置在显示矩形中心的右侧。

帧中心垂直偏移　frame_centre_vertical_offset

16位整数。以1/16样本为单位给出垂直偏移。正值表示重建图像的中心位置在显示矩形中心的下方。

显示矩形区域的尺寸在序列显示扩展中定义。编码图像内区域的坐标由图像显示扩展定义。

重建图像的中心指由horizontal_size和vertical_size定义的矩形的中心。

在隔行序列中，一幅编码图像可能与一个、两个或三个解码场有关，因此图像显示扩展最多可以定义三组偏移量。

7.1.3.3中NumberOfFrameCentreOffsets的值按以下方式定义：

```
if ( progressive_sequence == '1' ) {
    if ( repeat_first_field == '1' ) {
        if ( top_field_first == '1' )
            NumberOfFrameCentreOffsets = 3
        else
            NumberOfFrameCentreOffsets = 2
```

```
        } else {
            NumberOfFrameCentreOffsets = 1
        }
    } else {
        if ( picture_structure == '0' ) {
            NumberOfFrameCentreOffsets = 1
        } else {
            if ( repeat_first_field == '1' )
                NumberOfFrameCentreOffsets = 3
            else
                NumberOfFrameCentreOffsets = 2
        }
    }
```

如果前面的序列头之后没有序列显示扩展，那么位流中不应出现图像显示扩展。

如果一帧图像没有图像显示扩展，使用最近的解码图像的中心偏移量。注意：丢失帧的中心偏移量的值与前一非丢失帧的值相同。在序列头后，所有图像的中心偏移量置为0，直到出现图像显示扩展。

利用图像中心偏移量可定义一个矩形区域，这个区域在整个重建图像范围内移动来实现全景扫描。

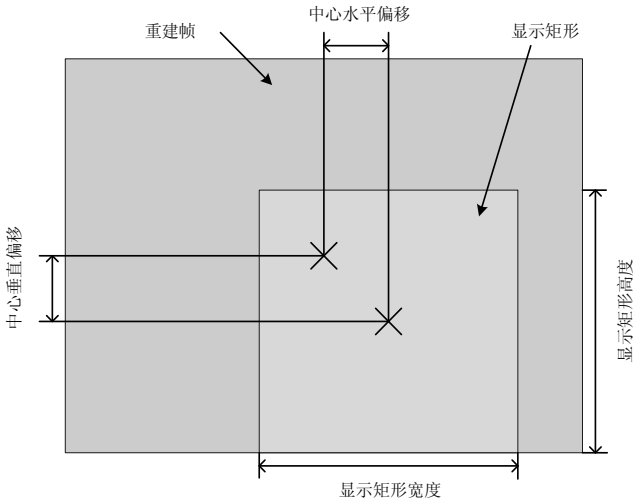


图10　帧中心偏移量参数

图像中心偏移量参数见图10。

注1：显示矩形的尺寸可能大于重建图像。

注2：在场图像中，frame_centre_vertical_offset 表示的中心偏移量以 1/16 帧行为单位。

注3：图 10 中，frame_centre_horizontal_offset 和 frame_centre_vertical_offset 均为负值。

7.2.4　条带

条带起始码　slice_start_code

32位位串，前24位是0x000001。后8位为slice_vertical_position，其范围是0x00～0xAF。

条带垂直位置　slice_vertical_position

8位无符号整数，给出条带的第一个宏块在图像中的垂直位置MbRow，以宏块为单位。slice_vertical_position的值的范围应为0x00～0xAF。

如果图像的vertical_size大于2800，MbRow由slice_vertical_position和slice_vertical_position_extension决定。

条带垂直位置扩展 slice_vertical_position_extension

3位无符号整数，如果图像的vertical_size小于或等于2800，位流中不应出现slice_vertical_position_extension。

MbRow按下面的方法计算：

```
if ( vertical_size > 2800 )
    MbRow = ( slice_vertical_position_extension << 7 ) + slice_vertical_position
else
    MbRow = slice_vertical_position
```

固定条带量化因子 fixed_slice_qp

标志。值为“1”说明在该条带内量化因子不变；值为“0”说明在该条带内量化因子可变。

条带量化因子 slice_qp

6位无符号整数。给出条带的量化因子。量化因子取值范围是0～63。

条带加权预测标志 slice_weighting_flag

标志。值为“1”表示宏块的运动补偿可使用加权预测；值为“0”表示宏块的运动补偿不应使用加权预测。

亮度缩放参数 luma_scale

8位无符号整数，表示预测亮度时的缩放参数。

亮度平移参数 luma_shift

8位有符号整数，表示预测亮度时的平移参数。

色度缩放参数 chroma_scale

8位无符号整数，表示预测色度时的缩放参数。

色度平移参数 chroma_shift

8位有符号整数，表示预测色度时的平移参数。

宏块加权预测标志 mb_weighting_flag

标志。mb_weighting_flag的值为“0”表示所有非帧内预测宏块都应采用加权运动补偿；值为“1”表示每个非帧内预测宏块由WeightingPrediction决定是否采用加权预测。

高级熵编码字节对齐填充位 aec_byte_alignment_bit

填充位。值应为“1”。

跳过宏块计数 mb_skip_run

跳过宏块计数。如果aec_enable的值为“0”，解析过程见8.2；如果aec_enable的值为“1”，解析过程见8.4。解码过程见9.3。

高级熵编码宏块填充位 aec_mb_stuffing_bit

填充位。条带的最后一个宏块的aec_mb_stuffing_bit的值应为“1”，解析过程见8.4。

7.2.5 宏块

宏块类型 mb_type

宏块的类型，其语义由图像类型、PictureStructure和skip_mode_flag决定。如果aec_enable的值为“0”，解析过程见8.2；如果aec_enable的值为“1”，解析过程见8.4。解码过程见9.4.2。

宏块子类型 mb_part_type

2位无符号整数。宏块的子类型。如果aec_enable的值为“1”，解析过程见8.4。解码过程见9.4.2。

预测模式标志 pred_mode_flag

标志。值为“1”表示根据预测模式的预测值确定帧内亮度预测模式；值为“0”表示根据intra_luma_pred_mode确定帧内亮度预测模式。

帧内亮度预测模式　intra_luma_pred_mode

2位无符号整数。用于确定亮度块的帧内预测模式。如果aec_enable的值为“1”，解析过程见8.4。解码过程见9.4.4。

帧内色度预测模式　intra_chroma_pred_mode

用于确定4:2:0和4:2:2格式下宏块中顺序号为4、5的两个色度块的帧内预测模式。如果aec_enable的值为“0”，解析过程见8.2；如果aec_enable的值为“1”，解析过程见8.4。解码过程见9.4.4。

4:2:2帧内色度预测模式　intra_chroma_pred_mode_422

用于确定4:2:2格式下宏块中顺序号为6、7的两个色度块的帧内预测模式。解析过程见8.2。

宏块参考索引　mb_reference_index

如果aec_enable的值为“1”，解析过程见8.4。

如果PictureStructure的值为0并且PictureType的值为1，mb_reference_index是2位无符号整数；否则，mb_reference_index是1位无符号整数。

解码过程见9.4.5。先解码全部前向参考索引，然后解码全部后向参考索引。

运动矢量水平分量差值　mv_diff_x

运动矢量垂直分量差值　mv_diff_y

运动矢量差值。单位为1/4样本，取值范围为-4096～4095（按亮度像素样本为-1024～1023.75）。先解码全部前向运动矢量，然后解码全部后向运动矢量。如果aec_enable的值为“0”，解析过程见8.2；如果aec_enable的值为“1”，解析过程见8.4。解码过程见9.4.6。

加权预测　weighting_prediction

标志。如果aec_enable的值为“1”，解析过程见8.4。

宏块编码模板　cbp

cbp确定宏块中顺序号为0到5的8×8亮度块和色度块是否包含编码数据。由cbp得到一个6位无符号整数MbCBP。

如果aec_enable的值为“0”，cbp的解析过程见8.2；如果aec_enable的值为“1”，cbp的解析过程见8.4。MbCBP的解码过程见9.4.7。

4:2:2 宏块编码模板　cbp_422

cbp_422确定4:2:2格式下宏块中顺序号为6、7的两个8×8色度块是否包含编码数据。由cbp_422得到一个2位无符号整数MbCBP422。如果aec_enable的值为“0”,cbp_422的解析过程见8.2；如果aec_enable的值为“1”，cbp_422的解析过程见8.4。MbCBP422的解码过程见9.4.7。

宏块量化参数增量　mb_qp_delta

给出当前宏块的量化参数相对预测量化参数的增量。如果aec_enable的值为“0”，解析过程见8.2；如果aec_enable的值为“1”，解析过程见8.4，解码过程见9.4.8。

7.2.6　块

变换系数　trans_coefficient

如果aec_enable的值为“0”，用于确定游程长度和非零量化系数值的联合索引，或用于确定转逸游程长度和转逸系数值的符号，解析过程见8.3，解码过程见9.5.1。

如果aec_enable的值为“1”，用于确定游程长度和非零量化系数值，解析过程见8.4，解码过程见9.5.2。

转逸系数差值　escape_level_diff

escape_level_diff用于确定转逸系数的绝对值，解析过程见8.3，解码过程见9.5.1。

8 解析过程

8.1 k 阶指数哥伦布码

解析k阶指数哥伦布码时，首先从位流的当前位置开始寻找第一个非零位，并将找到的零位个数记为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40给出了0阶、1阶、2阶和3阶指数哥伦布码的结构。指数哥伦布码的位串分为“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个位构成，即表中的x_i串，x_i的值为“0”或“1”。

表40 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 x_0	1～2
	0 0 1 x_1 x_0	3～6
	0 0 0 1 x_2 x_1 x_0	7～14
	.....	.....
k = 1	1 x_0	0～1
	0 1 x_1 x_0	2～5
	0 0 1 x_2 x_1 x_0	6～13
	0 0 0 1 x_3 x_2 x_1 x_0	14～29
	.....	.....
k = 2	1 x_1 x_0	0～3
	0 1 x_2 x_1 x_0	4～11
	0 0 1 x_3 x_2 x_1 x_0	12～27
	0 0 0 1 x_4 x_3 x_2 x_1 x_0	28～59
	.....	.....
k = 3	1 x_2 x_1 x_0	0～7
	0 1 x_3 x_2 x_1 x_0	8～23
	0 0 1 x_4 x_3 x_2 x_1 x_0	24～55
	0 0 0 1 x_5 x_4 x_3 x_2 x_1 x_0	56～119
	.....	.....

8.2 ue(v)、se(v)和 me(v)的解析过程

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

——ue(v)：语法元素的值等于 CodeNum；

——se(v)：根据表 41 给出的有符号指数哥伦布码的映射关系求语法元素的值；

——me(v)：分别根据表 42 和表 43 求 MbCBP 和 MbCBP422 的值（见 9.4.1 和 9.4.7）；

表41　se(v)与 CodeNum 的映射关系

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

表42　MbCBP 与 CodeNum 的映射关系

CodeNum值	MbCBP值 xxxxxx (543210)	
	宏块帧内编码模式	宏块帧间编码模式
0	63	0
1	15	15
2	31	63
3	47	31
4	0	16
5	14	32
6	13	47
7	11	13
8	7	14
9	5	11
10	10	12
11	8	5
12	12	10
13	61	7
14	4	48
15	55	3
16	1	2
17	2	8
18	59	4
19	3	1
20	62	61
21	9	55
22	6	59
23	29	62

表 42（续）

CodeNum值	MbCBP值 xxxxxx (543210)	
	宏块帧内编码模式	宏块帧间编码模式
24	45	29
25	51	27
26	23	23
27	39	19
28	27	30
29	46	28
30	53	9
31	30	6
32	43	60
33	37	21
34	60	44
35	16	26
36	21	51
37	28	35
38	19	18
39	35	20
40	42	24
41	26	53
42	44	17
43	32	37
44	58	39
45	24	45
46	20	58
47	17	43
48	18	42
49	48	46
50	22	36
51	33	33
52	25	34
53	49	40
54	40	52
55	36	49
56	34	50
57	50	56
58	52	25

表 42（续）

CodeNum值	MbCBP值 xxxxxx (543210)	
	宏块帧内编码模式	宏块帧间编码模式
59	54	22
60	41	54
61	56	57
62	38	41
63	57	38

表43　MbCBP422 与 CodeNum 的映射关系

CodeNum值	MbCBP422值 xx (10)	
	宏块帧内编码模式	宏块帧间编码模式
0	0	0
1	1	1
2	2	2
3	3	3

8.3　ce(v)的解析过程

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

——帧内编码块亮度系数的 escape_level_diff 采用 1 阶指数哥伦布码。

——帧间编码块亮度系数的 escape_level_diff 采用 0 阶指数哥伦布码。

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

——本部分定义了 19 个与 ce(v)有关的基本熵编码表，即 VLC0_Intra，VLC1_Intra，VLC2_Intra，VLC3_Intra，VLC4_Intra，VLC5_Intra，VLC6_Intra，VLC0_Inter，VLC1_Inter，VLC2_Inter，VLC3_Inter，VLC4_Inter，VLC5_Inter，VLC6_Inter，VLC0_Chroma，VLC1_Chroma，VLC2_Chroma，VLC3_Chroma，VLC4_Chroma，见附录 D。不同的码表决定了 ce(v)所用的指数哥伦布码的阶数。其中 VLC0_Inter 采用 3 阶指数哥伦布码，VLC2_Chroma 及 VLC3_Chroma 采用 1 阶指数哥伦布码，VLC1_Chroma 及 VLC4_Chroma 采用 0 阶指数哥伦布码，其他码表均采用 2 阶指数哥伦布码。

本条中上述19个码表的选择按照以下规则进行：

a)　编码块第 1 个解码量化系数的码表选择：

1)　帧内预测编码块的亮度系数，CurrentVLCTable = VLC0_Intra，见表 D.1。

2)　帧间预测编码块的亮度系数，CurrentVLCTable = VLC0_Inter，见表 D.8。

3)　色度系数，CurrentVLCTable = VLC0_Chroma，见表 D.15。

b)　其他解码量化系数的码表选择：

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

——如果 absLevel 等于 1，选择 CurrentVLCTable = VLC1_Intra，见表 D.2。

——如果 absLevel 等于 2，选择 CurrentVLCTable = VLC2_Intra，见表 D.3。
——如果 absLevel 等于 3 或 4，选择 CurrentVLCTable = VLC3_Intra，见表 D.4。
——如果 absLevel 等于 5、6 或 7，选择 CurrentVLCTable = VLC4_Intra，见表 D.5。
——如果 absLevel 等于 8、9 或 10，选择 CurrentVLCTable = VLC5_Intra，见表 D.6。
——如果 absLevel 大于 10，选择 CurrentVLCTable = VLC6_Intra，见表 D.7。

2) 对于帧间预测编码块的亮度系数，如果 absLevel 大于 maxAbsLevel，按如下方式进行码表切换：
——如果 absLevel 等于 1，选择 CurrentVLCTable = VLC1_Inter，见表 D.9。
——如果 absLevel 等于 2，选择 CurrentVLCTable = VLC2_Inter，见表 D.10。
——如果 absLevel 等于 3，选择 CurrentVLCTable = VLC3_Inter，见表 D.11。
——如果 absLevel 等于 4、5 或 6，选择 CurrentVLCTable = VLC4_Inter，见表 D.12。
——如果 absLevel 等于 7、8 或 9，选择 CurrentVLCTable = VLC5_Inter，见表 D.13。
——如果 absLevel 大于 9，选择 CurrentVLCTable = VLC6_Inter，见表 D.14。

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

ce(v)描述的语法元素解析过程如下：

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

8.4 ae(v)的解析过程

8.4.1 概述

ae(v)描述的语法元素解析过程如下：

a) 对条带进行解析前，首先进行初始化，见 8.4.2。
b) 对语法元素做二值化，见 8.4.3。
c) 对二值化得到的二元符号串进行解析，见 8.4.4。
   1) 二元符号串中每个二元符号的索引号是 binIdx，对应唯一的 ctxIdx，见 8.4.4.2；
   2) 根据 ctxIdx 解析二元符号，见 8.4.4.3；
   3) 完成一个二元符号的解析后，将得到的二元符号串与二值化过程得到的二元符号串集合进行比较。如果得到的二元符号串与集合中某个二元符号串相匹配，则输出相应语法元素值；否则继续解析二元符号。

上述解析过程用伪代码描述如下：

```
if ( 当前语法元素是条带的第1个语法元素 ) {
    初始化所有上下文模型
    初始化高级熵编码解码器
}
语法元素二值化
binIdx = -1
```

```
do {
    binIdx++
    得到与binIdx对应的ctxIdx
    得到与ctxIdx对应的上下文模型ctx
    根据ctx解析二元符号
} while ( ($b_0$,...,$b_{binIdx}$)不是语法元素的二元符号串 )
输出语法元素值
```

8.4.2 初始化

8.4.2.1 初始化上下文模型

每个上下文模型ctx需要初始化三个变量mps、cycno和lgPmps。mps的位宽为1位，cycno的位宽为2位，lgPmps的位宽为11位。mps和cycno的值应初始化为“0”；lgPmps的值应初始化为“1023”。

8.4.2.2 初始化高级熵编码解码器

rS1、rT1、valueS和valueT是用于高级熵编码解码器的变量。rS1、rT1的位宽是8位，valueS的位宽是32位，valueT的位宽是9位。rS1的值应初始化为“0”；rT1的值应初始化为“0xFF”。

valueS和valueT的初始化过程用伪代码描述如下：

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

8.4.3 二值化

不同语法元素的二值化过程如下：

a） mb_skip_run、mb_qp_delta、P 帧的 mb_reference_index 和 P 帧的 mb_type 的二值化过程：语法元素的值为 synElVal，synElVal 与二元符号串的关系见表 44。

表44 ynElVal 与二元符号串的关系

synElVal的值	二元符号串					
0	1					
1	0	1				
2	0	0	1			
3	0	0	0	1		
4	0	0	0	0	1	
5	0	0	0	0	0	1
…						
binIdx的值	0	1	2	3	4	5

b） B 帧的 mb_type 的二值化过程：语法元素的值为 synElVal，synElVal 与二元符号串的关系见表 45。

表45 synElVal 与二元符号串的关系

synElVal的值	二元符号串					
0	0					
1	1	1				
2	1	0	1			
3	1	0	0	1		
4	1	0	0	0	1	
5	1	0	0	0	0	1
…						
binIdx的值	0	1	2	3	4	5

c） mb_part_type 的二值化过程：语法元素的值为 synElVal，synElVal 与二元符号串的关系见表 46。

表46 synElVal 与二元符号串的关系

synElVal的值	二元符号串	
0	0	0
1	0	1
2	1	0
3	1	1
binIdx的值	0	1

d） intra_luma_pred_mode 的二值化过程：语法元素的值为 synElVal，synElVal 与二元符号串的关系见表 47。

表47 synElVal 与二元符号串的关系

synElVal的值	二元符号串			
0	1			
1	0	1		
2	0	0	1	
3	0	0	0	1
4	0	0	0	0
binIdx的值	0	1	2	3

e） intra_chroma_pred_mode 的二值化过程：语法元素的值为 synElVal，synElVal 与二元符号串的关系见表 48。

表48 synElVal 与二元符号串的关系

synElVal的值	二元符号串		
0	0		
1	1	0	
2	1	1	0
3	1	1	1
binIdx的值	0	1	2

f） mv_diff_x 和 mv_diff_y 的二值化过程:语法元素分为绝对值 mvdAbs 和符号位 mvdSign 两部分，先解析 mvdAbs，再解析 mvdSign。mvdAbs 的值为 synElVal，mvdSign 的解析过程见 8.4.4。synElVal 与二元符号串的关系见表 49。表 49 中,如果 synElVal 的值大于或等于 3 并且 synElVal 的值为奇数，二元符号串的前四位为“1110”，后续位为(synElVal-3)/2 对应的 0 阶指数哥伦布码（见表 40）；如果 synElVal 的值大于 3 并且 synElVal 的值为偶数，二元符号串的前四位为“1111”，后续位为(synElVal-3)/2 对应的 0 阶指数哥伦布码。如果 mvdAbs 为“0”，此时不应解析 mvdSign。mvdSign 的值为“0”表示 mvdAbs 的值为 synElVal；mvdSign 的值为 1 表示 mvdAbs 的值为-synElVal。

表49 synElVal 与二元符号串的关系

synElVal的值	二元符号串										
0	0										
1	1	0									
2	1	1	0								
3	1	1	1	0	1						
4	1	1	1	1	1						
5	1	1	1	0	0	1	0				
6	1	1	1	1	0	1	0				
7	1	1	1	0	0	1	1				
8	1	1	1	1	0	1	1				
9	1	1	1	0	0	0	1	0	0		
10	1	1	1	1	0	0	1	0	0		
11	1	1	1	0	0	0	1	0	1		
12	1	1	1	1	0	0	1	0	1		
13	1	1	1	0	0	0	1	1	0		
14	1	1	1	1	0	0	1	1	0		
…											
binIdx的值	0	1	2	3	4	5	6	7	8	9	10

g） cbp 的二值化过程:语法元素的值为 synElVal,如果 synElVal 小于 16,二元符号串为 synElVal 的四位前缀,后缀为“0”；如果 synElVal 大于 15 并且小于或等于 31,二元符号串为 synElVal 的四位前缀，后续位为“100”；如果 synElVal 大于 31 并且小于或等于 47，二元符号串为 synElVal 的四位前缀，后续位为“101”；如果 synElVal 大于 47，二元符号串为 synElVal 的四位前缀，后续位为“11”。synElVal 与二元符号串的关系见表 50，MbCBP 的值等于 synElVal 的值。

表50 synElVal 与二元符号串的关系

synElVal的值	二元符号串						
	四位前缀				后缀		
1	1	0	0	0	0		
2	0	1	0	0	0		
3	1	1	0	0	0		
…							
15	1	1	1	1	0		
16	0	0	0	0	1	0	0
17	1	0	0	0	1	0	0
…							
31	1	1	1	1	1	0	0
32	0	0	0	0	1	0	1
33	1	0	0	0	1	0	1
…							
47	1	1	1	1	1	0	1
48	0	0	0	0	1	1	
49	1	0	0	0	1	1	
…							
63	1	1	1	1	1	1	
binIdx的值	0	1	2	3	4	5	6

h） cbp_422 的二值化过程：语法元素的值为 synElVal。如果二元符号串为“00”，synElVal 的值为 0。如果二元符号串为“01”，synElVal 的值为 1。如果二元符号串为“10”，synElVal 的值为 2。如果二元符号串为“11”，synElVal 的值为 3。MbCBP422 的值等于 synElVal 的值。

i） trans_coefficient 的二值化过程：语法元素可分为 coeffLevel、coeffSign 和 coeffRun 三部分。首先由 trans_coefficient 得到 coeffLevel，如果是块的第一个 coeffLevel 或者 coeffLevel 的值不等于“0”，则继续得到 coeffSign 和 coeffRun。AbsLevel 的值等于根据 coeffLevel 查表 44 得到的 synElVal。coeffSign 是一个二元符号“0”或“1”。RunVal 的值等于根据 coeffRun 查表 44 得到的 synElVal。

j） B 帧的 mb_reference_index 的二值化过程：语法元素的值为 synElVal。如果二元符号串为“0”，synElVal 的值为“1”。如果二元符号串为“1”，synElVal 的值为“0”。

k） weighting_prediction 的二值化过程：语法元素的值为 synElVal。如果二元符号串为“0”，synElVal 的值为“0”。如果二元符号串为“1”，synElVal 的值为“1”。

