ICS 35.040
L 71

团 体 标 准

T/AVS 106—2018

高效音视频编码 视频

High efficiency coding of audio and video—
video

2018－05－28 发布 2018－05－28 实施

中关村视听产业技术创新联盟 发 布

目 录

前　　言

本部分按照GB/T 1.1—2009给出的规则起草。

本部分由中关村视听产业技术创新联盟归口。

本部分起草单位：国家新闻出版广电总局广播科学研究院、中央电视台、国家新闻出版广电总局广播电视规划院、中关村视听产业技术创新联盟、北京大学、浙江大学、深圳市海思半导体有限公司、清华大学、北京大学深圳研究生院、电子科技大学、上海大学、中国科学院大学、上海国茂数字技术有限公司、中国科技大学、中山大学、哈尔滨工业大学、中科院计算技术研究所、上海兆芯集成电路有限公司、武汉大学、上海交通大学。

本部分主要起草人：高文、黄铁军、盛志凡、潘晓菲、虞露、郭晓强、郑萧桢、董文辉、马思伟、张伟民、郑建铧、梁凡、李跃山、周芸、邓向冬、王强、何芸、何至初、余琴、李斐、李蔚然、陈杰、李善一、朴银姬、邵振江、王荣刚、童怡新、林和源、赵海武、张贤国、姜晓龙、凌勇、朱兴国、董思维、余全合、何大可、刘凌志、周敏华、曾伟民。

高效音视频编码　视频

1　范围

本部分规定了适应多种比特率、分辨率和质量要求的高效视频压缩方法的解码过程。

本部分适用于广播电视、数字电影、网络电视、网络视频、数字存储媒体、静止图像等视频编码。

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，仅所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。

ISO 11664-1:2007/CIE S 014-1:2006　色度　第1部分：标准比色观测器（Colorimetry – Part 1: Standard Colorimetric Observers）

ISO 11664-3:2012/CIE S 014-3:2011　色度　第3部分：CIE三色值（Colorimetry - Part 3: CIE Tristimulus Values）