8.4.4 二元符号串解析

8.4.4.1 概述

对每个二元符号，置contextWeighting的值为“0”。如果二元符号是mvdSign或coeffSign，或该二元符号属于mv_diff_x和mv_diff_y的0阶指数哥伦布码部分，或该二元符号属于cbp_422，则BypassFlag的值为“1”；否则BypassFlag的值为“0”。解析二元符号时，如果BypassFlag的值为“0”，

则先由binIdx确定ctxIdx（见8.4.4.2），然后再解析二元符号（见8.4.4.3）；否则，直接解析二元符号（见8.4.4.3）。

在解析二元符号串的过程中，binIdx从0开始，每解析一个二元符号，binIdx加1，并且将得到的二元符号串与表44到表50中相应的二元符号串比对，得到synElVal。

8.4.4.2 确定 ctxIdx

ctxIdx由起始索引号和ctxIdxInc决定。语法元素的起始索引号见表51。

表51 语法元素起始索引号

语法元素	类型	起始索引号
mb_skip_run		0
mb_type		4
mb_part_type		19
intra_luma_pred_mode		22
intra_chroma_pred_mode		26
mb_reference_index		30
mv_diff_x		36
mv_diff_y		42
cbp		48
mb_qp_delta		54
trans_coefficient	帧模式亮度	58
	帧模式色度	124
	场模式亮度	190
	场模式色度	256
weighting_prediction		322

确定各种语法元素的ctxIdx的步骤如下：

a) 确定 mb_skip_run 的 ctxIdx：

1) 第一步，根据 binIdx 得到 ctxIdxInc：

ctxIdxInc = Min(binIdx, 3)

2) 第二步，计算 ctxIdx：

ctxIdx = 起始索引号 + ctxIdxInc

b) 确定 mb_type 的 ctxIdx：

1) 第一步，根据 binIdx 得到 ctxIdxInc：

——如果当前帧是 P 帧，则：

ctxIdxInc = Min(binIdx, 4)

——否则，如果当前帧是 B 帧，并且 binIdx 等于 0，则：

ctxIdxInc = 5 + a + b

——否则，如果当前帧是 B 帧，并且 binIdx 小于或等于 7，则：

ctxIdxInc = 7 + binIdx

——否则，如果当前帧是 B 帧，并且 binIdx 大于 7，则：

ctxIdxInc = 14

2) 第二步，计算 ctxIdx：

ctxIdx = 起始索引号 + ctxIdxInc

如果当前块 E 的左边块 A（或上边块 B）“可用”，并且块 A（或块 B）的宏块类型不是 P_Skip、B_Skip 或 B_Direct_16x16，则 a（或 b）的值为“1”；否则 a（或 b）的值为“0”。块 E 和块 A、块 B 的关系见 9.4.3。

c) 确定 mb_part_type 的 ctxIdx：

1) 第一步，根据 binIdx 得到 ctxIdxInc：

——如果 binIdx 等于 0，则：

ctxIdxInc = 0

——否则，如果二元符号串的第一个二元符号为“0”，则：

ctxIdxInc = 1

——否则：

ctxIdxInc = 2

2) 第二步，计算 ctxIdx：

ctxIdx = 起始索引号 + ctxIdxInc

d) 确定 intra_luma_pred_mode 的 ctxIdx：

1) 第一步，根据 binIdx 得到 ctxIdxInc：

ctxIdxInc = Min(binIdx, 3)

2) 第二步，计算 ctxIdx：

ctxIdx = 起始索引号 + ctxIdxInc

e) 确定 intra_chroma_pred_mode 的 ctxIdx：

1) 第一步，根据 binIdx 得到 ctxIdxInc：

——如果 binIdx 等于 0，则：

ctxIdxInc = a + b

——否则：

ctxIdxInc = 3

2) 第二步，计算 ctxIdx：

ctxIdx = 起始索引号 + ctxIdxInc

如果当前块 E 的左边块 A（或上边块 B）“可用”，并且块 A（或块 B）的预测模式不是 Intra_Chroma_DC，则 a（或 b）的值为 1；否则 a（或 b）的值为“0”。块 E 和块 A、块 B 的关系见 9.4.3。

f) 确定 mb_reference_index 的 ctxIdx：

1) 第一步，根据 binIdx 得到 ctxIdxInc：

——如果 binIdx 等于 0，则：

ctxIdxInc = a + 2 × b

——否则，如果 binIdx 等于 1，则：

ctxIdxInc = 4

——否则：

ctxIdxInc = 5

2) 第二步，计算 ctxIdx：

ctxIdx = 起始索引号 + ctxIdxInc

如果当前块 E 的左边块 A（或上边块 B）“可用”，并且块 A（或块 B）的参考索引值大于 0，则 a（或 b）的值为 1；否则 a（或 b）的值为“0”。块 E 和块 A、块 B 的关系见 9.4.3。块 A 和块 B 的参考索引值的定义和赋值过程见 9.4.5 和 9.4.6.2。

g) 确定 mv_diff_x、mv_diff_y 的 ctxIdx：

1) 第一步，根据 binIdx 得到 ctxIdxInc：

——如果 binIdx 等于 0，则：

```
if ( mvda < 2 )
    ctxIdxInc = 0
else if ( mvda < 16 )
    ctxIdxInc = 1
else
    ctxIdxInc = 2
```

——否则，如果 binIdx 等于 1，则：

```
ctxIdxInc = 3
```

——否则，如果 binIdx 等于 2，则：

```
ctxIdxInc = 4
```

——否则，如果 binIdx 等于 3，则：

```
ctxIdxInc = 5
```

——否则，BypassFlag 的值为 1。

2) 第二步，计算 ctxIdx：

```
ctxIdx = 起始索引号 + ctxIdxInc
```

如果当前块 E 的左边 8×8 块 A“可用”，并且块 A 的 mv_diff_x 或 mv_diff_y 在位流中存在，则 mvda 的值按以下方法决定：

——如果当前块 E 是前向预测块，并且块 A 是前向或双向预测块，则 mvda 的值为块 A 的 mv_diff_x 或 mv_diff_y 的绝对值；

——否则，如果当前块 E 是双向预测块，并且块 A 是前向或双向预测块，则 mvda 的值为块 A 的 mv_diff_x 或 mv_diff_y 的绝对值；

——否则，如果当前块 E 是后向预测块，并且块 A 是后向预测块，则 mvda 的值为块 A 的 mv_diff_x 或 mv_diff_y 的绝对值；

——否则，mvda 的值为“0”。

如果当前块 E 的左边 8×8 块 A“不可用”，则 mvda 的值为“0”。块 E 和块 A 的关系见 9.4.3。

h) 确定 cbp 的 ctxIdx：

1) 第一步，根据 binIdx 得到 ctxIdxInc：

——如果 binIdx 的值小于或等于 3，则：

```
ctxIdxInc = a + 2 × b
```

——否则，如果 binIdx 的值等于 4，则：

```
ctxIdxInc = 4
```

——否则：

```
ctxIdxInc = 5
```

2) 第二步，计算 ctxIdx：

```
ctxIdx = 起始索引号 + ctxIdxInc
```

如果当前块 E 的左边 8×8 块 A（或上边 8×8 块 B）“可用”，并且块 A（或块 B）中不包含非零系数，则 a（或 b）的值为“1”；否则 a（或 b）的值为“0”。块 E 和块 A、块 B 的关系见 9.4.3。

i) 确定 mb_qp_delta 的 ctxIdx：

1) 第一步，根据 binIdx 得到 ctxIdxInc：

——如果 binIdx 等于 0，则：

```
if (PreviousDeltaQP != 0)
    ctxIdxInc = 1
else
    ctxIdxInc = 0
```

——否则，如果 binIdx 等于 1，则：

```
ctxIdxInc = 2
```

——否则：

```
ctxIdxInc = 3
```

2) 第二步，计算 ctxIdx：

ctxIdx = 起始索引号 + ctxIdxInc

j) 确定 trans_coefficient 的 coeffLevel 的 ctxIdx：

1) 第一步，根据 binIdx 得到 ctxIdxInc：

——如果 binIdx 不等于 0 或 lMax 等于 0，则：

```
contextWeighting = 0
ctxIdxInc = priIdx × 3 + secIdx – (priIdx != 0)
```

——否则：

```
contextWeighting = 1
ctxIdxInc = priIdx × 3 + secIdx - 1
ctxIdxIncW = 14 + (pos >> 5) × 16 + ((pos >> 1) & 0x0F)
```

2) 第二步，计算 ctxIdx：

ctxIdx = 起始索引号 + ctxIdxInc

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

其中 priIdx 和 secIdx 分别见表 52 和表 53；pos 记录了待解码的量化系数幅值在逆扫描中的位置。对每个 8×8 块进行解码前，pos 应初始化为“0”。lMax 是当前 8×8 块已解码系数的最大幅值。对每个 8×8 块进行解码前，lMax 应初始化为“0”。

k) 确定 trans_coefficient 的 coeffRun 的 ctxIdx：

1) 第一步，根据 binIdx 和 AbsLevel 得到 ctxIdxInc：

ctxIdxInc = priIdx × 4 + secIdx

2) 第二步，计算 ctxIdx：

ctxIdx = 起始索引号 + 46 + ctxIdxInc

其中 priIdx 和 secIdx 分别见表 52 和表 53。

l) 确定 weighting_prediction：

1) 第一步，ctxIdxInc 赋值为“0”；

2) 第二步，计算 ctxIdx：

ctxIdx = 起始索引号 + ctxIdxInc

表52 prildx 与 lMax 的关系

lMax的值	priIdx的值
0	0
1	1
2	2
3	3
4	3
>= 5	4

表53 secIdx 与 coeffLevel、coeffRun 的关系

secIdx的值	含 义	
	coeffLevel的含义	coeffRun的含义
0	binIdx等于0	AbsLevel的值等于1，并且binIdx等于0
1	binIdx等于1	AbsLevel的值等于1，并且binIdx大于或等于1
2	binIdx大于或等于2	AbsLevel的值大于1，并且binIdx等于0
3	-	AbsLevel的值大于1，并且binIdx大于或等于1

8.4.4.3 二元符号解析

8.4.4.3.1 解析过程

二元符号的解析过程如下：

a) 首先，解析二元符号值 binVal
   1) 如果 BypassFlag 的值为“1”，执行 decode_bypass 过程（见 8.4.4.3.3）；
   2) 否则,如果当前解码语法元素为 aec_mb_stuffing_bit,则执行 decode_aec_stuffing_bit 过程（见 8.4.4.3.4）；
   3) 否则，执行 decode_decision 过程，见 8.4.4.3.2。
b) 第二步，如果 binVal 的值为“0”，则二元符号为‘0’；如果 binVal 的值为“1”，则二元符号为‘1’。

8.4.4.3.2 decode_decision

如果contextWeighting的值为1，decode_decision过程的输入是rS1、rT1、valueS、valueT以及上下文模型ctx1、ctx2；否则decode_decision过程的输入是rS1、rT1、valueS、valueT以及上下文模型ctx。decode_decision过程的输出是二元符号值binVal。decode_decision过程用伪代码描述如下：

```
decode_decision( )
{
    if ( contextWeighting == 1 ) {
        if  ( ctx1->mps == ctx2->mps ) {
            predMps = ctx1->mps
            lgPmps = (ctx1->lgPmps + ctx2->lgPmps) / 2
        }
        else {
```

```
        if ( ctx1->lgPmps < ctx2->lgPmps ) {
            predMps = ctx1->mps
            lgPmps = 1023 - ((ctx2->lgPmps – ctx1->lgPmps) >> 1)
        }
        else {
            predMps = ctx2->mps
            lgPmps = 1023 - ((ctx1->lgPmps – ctx2->lgPmps) >> 1)
        }
    }
}
else {
    predMps = ctx->mps
    lgPmps =  ctx->lgPmps
}

if ( rT1 >= (lgPmps >> 2) ) {
    rS2 = rS1
    rT2 = rT1 - (lgPmps >> 2 )
    sFlag = 0
}
else {
    rS2 = rS1 + 1
    rT2 = 256 + rT1 - (lgPmps >> 2)
    sFlag = 1
}

if( rS2 > valueS || (rS2 == valueS && valueT >= rT2) ) {
    binVal = ! predMps
    if ( sFlag == 0 )
        tRlps = lgPmps >> 2
    else
        tRlps = rT1 + (lgPmps >> 2)
    if ( rS2 == valueS )
        valueT = valueT - rT2
    else
        valueT = 256 + ((valueT << 1 ) | read_bits(1)) - rT2
    while ( tRlps < 0x100 ) {
        tRlps = tRlps << 1
        valueT = (valueT << 1 ) | read_bits(1)
    }
    rS1 = 0
    rT1 = tRlps & 0xFF
    valueS = 0
```

```
            while ( valueT < 0x100 ) {
                valueS++
                valueT = (valueT << 1 ) | read_bits(1)
            }
            valueT = valueT & 0xFF
        }
        else {
            binVal = predMps
            rS1 = rS2
            rT1 = rT2
        }

        if ( contextWeighting == 1 ) {
            ctx1 = update_ctx(binVal, ctx1)
            ctx2 = update_ctx(binVal, ctx2)
        }
        else
            ctx = update_ctx(binVal, ctx)
        return (binVal)
    }
```

8.4.4.3.3 decode_bypass

decode_bypass过程的输入是rS1、rS2、valueS和valueT。decode_bypass过程的输出是二元符号值binVal。decode_bypass过程用伪代码描述如下：

```
    decode_bypass( )
    {
        predMps = 0
        lgPmps =  1023
        if ( rT1 >= (lgPmps >> 2) ) {
            rS2 = rS1
            rT2 = rT1 - (lgPmps >> 2 )
            sFlag = 0
        }
        else {
            rS2 = rS1 + 1
            rT2 = 256 + rT1 - (lgPmps >> 2)
            sFlag = 1
        }
        if( rS2 > valueS || (rS2 == valueS && valueT >= rT2) ) {
            binVal = ! predMps
            if ( sFlag == 0 )
                tRlps = lgPmps >> 2
            else
```

```
            tRlps = rT1 + (lgPmps >> 2)
        if ( rS2 == valueS )
            valueT = valueT - rT2
        else
            valueT = ((valueT << 1 ) | read_bits(1)) - rT2 + 256
        while ( tRlps < 0x100 ) {
            tRlps = tRlps << 1
            valueT = (valueT << 1 ) | read_bits(1)
        }
        rS1 = 0
        rT1 = tRlps & 0xFF
        valueS = 0
        while ( valueT < 0x100 ) {
            valueS++
            valueT = (valueT << 1 ) | read_bits(1)
        }
        valueT = valueT & 0xFF
    }
    else {
        binVal = predMps
        rS1 = rS2
        rT1 = rT2
    }
    return (binVal)
}
```

8.4.4.3.4 decode_aec_stuffing_bit

decode_aec_stuffing_bit过程的输入是rS1、rS2、valueS和valueT。decode_aec_stuffing_bit过程的输出是二元符号值binVal。decode_aec_stuffing_bit过程用伪代码描述如下：

```
decode_aec_stuffing_bit( )
{
    predMps = 0
    lgPmps =  4
    if ( rT1 >= (lgPmps >> 2) ) {
        rS2 = rS1
        rT2 = rT1 - (lgPmps >> 2 )
        sFlag = 0
    }
    else {
        rS2 = rS1 + 1
        rT2 = 256 + rT1 - (lgPmps >> 2)
        sFlag = 1
    }
```

```
        if( rS2 > valueS || (rS2 == valueS && valueT >= rT2) ) {
            binVal = ! predMps
            if ( sFlag == 0 )
                tRlps = lgPmps >> 2
            else
                tRlps = rT1 + (lgPmps >> 2)
            if ( rS2 == valueS )
                valueT = valueT – rT2
            else
                valueT = 256 + ((valueT << 1) | read_bits(1)) – rT2
            while ( tRlps < 0x100 ) {
                tRlps = tRlps << 1
                valueT = (valueT << 1) | read_bits(1)
            }
            rS1 = 0
            rT1 = tRlps & 0xFF
            valueS = 0
            while ( valueT < 0x100 ) {
                valueS++
                valueT = (valueT << 1) | read_bits(1)
            }
            valueT = valueT & 0xFF
        }
        else {
            binVal = predMps
            rS1 = rS2
            rT1 = rT2
        }
        return (binVal)
    }
```

8.4.4.3.5 update_ctx

update_ctx过程的输入是binVal和ctx。update_ctx过程的输出是更新后的ctx。update_ctx过程用伪代码描述如下：

```
    update_ctx( )
    {
        if ( ctx->cycno <= 1 )
            cwr = 3
        else if ( ctx->cycno == 2 )
            cwr = 4
        else
            cwr = 5
        if ( binVal != ctx->mps ) {
```

```
            if ( ctx->cycno <= 2 )
                ctx->cycno = ctx->cycno + 1
            else
                ctx->cycno = 3
        }
        else if ( ctx->cycno == 0)
            ctx->cycno = 1

        if ( binVal == ctx->mps )
            ctx->lgPmps = ctx->lgPmps – (ctx->lgPmps >> cwr) - (ctx->lgPmps >> (cwr+2))
        else {
            switch (cwr) {
                case 3:
                    ctx->lgPmps = ctx->lgPmps + 197
                    break
                case 4:
                    ctx->lgPmps = ctx->lgPmps + 95
                    break
                default:
                    ctx->lgPmps = ctx->lgPmps + 46
            }
            if ( ctx->lgPmps > 1023 ) {
                ctx->lgPmps = 2047 - ctx->lgPmps
                ctx->mps = ! (ctx->mps)
            }
        }
        return (ctx)
    }
```

9 解码过程

9.1 高层语法结构

如果progressive_sequence的值为“1”，解码器以帧周期的整数倍为时间间隔输出重建图像。

如果progressive_sequence的值为“0”，则解码器将重建图像拆成两场，以场周期的整数倍为时间间隔输出。

9.2 图像头解码

图像头解码过程如下：

如果当前图像起始码是0x00001B3，则表示图像是I帧，PictureType等于“0”。

如果当前图像起始码是0x00001B6，并且picture_coding_type等于“01”，则表示图像是P帧，PictureType等于“1”。

如果当前图像起始码是0x00001B6，并且picture_coding_type等于“10”，则表示图像是B帧，PictureType等于“2”。

预测量化参数PreviousQP初始化为picture_qp。预测量化参数增量PreviousDeltaQP初始化为“0”。固定量化因子标志FixedQP等于fixed_picture_qp。当前图像的宏块索引MbIndex初始化为“0”。

CurrentSceneModel等于weighting_quant_model。

加权量化参数WeightingQuantParamDefault[i]＝{128，98，106，116，116，128}；加权量化参数WeightingQuantParamBase1[i] ＝ {135， 143， 143， 160， 160， 213}；加权量化参数WeightingQuantParamBase2[i]＝{128，98，106，116，116，128}。

加权量化矩阵WeightingQuantMatrix8x8的元素WeightingQuantMatrix8x8[i,j]（i,j=0～7）的值均初始化为“128”。

如果weighting_quant_flag的值为“1”，按以下步骤确定加权量化矩阵：

a) 首先，初始化 wqP，wqP[0][i]=128（i=0～5），wqP[1][i]= WeightingQuantParamBase1[i]（i=0～5），wqP[2][i]= WeightingQuantParamBase2[i]（i=0～5）。

b) 第二步，确定当前图像的 wqP（wqP 的元素的取值范围应是 0～255）：

1) 如果 weighting_quant_param_index 的值为“00”，则 wqP[0][i]=WeightingQuantParamDefault[i]（i=0～5）。

2) 如果 weighting_quant_param_index 的值为“01”，从 weighting_quant_param_delta1[i] 解析得到 wqPDelta[1][i]。wqP[1][i]=wqPDelta[1][i]＋WeightingQuantParamBase1[i]（i=0～5）。

3) 如果 weighting_quant_param_index 的值为“10”，从 weighting_quant_param_delta2[i] 解析得到 wqPDelta[2][i]。wqP[2][i]=wqPDelta[2][i]＋WeightingQuantParamBase2[i]（i=0～5）。

c) 第三步，根据 CurrentSceneModel 确定 8×8 变换块的加权量化矩阵 wqM8x8[m]：

1) 如果 CurrentSceneModel 的值为“0”，则 wqM8x8[m]如下：

$$wqM8x8[m]=\begin{bmatrix} wqP[m][0] & wqP[m][0] & wqP[m][0] & wqP[m][4] & wqP[m][4] & wqP[m][4] & wqP[m][5] & wqP[m][5] \\ wqP[m][0] & wqP[m][0] & wqP[m][3] & wqP[m][3] & wqP[m][3] & wqP[m][3] & wqP[m][5] & wqP[m][5] \\ wqP[m][0] & wqP[m][3] & wqP[m][2] & wqP[m][2] & wqP[m][1] & wqP[m][1] & wqP[m][5] & wqP[m][5] \\ wqP[m][4] & wqP[m][3] & wqP[m][2] & wqP[m][2] & wqP[m][1] & wqP[m][5] & wqP[m][5] & wqP[m][5] \\ wqP[m][4] & wqP[m][3] & wqP[m][1] & wqP[m][1] & wqP[m][5] & wqP[m][5] & wqP[m][5] & wqP[m][5] \\ wqP[m][4] & wqP[m][3] & wqP[m][1] & wqP[m][5] & wqP[m][5] & wqP[m][5] & wqP[m][5] & wqP[m][5] \\ wqP[m][5] & wqP[m][5] & wqP[m][5] & wqP[m][5] & wqP[m][5] & wqP[m][5] & wqP[m][5] & wqP[m][5] \\ wqP[m][5] & wqP[m][5] & wqP[m][5] & wqP[m][5] & wqP[m][5] & wqP[m][5] & wqP[m][5] & wqP[m][5] \end{bmatrix},\ m=0\sim2$$

2) 如果 CurrentSceneModel 的值为“1”，则 wqM8x8[m]如下：

$$wqM8x8[m]=\begin{bmatrix} wqP[m][0] & wqP[m][0] & wqP[m][0] & wqP[m][4] & wqP[m][4] & wqP[m][4] & wqP[m][5] & wqP[m][5] \\ wqP[m][0] & wqP[m][0] & wqP[m][4] & wqP[m][4] & wqP[m][4] & wqP[m][4] & wqP[m][5] & wqP[m][5] \\ wqP[m][0] & wqP[m][3] & wqP[m][2] & wqP[m][2] & wqP[m][2] & wqP[m][1] & wqP[m][5] & wqP[m][5] \\ wqP[m][3] & wqP[m][3] & wqP[m][2] & wqP[m][2] & wqP[m][1] & wqP[m][5] & wqP[m][5] & wqP[m][5] \\ wqP[m][3] & wqP[m][3] & wqP[m][2] & wqP[m][1] & wqP[m][5] & wqP[m][5] & wqP[m][5] & wqP[m][5] \\ wqP[m][3] & wqP[m][3] & wqP[m][1] & wqP[m][5] & wqP[m][5] & wqP[m][5] & wqP[m][5] & wqP[m][5] \\ wqP[m][5] & wqP[m][5] & wqP[m][5] & wqP[m][5] & wqP[m][5] & wqP[m][5] & wqP[m][5] & wqP[m][5] \\ wqP[m][5] & wqP[m][5] & wqP[m][5] & wqP[m][5] & wqP[m][5] & wqP[m][5] & wqP[m][5] & wqP[m][5] \end{bmatrix},\ m=0\sim2$$

3) 如果 CurrentSceneModel 的值为“2”，则 wqM8x8[m]如下：

$$wqM8x8[m]=\begin{bmatrix} wqP[m][0] & wqP[m][0] & wqP[m][0] & wqP[m][4] & wqP[m][4] & wqP[m][3] & wqP[m][5] & wqP[m][5] \\ wqP[m][0] & wqP[m][0] & wqP[m][4] & wqP[m][4] & wqP[m][3] & wqP[m][2] & wqP[m][5] & wqP[m][5] \\ wqP[m][0] & wqP[m][4] & wqP[m][4] & wqP[m][3] & wqP[m][2] & wqP[m][1] & wqP[m][5] & wqP[m][5] \\ wqP[m][4] & wqP[m][4] & wqP[m][3] & wqP[m][2] & wqP[m][1] & wqP[m][5] & wqP[m][5] & wqP[m][5] \\ wqP[m][4] & wqP[m][3] & wqP[m][2] & wqP[m][1] & wqP[m][5] & wqP[m][5] & wqP[m][5] & wqP[m][5] \\ wqP[m][3] & wqP[m][2] & wqP[m][1] & wqP[m][5] & wqP[m][5] & wqP[m][5] & wqP[m][5] & wqP[m][5] \\ wqP[m][5] & wqP[m][5] & wqP[m][5] & wqP[m][5] & wqP[m][5] & wqP[m][5] & wqP[m][5] & wqP[m][5] \\ wqP[m][5] & wqP[m][5] & wqP[m][5] & wqP[m][5] & wqP[m][5] & wqP[m][5] & wqP[m][5] & wqP[m][5] \end{bmatrix}, m=0\sim 2$$

9.3 条带解码

条带解码过程如下：

宏块索引MbIndex等于MbRow × MbWidth。

如果fixed_picture_qp等于“0”，预测量化参数PreviousQP等于slice_qp，预测量化参数增量PreviousDeltaQP初始化为“0”。固定量化因子标志FixedQP等于fixed_slice_qp。

如果mb_skip_run存在，由mb_skip_run解析得到SkipMbCount，否则SkipMbCount等于“0”。若SkipMbCount不等于“0”，从MbIndex至MbIndex+SkipMbCount-1的所有宏块的MbType根据图像类型设为P_Skip或B_Skip,这些宏块按照所设置的宏块类型进行处理。处理完成后，MbIndex应加上SkipMbCount。

如果当前条带的位流中存在mb_weighting_flag，则MbWeightingFlag等于mb_weighting_flag；否则MbWeightingFlag等于“0”。

9.4 宏块解码

9.4.1 初始化

解码索引值为MbIndex的宏块，完成后MbIndex加1。

WeightingPrediction的值等于slice_weighting_flag的值。

如果slice_weighting_flag和mb_weighting_flag的值均为“1”，则：

a）如果当前宏块的位流中存在 weighting_prediction，则当前宏块的 WeightingPrediction 等于 weighting_prediction；

b）如果当前宏块的位流中不存在 weighting_prediction，则当前宏块的 WeightingPrediction 等于“0”。

9.4.2 宏块类型

宏块类型MbType和宏块子类型MbPartType的解码过程如下：

a） 如果当前图像是 I 帧：

1） 如果满足以下条件之一，则 MbType 等于“I_8x8”（称为 I 宏块类型），MvNum 等于 0：

——PictureStructure 的值为“1”。

——PictureStructure 的值为“0”并且 MbIndex < MbWidth × MbHeight / 2。

2） 如果 PictureStructure 的值为“0”并且 MbIndex >= MbWidth × MbHeight / 2：

——aec_enable 的值为“0”时，如果 skip_mode_flag 的值为“1”，MbTypeIndex 等于 mb_type 加 1；否则 MbTypeIndex 等于 mb_type。如果 MbTypeIndex 大于或等于 5，则 mb_type 中包含的 CBP 信息 CBPCodeNum 等于 MbTypeIndex 减 5,然后令 MbTypeIndex 等于 5。

——aec_enable 的值为“1”时，mb_type 和 MbTypeIndex 的关系见表 54。

b） 如果当前图像是 P 帧：

1) aec_enable 的值为“0”时，如果 skip_mode_flag 的值为“1”，MbTypeIndex 等于 mb_type 加 1；否则 MbTypeIndex 等于 mb_type。如果 MbTypeIndex 大于或等于 5，则 mb_type 中包含的 CBP 信息 CBPCodeNum 等于 MbTypeIndex 减 5，然后令 MbTypeIndex 等于 5。
2) aec_enable 的值为“1”时， mb_type 和 MbTypeIndex 的关系见表 54。

c) 如果当前图像是 B 帧：
1) aec_enable 的值为“0”时，如果 skip_mode_flag 的值为“1”，MbTypeIndex 等于 mb_type 加 1；否则 MbTypeIndex 等于 mb_type。如果 MbTypeIndex 大于或等于 24，则 mb_type 中包含的 CBP 信息 CBPCodeNum 等于 MbTypeIndex 减 24，然后令 MbTypeIndex 等于 24。
2) aec_enable 的值为“1”时，如果 skip_mode_flag 的值为“1”，MbTypeIndex 等于 mb_type 加 1；否则 MbTypeIndex 等于 mb_type。

如果当前图像是P帧，或者当前图像是I帧并且PictureStructure的值为0同时MbIndex >= MbWidth × MbHeight / 2，MbType和MvNum的值见表55；如果当前图像是B帧，MbType和MvNum的值见表56。如果MbType等于“B_8x8”，MbPartType和MbPartMvNum的值用mb_part_type查表57，MvNum是该宏块所有块的MbPartMvNum的和。

表54 mb_type 和 MbTypeIndex 的关系

mb_type的值	MbTypeIndex的值	
	skip_mode_flag的值为“1”	skip_mode_flag的值为“0”
0	5	5
1	1	0
2	2	1
3	3	2
4	4	3
5	-	4

表55 P 帧的宏块类型

MbTypeIndex的值	宏块类型（MbType）	运动矢量数（MvNum）	预测模式（MbPredMode）	宏块分类
0	P_Skip	0	前向	P宏块类型
1	P_16x16	1	前向	P宏块类型
2	P_16x8	2	前向	P宏块类型
3	P_8x16	2	前向	P宏块类型
4	P_8x8	4	前向	P宏块类型
5	I_8x8	0	无	I宏块类型

表56 B 帧的宏块类型

MbTypeIndex的值	宏块类型（MbType）	运动矢量数（MvNum）	预测模式（MbPredMode）	宏块分类
0	B_Skip	0	双向	B宏块类型
1	B_Direct_16x16	0	双向	B宏块类型
2	B_Fwd_16x16	1	前向	B宏块类型
3	B_Bck_16x16	1	后向	B宏块类型

表 56（续）

MbTypeIndex的值	宏块类型（MbType）	运动矢量数（MvNum）	预测模式（MbPredMode）	宏块分类
4	B_Sym_16x16	1	双向	B宏块类型
5	B_Fwd_Fwd_16x8	2	上前、下前	B宏块类型
6	B_Fwd_Fwd_8x16	2	左前、右前	B宏块类型
7	B_Bck_Bck_16x8	2	上后、下后	B宏块类型
8	B_Bck_Bck_8x16	2	左后、右后	B宏块类型
9	B_Fwd_Bck_16x8	2	上前、下后	B宏块类型
10	B_Fwd_Bck_8x16	2	左前、右后	B宏块类型
11	B_Bck_Fwd_16x8	2	上后、下前	B宏块类型
12	B_Bck_Fwd_8x16	2	左后、右前	B宏块类型
13	B_Fwd_Sym_16x8	2	上前、下双	B宏块类型
14	B_Fwd_Sym_8x16	2	左前、右双	B宏块类型
15	B_Bck_Sym_16x8	2	上后、下双	B宏块类型
16	B_Bck_Sym_8x16	2	左后、右双	B宏块类型
17	B_Sym_Fwd_16x8	2	上双、下前	B宏块类型
18	B_Sym_Fwd_8x16	2	左双、右前	B宏块类型
19	B_Sym_Bck_16x8	2	上双、下后	B宏块类型
20	B_Sym_Bck_8x16	2	左双、右后	B宏块类型
21	B_Sym_Sym_16x8	2	上双、下双	B宏块类型
22	B_Sym_Sym_8x16	2	左双、右双	B宏块类型
23	B_8x8	0～4	前向、后向、双向	B宏块类型
24	I_8x8	0	无	I宏块类型

表57　B_8x8 的宏块子类型

mb_part_type的值	宏块子类型（MbPartType）	运动矢量数（MbPartMvNum）	预测模式（MbPartPredMode）
0	SB_Direct_8x8	0	双向
1	SB_Fwd_8x8	1	前向
2	SB_Bck_8x8	1	后向
3	SB_Sym_8x8	1	双向

表55、表56和表57定义的MbType和MbPartType中，有Skip字样的称为跳过模式，有Direct字样的称为直接模式，有Sym字样的称为对称模式。表55到表57中的MvNum表示当前宏块在位流中的运动矢量数。对称模式是一种双向预测模式，此时位流中只包含前向参考索引和前向运动矢量，对称模式的后向参考索引和后向运动矢量的推导见9.9.1。

9.4.3　相邻块

一个块E和它的相邻块A、B、C和D之间的空间位置如图11所示。图11中E的大小可以是16×16、16×8、8×16或8×8。块A是块E的左边块，块B是块E的上边块。设块E左上角样本在图像中的坐标是(x_0, y_0)，右上角样本在图像中的坐标是(x_1, y_1)，块X（X是A、B、C或D）是表58给出的样本所在的编码块。一个宏块K和它的相邻块A、B、C、D、F和G之间的空间位置如图12所示。设宏块K左上角样本在图像中的坐标是

(x_0, y_0)，右上角样本在图像中的坐标是(x_1, y_1)，块Y（Y是A、B、C、D、F和G）是表58给出的样本所在的编码块。表58中坐标均为样本在图像中的坐标。

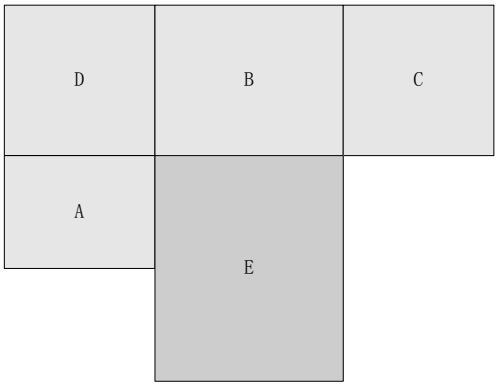


图11　块 E 和相邻块的空间位置关系

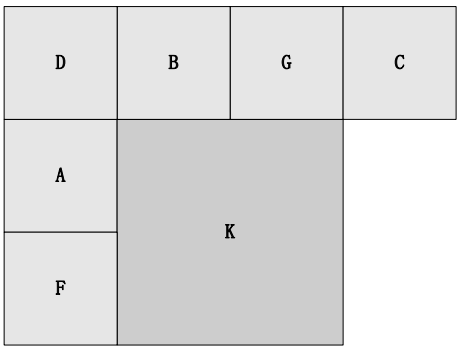


图12　宏块 K 和相邻块的空间位置关系

表58　相邻块的位置

PictureStructure 的值	块 A 右上角样本位置	块 B 左下角样本位置	块 C 左下角样本位置	块 D 右下角样本位置	块 F 右上角样本位置	块 G 左下角样本位置
0	(x_0-1, y_0)	(x_0, y_0-2)	(x_1+1, y_1-2)	(x_0-1, y_0-2)	(x_0-1, y_0+16)	(x_0+8, y_0-2)
1	(x_0-1, y_0)	(x_0, y_0-1)	(x_1+1, y_1-1)	(x_0-1, y_0-1)	(x_0-1, y_0+8)	(x_0+8, y_0-1)

相邻块X（X为A、B、C或D）及相邻块Y（Y为F或G）“存在”指该块应在图像内并且该块应与块E或K属于同一条带；否则相邻块“不存在”。

如果块“不存在”或者尚未解码，则此块“不可用”；否则此块“可用”。如果图像样本所在的块“不存在”或者此样本尚未解码，则此样本“不可用”；否则此样本“可用”。

9.4.4　帧内预测模式

9.4.4.1　概述

当前宏块的每一个8×8块使用9.4.4.2定义的方法确定其预测模式。

9.4.4.2　8×8 帧内预测模式

8×8块采用以下方法确定其预测模式：

a) 如果当前块 E 是亮度块：

1) 计算当前块预测模式的预测值：

——如果左边块 A“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 intraPredModeA；否则 intraPredModeA 等于“-1”。

——如果上边块 B“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 intraPredModeB；否则 intraPredModeB 等于“-1”。

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

2) aec_enable 的值为“1”时，置 BypassFlag 的值为“0”，解析得到 intra_luma_pred_mode。如果 intra_luma_pred_mode 的值为“0”,则 IntraLumaPredMode 等于 predIntraPredMode；否则：

——如果 intra_luma_pred_mode 的值为“4”，则 intra_luma_pred_mode 的值置为“0”。

——如果 intra_luma_pred_mode 小于 predIntraPredMode，则 IntraLumaPredMode 等于 intra_luma_pred_mode；否则 IntraLumaPredMode 等于 intra_luma_pred_mode 加 1。

3) aec_enable 的值为“0”时，如果 pred_mode_flag 的值为“1”，则预测模式 IntraLumaPredMode 等于 predIntraPredMode；否则，如果 intra_luma_pred_mode 小于 predIntraPredMode，则 IntraLumaPredMode 等于 intra_luma_pred_mode；否则 IntraLumaPredMode 等于 intra_luma_pred_mode 加 1。

b) 如果当前块 E 是色度块，块的顺序号为 4、5 时预测模式 IntraChromaPredMode 等于 intra_chroma_pred_mode；块的顺序号为 6、7 时 IntraChromaPredMode 等于 intra_chroma_pred_mode_422。

IntraLumaPredMode的值与8×8亮度块预测模式的关系见表59。IntraChromaPredMode的值与8×8色度块预测模式的关系见表60。

表59 8×8 亮度块帧内预测模式

IntraLumaPredMode的值	帧内预测模式
0	Intra_8x8_Vertical
1	Intra_8x8_Horizontal
2	Intra_8x8_DC
3	Intra_8x8_Down_Left
4	Intra_8x8_Down_Right

表60 8×8 色度块帧内预测模式

IntraChromaPredMode的值	帧内预测模式
0	Intra_Chroma_DC
1	Intra_Chroma_Horizontal
2	Intra_Chroma_Vertical
3	Intra_Chroma_Plane

表59所示的8×8亮度块帧内预测模式见图13。

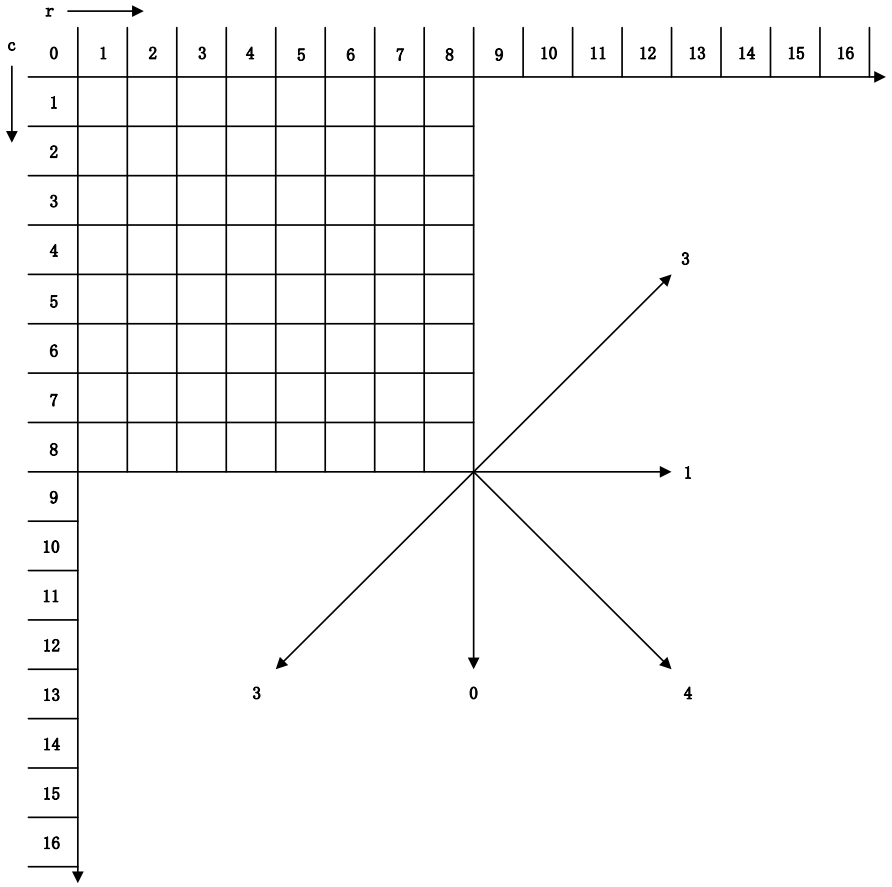


图13　8×8 亮度块帧内预测模式

9.4.5　参考图像选择

每帧图像使用的参考图像除该帧本身外不应超过两个，它们应是最近解码的I帧或P帧。

参考索引值用来确定对当前图像进行解码处理所用的参考图像，参考索引的取值范围是0～3。参考图像的两场可有不同的参考索引值。参考索引值随着参考图像与当前图像距离（显示顺序）的增加而增加，参考索引值为“0”的参考图像与当前图像的距离最近，参考索引值为1的图像距离其次，参考索引值为3的图像距离最远。前向预测和后向预测的参考索引值单独处理。参考图像（或场）的参考索引值的标记方法如下[图14到图20中的数字表示参考索引值，箭头所指的是参考图像（或场）]：

a) 如果当前图像是 I 帧并且 PictureStructure 等于“0”，同时当前解码场在显示顺序上是第二场，标记方法如图 14 所示。此时参考图像（或场）的个数 NumberOfReference 等于“1”。

b) 如果当前图像是 P 帧并且 PictureStructure 等于“1”，标记方法如图 15 所示。此时 NumberOfReference 等于“2”。

c) 如果当前图像是 P 帧并且 PictureStructure 等于“0”，同时当前解码场在显示顺序上是第一场，标记方法如图 16 所示。此时 NumberOfReference 等于“4”。

d) 如果当前图像是 P 帧并且 PictureStructure 等于“0”，同时当前解码场在显示顺序上是第二场，标记方法如图 17 所示。此时 NumberOfReference 等于“4”。

e) 如果当前图像是 B 帧并且 PictureStructure 等于“1”，标记方法如图 18 所示。此时 NumberOfReference 等于“2”。

f) 如果当前图像是 B 帧并且 PictureStructure 等于“0”，同时当前解码场在显示顺序上是第一

场，标记方法如图 19 所示。此时 NumberOfReference 等于“4”。

g) 如果当前图像是 B 帧并且 PictureStructure 等于“0”，同时当前解码场在显示顺序上是第二场，标记方法如图 20 所示。此时 NumberOfReference 等于“4”。

图14　参考索引标记方法 1

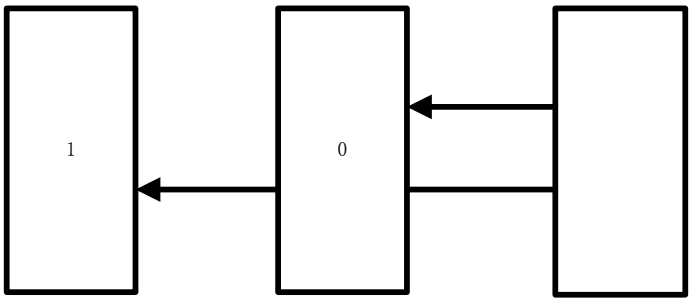


图15　参考索引标记方法 2

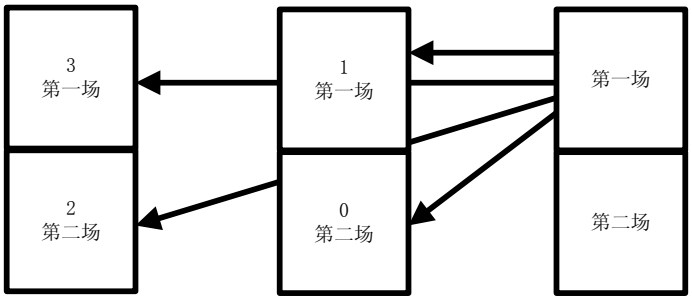


图16　参考索引标记方法 3

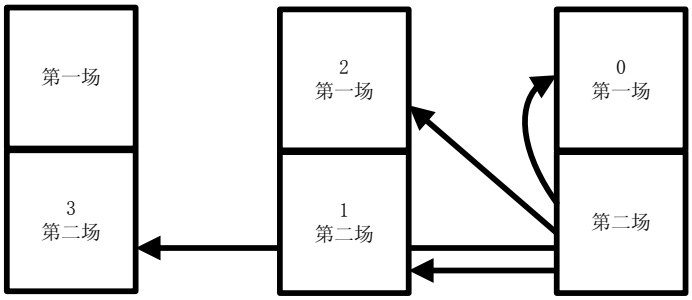


图17　参考索引标记方法 4

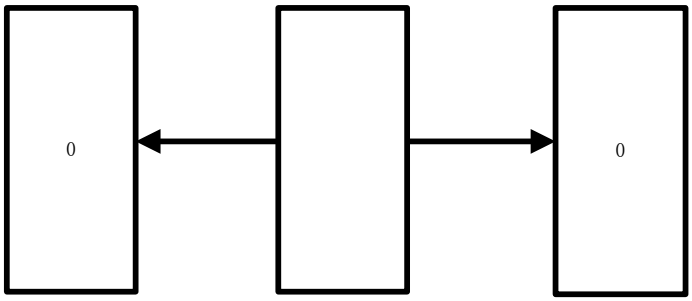


图18　参考索引标记方法 5

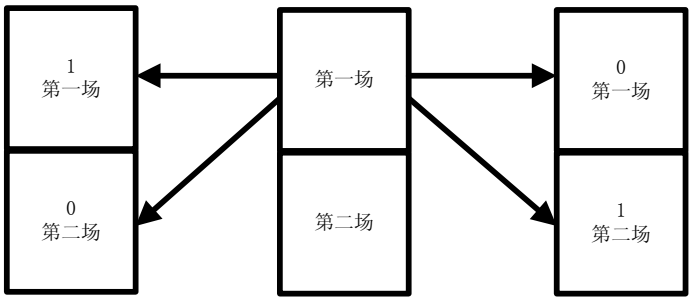


图19　参考索引标记方法 6

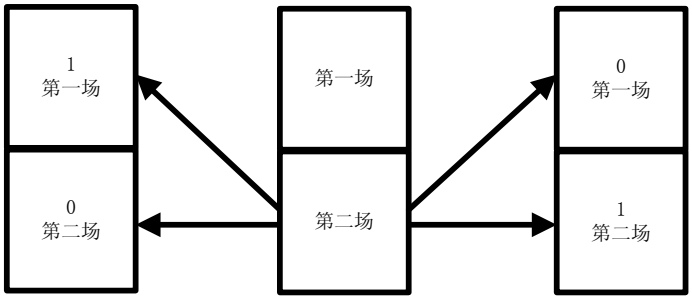


图20　参考索引标记方法 7

如果当前图像的位流中不出现mb_reference_index，参考索引值为0；否则参考索引值等于mb_reference_index的值。

如果使用加权预测，解码的加权预测参数按参考索引值由小到大依次分配给各参考图像（或场）。如果当前图像是B帧，对参考索引值相同的前后参考图像（或场），解码的加权预测参数先分配给前向参考图像（或场），再分配给后向参考图像（或场）。