CIE S 015:2005　室外工作场景照明（Lighting of Outdoor Workplaces）

3　术语和定义

下列术语和定义适用于本部分。

3.1

B图像　B picture

帧间预测中使用显示顺序上过去和将来的参考图像进行解码的图像。

3.2

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。

3.3

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

编码块　coding block
一个M×M的样值块。编码块由最大编码块划分得到。

3.6

编码单元　coding unit
包括一个亮度编码块和对应的色度编码块。编码单元由最大编码单元划分得到。

3.7

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。

3.8

编码图像　coded picture
一幅图像的编码表示。

3.9

补偿后样本　compensated sample
经预测补偿得到的样本。

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

参考图像队列　reference picture list
当前图像的参考图像所组成的队列。

3.12

参考图像缓冲区　reference picture buffer
保存解码图像并用于非场景预测的缓冲区。

3.13

参考索引　reference index
参考图像队列中参考图像或其中场的编号。

3.14

参考图像　reference picture
解码过程中用于后续图像帧间预测的图像。

3.15

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、最大编码单元、编码单元和编码块。

3.16

场　field

由构成帧的样本矩阵中相间的行构成。

3.17

场景图像　scene picture

场景图像包括G图像和GB图像。

3.18

场景图像缓冲区　scene reference picture buffer

保存G或GB图像并用于预测的缓冲区。

3.19

重建样本　reconstructed sample

构成解码图像的样本，由解码器根据位流解码得到。

3.20

二元符号　bin

组成二元符号串的符号，包括‘0’和‘1’。

3.21

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，最右边符号是最低有效位。

3.22

F图像　F picture

帧间预测中应使用单前向和双前向预测解码的图像。

3.23

反变换　inverse transform

将变换系数矩阵转换成空域样值矩阵的过程。

3.24

反量化　dequantization

对量化系数缩放后得到变换系数的过程。

3.25

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。

3.26

G图像　G picture
一种只使用帧内预测解码的场景图像，G图像应被输出。

3.27

GB图像　GB picture
一种只使用帧内预测解码的场景图像，GB图像不应被输出。

3.28

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。

3.29

划分　partition
将一个集合分为子集。集合中的每个元素属于且只属于某一个子集。

3.30

划分方式　partition type
划分获得的子集的组织方式。

3.31

I图像　I picture
只使用帧内预测解码的图像。

3.32

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。

3.33

解码顺序　decoding order
解码过程根据图像之间的预测关系，对每幅图像解码的顺序。

3.34

解码图像　decoded picture
解码器根据位流重建的图像。

3.35

解码图像缓冲区　decoded picture buffer
保存解码图像并用于输出重排序和输出定时的缓冲区，包括参考图像缓冲区和场景图像缓冲区。

3.36

解析过程　parse
由位流获得语法元素的过程。

3.37

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。

3.38

块　block

一个M×N（M列N行）的样值矩阵或者变换系数矩阵。

3.39

块扫描　block scan

量化系数的特定串行排序方式。

3.40

类　profile

语法、语义及算法的子集。

3.41

亮度　luma

表示亮度信号的样值矩阵或单个样值，符号为Y。

3.42

量化系数　quantization coefficient

反量化前变换系数的值。

3.43

量化参数　quantization parameter

在解码过程对量化系数进行反量化的参数。

3.44

滤波后样本　filtered sample

经去块效应滤波得到的样本。

3.45

P图像　P picture

帧间预测中只使用单前向预测进行解码的图像。

3.46

偏移后样本　offseted sample

经样值偏移补偿得到的样本。

3.47

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。

3.48

S图像 S picture
使用帧内预测和单前向预测解码的图像。S图像的参考图像应是最近解码的G或GB图像。

3.49

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。

3.50

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。

3.51

输出顺序 output order
输出解码图像的顺序，与显示顺序相同。

3.52

随机访问 random access
允许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。

3.53

填充位 stuffing bits
编码时插入位流中的位串，在解码时被丢弃。

3.54

条带 slice
按光栅扫描顺序排列的若干连续最大编码单元。

3.55

图像 picture
一幅图像可是一帧或一场。

3.56

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。

3.57

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）。

3.58

位流　bitstream
编码图像所形成的二进制数据流。

3.59

位流缓冲区　bitstream buffer
存储位流的缓冲区。

3.60

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。

3.61

显示顺序　display order
显示解码图像的顺序，不应被输出的图像是不具有显示顺序的图像。

3.62

样本　sample
构成图像的基本元素。

3.63

样本宽高比　width height ratio
一幅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：$h \div v$
式中h为样本水平方向尺寸，v为样本垂直方向尺寸。

3.64

样值　sample value
样本的幅值。

3.65

游程　run
在解码过程中若干连续的相同数据元素个数。在块扫描中一个非0系数前（沿块扫描顺序）连续的值为0的系数的个数。

3.66

预测　prediction
根据已知样本推测当前样本。

3.67

预测补偿　prediction 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。

3.68

预测单元　prediction unit
包括一个亮度预测块和对应的色度预测块。预测单元由编码单元划分得到。

3.69

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。

3.70

预测划分方式　prediction partition type
编码单元划分为帧内预测块或帧间预测单元的方式。

3.71

预测块　prediction block
一个使用相同预测过程的M×N的样值块。预测块由编码单元划分得到。

3.72

预测值　prediction value
在样值或数据元素的解码过程中，用先前已解码的样值或数据元素对当前样值的估计值。

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
视频信号中的一幅图像，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
使用帧间预测对编码单元或图像进行编码。

3.78

帧间预测　inter prediction
使用先前解码图像生成当前图像样本预测值的过程。

3.79

帧内编码　intra coding

使用帧内预测对编码单元或图像进行编码。

3.80

帧内预测　intra prediction

在相同解码图像中使用先前解码的样值生成当前样本预测值的过程。

3.81

字节　byte

8位的位串。

3.82

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。

3.83

最大编码单元　largest coding unit

包括一个L×L的亮度样值块和对应的色度样值块。最大编码单元由图像划分得到。

3.84

最大编码块　largest coding block

一个K×K的样值块。最大编码块由图像的三个样值矩阵（亮度和两个色度）中的一个矩阵划分得到。

4　缩略语

下列缩略语适用于本部分。

AEC　高级熵编码（Advanced Entropy Code）
ALF　自适应修正滤波（Adaptive Leveling Filter）
BBV　位流参考缓冲区管理（Bitstream Buffer Verifier）
CBR　恒定比特率（Constant Bit Rate）
CB　编码块（Coding Block）
CU　编码单元（Coding Unit）
CUT　编码树（Coding Unit Tree）
LCU　最大编码单元（Largest Coding Unit）
LSB　最低有效位（Least Significant Bit）
MSB　最高有效位（Most Significant Bit）
PB　预测块（Prediction Block）
PU　预测单元（Prediction Unit）
ROI　感兴趣区域（Region of Interesting）
SAO　样值偏移补偿（Sample Adaptive Offset）

5 约定

5.1 概述

本部分中使用的数学运算符和优先级参照C语言。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。

5.2 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
$\lceil \cdot \rceil$	上取整