9.4.6 运动矢量

9.4.6.1 概述

块的距离索引DistanceIndex定义如下：如果块的所有像素都属于所在隔行扫描图像的第二场（显示顺序）或者都属于所在逐行扫描图像的底场，DistanceIndex等于picture_distance乘2加1；否则DistanceIndex等于picture_distance乘2。

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

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

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

9.4.6.2 亮度运动矢量预测

图11中块A、B、C、D的原始运动矢量或转换后的运动矢量为mvA、mvB、mvC、mvD，对应的BlockDistance为 BlockDistanceA 、 BlockDistanceB 、 BlockDistanceC 、 BlockDistanceD ， 对 应 的 参 考 索 引 为 referenceIndexA、referenceIndexB、referenceIndexC、referenceIndexD。

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

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

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

d) 如果 C“不可用”，那么 mvC 等于 mvD，BlockDistanceC 等于 BlockDistanceD，C 的参考索引值等于 D 的参考索引值。

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

当前块E的运动矢量预测值MVEPred计算过程如下:

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

b) 第二步，如果 E 所在宏块按 16×8 或 8×16 模式编码，计算过程如下（见图 21）：

1) 8×16 模式：

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

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

2) 16×8 模式：

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

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

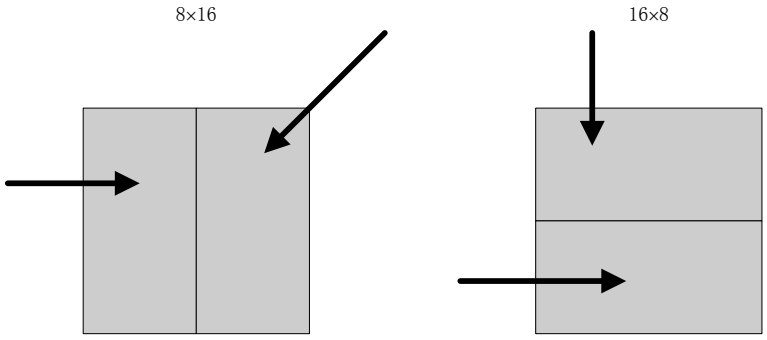


图21　8×16 或 16×8 模式预测

c)　第三步，根据 BlockDistanceA、BlockDistanceB、BlockDistanceC、BlockDistanceE 对 mvA(mvA_x,mvA_y)、mvB(mvB_x,mvB_y)、mvC(mvC_x,mvC_y)进行缩放，得到 MVA(MVA_x,MVA_y)、MVB(MVB_x,MVB_y)、MVC(MVC_x,MVC_y)。其中 BlockDistanceE 是当前块 E 对应的 BlockDistance。

计算 MVA(MVA_x,MVA_y)、MVB(MVB_x,MVB_y)、MVC(MVC_x,MVC_y)：

1)　如果 BlockDistanceA 等于 0，MVA_x = mvA_x，MVA_y = mvA_y；

2)　否则，

$$MVA\_x = Sign(mvA\_x) \times ((Abs(mvA\_x) \times BlockDistanceE \times (512 / BlockDistanceA) + 256) >> 9)$$

$$MVA\_y = Sign(mvA\_y) \times ((Abs(mvA\_y) \times BlockDistanceE \times (512 / BlockDistanceA) + 256) >> 9)$$

3)　如果 BlockDistanceB 等于 0，MVB_x = mvB_x，MVB_y = mvB_y；

4)　否则，

$$MVB\_x = Sign(mvB\_x) \times ((Abs(mvB\_x) \times BlockDistanceE \times (512 / BlockDistanceB) + 256) >> 9)$$

$$MVB\_y = Sign(mvB\_y) \times ((Abs(mvB\_y) \times BlockDistanceE \times (512 / BlockDistanceB) + 256) >> 9)$$

5)　如果 BlockDistanceC 等于 0，MVC_x = mvC_x，MVC_y = mvC_y；

6)　否则，

$$MVC\_x = Sign(mvC\_x) \times ((Abs(mvC\_x) \times BlockDistanceE \times (512 / BlockDistanceC) + 256) >> 9)$$

$$MVC\_y = Sign(mvC\_y) \times ((Abs(mvC\_y) \times BlockDistanceE \times (512 / BlockDistanceC) + 256) >> 9)$$

d)　第四步，定义距离 Dist(MV1, MV2) = Abs(x1 − x2) + Abs(y1 − y2)，其中运动矢量 MV1=[x1, y1],MV2=[x2, y2]。定义 VAB 等于 Dist(MVA, MVB),VBC 等于 Dist(MVB, MVC),VCA 等于 Dist(MVC, MVA)。MVEPred 的计算如下：

1)　计算 FMV 等于 Median(VAB, VBC, VCA)；

2)　如果 FMV 和 VAB 相等，MVEPred 等于 MVC；

3)　否则，如果 FMV 和 VBC 相等，MVEPred 等于 MVA；

4)　否则，MVEPred 等于 MVB。

9.4.6.3 亮度运动矢量解码

当前块E的运动矢量mvE等于MVEPred加上由mv_diff_x和mv_diff_y解码得到的运动矢量增量的和。运动矢量基本单位为1/4样本。当前块E的运动矢量应符合6.2.5的范围规定。

9.4.7 宏块编码模板

当aec_enable的值为“0”时，如果当前宏块的cbp在位流中，解析cbp得到CodeNum；否则，如果当前宏块的类型不等于P_Skip或B_Skip，令CodeNum的值等于CBPCodeNum（见9.4.2）。由CodeNum查表42得到MbCBP（见8.2）。MbCBP表示宏块中顺序号为0到5的4个8×8亮度块和2个8×8色度块是否包含非零变换系数。MbCBP的第n位为“0”表示顺序号为n的8×8块没有非零系数，等于“1”表示该8×8块至少有一个非零系数（顺序号见6.5）。

当aec_enable的值为“1”时，如果当前宏块的cbp在位流中，解析cbp得到MbCBP（见8.4）。MbCBP表示宏块中顺序号为0到5的4个8×8亮度块和2个8×8色度块是否包含非零变换系数。MbCBP的第n位为“0”表示顺序号为n的8×8块没有非零系数，等于“1”表示该8×8块至少有一个非零系数（顺序号见6.5）。

当chroma_format的值为“10”并且aec_enable的值为“0”时，如果当前宏块的cbp_422在位流中，解析cbp_422得到CodeNum。由CodeNum查表43得到MbCBP422（见8.2）。MbCBP422表示4:2:2格式的宏块中顺序号为6和7的2个8×8色度块是否包含非零变换系数。MbCBP422的第*n*位为“0”表示顺序号为n+6的8×8色度块没有非零系数，等于“1”表示该8×8色度块至少有一个非零系数（顺序号见6.5）。

当chroma_format的值为“10”并且aec_enable的值为“1”时，如果当前宏块的cbp_422在位流中，解析cbp_422得到MbCBP422（见8.4）。MbCBP422表示4:2:2格式的宏块中顺序号为6和7的2个8×8色度块是否包含非零变换系数。MbCBP422的第*n*位为“0”表示顺序号为n+6的8×8色度块没有非零系数，等于“1”表示该8×8色度块至少有一个非零系数（顺序号见6.5）。

9.4.8 量化参数

如果当前宏块的mb_qp_delta不在位流中，当前宏块的量化参数CurrentQP等于PreviousQP。

如果aec_enable的值为“0”并且当前宏块的mb_qp_delta在位流中，则解析mb_qp_delta得到mbQpDelta，CurrentQP等于PreviousQP加上mbQpDelta。

如果aec_enable的值为“1”并且当前宏块的mb_qp_delta在位流中，则置BypassFlag的值为“0”，解析mb_qp_delta得到mbQpDelta。如果mbQpDelta的值不为“0”，则按以下方式计算：

```
if ( mbQpDelta % 2 == 0 )
    mbQpDelta = -(mbQpDelta + 1) / 2
else
    mbQpDelta = (mbQpDelta + 1) / 2
```

CurrentQP等于PreviousQP加上mbQpDelta。

宏块解码完成后，将PreviousQP的值设为CurrentQP，将PreviousDeltaQP的值设为mbQpDelta。mbQpDelta的取值范围应是-32～31。CurrentQP的取值范围应是0～63。

9.4.9 加权量化矩阵

如果weighting_quant_flag的值为“1”，加权量化矩阵WeightingQuantMatrix8x8等于wqM8x8[QMode]。其中，QMode等于weighting_quant_param_index的值。

9.5 块解码

9.5.1 基本熵编码的解码

基本熵编码的解码过程定义了aec_enable的值为“0”时由语法元素生成量化系数值数组（Level数组）和量化系数游程数组（Run数组）的过程。Level数组的元素为非0量化系数的幅值，Run数组的元素为当前非0量化系数前的连续0的个数。生成量化系数数组所需语法元素的解析过程见8.3。

生成Run数组和Level数组的过程如下（数组地址初始值为0，maxAbsLevel初始值为0）：

a) 如果 trans_coefficient 小于“59”：
   1) 如果 CurrentVLCTable 中包含此 trans_coefficient 值的索引项，则以 trans_coefficient 为索引查 CurrentVLCVTable，得到量化系数和游程，把它们分别放入 Level 数组和 Run 数组；
   2) 如果表中不包含此 trans_coefficient 值的索引项，则以(trans_coefficient-1)为索引查 CurrentVLCTable，得到量化系数和游程，absLevel 等于量化系数的绝对值，并将量化系数符号取反后放入 Level 数组，将游程放入 Run 数组。
b) 如果 trans_coefficient 大于或等于“59”：
   1) 将(trans_coefficient-59)/2 放入 Run 数组；
   2) 由 CurrentVLCTable 查表 D.20 得到 MaxRun。如果(trans_coefficient-59)/2 大于 MaxRun，RefAbsLevel 等于“1”；否则以(trans_coefficient-59)/2 为索引查 CurrentVLCTable 得到 RefAbsLevel；
   3) 如果trans_coefficient是奇数,将-(RefAbsLevel+escape_level_diff)放入Level数组；否则，将(RefAbsLevel+escape_level_diff)放入 Level 数组。absLevel 等于(RefAbsLevel+escape_level_diff)。
c) 如果 trans_coefficient 等于“EOB”，结束块系数解码；否则数组地址加 1，首先按 8.3 定义的方法更新 CurrentVLCTable，然后如果 absLevel 大于 maxAbsLevel，则 maxAbsLevel 等于 absLevel，最后解码下一个系数值和游程。

9.5.2 高级熵编码的解码

本条定义aec_enable的值为“1”时由语法元素生成量化系数值数组（Level数组）和量化系数游程数组（Run数组）的过程。Level数组的元素为非0量化系数的幅值，Run数组的元素为当前非0量化系数前的连续0的个数。生成量化系数数组所需语法元素的解析过程见8.4。

生成Run数组和Level数组的过程如下（数组地址初始值为“0”）：

a) 由 trans_coefficient 解析得到 AbsLevel。如果是块的第一个 AbsLevel，则 AbsLevel = AbsLevel + 1。
b) 如果 AbsLevel 不等于“0”：
   1) level 等于 AbsLevel。
   2) 由 trans_coefficient 解析得到 coeffSign。如果 coeffSign 为“0”，将 level 放入 Level 数组；否则，将-level 放入 Level 数组。
   3) 由 trans_coefficient 解析得到 RunVal，将 RunVal 放入 Run 数组。
c) 如果 AbsLevel 等于“0”，结束块系数解码（即 trans_coefficient 等于 EOB）；否则数组地址加 1，解码下一个系数值和游程。

9.5.3 逆扫描

由解码的Level数组和Run数组生成数组QuantCoeffArray（8×8块包含64个量化系数）的步骤如下：

a) 首先，将 QuantCoeffArray 数组的所有元素初始化为“0”；

b) 第二步，将非 0 量化系数的值赋给 QuantCoeffArray 数组中相应的元素。定义 j 和 coeffNum，令 j 等于 9.5.1 或 9.5.2 中 Run 数组或 Level 数组中最后一个赋值元素的地址索引，coeffNum 等于“-1”，

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

c) 第三步，确定逆块扫描方法：

1) 如果 progressive_frame 的值为“1”，或者 progressive_frame 的值为“0”并且 picture_structure 的值为“1”，逆块扫描方法 1 见图 22。

2) 如果 progressive_frame 的值为“0”并且 picture_structure 的值为“0”，逆块扫描方法 2 见图 23。

d) 第四步，通过逆块扫描将 QuantCoeffArray 数组映射为 QuantCoeffMatrix 数组。设 QuantCoeffArray 数组中某元素的地址为 k，在扫描表中找到值为 k 的单元对应的列号 i 和行号 j，然后将 QuantCoeffArray[k]赋给 QuantCoeffMatrix[i,j]。

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

图22　8×8 逆块扫描方法 1

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

图23　8×8 逆块扫描方法 2

9.6　反量化

9.6.1　确定量化参数

本条确定的亮度量化参数和色度量化参数的取值范围应是0～63。

如果当前块是8×8亮度块，该块的量化参数QP等于其所在宏块的CurrentQP。

如果当前块是色度块，按下式确定Cb、Cr色度块的CurrentQPCb、CurrentQPCr：

CurrentQPCb = CurrentQP + chroma_quant_param_delta_cb

CurrentQPCr = CurrentQP + chroma_quant_param_delta_cr

然后分别以CurrentQPCb和CurrentQPCr为索引查表61得到Cb、Cr色度块的量化参数QP。

表61　色度块的 QP 与 CurrentQPCb、CurrentQPCr 的映射关系

CurrentQPCb、CurrentQPCr的值	QP的值
< 43	CurrentQPCb、CurrentQPCr
43	42
44	43
45	43
46	44
47	44
48	45

表 61（续）

CurrentQPCb、CurrentQPCr的值	QP的值
49	45
50	46
51	46
52	47
53	47
54	48
55	48
56	48
57	49
58	49
59	49
60	50
61	50
62	50
63	51

9.6.2 反量化过程

本条定义将二维量化系数矩阵QuantCoeffMatrix转换为二维变换系数矩阵CoeffMatrix的过程。从符合本部分的位流中解码得到的8×8块量化系数的取值范围应为-2^{n+3}～$2^{n+3}-1$（n是样本点精度）。

8×8块的二维变换系数矩阵CoeffMatrix8x8由下式得到：

$$CoeffMatrix8x8[i,j] = (((((QuantCoeffMatrix[i,j] \times WeightingQuantMatrix8x8[i,j]) >> 3) \times DequantTable(QP)) >> 4) + 2^{ShiftTable(QP)-1}) >> ShiftTable(QP)\quad i,j=0\sim7$$

从符合本部分的位流中解码得到的QuantCoeffMatrix[i,j]（i,j=0～7）的值的范围应为-2^{n+3}～$2^{n+3}-1$（n是样本点精度）。

QP与DequantTable和ShiftTable的关系见表62。

表62 QP 与 DequantTable 和 ShiftTable 的关系

QP 的值	DequantTable(QP)的值	ShiftTable(QP)的值
0	32768	14
1	36061	14
2	38968	14
3	42495	14
4	46341	14
5	50535	14
6	55437	14
7	60424	14
8	32932	13
9	35734	13

表 62（续）

QP 的值	DequantTable(QP)的值	ShiftTable(QP)的值
10	38968	13
11	42495	13
12	46177	13
13	50535	13
14	55109	13
15	59933	13
16	65535	13
17	35734	12
18	38968	12
19	42577	12
20	46341	12
21	50617	12
22	55027	12
23	60097	12
24	32809	11
25	35734	11
26	38968	11
27	42454	11
28	46382	11
29	50576	11
30	55109	11
31	60056	11
32	65535	11
33	35734	10
34	38968	10
35	42495	10
36	46320	10
37	50515	10
38	55109	10
39	60076	10
40	65535	10
41	35744	9
42	38968	9
43	42495	9
44	46341	9
45	50535	9
46	55099	9
47	60087	9

表 62（续）

QP 的值	DequantTable(QP)的值	ShiftTable(QP)的值
48	65535	9
49	35734	8
50	38973	8
51	42500	8
52	46341	8
53	50535	8
54	55109	8
55	60097	8
56	32771	7
57	35734	7
58	38965	7
59	42497	7
60	46341	7
61	50535	7
62	55109	7
63	60099	7

从符合本部分的位流中解码得到的CoeffMatrix8x8的元素的取值范围应为 -2^{n+5}～$2^{n+5}-1$（n是样本点精度）。

9.7 反变换

本条定义将8×8变换系数矩阵CoeffMatrix8x8转换为8×8残差样值矩阵ResidueMatrix的过程，步骤如下：

a) 首先，对变换系数矩阵进行如下水平反变换：

$$H' = \text{CoeffMatrix8x8} \times T_8^{\mathrm{T}}$$

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

$$T_8 = \begin{bmatrix} 8 & 10 & 10 & 9 & 8 & 6 & 4 & 2 \\ 8 & 9 & 4 & -2 & -8 & -10 & -10 & -6 \\ 8 & 6 & -4 & -10 & -8 & 2 & 10 & 9 \\ 8 & 2 & -10 & -6 & 8 & 9 & -4 & -10 \\ 8 & -2 & -10 & 6 & 8 & -9 & -4 & 10 \\ 8 & -6 & -4 & 10 & -8 & -2 & 10 & -9 \\ 8 & -9 & 4 & 2 & -8 & 10 & -10 & 6 \\ 8 & -10 & 10 & -9 & 8 & -6 & 4 & -2 \end{bmatrix}$$

b) 第二步，矩阵 H'的元素 h''_{ij}计算如下：

$$h''_{ij} = \text{Clip3}(-2^{n+7},\ 2^{n+7}-1,\ (h'_{ij} + 4)) >> 3 \quad i,j=0\sim7;n\text{是样本点精度}$$

c) 第三步，对矩阵 H'进行如下垂直反变换：

$$H = T_8 \times H''$$

其中，H 表示反变换后的8×8矩阵。

d) 第四步，残差样值矩阵 ResidueMatrix 的元素 r_{ij}计算如下：

$$r_{ij} = \mathrm{Clip3}(-2^{n+7},\ 2^{n+7}-1,\ (h_{ij} + 2^{6})) >> 7 \quad i,j=0\sim7;n\text{是样本点精度}$$

其中h_{ij}是H 矩阵的元素。

9.8 帧内预测

9.8.1 概述

帧内预测完成后得到预测样本矩阵predMatrix。应使用9.8.2定义的方法获得当前8×8块的参考样本，并应使用9.8.3、9.8.4定义的方法进行帧内预测。

当前8×8块由其上边和左边的参考样本$r[i]$（i=0～16）和$c[i]$（i=0～16）来预测（r、c可表示亮度或色度参考样本），见图13，其中$r[0]$等于$c[0]$。如果帧内预测需要用到i大于16的上边和左边的参考样本，则$r[i]=r[16]$，$c[i]=c[16]$（i>16）。

9.8.2 8×8 块参考样本的获得

如果当前块所在的图像的PictureStructure的值为“1”，则当前块所在图像的样本矩阵为I；否则当前块所在的场的样本矩阵为I。I可表示亮度或色度矩阵。

设当前块左上角样本在样本矩阵I中的坐标是(x_0, y_0)，其参考样本按以下规则获得：

a) 如果坐标为(x_0+i-1, y_0-1)（i=1～8）的样本“可用”，则 $r[i]$等于 $I[x_0+i-1, y_0-1]$，$r[i]$“可用”；否则 r[i]“不可用”；

b) 如果坐标为(x_0+i-1, y_0-1)（i=9～16）的样本“可用”，则 $r[i]$等于 $I[x_0+i-1, y_0-1]$，$r[i]$“可用”；否则 $r[i]$等于 $r[8]$，$r[i]$是否“可用”由 $r[8]$是否“可用”决定；

c) 如果坐标为(x_0-1, y_0+i-1)（i=1～8）的样本“可用”，则 $c[i]$等于 $I[x_0-1, y_0+i-1]$，$c[i]$“可用”；否则 $c[i]$“不可用”；

d) 如果坐标为(x_0-1, y_0+i-1)（i=9～16）的样本“可用”，则 $c[i]$等于 $I[x_0-1, y_0+i-1]$，$c[i]$“可用”；否则 $c[i]$等于 $c[8]$，$c[i]$是否“可用”由 $c[8]$是否“可用”决定；

e) 如果坐标为(x_0-1, y_0-1)的样本“可用”，则 $r[0]$等于 $I[x_0-1, y_0-1]$，$r[0]$“可用”；否则

   1) 如果 $r[1]$“可用”并且 $c[1]$“不可用”，则 $r[0]$等于 $r[1]$，$r[0]$“可用”；否则

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

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

9.8.3 8×8 亮度块帧内预测

根据IntraLumaPredMode决定亮度块帧内预测方法。

a) IntraLumaPredMode 等于“0”（Intra_8x8_Vertical 预测）

   当$r[i]$（i=1～8）“可用”时，该模式才被使用，此时

   $\mathrm{predMatrix}[x,y] = r[x+1]$（$x,y$=0～7）

b) IntraLumaPredMode 等于“1”（Intra_8x8_Horizontal 预测）

   当$c[i]$（i=1～8）“可用”时，该模式才被使用，此时

   $\mathrm{predMatrix}[x,y] = c[y+1]$（$x,y$=0～7）

c) IntraLumaPredMode 等于“2”（Intra_8x8_DC 预测）

   1) 如果 $r[i]$、$c[i]$（i=0～9）都“可用”,则

      $\mathrm{predMatrix}[x,y]= ((r[x]+2\times r[x+1]+r[x+2]+2)>>2+(c[y]+2\times c[y+1]+c[y+2]+2)>>2)>>1$