5.3 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*之间的逻辑与运算
a||*b*	*a*和*b*之间的逻辑或运算
!	逻辑非运算

5.4 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

5.5　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

5.6　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$x=x-(-3)$

5.7　数学函数

数学函数定义见式（1）～式（9）。

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

式中：

x——自变量。

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

式中：

x——自变量。

$$\mathrm{Clip1}(x) = \mathrm{Clip3}(0, 2^{BitDepth} - 1, x) \quad (3)$$

式中：

x ——自变量；

$BitDepth$——编码样本精度。

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

式中：

x——自变量；

i——下界；

j——上界。

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

式中：

x——自变量x；

y——自变量y；

z——自变量z。

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

式中：

x——自变量x；

y——自变量y。

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

式中：

x——自变量x；

y——自变量y。

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

式中：

x——自变量。

$$\mathrm{Log}(x) = \log_2 x \quad (9)$$

式中：

x——自变量。

$$\mathrm{Ln}(x) = \log_e x \quad (10)$$

式中：

x——自变量；

e——自然对数的底，其值为2.718281828...。

5.8 结构关系符

结构关系符定义见表6。

表6 结构关系符

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

5.9 位流语法、解析过程和解码过程的描述方法

5.9.1 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	
…	
}	

表 7（续）

伪代码	描述符
/*“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	
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。

5.9.2 函数

5.9.2.1 概述

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。

5.9.2.1 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。

5.9.2.2 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。

5.9.2.3 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。

5.9.2.4 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。

5.9.2.5 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;	
}	

5.9.2.6 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;	
}	

5.9.2.7　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。

5.9.3　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8.3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
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
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
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

5.9.4　保留、禁止和标记位

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。

“标记位”（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。视频序列由第一个序列头开始，序列结束码或视频编辑码表明了一个视频序列的结束。视频序列的第一个序列头到第一个出现的序列结束码或视频编辑码之间的序列头为重复序列头。每个序列头后面跟着一个或多个编码图像，每幅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。

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图像。两帧之间存在着一个帧时间间隔。对隔行序列而言，每帧的两场之间存在着一个场时间间隔。对逐行序列而言，每帧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。

序列头后的第一个解码图像应是I、G或GB图像。如果序列头后的第一个解码图像是GB图像，则该序列头后的第二个解码图像应是S图像。

当前图像对应的序列头为解码顺序在当前图像之前的最近的序列头。

如果当前图像对应的序列头后的第一个解码图像为I图像，并且当前图像的显示顺序在该I图像之后，则当前图像的参考图像应在以下范围内：该I图像、显示顺序在这个I图像之后的图像，和解码顺序在当前图像对应的序列头之后的G或GB图像。

如果当前图像对应的序列头后的第一个解码图像为G图像，并且当前图像的显示顺序在该G图像之后，则当前图像的参考图像应在以下范围内：该G图像、参考图像的显示顺序在该G图像之后的图像、和解码顺序在当前图像对应的序列头之后的G或GB图像。

如果当前图像对应的序列头后的第一个解码图像为GB图像、第二个解码图像为S图像，并且当前图像的显示顺序在该S图像之后，则当前图像的参考图像应在以下范围内：该S图像、显示顺序在这个S图像之后的图像、解码顺序在当前图像对应的序列头之后的G或GB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