      （x,y=0～7）；否则

   2) 如果 $r[i]$（i=0～9）“可用”，则

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

3) 如果 c[i]（i=0～9）“可用”，则

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

4) predMatrix[x,y] = 2^{n-1}（x,y=0～7；n是样本点精度）。

d) IntraLumaPredMode 等于“3”（Intra_8x8_Down_Left 预测）

当 r[i] 、c[i]（i=1～16）均“可用”时，该模式才被使用，此时

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

e) IntraLumaPredMode 等于“4”（Intra_8x8_Down_Right 预测）

当 r[i] 、c[i]（i=0～16）均“可用”时，该模式才被使用，此时

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

2) 如果 x 大于 y，predMatrix[x,y] = (r[x-y+1]+ 2×r[x-y]+r[x-y-1]+2)>>2（x,y=0～7）；否则

3) 如果 y 大于 x，predMatrix[x,y] = (c[y-x+1]+2×c[y-x]+c[y-x-1]+2)>>2（x,y=0～7）。

9.8.4 8×8 色度块帧内预测

根据IntraChromaPredMode决定色度块帧内预测方法。

a) IntraChromaPredMode 等于“0”（Intra_Chroma_DC 预测）

1) 如果 r[i]、c[i]（i=0～9）都“可用”,则

predMatrix[x,y]= ((r[x]+2×r[x+1]+r[x+2]+2)>>2+(c[y]+2×c[y+1]+c[y+2]+2)>>2)>>1（x,y=0～7）；否则

2) 如果 r[i]（i=0～9）“可用”，则

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

3) 如果 c[i]（i=0～9）“可用”，则

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

4) predMatrix[x,y] = 2^{n-1}（x,y=0～7；n是样本点精度）。

b) IntraChromaPredMode 等于“1”（Intra_Chroma_Horizontal 预测）

当c[i] （i=1～8）“可用”时，该模式才被使用，此时

predMatrix[x,y] = c[y+1]（x,y=0～7）。

c) IntraChromaPredMode 等于“2”（Intra_Chroma_Vertical 预测）

当r[i] （i=1～8）“可用”时，该模式才被使用，此时

predMatrix[x,y] = r[x+1]（x,y=0～7）。

d) IntraChromaPredMode 等于“3”（Intra_Chroma_Plane 预测）

当r[i]、c[i] （i=1～8）均“可用”时，该模式才被使用，此时

predMatrix[x,y] = Clip1((ia+(x-3)×ib+(y-3)×ic+16)>>5)（x,y=0～7）。

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

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

9.9 帧间预测

9.9.1 亮度运动矢量导出

如果当前宏块类型或者当前宏块子类型是跳过模式、直接模式或对称模式，其运动矢量按照下面定义的方法导出（具体见图24到图29），否则将9.4.6解码得到的运动矢量，按照宏块划分顺序(在图6中定义)分配给相应的子块。

a) 如果当前宏块类型为 P_Skip：
   1) 如果 PFieldSkip 的值等于 1，picture_coding_type 等于“01”，并且 picture_structure 的值等于“0”，当前块的前向参考图像为图 14 到图 20 中标记为 1 的图像；否则，当前块的前向参考图像为缺省参考图像，即图 14 到图 20 中标记为 0 的图像；
   2) 如果当前宏块的上边宏块 B 或左边宏块 A“不可用”，mvE 等于零矢量；
   3) 通过 9.4.6.2 定义的方法得到 mvA、mvB，如果 mvA 为零矢量并且宏块 A 的参考图像与当前块的参考图像相同，或 mvB 为零矢量并且宏块 B 的参考图像与当前块的参考图像相同，mvE 等于零矢量；
   4) 对于其他情况，通过 9.4.6.2 定义的方法得到 MVEPred，mvE = MVEPred。
b) 如果当前宏块类型为 B_Skip 或 B_Direct_16x16，或当前宏块子类型为 SB_Direct_8x8，对每个 8×8 块分别进行以下步骤：
   1) 第一步：

——如果当前块符合以下条件之一， 则当前块的前后向参考图像均为缺省参考图像，即图 14 到图 20 中标记为 0 的图像，当前块的前后向运动矢量是分别根据 9.4.6 得到的当前块所在宏块（9.4.6 中块 E 为当前块所在宏块）的前后向运动矢量预测值。

- 当前块所在图像的 PictureStructure 等于 1，且后向参考图像中与当前 8×8 块的左上角样本位置对应的样本所在的编码宏块类型为“I_8x8”。
- 当前块所在图像的 PictureStructure 等于 0，其左上角样本位置为(x0, y0)。
  - 如果 BFieldEnhanced 的值等于 1，且 MbIndex >= MbWidth × MbHeight / 2，并且在后向参考图像中与(x0, y0-2×top_field_first+1)样本位置对应的样本所在的编码块的宏块类型为“I_8x8”。
  - 如果 BFieldEnhanced 的值等于 0，且后向参考图像中与当前 8×8 块的左上角样本位置对应的样本所在的编码宏块类型为“I_8x8”。
  - MbIndex < MbWidth × MbHeight / 2 且后向参考图像中与当前 8×8 块的左上角样本位置对应的样本所在的编码宏块类型为“I_8x8”。

——否则，如果当前块所在图像的 PictureStructure 等于 1：

- 当前块的前后向参考图像均为缺省参考图像，即图 18 中标记为 0 的图像，其前后向距离索引分别为 DistanceIndexFw 和 DistanceIndexBw；当前块的前后向 BlockDistance 分别为 BlockDistanceFw 和 BlockDistanceBw。
- 在后向参考图像中与当前块的左上角样本位置对应的样本所在的编码块的运动矢量为 mvRef（mvRef_x,mvRef_y），该编码块的距离索引为 DistanceIndexCol，该运动矢量指向的参考块的距离索引为 DistanceIndexRef。

——否则，如果当前块所在图像的 PictureStructure 等于 0：

- 当前块的左上角样本位置为(x0, y0)。如果 BFieldEnhanced 的值等于 1 且 MbIndex >= MbWidth × MbHeight / 2，则在后向参考图像中与(x0, y0-2×top_field_first+1)样本位置对应的样本所在的编码块的运动矢量为 mvRef（mvRef_x,mvRef_y），该编码块的距离索引为 DistanceIndexCol，该运动矢量指向的参考块的距离索引为 DistanceIndexRef；否则,在后向参考图像中与(x0, y0)样本位置对应的样本所在的编码块的运动矢量为 mvRef

（mvRef_x,mvRef_y），该编码块的距离索引为 DistanceIndexCol，该运动矢量指向的参考块的距离索引为 DistanceIndexRef。

- 当前块的参考索引标记为 0 和 1 的前向参考场中的参考块的距离索引分别为 DistanceIndexFw0 和 DistanceIndexFw1。当 BFieldEnhanced 的值等于 0，或者当前块的 MbIndex < MbWidth × MbHeight / 2 时，如果 DistanceIndexRef 等于 DistanceIndexFw0，则当前块的前向参考场是参考索引标记为 0 的场，DistanceIndexFw 等于 DistanceIndexFw0;否则当前块的前向参考场是参考索引标记为 1 的场，DistanceIndexFw 等于 DistanceIndexFw1。当 BFieldEnhanced 的值等于 1，并且当前块的 MbIndex >= MbWidth × MbHeight / 2 时，当前块的当前块的前向参考场是参考索引标记为 0 的场，DistanceIndexFw 等于 DistanceIndexFw0。
- 当前块的参考索引标记为 0 和 1 的后向参考场中的参考块的距离索引分别为 DistanceIndexBw0 和 DistanceIndexBw1。当 BFieldEnhanced 的值等于 0 时，如果当前块的 MbIndex < MbWidth × MbHeight / 2，则当前块的后向参考场是参考索引标记为 0 的场，DistanceIndexBw 等于 DistanceIndexBw0；否则当前块的后向参考场是参考索引标记为 1 的场，DistanceIndexBw 等于 DistanceIndexBw1。当 BFieldEnhanced 的值等于 1 时，当前块的后向参考场是参考索引标记为 0 的场，DistanceIndexBw 等于 DistanceIndexBw0。

2) 第二步：

——BlockDistanceRef = (DistanceIndexCol – DistanceIndexRef + 512) % 512

当前块的距离索引为 DistanceIndexCur，则

BlockDistanceFw = (DistanceIndexCur – DistanceIndexFw + 512) % 512

BlockDistanceBw = (DistanceIndexBw – DistanceIndexCur + 512) % 512

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

——如果当前块所在图像的 PictureStructure 等于 0，并且后向参考图像的 PictureStructure 等于 1，则 mvRef_y = mvRef_y / 2。

3) 第三步：

——计算 delta1，初始化 deltaFw、deltaBw：

- 置 delta1、deltaFw、deltaBw 的值为“0”。
- 如果当前图像的 PictureStructure 的值为“0”，并且 BfieldEnhanced 值为 1：
  — 如果后向参考场为顶场，mvRef 指向的场为底场，则 delta1 = 2；
  — 如果后向参考场为底场，mvRef 指向的场为顶场，则 delta1 = -2；

——当前块的前向运动矢量 mvFw(mvFw_x, mvFw_y)为：

- 如果当前图像的 PictureStructure 的值为“0”，并且 BFieldEnhanced 值为 1：
  — 如果当前块所在的场为顶场，并且当前块的前向参考场为底场，则 deltaFw=2。
  — 如果当前块所在的场为底场，并且当前块的前向参考场为顶场，则 deltaFw=-2。
- 如果 mvRef_x 小于 0，则 mvFw_x = -(((16384/BlockDistanceRef) × (1-mvRef_x × BlockDistanceFw)-1)>>14)；否则 mvFw_x = ((16384/ BlockDistanceRef)×(1+ mvRef_x×BlockDistanceFw)-1)>>14。

- 如果(mvRef_y + delta1)小于 0，则 mvFw_y = -(((16384/BlockDistanceRef)×(1-(mvRef_y+delta1)×BlockDistanceFw)-1)>>14) - deltaFw；否则 mvFw_y = (((16384/BlockDistanceRef) × (1+(mvRef_y+ delta1)× BlockDistanceFw)-1)>>14) - deltaFw。

——当前块的后向运动矢量 mvBw(mvBw_x, mvBw_y)为：

- 如果当前图像的 PictureStructure 的值为“0”，并且 BFieldEnhanced 值为 1：
  — 如果当前块所在的场为顶场，并且当前块的后向参考场为底场，则 deltaBw=2。
  — 如果当前块所在的场为底场，并且当前块的后向参考场为顶场，则 deltaBw=-2。
- 如果 mvRef_x 小于 0，则 mvBw_x = ((16384/BlockDistanceRef)×(1-mvRef_x×BlockDistanceBw)-1)>>14；否则 mvBw_x = -(((16384/BlockDistanceRef) × (1+mvRef_x×BlockDistanceBw)-1)>>14)。
- 如果(mvRef_y + delta1)小于 0，则 mvBw_y = (((16384/BlockDistanceRef)×(1-(mvRef_y+ delta1)×BlockDistanceBw)-1)>>14) - deltaBw；否则 mvBw_y = -(((16384/BlockDistanceRef) × (1+(mvRef_y+ delta1)×BlockDistanceBw)-1)>>14) - deltaBw。

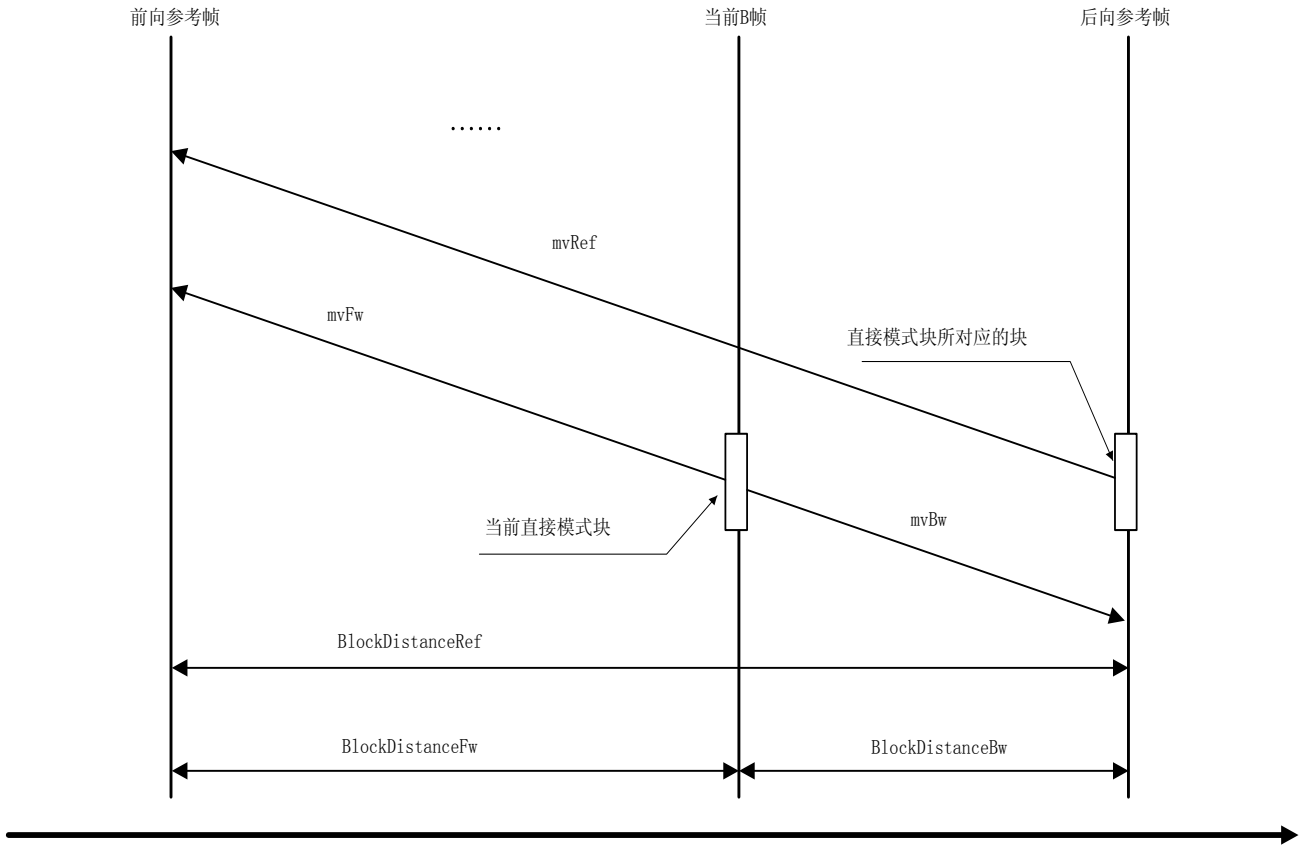
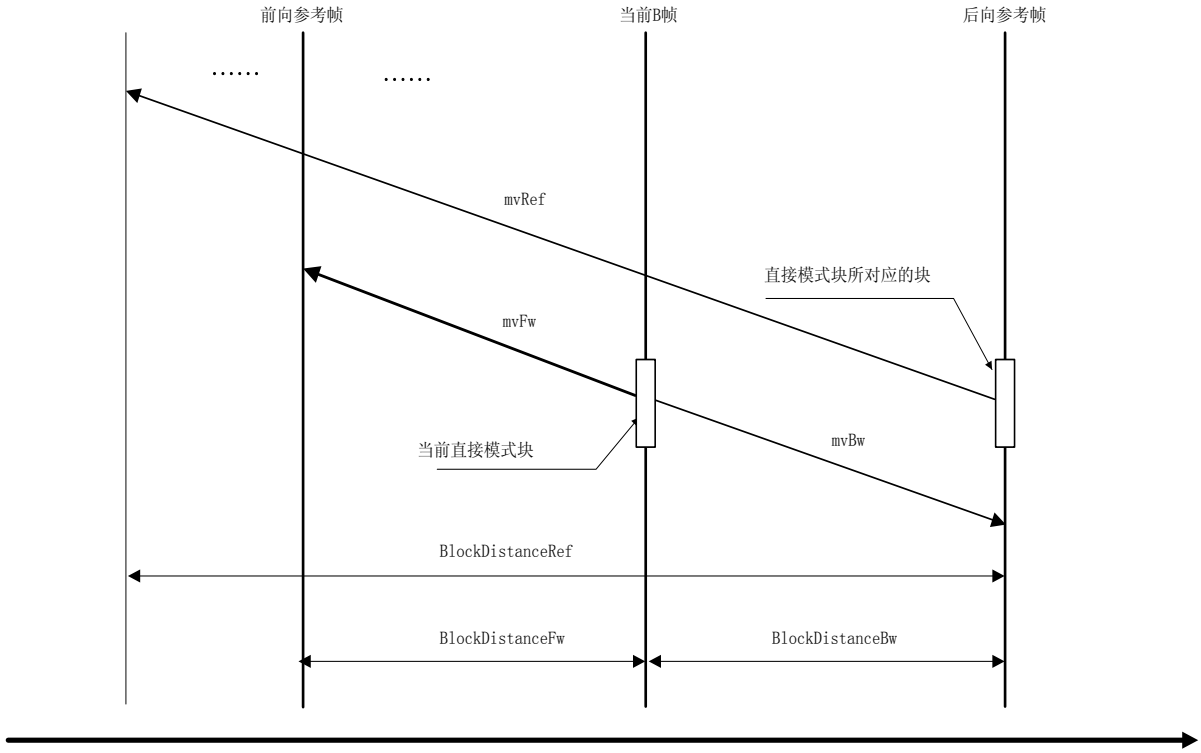
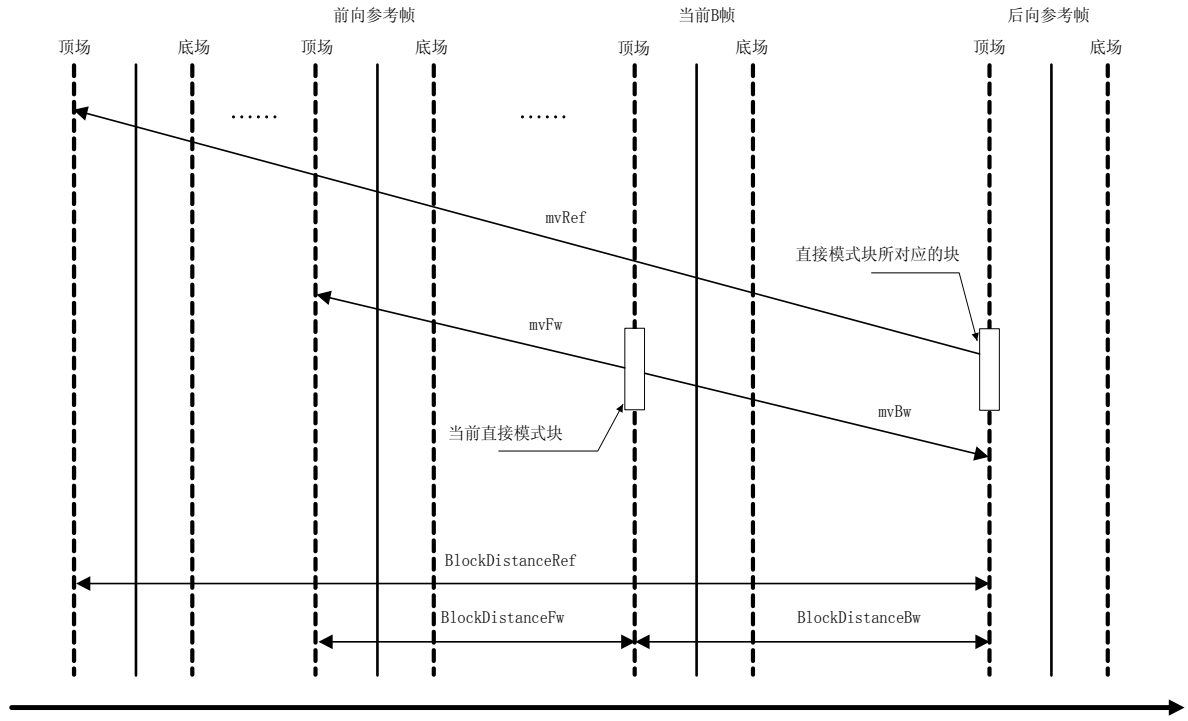


图24　直接模式的运动矢量推导过程 1



当前帧的PictureStructure等于1时，直接或跳过模式块所对应的块指向标记为1的参考图像

图25　直接模式的运动矢量推导过程 2



当前帧的PictureStructure等于0时，直接或跳过模式块所对应的块指向标记为3的参考图像

图26　直接模式的运动矢量推导过程 3

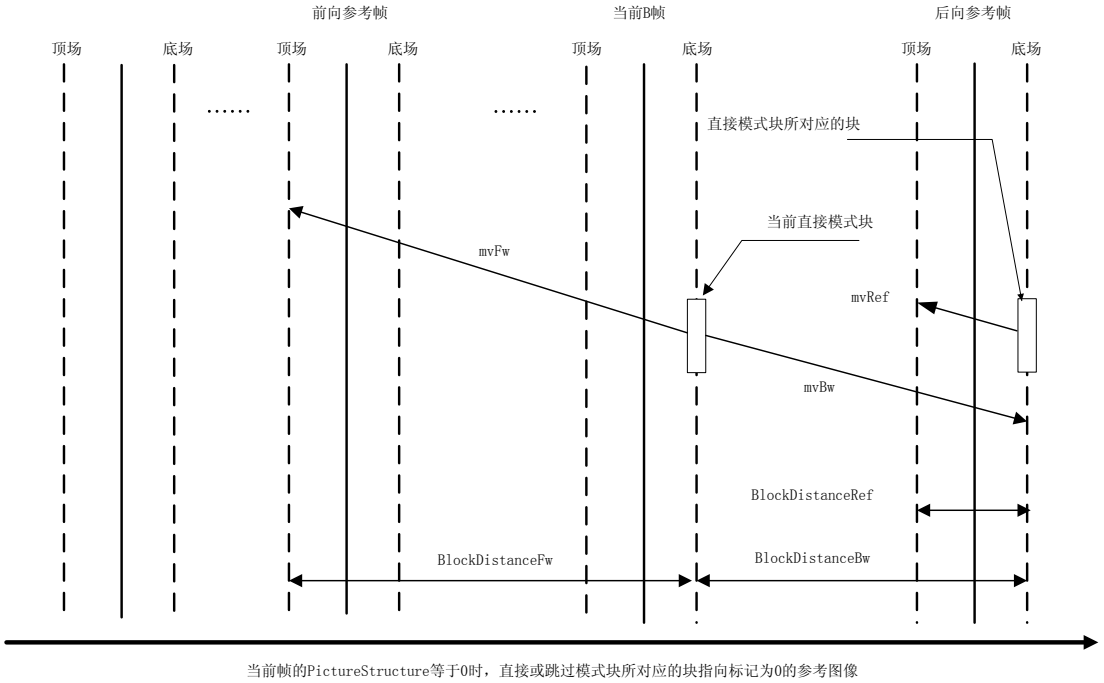


图27　直接模式的运动矢量推导过程 4

c)　如果当前块类型为对称模式，运动矢量按照下面定义的方法导出：

前向参考索引由位流中的 mb_reference_index 解码得到。如果 PictureStructure 等于“1”，则后向参考索引与前向参考索引相等；否则，后向参考索引等于 1 减去前向参考索引。对称模式块的前向运动矢量 mvFw 按照 9.4.6 的方法得到，当前块的后向运动矢量 mvBw(mvBw_x, mvBw_y)为：

$$mvBw\_x = -((mvFw\_x \times BlockDistanceBw \times (512 / BlockDistanceFw) + 256) >> 9)$$

$$mvBw\_y = -((mvFw\_y \times BlockDistanceBw \times (512 / BlockDistanceFw) + 256) >> 9)$$

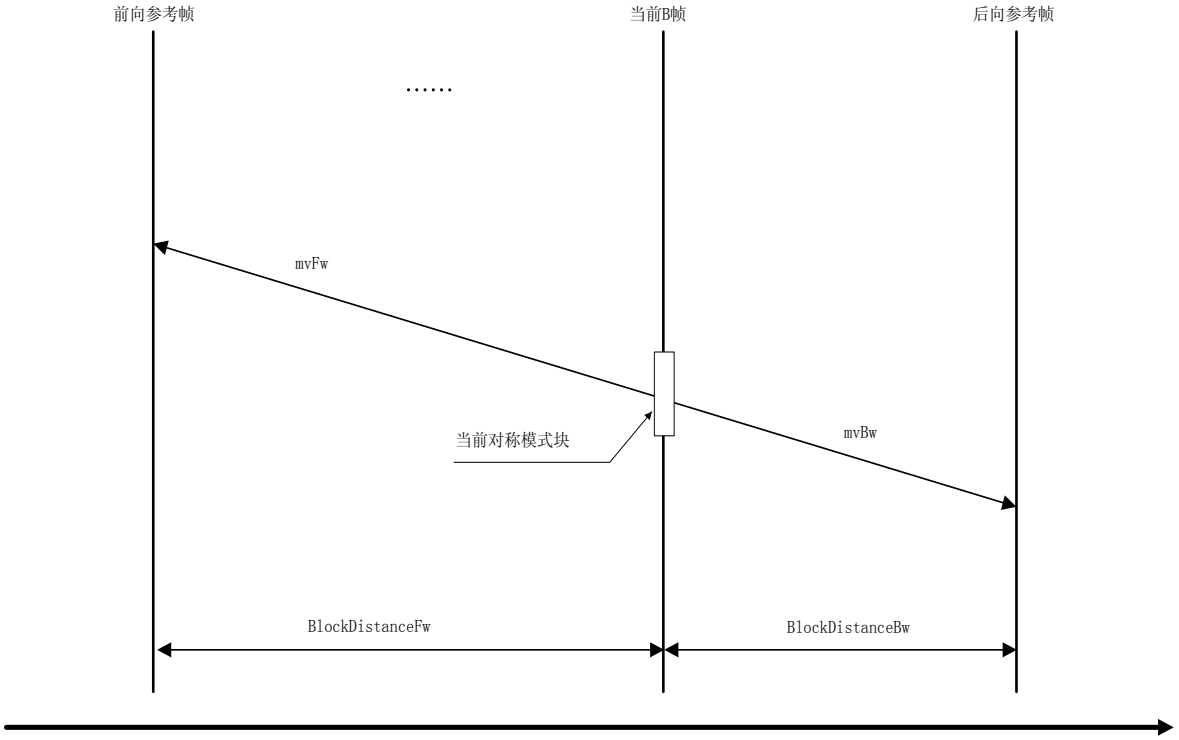


图28　对称模式的运动矢量推导过程 1

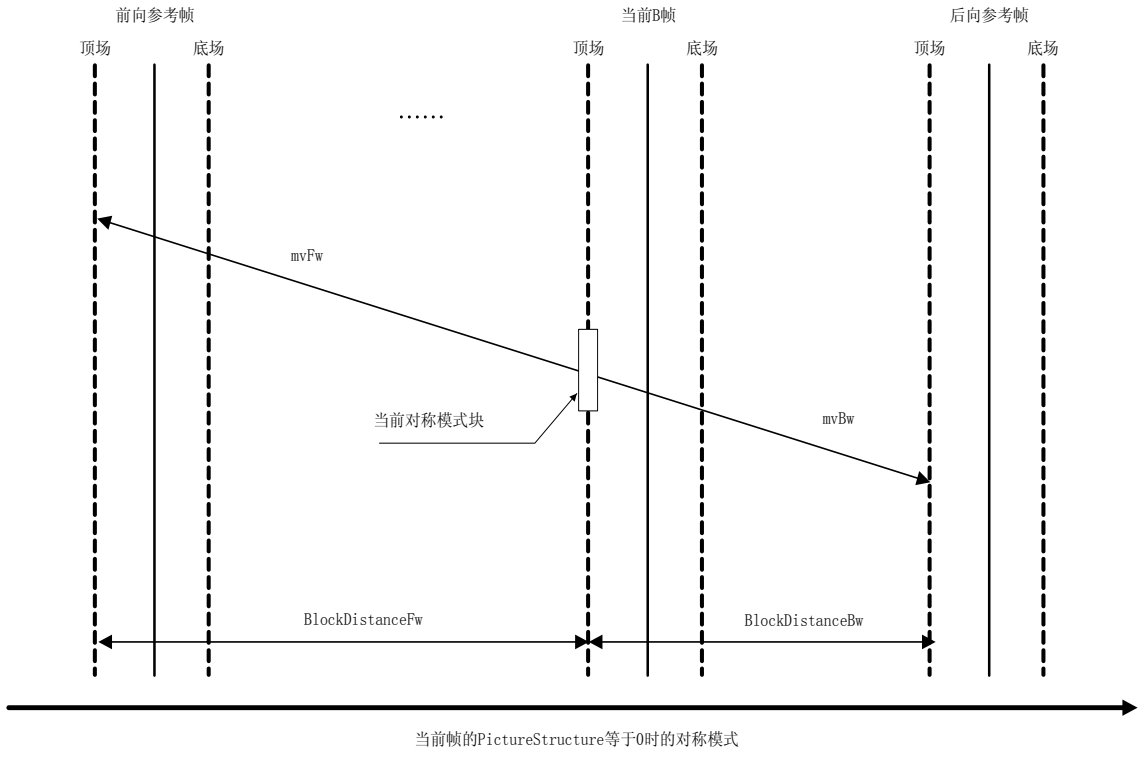


图29　对称模式的运动矢量推导过程 2

按本条定义的方法导出的亮度运动矢量的范围应符合6.2.5规定。

9.9.2 参考样本的导出过程

9.9.2.1 概述

根据运动矢量对亮度进行1/2样本、1/4样本的插值，得到相应的参考样本。

如果在插值过程中所参考的整数样本在参考图像外，应用该图像内距离参考样本最近的整数样本（边缘或角样本）代替，即运动矢量能指向参考图像外的样本。

9.9.2.2 亮度样本插值过程

图30给出了参考图像（或场）整数样本、1/2样本和1/4样本的位置，其中用大写字母标记的为整数样本位置，用小写字母标记的为1/2和1/4样本位置。

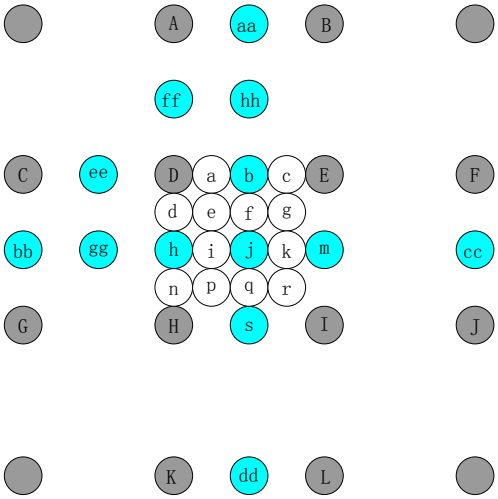


图30 整数样本、1/2 样本和 1/4 样本的位置

1/2样本位置的预测值通过4抽头滤波器F_1(-1, 5, 5, -1)计算得到。1/4样本位置的预测值通过4抽头滤波器F_2(1, 7, 7, 1)计算得到。

1/2样本的计算过程如下：

a) 1/2 样本 b：首先用 F1 对水平方向上最近的 4 个整数样本滤波，得到中间值 b′=(-C+5D+5E-F)；最终的预测值 b=Clip1((b′+4)>>3)。

b) 1/2 样本 h：首先用 F1 对垂直方向上最近的 4 个整数样本滤波，得到中间值 h=(-A+5D+5H-K)；最终的预测值 h=Clip1((h+4)>>3)。

c) 1/2 样本 j：首先用 F1 在水平或垂直方向上对最近的 4 个 1/2 样本中间值滤波，得到中间值 j=(-bb+5h+5m-cc)，或者 j=(-aa+5b+5s-dd)。其中 aa，dd 和 s 是相应位置 1/2 样本中间值（用 F1 在水平方向滤波得到），bb′，cc′和 m′是相应位置 1/2 样本中间值（用 F1 在垂直方向滤波得到）。最终的预测值 j=Clip1((j+32)>>6)。采用水平方向或垂直方向滤波得到的值相同。

1/4样本的计算过程如下：

a) 1/4样本a：首先用F_2在水平方向上对ee′，D′，b′和E′四个值滤波，得到中间值a′=(ee′+7D′+7b′+E′)；最终的预测值 a=Clip1((a′+64)>>7)。其中 ee′和 b′是相应位置 1/2 样本中间值，D′和 E′是相应位置整数样本放大 8 倍的值。1/4 样本 c 的插值过程与 a 的插值过程相同。

b) 1/4 样本 d：首先用 F2 在垂直方向上对 ff′，D′，h′和 H′四个值滤波，得到中间值

d′=(ff′+7D′+7h′+H′)；最终的预测值 d=Clip1((d′+64)>>7)。其中 ff′和 h′是相应位置 1/2 样本中间值，D′和 H′是相应位置整数样本放大 8 倍的值。1/4 样本 n 的插值过程与 d 的插值过程相同。

c) 1/4 样本 i：首先用 F2 在水平方向上对 gg′，h″，j′和 m″四个值滤波，得到中间值 i′=(gg′+7h″+7j′+m″)；最终的预测值 i=Clip1((i′+512)>>10)。其中 gg′和 j′是相应位置 1/2 中间值，h″和 m″是相应位置 1/2 样本中间值放大 8 倍的值。1/4 样本 k 的插值过程与 i 的插值过程相同。

d) 1/4 样本 f：首先用 F2 在垂直方向上对 hh′，b″，j′和 s″四个值滤波，得到中间值 f′=(hh′+7b″+7j′+s″)；最终的预测值 f=Clip1((f′+512)>>10)。其中 hh′和 j′是相应位置 1/2 中间值，b″和 s″是相应位置 1/2 样本中间值放大 8 倍的值。1/4 样本 q 的插值过程与 f 的插值过程相同。

e) 1/4 样本 e，g，p 和 r：

e = Clip1((D″ + j′+ 64) >> 7)

g = Clip1((E″ + j′+ 64) >> 7)

p = Clip1((H″ + j′ + 64) >> 7)

r = Clip1((I″ + j′+ 64) >> 7)

其中D',E',H'和I'是相应位置整数样本放大64倍的值，j'是相应位置1/2样本中间值。

根据当前块运动矢量mvE查表63得到预测样本矩阵的元素predMatrix[x,y]。表63中xFracL等于mvE的水平分量mvE_x&3, yFracL等于mvE的垂直分量mvE_y&3。

表63　预测样本矩阵元素

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

9.9.2.3　色度样本插值过程

色度样本插值使用与对应亮度块的运动矢量mvE（mvE的水平分量为mvE_x，垂直分量为mvE_y）对应的运动矢量mvC。mvC的水平分量为mvC_x，垂直分量为mvC_y。mvC的基本单位为1/8样本。mvC的范围应符合6.2.4规定。如果chroma_format的值为“01”，则mvC_x的值等于mvE_x，mvC_y的值等于mvE_y；如

果chroma_format的值为“10”，则mvC_x的值等于mvE_x，mvC_y的值等于mvE_y × 2。色度样本插值如图31所示，A，B，C，D是被插值样本周围的整数样本值，dx与dy分别是预测样本与*A*的水平和垂直距离。dx等于mvC_x & 7，dy等于mvC_y & 7。这里各变量与参考样本的位置关系如图31所示。

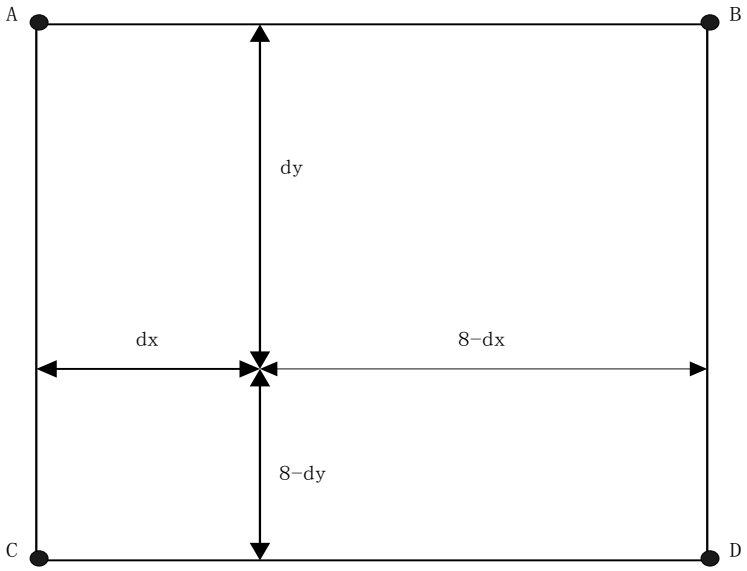


图31　色度插值

预测样本矩阵的元素predMatrix[*x*,*y*]根据下式计算：

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

9.9.3　加权预测

如果当前宏块的WeightingPrediction的值为1并且不是I宏块类型，应使用本条定义的方法进行加权预测。

如果当前宏块是P宏块类型，当前块的参考索引值为i，亮度预测样本矩阵计算如下：

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

如果当前宏块是P宏块类型，当前块的参考索引值为i，色度预测样本矩阵计算如下：

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

如果当前宏块是B宏块类型，当前块的前向参考索引值为i，前向亮度预测样本矩阵计算如下：

predMatrixFw[*x*,*y*]=Clip1(((predMatrixFw[*x*,*y*]×luma_scale[i×2]+16)>>5)+luma_shift[i×2])

如果当前宏块是B宏块类型，当前块的后向参考索引值为j，后向亮度预测样本矩阵计算如下：

predMatrixBw[*x*,*y*]=Clip1(((predMatrixBw[*x*,*y*]×luma_scale[j×2+1]+16)>>5)+
luma_shift[j×2+1])

如果当前宏块是B宏块类型，当前块的前向参考索引值为i，前向色度预测样本矩阵计算如下：

predMatrixFw[*x*,*y*]=Clip1(((predMatrixFw[*x*,*y*]×chroma_scale[i×2]+16)>>5)+
chroma_shift[i×2])

如果当前宏块是B宏块类型，当前块的后向参考索引值为j，后向色度预测样本矩阵计算如下：

predMatrixBw[*x*,*y*]=Clip1(((predMatrixBw[*x*,*y*]×chroma_scale[j×2+1]+16)>>5)+
chroma_shift[j×2+1])

9.10　重建

如果当前块是I或P宏块类型，重建系数矩阵RecMatrix计算如下：

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

如果当前块是B宏块类型，并且使用两帧参考图像，重建系数矩阵RecMatrix计算如下：

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

如果当前块是B宏块类型，并且只使用前向参考图像，重建系数矩阵RecMatrix计算如下：

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

如果当前块是B宏块类型，并且只使用后向参考图像，重建系数矩阵RecMatrix计算如下：

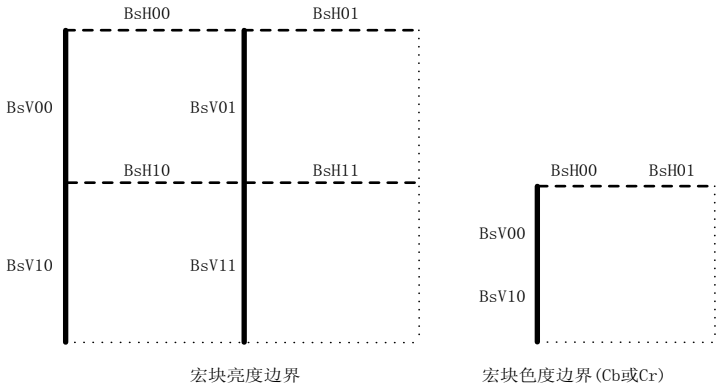
RecMatrix[*x*,*y*] = Clip1(predMatrixBw[*x*,*y*] + ResidueMatrix[*x*,*y*])

其中predMatrixFw是前向参考图像的预测样本矩阵，predMatrixBw是后向参考图像的预测样本矩阵。

9.11 环路滤波

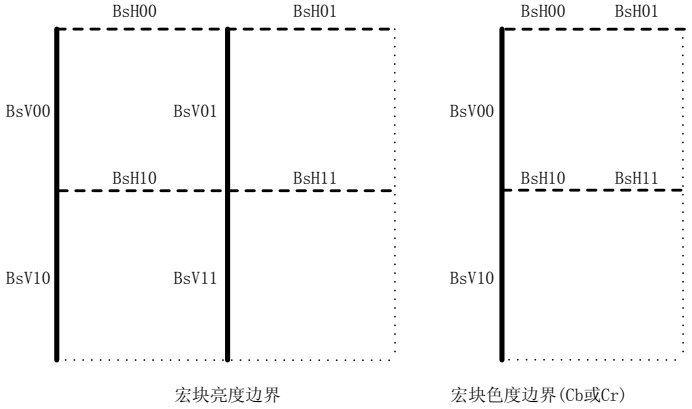
9.11.1 概述

8×8亮度块之间的每条边界有一个“边界强度”Bs，色度块的边界强度用对应位置亮度块边界的Bs代替，如图32和图33所示。如果Bs等于0则不要对边界滤波，否则根据局部样本值的特性和Bs值对边界滤波。



注：实线为块垂直边界，粗虚线为水平边界，细虚线为下一宏块待滤波的边界。

图32 4:2:0 宏块中需要滤波的边界示意



注：实线为块垂直边界，粗虚线为水平边界，细虚线为下一宏块待滤波的边界。

图33 4:2:2 宏块中需要滤波的边界示意

除图像边界及条带的边界之外，宏块与相邻宏块的上边界和左边界，以及宏块内部各个8×8块的边界都应进行滤波。

环路滤波以宏块为单位，按照解码顺序依次处理。图像中每个宏块的滤波过程如下：

对亮度和色度分别做环路滤波，如图32和图33所示，首先从左到右对垂直边界滤波，然后从上到下对水平边界滤波。当前宏块的上边或者左边的样本值可能在以前的宏块环路滤波过程中已经被修改，当前宏块的环路滤波的输入为这些可能被修改的样本值，并且当前宏块环路滤波可能进一步修改这些样本值。当前宏块垂直边界滤波过程中修改的样本值作为水平边界滤波过程的输入。

帧内预测使用环路滤波前的重建图像样本值。

9.11.2 边界滤波强度的推导过程

根据宏块类型、宏块中8×8亮度块的运动矢量，按如下方法计算Bs的值：

a) 如果边界两边的两个 8×8 块有一个或两个都属于帧内预测宏块，则 Bs 等于“2”。

b) 否则，如果当前图像是 P 帧、I 帧的第二场，并且满足以下两个条件中的任一个，则 Bs 等于“1”；如果 当前图像是 P 帧、I 帧的第二场，并且以下两个条件都不满足，则 Bs 等于“0”：

1) 两个块的参考图像不同；

2) 两个块的参考图像相同，但是两个运动矢量分量中任一个分量的差值大于或等于一个整像素。

c) 否则，按以下方法计算 Bs 的值：

1) 如果两个块 p 和 q 的前向索引参考值和后向索引参考值分别相等：

——以下两个条件中任一个成立，则 Bs 等于“1”：

- 两个块的前向运动矢量分量中任一个分量的差值大于或等于一个整像素；
- 两个块的后向运动矢量分量中任一个分量的差值大于或等于一个整像素。

——否则，Bs 等于“0”。

2) 否则，Bs 等于“1”。

9.11.3 块边界阈值的推导过程

图34表示块p和块q在水平或垂直边界两侧的6个样本点（边界用黑色粗线表示）。用P_0、P_1、Q_0和Q_1分别表示p_0、p_1、q_0和q_1滤波后的样本值。

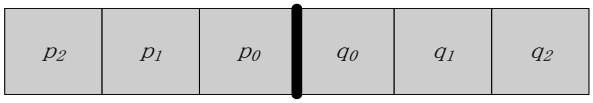


图34 8×8 块水平或垂直边界样本

如果下式为真，对边界样本滤波：

$$Bs\ !=\ 0\ \&\&\ Abs(p_0 - q_0) < \alpha\ \&\&\ Abs(p_1 - p_0) < \beta\ \ \&\&\ Abs(q_1 - q_0) < \beta$$

其中α和β为块边界阈值，可以根据两个块的QP平均值QP_{av}，以及AlphaCOffset和BetaOffset计算查表索引IndexA和IndexB。

两个块的QP平均值QP_{av}为：

$$QP_{av} = (QP_p + QP_q + 1) >> 1$$

对于亮度分量，QP_p和QP_q应使用宏块的量化参数；对于色度分量，QP_p和QP_q应使用色度块的量化参数。

索引IndexA和IndexB为：

$$IndexA = Clip3(0,\ 63,\ QP_{av} + AlphaCOffset)$$

$$IndexB = Clip3(0,\ 63,\ QP_{av} + BetaOffset)$$

根据索引IndexA和IndexB与阈值α和β间的对应关系，由表64得到α、β的取值，根据IndexA查表64得到α，根据IndexB查表64得到β。

表64　块边界阈值α和β与 IndexA 和 IndexB 的关系

索引	α的值	β的值
0	0	0
1	0	0
2	0	0
3	0	0
4	0	0
5	0	0
6	1	1
7	1	1
8	1	1
9	1	1
10	1	1
11	2	1
12	2	1
13	2	2
14	3	2
15	3	2
16	4	2
17	4	2
18	5	3
19	5	3
20	6	3
26	12	5
27	13	5
28	15	5
29	16	5
30	18	6
31	20	6
32	22	6
33	24	7
34	26	7
35	28	7
36	30	8
37	33	8
38	33	8
39	35	9

表 64（续）

索引	α的值	β的值
40	35	9
41	36	10
42	37	10
43	37	11
44	39	11
45	39	12
46	42	13
47	44	14
48	46	15
49	48	16
50	50	17
51	52	18
52	53	19
53	54	20
54	55	21
55	56	22
56	57	23
57	58	23
58	59	24
59	60	24
60	61	25
61	62	25
62	63	26
63	64	27

9.11.4 Bs 等于 2 时的边界滤波过程

首先定义ap=Abs(p_2 - p_0)，aq=Abs(q_2 - q_0)。

对亮度块边界两边的样本p_0、p_1、q_0和q_1的滤波过程如下：

```
if ( ap < β  &&  Abs( p0  -  q0 ) < ((α >> 2) + 2) ) {
    P0 = ( p1 + 2 × p0 + q0 + 2 ) >> 2
    P1 = ( 2 × p1 + p0 + q0 + 2 ) >> 2
}
else
    P0 = ( 2 × p1 + p0 + q0 + 2 ) >> 2
if ( aq < β  &&  Abs( p0 - q0 ) < ((α >> 2) + 2) ) {
    Q0 = (q1 + 2 × q0 + p0 + 2) >> 2
    Q1 = (2 × q1 + q0 + p0 + 2) >> 2
}
else
```

$$Q_0 = (2 \times q_1 + q_0 + p_0 + 2) >> 2$$

其中P_0和Q_0分别为p_0和q_0滤波后的值。如果在上面的滤波过程中P_1（或Q_1）不被赋值，则不对p_1（或q_1）滤波；否则，P_1或Q_1为p_1或q_1滤波后的值。

色度块边界两边的样本p_0和q_0的采用同样的方法滤波，不对p_1和q_1滤波。

9.11.5 Bs 等于 1 时的边界滤波过程

边界滤波强度Bs的值为1时，对p_0和q_0滤波的计算过程如下(P_0和Q_0分别为p_0和q_0滤波后的值)：

$$delta = \text{Clip3}(-C, C, (((q_0 - p_0) \times 3 + (p_1 - q_1) + 4) >> 3))$$

$$P_0 = \text{Clip1}(p_0 + delta)$$

$$Q_0 = \text{Clip1}(q_0 - delta)$$

然后判断是否需要对 p_1和 q_1滤波，计算过程如下：

a) 如果为色度边界，不对 p1 和 q1 滤波。

b) 如果在亮度边界处有 ap 小于β，则对 p1 滤波，滤波后的值为

$$P_1 = \text{Clip1}(p_1 + \text{Clip3}(-C, C, (((P_0 - p_1) \times 3 + (p_2 - Q_0) + 4) >> 3)))$$

c) 如果在亮度边界处有 aq 小于β，则对 q1 滤波，滤波后的值为

$$Q_1 = \text{Clip1}(q_1 - \text{Clip3}(-C, C, (((q_1 - Q_0) \times 3 + (P_0 - q_2) + 4) >> 3)))$$

上述滤波过程中，*ap*和*aq*的定义见9.11.3，*C*称为滤波裁减参数，*C*与*IndexA*之间的关系见表65。

表65 滤波裁减参数 *C* 与 *IndexA* 的关系

IndexA	*C*的值
0	0
1	0
2	0
3	0
4	0
5	0
6	0
7	0
8	0
9	0
10	0
11	0
12	0
13	0
14	0
15	0
16	1
17	1
18	1
19	1
20	1
21	1

表 65（续）

IndexA	*C*的值
22	1
23	1
24	1
25	1
26	1
27	1
28	1
29	1
30	2
31	2
32	2
33	2
34	2
35	2
36	2
37	2
38	3
39	3
40	3
41	3
42	3
43	3
44	3
45	4
46	4
47	4
48	5
49	5
50	5
51	6
52	6
53	6
54	7
55	7
56	7
57	7
58	8
59	8

表 65（续）

IndexA	*C*的值
60	8
61	9
62	9
63	9

附 录 A
（规范性附录）
伪起始码

本附录定义防止在位流中出现伪起始码的方法。起始码的形式、含义，以及为了使起始码字节对齐而进行填充的方法见7.1.1和5.8.2。

为了防止出现伪起始码，编码时应按照以下方法处理：写入一位时，如果该位是一个字节的第二最低有效位，检查该位之前写入的22位，如果这22位都是“0”，在该位之前插入“10”，该位成为下一个字节的最高有效位。

解码时应按以下方法处理：每读入一个字节时，检查前面读入的两个字节和当前字节，如果这三个字节构成位串“0000 0000 0000 0000 0000 0010”，丢弃当前字节的最低两个有效位。丢弃一个字节最低两个有效位可采用任意等效的方式，本部分不做规定。

在编码和解码时对于序列头、序列显示扩展、版权扩展、用户数据、摄像机参数扩展、图像显示扩展、区域扩展、视频扩展数据保留字节中的数据不应采用上述方法。

附 录 B
（规范性附录）
类和级

B.1 概述

类和级提供了一种定义本部分的语法和语义的子集的手段。类和级对位流进行了各种限制，同时也就规定了对某一特定位流解码所需要的解码器能力。类是本部分规定的语法、语义及算法的子集。符合某个类规定的解码器应完全支持该类定义的子集。级是在某一类下对语法元素和语法元素参数值的限定集合。在给定类的情况下，不同级往往意味着对解码器能力和存储器容量的不同要求。

本附录描述了不同类和级所对应的各种限制。所有未被限定的语法元素和参数可以取任何本部分所允许的值。如果一个解码器能对某个类和级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类和级上符合本部分。如果一个位流中不存在某个类和级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类和级所允许的范围，则认为此位流在这个类和级上符合本部分。

profile_id和level_id定义了位流的类和级。

B.2 类

本部分定义的类见表B.1。

表B.1 类

profile_id的值	类
0x00	禁止
0x20	见GB/T 20090.2-2006的定义
0x48	广播类（Broadcasting profile）
其他	保留

对于一个给定的类，不同的级支持相同的语法子集。

广播类的位流应满足以下条件：

a) profile_id 的值应为 0x48；
b) progressive_sequence 的值为“0”时，整个视频序列中所有图像的 top_field_first 的值均应相同；
c) advanced_pred_mode_disable 的值应为“1”；
d) chroma_format 的值应为“01”或“10”；
e) sample_precision 的值应为“001”；
f) 视频序列起始码与随后第一个视频序列结束码之间，或视频序列起始码与随后第一个视频编辑码之间所有编码图像的 PictureStructure 的值均应相同；
g) B.3.2 规定的级限制；
h) 广播类支持的级包括：2.0.0.08.30、2.1.0.08.30、2.1.0.08.15、4.0.0.08.30、4.0.2.08.60、4.2.0.08.30、6.0.0.08.60、6.0.1.08.60、6.0.3.08.60、6.0.5.08.60 和 6.2.0.08.60。

B.3 级

B.3.1 本部分定义的级

本部分定义的级见表B.2。

表B.2 级

level_id的值	级
0x00	禁止
0x10	2.0.0.08.30
0x12	2.1.0.08.15
0x14	2.1.0.08.30
0x20	4.0.0.08.30
0x22	4.2.0.08.30
0x2A	4.0.2.08.60
0x40	6.0.0.08.60
0x41	6.0.1.08.60
0x42	6.2.0.08.60
0x44	6.0.3.08.60
0x46	6.0.5.08.60
其他	保留

B.3.2 与类无关的级限制

对于所有类，每个宏块编码后最大二进制位数的限制见表B.3，其中n是样本点精度。

表B.3 宏块编码后最大二进制位数

图像格式	宏块编码后最大二进制位数
4:2:0	128 + 256×n×1.5
4:2:2	128 + 256×n×2

级2.0.0.08.30、2.1.0.08.15和2.1.0.08.30的参数限制见表B.4。

表B.4 级 2.0.0.08.30、2.1.0.08.15 和 2.1.0.08.30 的参数限制

参 数	级		
	2.0.0.08.30	2.1.0.08.15	2.1.0.08.30
每行最大样本数	352	352	352
每帧最大行数	288	288	288
每秒最大帧数	30	15	30
样本精度（位）	8	8	8
亮度样本速率	3 041 280	1 520 640	3 041 280
最大比特率（位每秒）	1 000 000	1 500 000	2 500 000
每秒最大二元符号数	8 110 080	12 165 120	20 275 200

表B.4（续）

参　数	级		
	2.0.0.08.30	2.1.0.08.15	2.1.0.08.30
BBV缓冲区大小（位）	122 880	196 608	311 296
每帧最大宏块个数	396	396	396
每秒最大宏块个数	11 880	5 940	11 880
帧编码时最大垂直运动矢量范围（亮度样本数）	[-128,+127.875]	[-128,+127.875]	[-128,+127.875]
场编码时最大垂直运动矢量范围（亮度样本数）	-	-	-
最大水平运动矢量范围（亮度样本数）	[-2048,+2047.875]	[-2048,+2047.875]	[-2048,+2047.875]
图像格式	4:2:0	4:2:0	4:2:0

级4.0.0.08.30和4.2.0.08.30的参数限制见表B.5。

表B.5　级 4.0.0.08.30 和 4.2.0.08.30 的参数限制

参　数	级　别	
	4.0.0.08.30	4.2.0.08.30
每行最大样本数	720	720
每帧最大行数	576	576
每秒最大帧数	30	30
样本精度（位）	8	8
亮度样本速率	10 368 000	10 368 000
最大比特率（位每秒）	10 000 000	15 000 000
每秒最大二元符号数	82 944 000	110 592 000
BBV缓冲区大小（位）	1 228 800	1 851 392
每帧最大宏块个数	1 620	1 620
每秒最大宏块个数	40 500	40 500
帧编码时最大垂直运动矢量范围（亮度样本数）	[-256,+255.875]	[-256,+255.875]
场编码时最大垂直运动矢量范围（亮度样本数）	[-128,+127.875]	[-128,+127.875]
最大水平运动矢量范围（亮度样本数）	[-2048,+2047.875]	[-2048,+2047.875]
图像格式	4:2:0	4:2:2

级4.0.0.10.30、4.0.0.12.30和4.0.2.08.60的参数限制见表B.6。

表B.6　级 4.0.2.08.60 的参数限制

参　数	级　别
	4.0.2.08.60
每行最大样本数	720
每帧最大行数	576
每秒最大帧数	60
样本精度（位）	8
亮度样本速率	20 736 000

表B.6（续）

参　数	级　别
	4.0.2.08.60
最大比特率（位每秒）	20 000 000
每秒最大二元符号数	165 888 000
BBV缓冲区大小（位）	10 485 760
每帧最大宏块个数	1 620
每秒最大宏块个数	81 000
帧编码时最大垂直运动矢量范围（亮度样本数）	[-256,+255.875]
场编码时最大垂直运动矢量范围（亮度样本数）	[-128,+127.875]
最大水平运动矢量范围（亮度样本数）	[-2048,+2047.875]
图像格式	4:2:0

级6.0.0.08.60和6.0.1.08.60的参数限制见表B.7。

表B.7　级 6.0.0.08.60 和 6.0.1.08.60 的参数限制

参　数	级　别	
	6.0.0.08.60	6.0.1.08.60
每行最大样本数	1 920	1 920
每帧最大行数	1 152	1 152
每秒最大帧数	60	60
样本精度（位）	8	8
亮度样本速率	62 668 800	62 668 800
最大比特率（位每秒）	20 000 000	50 000 000
每秒最大二元符号数	200 540 160	501 350 400
BBV缓冲区大小（位）	2 457 600	62 488 576
每帧最大宏块个数	8 160	8 160
每秒最大宏块个数	244 800	244 800
帧编码时最大垂直运动矢量范围（亮度样本数）	[-512,+511.875]	[-512,+511.875]
场编码时最大垂直运动矢量范围（亮度样本数）	[-256,+255.875]	[-256,+255.875]
最大水平运动矢量范围（亮度样本数）	[-2048,+2047.875]	[-2048,+2047.875]
图像格式	4:2:0	4:2:0

级6.0.3.08.60、6.0.5.08.60和6.2.0.08.60的参数限制见表B.8。

表B.8　级 6.0.3.08.60、6.0.5.08.60 和 6.2.0.08.60 的参数限制

参　数	级　别		
	6.0.3.08.60	6.0.5.08.60	6.2.0.08.60
每行最大样本数	1 920	4 096	1 920
每帧最大行数	1 152	2 048	1 152
每秒最大帧数	60	60	60
样本精度（位）	8	8	8

表 B.8（续）

参　数	级　别		
	6.0.3.08.60	6.0.5.08.60	6.2.0.08.60
亮度样本速率	125 337 600	251 658 240	62 668 800
最大比特率（位每秒）	100 000 000	200 000 000	30 000 000
每秒最大二元符号数	1 002 700 800	2 005 401 600	401 080 320
BBV缓冲区大小（位）	62 488 576	249 954 304	3 686 400
每帧最大宏块个数	8 160	32 768	8 160
每秒最大宏块个数	489 600	983 040	244 800
帧编码时最大垂直运动矢量范围（亮度样本数）	[-512,+511.875]	[-1024,+1023.875]	[-512,+511.875]
场编码时最大垂直运动矢量范围（亮度样本数）	[-256,+255.875]	[-512,+511.875]	[-256,+255.875]
最大水平运动矢量范围（亮度样本数）	[-2048,+2047.875]	[-4096,+4095.875]	[-2048,+2047.875]
图像格式	4:2:0	4:2:0	4:2:0或4:2:2

某一级别下 BBV 缓冲区大小是 16384 位的倍数。

与表 B.4、表 B.5、表 B.6、表 B.7 和表 B.8 有关的语法元素包括：horizontal_size、vertical_size、frame_rate_code、bbv_buffer_size、chroma_format。

如果 aec_enable 的值为“1”，一帧图像中二元符号的最大个数应满足表 B.4、表 B.5、表 B.6、表 B.7 和表 B.8 的限制。

附 录 C
（规范性附录）
位流虚拟参考解码器

C.1 概述

本附录定义了位流虚拟参考解码器（以下简称BBV）。

BBV有一个输入缓冲区，称为BBV缓冲区。编码数据按照C.3.1定义的方式进入BBV缓冲区，按照C.3.2定义的方式移出BBV缓冲区。符合本部分的位流不应导致BBV缓冲区上溢。如果low_delay的值为“0”，符合本部分的位流不应导致BBV缓冲区下溢；如果low_delay的值为“1”，符合本部分的位流可能导致BBV缓冲区下溢，此时应按C.3.2.2定义的方式处理。

本附录中所有的运算都是实数运算，不存在舍入误差，例如BBV缓冲区中的位数不必是整数值。

C.2 约定

C.2.1 约定一

BBV和视频编码器的时钟频率和帧率相同，并且同步操作。

C.2.2 约定二

BBV缓冲区的大小为BBS，单位为二进制位。

C.2.3 约定三

编码数据输入BBV缓冲区的最大速率R_{max}（单位为位每秒）按式（C.1）计算。

$$R_{max} = \text{BitRate} \times 400 \quad \text{(C.1)}$$

式中：

BitRate——以400bit/s为单位计算视频位流的比特率；

R_{max} ——编码数据输入BBV缓冲区的最大速率。

C.3 基本操作

C.3.1 数据输入

本附录定义了两种方法来计算编码数据进入BBV缓冲区的速率。这两种方法不应同时使用。

C.3.1.1 方法一

C.3.1.1.1 操作过程

在BBV中，第n帧图像的编码数据$f(n)$包括以下数据（如果存在）：

a) 序列头、跟在序列头之后的扩展和用户数据、视频编辑码。这些数据定义为 $b(n)$，$b(n)$与本帧图像起始码之间不应有其他图像的数据；

b) 本帧图像的所有编码数据；

c) 跟在本帧图像头后的图像显示扩展；

d) 跟在本帧图像后的填充数据、视频序列结束码。

如果BbvDelay的值不等于BbvDelayMax，第n帧图像进入BBV缓冲区的速率$R(n)$按式（C.2）计算。

$$R(n)=d_n^*\div(\tau(n)-\tau(n+1)+t(n+1)-t(n)) \quad \text{.....(C.2)}$$

式中：

d^*_n ——从第n帧的起始码后第1个位到第n+1帧的起始码后第1个位之间所有的位数；

$\tau(n)$ ——第n帧的BbvDelay的值，单位为秒（s）；

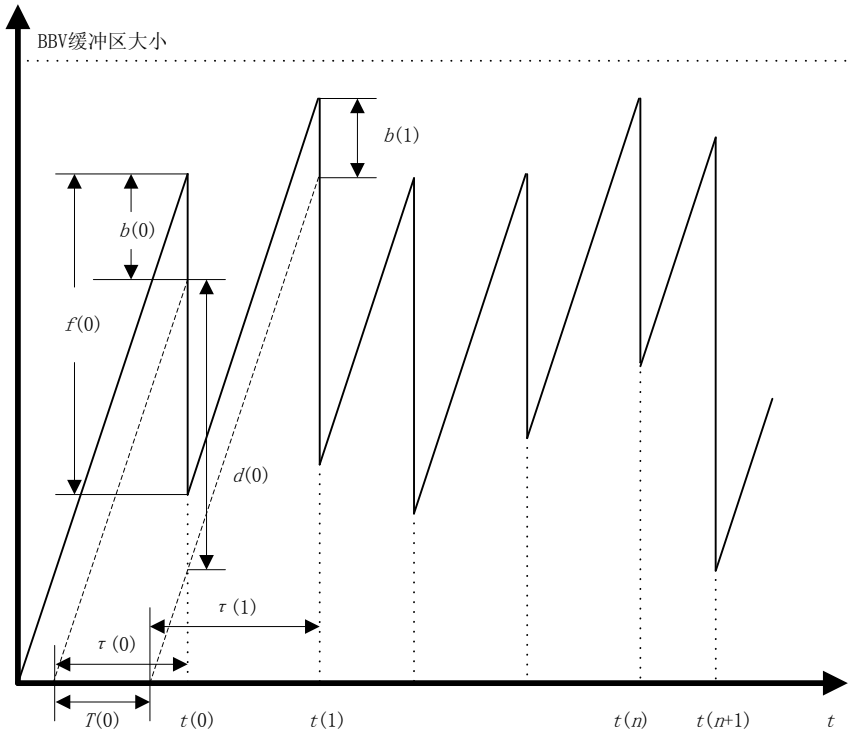
$t(n)$ ——第n帧图像的编码数据从BBV缓冲区移出的时间，单位为秒（s）；

$t(n+1)-t(n)$——第n+1帧和第n帧图像的解码时间间隔，单位为秒（s）；

$R(n)$ ——第n帧图像进入BBV缓冲区的速率。

在视频序列开始和结束时，可能缺少参数来无歧义地确定这个时间间隔，此时可采用下面的方法来处理。

图C.1为第n帧图像的编码数据按方法一进入BBV缓冲区的示意图。



图C.1 BBV 缓冲区占用情况一

C.3.1.1.2 视频序列开始时的歧义性

对视频序列进行随机访问时，序列头后的第1帧图像缺少在此之前的I帧或P帧。在这种情况下，如果位流是系统流的一部分，可从系统流来确定解码时间间隔。

如果还不能无歧义地确定解码时间间隔，就无法确定$R(n)$。此时在一段有限的时间内（这个时间总是小于BbvDelay的最大值）BBV不能准确地确定充满度的变化轨迹，从而无法在整个位流中严格地验证

BBV缓冲区。编码器总是知道在每个重复序列头后$t(n+1)-t(n)$的值，因此也知道如何生成不违反BBV限制的位流。

C.3.1.1.3 视频序列结束时的歧义性

如果一帧图像的编码数据后第1个起始码是视频序列结束码，此时无法确定该帧图像编码数据的位数。在这种情况下，应存在一个输入速率，这个速率不应导致BBV缓冲区上溢；在low_delay的值为“1”时，这个速率也不应导致缓冲区下溢。该速率应小于在视频序列头中定义的最大速率。

视频序列的第1帧图像的起始码前的所有数据和这个起始码输入BBV缓冲区后，每一帧图像的编码数据应在由位流中的BbvDelay规定的时间内输入BBV缓冲区，并在这个时间开始解码处理。输入速率由公式C.2确定。

所有位流的$R(n)$均不应大于R_{max}。

在CBR情况下，视频序列中的$R(n)$在BbvDelay允许的精度范围内是一个常数。

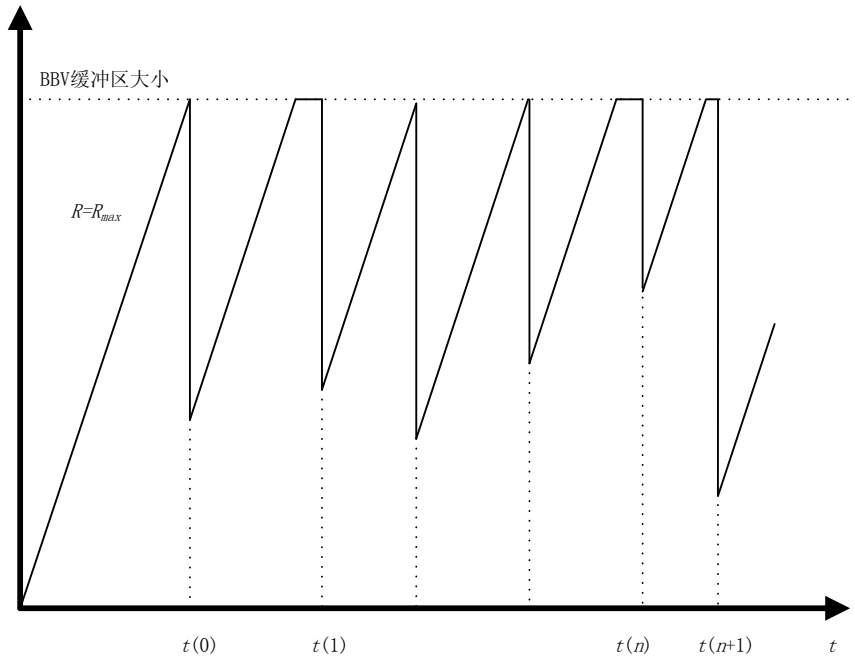
C.3.1.2 方法二

如果BbvDelay的值为BbvDelayMax，编码数据按以下方式输入BBV缓冲区：

——如果BBV缓冲区没有充满，数据以速率R_{max}输入缓冲区；

——如果BBV缓冲区充满，则数据不能进入该缓冲区直到缓冲区中的部分数据被移出。

视频序列的第1帧图像的起始码前的所有数据和这个起始码输入BBV缓冲区后，数据继续输入BBV缓冲区直到缓冲区充满，并在这个时间开始解码处理，见图C.2。



图C.2 BBV 缓冲区占用情况二

C.3.2 数据移出

C.3.2.1 非低延迟

low_delay的值为“0”时，数据按照下面的方式从BBV缓冲区中移出：在每帧图像的解码时刻$t(n)$，如果BBV缓冲区充满度$B(n)$小于$f(n)$则发生下溢，否则移出该帧图像编码数据并解码。

在$t(n)$时刻，即移出第n帧图像的位之前和检测BBV缓冲区之时，该图像的所有编码数据均应在缓冲区中，然后该图像被瞬时移出。

如果low_delay的值为“0”，BBV缓冲区不应发生下溢。

C.3.2.2 低延迟

low_delay的值为“1”时，在移出一帧图像的编码数据之前，要检测BBV缓冲区BbvCheckTimes加1次。如果BbvCheckTimes大于“0”，当前解码图像定义为大图像。

第n帧图像的数据按照下面的方式从BBV缓冲区中移出:每隔一个检测时间间隔检测一次BBV缓冲区，完成BbvCheckTimes加1次检测，此时BBV缓冲区充满度应小于BBS，该图像的所有编码数据均应在缓冲区中，然后该图像被瞬时移出。

如果当前解码图像不是大图像，检测时间间隔在C.4定义。

如果当前解码图像是大图像，检测时间间隔是一个帧时间间隔。

一个视频序列的最后一帧不应是大图像。

C.4 缓冲区检测时间间隔

C.4.1 非低延迟

当low_delay的值为“0”时，视频位流中可包含B帧，会产生进行图像重排序而造成的延迟。

当progressive_sequence的值为“1”并且low_delay的值为“0”时，BBV缓冲区检测时间间隔$t(n+1)-t(n)$是帧率的倒数T的倍数：

a) 如果第 n 帧图像是 repeat_first_field 的值为“0”的 B 帧，检测时间间隔等于 T；
b) 如果第 n 帧图像是 repeat_first_field 的值为“1”的 B 帧，并且 top_field_first 的值为“0”，检测时间间隔等于 2T；
c) 如果第 n 帧图像是 repeat_first_field 的值为“1”的 B 帧，并且 top_field_first 的值为“1”，检测时间间隔等于 3T；
d) 如果第 n 帧图像是 P 帧或 I 帧，并且前一个 P 帧或 I 帧的 repeat_first_field 的值为“0”，检测时间间隔等于 T；
e) 如果第 n 帧图像是 P 帧或 I 帧，并且前一个 P 帧或 I 帧的 repeat_first_field 的值为“1”，top_field_first 的值为“0”，检测时间间隔等于 2T；
f) 如果第 n 帧图像是 P 帧或 I 帧，并且前一个 P 帧或 I 帧的 repeat_first_field 的值为“1”，top_field_first 的值为“1”，检测时间间隔等于 3T。

当progressive_sequence的值为“0”并且low_delay的值为“0”时，BBV缓冲区检测时间间隔$t(n+1)-t(n)$是帧率的倒数T的倍数：

a) 如果第 n 帧图像是 repeat_first_field 的值为“0”的 B 帧，检测时间间隔等于 T；
b) 如果第 n 帧图像是 repeat_first_field 的值为“1”的 B 帧，检测时间间隔等于 1.5T；
c) 如果第 n 帧图像是 P 帧或 I 帧，并且前一个 P 帧或 I 帧的 repeat_first_field 的值为“0”，检测时间间隔等于 T；
d) 如果第 n 帧图像是 P 帧或 I 帧，并且前一个 P 帧或 I 帧的 repeat_first_field 的值为“1”，检测时间间隔等于 1.5T。

C.4.2 低延迟

当progressive_sequence的值为“1”并且low_delay的值为“1”时，BBV缓冲区检测时间间隔$t(n+1)-t(n)$是帧率的倒数T的倍数：

a) 如果第 n 帧图像是 repeat_first_field 的值为“0”的 P 帧或 I 帧，检测时间间隔等于 T；

b) 如果第 n 帧图像是 repeat_first_field 的值为“1”并且 top_field_first 等于“0”的 P 帧或 I 帧，检测时间间隔等于 2T；

c) 如果第 n 帧图像是 repeat_first_field 的值为“1”并且 top_field_first 等于“1”的 P 帧或 I 帧，检测时间间隔等于 3T。

当progressive_sequence的值为“0”并且low_delay的值为“1”时，BBV缓冲区检测时间间隔$t(n+1)-t(n)$是帧率的倒数T的倍数：

a) 如果第 n 帧图像是 repeat_first_field 的值为“0”的 P 帧或 I 帧，检测时间间隔等于 T；

b) 如果第 n 帧图像是 repeat_first_field 的值为“1”的 P 帧或 I 帧，检测时间间隔等于 1.5T。

附 录 D
（规范性附录）
基本熵编码码表

本附录给出了基本熵编码码表。表D.1到表D.20中Run表示量化系数游程，Level表示量化系数值，EOB栏及Level栏的各列数据表示语法元素trans_coefficient的值。解码时根据trans_coefficient可以查表得到Level和Run。表中EOB栏对应的数字表示代表EOB的trans_coefficient的值。

用于解码帧内编码亮度块的游程和非零量化系数值的映射表VLC0_Intra见表D.1。

表 D.1 VLC0_Intra（用于解码帧内编码亮度块的游程和非零量化系数值的映射表）

Run的值	Level>0			RefAbsLevel的值	Run的值	Level>0			RefAbsLevel的值
	1	2	3			1	2	3	
0	0	22	38	4	12	26	–	–	2
1	2	32	–	3	13	28	–	–	2
2	4	44	–	3	14	30	–	–	2
3	6	50	–	3	15	34	–	–	2
4	8	54	–	3	16	36	–	–	2
5	10	–	–	2	17	40	–	–	2
6	12	–	–	2	18	42	–	–	2
7	14	–	–	2	19	46	–	–	2
8	16	–	–	2	20	48	–	–	2
9	18	–	–	2	21	52	–	–	2
10	20	–	–	2	22	56	–	–	2
11	24	–	–	2					

用于解码帧内编码亮度块的游程和非零量化系数值的映射表VLC1_Intra见表D.2。

表 D.2 VLC1_Intra（用于解码帧内编码亮度块的游程和非零量化系数值的映射表）

Run的值	EOB	Level > 0						RefAbsLevel的值
		1	2	3	4	5	6	
	8	–	–	–	–	–	–	
0	–	0	4	15	27	41	55	7
1	–	2	17	35	–	–	–	4
2	–	6	25	53	–	–	–	4
3	–	9	33	–	–	–	–	3
4	–	11	39	–	–	–	–	3
5	–	13	45	–	–	–	–	3
6	–	19	49	–	–	–	–	3

表 D.2（续）

Run的值	EOB	Level > 0						RefAbsLevel的值
		1	2	3	4	5	6	
	8	–	–	–	–	–	–	
7	–	21	51	–	–	–	–	3
8	–	23	–	–	–	–	–	2
9	–	29	–	–	–	–	–	2
10	–	31	–	–	–	–	–	2
11	–	37	–	–	–	–	–	2
12	–	43	–	–	–	–	–	2
13	–	47	–	–	–	–	–	2
14	–	57	–	–	–	–	–	2

用于解码帧内编码亮度块的游程和非零量化系数值的映射表VLC2_Intra见表D.3。

表 D.3 VLC2_Intra（用于解码帧内编码亮度块的游程和非零量化系数值的映射表）

Run的值	EOB	Level > 0									RefAbsLevel的值
		1	2	3	4	5	6	7	8	9	
	8	–	–	–	–	–	–	–	–	–	
0	–	0	2	6	13	17	27	35	45	55	10
1	–	4	11	21	33	49	–	–	–	–	6
2	–	9	23	37	–	–	–	–	–	–	4
3	–	15	29	51	–	–	–	–	–	–	4
4	–	19	39	–	–	–	–	–	–	–	3
5	–	25	43	–	–	–	–	–	–	–	3
6	–	31	53	–	–	–	–	–	–	–	3
7	–	41	–	–	–	–	–	–	–	–	2
8	–	47	–	–	–	–	–	–	–	–	2
9	–	57	–	–	–	–	–	–	–	–	2

用于解码帧内编码亮度块的游程和非零量化系数值的映射表VLC3_Intra见表D.4。

表 D.4 VLC3_Intra（用于解码帧内编码亮度块的游程和非零量化系数值的映射表）

Run的值	EOB	Level > 0												RefAbsLevel的值
		1	2	3	4	5	6	7	8	9	10	11	12	
	8	–	–	–	–	–	–	–	–	–	–	–	–	
0	–	0	2	4	9	11	17	21	25	33	39	45	55	13
1	–	6	13	19	29	35	47	–	–	–	–	–	–	7
2	–	15	27	41	57	–	–	–	–	–	–	–	–	5
3	–	23	37	53	–	–	–	–	–	–	–	–	–	4
4	–	31	51	–	–	–	–	–	–	–	–	–	–	3
5	–	43	–	–	–	–	–	–	–	–	–	–	–	2
6	–	49	–	–	–	–	–	–	–	–	–	–	–	2

用于解码帧内编码亮度块的游程和非零量化系数值的映射表VLC4_Intra见表D.5。

表 D.5　VLC4_Intra（用于解码帧内编码亮度块的游程和非零量化系数值的映射表）

Run的值	EOB	Level > 0																	RefAbsLevel的值
		1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	
	6	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	
0	–	0	2	4	7	9	11	15	17	21	23	29	33	35	43	47	49	57	18
1	–	13	19	27	31	37	45	55	–	–	–	–	–	–	–	–	–	–	8
2	–	25	41	51	–	–	–	–	–	–	–	–	–	–	–	–	–	–	4
3	–	39	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	2
4	–	53	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	2

用于解码帧内编码亮度块的游程和非零量化系数值的映射表VLC5_Intra见表D.6。

表 D.6　VLC5_Intra（用于解码帧内编码亮度块的游程和非零量化系数值的映射表）

Run的值	EOB	Level > 0																					RefAbsLevel的值
		1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	
	0	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	
0	–	1	3	5	7	9	11	13	15	17	19	23	25	27	31	33	37	41	45	49	51	55	22
1	–	21	29	35	43	47	53	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	7
2	–	39	57	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	3

用于解码帧内编码亮度块的游程和非零量化系数值的映射表VLC6_Intra见表D.7。

表 D.7　VLC6_Intra（用于解码帧内编码亮度块的游程和非零量化系数值的映射表）

Run的值	EOB	Level > 0																					RefAbsLevel的值
		1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	
	0	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	
0	–	1	3	5	7	9	11	13	15	17	19	21	23	25	27	29	31	35	37	39	41	43	27
1	–	33	45	55	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	4
		Level > 0																					
		22	23	24	25	26	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	
0	–	47	49	51	53	57	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	27
1	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–

用于解码帧间编码亮度块的游程和非零量化系数值的映射表VLC0_Inter见表D.8。

表 D.8 VLC0_Inter（用于解码帧间编码亮度块的游程和非零量化系数值的映射表）

Run的值	Level>0			RefAbsLevel的值	Run 的值	Level>0			RefAbsLevel的值
	1	2	3			1	2	3	
0	0	26	40	4	13	28	–	–	2
1	2	46	–	3	14	30	–	–	2
2	4	–	–	2	15	32	–	–	2
3	6	–	–	2	16	34	–	–	2
4	8	–	–	2	17	36	–	–	2
5	10	–	–	2	18	38	–	–	2
6	12	–	–	2	19	42	–	–	2
7	14	–	–	2	20	44	–	–	2
8	16	–	–	2	21	48	–	–	2
9	18	–	–	2	22	50	–	–	2
10	20	–	–	2	23	52	–	–	2
11	22	–	–	2	24	54	–	–	2
12	24	–	–	2	25	56	–	–	2

用于解码帧间编码亮度块的游程和非零量化系数值的映射表VLC1_Inter见表D.9。

表 D.9 VLC1_Inter（用于解码帧间编码亮度块的游程和非零量化系数值的映射表）

Run的值	EOB	Level > 0				RefAbsLevel的值
		1	2	3	4	
	2	–	–	–	–	
0	–	0	13	29	47	5
1	–	3	23	57	–	4
2	–	5	35	–	–	3
3	–	7	39	–	–	3
4	–	9	43	–	–	3
5	–	11	49	–	–	3
6	–	15	55	–	–	3
7	–	17	–	–	–	2
8	–	19	–	–	–	2
9	–	21	–	–	–	2
10	–	25	–	–	–	2
11	–	27	–	–	–	2
12	–	31	–	–	–	2
13	–	33	–	–	–	2
14	–	37	–	–	–	2
15	–	41	–	–	–	2
16	–	45	–	–	–	2
17	–	51	–	–	–	2
18	–	53	–	–	–	2

用于解码帧间编码亮度块的游程和非零量化系数值的映射表VLC2_Inter见表D.10。

表 D.10 VLC2_Inter（用于解码帧间编码亮度块的游程和非零量化系数值的映射表）

Run的值	EOB	Level > 0						RefAbsLevel的值
		1	2	3	4	5	6	
	2	–	–	–	–	–	–	
0	–	0	5	11	23	35	47	7
1	–	3	13	27	49	–	–	5
2	–	7	21	45	–	–	–	4
3	–	9	29	55	–	–	–	4
4	–	15	37	–	–	–	–	3
5	–	17	41	–	–	–	–	3
6	–	19	53	–	–	–	–	3
7	–	25	–	–	–	–	–	2
8	–	31	–	–	–	–	–	2
9	–	33	–	–	–	–	–	2
10	–	39	–	–	–	–	–	2
11	–	43	–	–	–	–	–	2
12	–	51	–	–	–	–	–	2
13	–	57	–	–	–	–	–	2

用于解码帧间编码亮度块的游程和非零量化系数值的映射表VLC3_Inter见表D.11。

表 D.11 VLC3_Inter（用于解码帧间编码亮度块的游程和非零量化系数值的映射表）

Run的值	EOB	Level > 0									RefAbsLevel的值
		1	2	3	4	5	6	7	8	9	
	2	–	–	–	–	–	–	–	–	–	
0	–	0	3	7	13	17	27	35	43	55	10
1	–	5	11	21	33	51	–	–	–	–	6
2	–	9	23	37	57	–	–	–	–	–	5
3	–	15	29	47	–	–	–	–	–	–	4
4	–	19	41	–	–	–	–	–	–	–	3
5	–	25	49	–	–	–	–	–	–	–	3
6	–	31	–	–	–	–	–	–	–	–	2
7	–	39	–	–	–	–	–	–	–	–	2
8	–	45	–	–	–	–	–	–	–	–	2
9	–	53	–	–	–	–	–	–	–	–	2

用于解码帧间编码亮度块的游程和非零量化系数值的映射表VLC4_Inter见表D.12。

表 D.12　VLC4_Inter（用于解码帧间编码亮度块的游程和非零量化系数值的映射表）

Run的值	EOB	Level > 0												RefAbsLevel的值
		1	2	3	4	5	6	7	8	9	10	11	12	
	2	–	–	–	–	–	–	–	–	–	–	–	–	
0	–	0	3	5	9	11	17	21	25	33	41	45	55	13
1	–	7	13	19	29	35	49	–	–	–	–	–	–	7
2	–	15	27	43	57	–	–	–	–	–	–	–	–	5
3	–	23	37	51	–	–	–	–	–	–	–	–	–	4
4	–	31	53	–	–	–	–	–	–	–	–	–	–	3
5	–	39	–	–	–	–	–	–	–	–	–	–	–	2
6	–	47	–	–	–	–	–	–	–	–	–	–	–	2

用于解码帧间编码亮度块的游程和非零量化系数值的映射表VLC5_Inter见表D.13。

表 D.13　VLC5_Inter（用于解码帧间编码亮度块的游程和非零量化系数值的映射表）

Run的值	EOB	Level > 0																RefAbsLevel的值
		1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	
	0	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	
0	–	1	3	5	7	9	13	15	17	21	25	29	33	39	43	49	53	17
1	–	11	19	27	31	41	45	57	–	–	–	–	–	–	–	–	–	8
2	–	23	37	51	–	–	–	–	–	–	–	–	–	–	–	–	–	4
3	–	35	55	–	–	–	–	–	–	–	–	–	–	–	–	–	–	3
4	–	47	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	2

用于解码帧间编码亮度块的游程和非零量化系数值的映射表VLC6_Inter见表D.14。

表 D.14　VLC6_Inter（用于解码帧间编码亮度块的游程和非零量化系数值的映射表）

Run的值	EOB	Level > 0																				RefAbsLevel的值
		1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	
	0	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	
0	–	1	3	5	7	9	11	13	17	19	21	23	25	29	33	35	39	41	43	47	49	57	22
1	–	15	27	37	45	55	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	6
2	–	31	51	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	3
3	–	53	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	2

用于解码色度块的游程和非零量化系数值的映射表VLC0_Chroma见表D.15。

表 D.15 VLC0_Chroma（用于解码色度块的游程和非零量化系数值的映射表）

Run的值	Level>0				RefAbsLevel的值	Run 的值	Level>0				RefAbsLevel 的值
	1	2	3	4			1	2	3	4	
0	0	14	32	56	5	13	28	–	–	–	2
1	2	48	–	–	3	14	30	–	–	–	2
2	4	–	–	–	2	15	34	–	–	–	2
3	6	–	–	–	2	16	36	–	–	–	2
4	8	–	–	–	2	17	38	–	–	–	2
5	10	–	–	–	2	18	40	–	–	–	2
6	12	–	–	–	2	19	42	–	–	–	2
7	16	–	–	–	2	20	44	–	–	–	2
8	18	–	–	–	2	21	46	–	–	–	2
9	20	–	–	–	2	22	50	–	–	–	2
10	22	–	–	–	2	23	52	–	–	–	2
11	24	–	–	–	2	24	54	–	–	–	2
12	26	–	–	–	2						

用于解码色度块的游程和非零量化系数值的映射表VLC1_Chroma见表D.16。

表 D.16 VLC1_Chroma（用于解码色度块的游程和非零量化系数值的映射表）

Run的值	EOB	Level > 0					RefAbsLevel的值
		1	2	3	4	5	
	0	–	–	–	–	–	
0	–	1	5	15	29	43	6
1	–	3	21	45	–	–	4
2	–	7	37	–	–	–	3
3	–	9	41	–	–	–	3
4	–	11	53	–	–	–	3
5	–	13	–	–	–	–	2
6	–	17	–	–	–	–	2
7	–	19	–	–	–	–	2
8	–	23	–	–	–	–	2
9	–	25	–	–	–	–	2
10	–	27	–	–	–	–	2
11	–	31	–	–	–	–	2
12	–	33	–	–	–	–	2
13	–	35	–	–	–	–	2
14	–	39	–	–	–	–	2
15	–	47	–	–	–	–	2
16	–	49	–	–	–	–	2
17	–	51	–	–	–	–	2
18	–	55	–	–	–	–	2
19	–	57	–	–	–	–	2

用于解码色度块的游程和非零量化系数值的映射表VLC2_Chroma见表D.17。

表 D.17 VLC2_Chroma（用于解码色度块的游程和非零量化系数值的映射表）

Run的值	EOB	Level > 0									RefAbsLevel的值
		1	2	3	4	5	6	7	8	9	
	2	–	–	–	–	–	–	–	–		
0	–	0	3	7	11	17	27	33	47	53	10
1	–	5	13	21	37	55	–	–	–	–	6
2	–	9	23	41	–	–	–	–	–	–	4
3	–	15	31	57	–	–	–	–	–	–	4
4	–	19	43	–	–	–	–	–	–	–	3
5	–	25	45	–	–	–	–	–	–	–	3
6	–	29	–	–	–	–	–	–	–	–	2
7	–	35	–	–	–	–	–	–	–	–	2
8	–	39	–	–	–	–	–	–	–	–	2
9	–	49	–	–	–	–	–	–	–	–	2
10	–	51	–	–	–	–	–	–	–	–	2

用于解码色度块的游程和非零量化系数值的映射表VLC3_Chroma见表D.18。

表 D.18 VLC3_Chroma（用于解码色度块的游程和非零量化系数值的映射表）

Run的值	EOB	Level > 0													RefAbsLevel的值
		1	2	3	4	5	6	7	8	9	10	11	12	13	
	0	–	–	–	–	–	–	–	–	–	–	–	–	–	
0	–	1	3	5	7	11	15	19	23	29	35	43	47	53	14
1	–	9	13	21	31	39	51	–	–	–	–	–	–	–	7
2	–	17	27	37	–	–	–	–	–	–	–	–	–	–	4
3	–	25	41	–	–	–	–	–	–	–	–	–	–	–	3
4	–	33	55	–	–	–	–	–	–	–	–	–	–	–	3
5	–	45	–	–	–	–	–	–	–	–	–	–	–	–	2
6	–	49	–	–	–	–	–	–	–	–	–	–	–	–	2
7	–	57	–	–	–	–	–	–	–	–	–	–	–	–	2

用于解码色度块的游程和非零量化系数值的映射表VLC4_Chroma见表D.19。

表 D.19 VLC4_Chroma（用于解码色度块的游程和非零量化系数值的映射表）

Run的值	EOB	Level > 0																		RefAbsLevel的值
		1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	
	0	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–			
0	–	1	3	5	7	9	11	13	15	19	21	23	27	29	33	37	41	43	51	55	20
1	–	17	25	31	39	45	53	–	–	–	–	–	–	–	–	–	–	–	–	–	7
2	–	35	49	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	3
3	–	47	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	2
4	–	57	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	2

CurrentVLCTable和MaxRun的映射关系见表D.20。

表 D.20 CurrentVLCTable 和 MaxRun 的映射关系

CurrentVLCTable	MaxRun的值	CurrentVLCTable	MaxRun 的值	CurrentVLCTable	MaxRun 的值
VLC0_Intra	22	VLC0_Inter	25	VLC0_Chroma	24
VLC1_Intra	14	VLC1_Inter	18	VLC1_Chroma	19
VLC2_Intra	9	VLC2_Inter	13	VLC2_Chroma	10
VLC3_Intra	6	VLC3_Inter	9	VLC3_Chroma	7
VLC4_Intra	4	VLC4_Inter	6	VLC4_Chroma	4
VLC5_Intra	2	VLC5_Inter	4		
VLC6_Intra	1	VLC6_Inter	3		

中 华 人 民 共 和 国
广播电影电视行业标准
广播电视先进音视频编解码
第 1 部分:视频
GY/T 257.1－2012
*
国家广播电影电视总局广播电视规划院出版发行
责任编辑：王佳梅
查询网址：www.abp.gov.cn
北京复兴门外大街二号
联系电话：（010）86093424　　86092923
邮政编码：100866