一幅图像可是一帧或一场，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 4:0:0 格式

对于4:0:0格式，只包括Y矩阵。

亮度样本位置见图1。

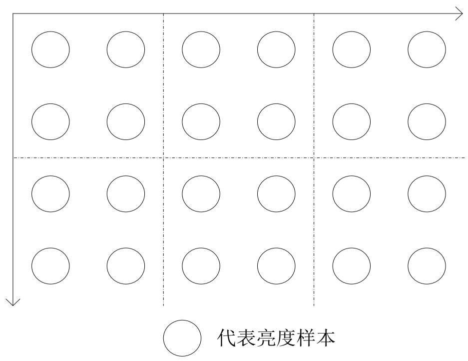


图1 4:0:0 格式下亮度样本位置

6.2.2.2 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2。

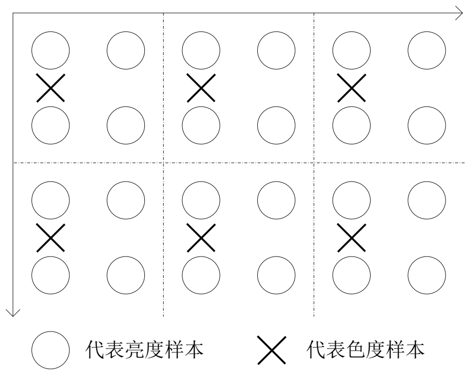


图2 4:2:0 格式下亮度和色度样本位置

6.2.2.3 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3。

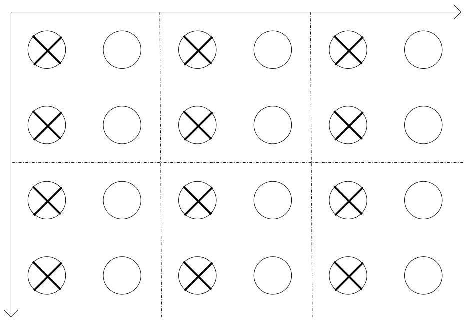


图3　4:2:2 格式下亮度和色度样本位置

6.2.2.4　4:4:4 格式

对于4:4:4格式，Cb和Cr矩阵在水平和垂直方向的尺寸都和Y矩阵一样。如果图像两场的编码数据依次出现，则Y矩阵的行数应是偶数。

亮度和色度样本位置见图4。

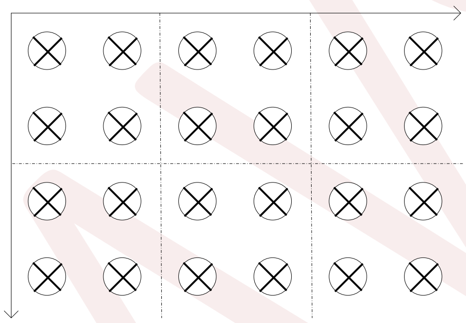


图4　4:4:4 格式下亮度和色度样本位置

6.2.3　图像类型

本部分定义了以下七种解码图像：

a)　I 图像；
b)　P 图像；
c)　B 图像；
d)　F 图像；
e)　G 图像；
f)　GB 图像；
g)　S 图像。

6.2.4　图像间的顺序

如果视频序列中没有B图像，解码顺序与显示顺序相同。如果视频序列中包含B图像，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。

序列头后的第一个解码图像应是I、G或GB图像。如果序列头后的第一个解码图像是GB图像，则该序列头后的第二个解码图像应是S图像。码流中显示顺序在该I、G或S图像之后的图像的解码顺序应在该I、G或者S图像之后。

下面举例说明图像重排序。

示例1：I图像和P图像之间有3个B图像，两个连续的P图像之间也有3个B图像。按照解码顺序用图像0I预测图像4P，用图像4P和0I预测图像2B，用图像2B和0I预测图像1B，用图像4P和2B预测图像3B。解码顺序是0I，4P，2B，1B，3B；显示顺序是0I，1B，2B，3B，4P。

按照解码顺序排序：

解码顺序	0	1	2	3	4	5	6	7	8	9	10	11	12
类型	I	P	B	B	B	P	B	B	B	P	B	B	B
显示顺序	0	4	2	1	3	8	6	5	7	12	10	9	11

按照显示顺序排序：

显示顺序	0	1	2	3	4	5	6	7	8	9	10	11	12
类型	I	B	B	B	P	B	B	B	P	B	B	B	P
解码顺序	0	3	2	4	1	7	6	8	5	11	10	12	9

示例2：I图像和P图像之间有7个B图像，两个连续的P图像之间也有7个B图像。按照解码顺序用图像0I预测图像8P，用图像8P和0I预测图像4B，用图像4B和0I预测图像2B，用图像2B和0I预测图像1B，用图像4B和2B预测图像3B，用图像8P和4B预测图像6B，用图像6B和4B预测图像5B，用图像8P和6B预测图像7B。解码顺序是0I，8P，4B，2B，1B，3B，6B，5B，7B；显示顺序是0I，1B，2B，3B，4B，5B，6B，7B，8P。

按照解码顺序排序：

解码顺序	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
类型	I	P	B	B	B	B	B	B	B	P	B	B	B	B	B	B	B
显示顺序	0	8	4	2	1	3	6	5	7	16	12	10	9	11	14	13	15

按照显示顺序排序：

显示顺序	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
类型	I	B	B	B	B	B	B	B	P	B	B	B	B	B	B	B	P
解码顺序	0	4	3	5	2	7	6	8	1	12	11	13	10	15	14	16	9

示例3：I图像和P图像之间有2个B图像，两个连续的P图像之间也有2个B图像。按照解码顺序用图像0I预测图像3P，用图像3P和0I预测图像1B和2B。解码顺序是0I，3P，1B，2B；显示顺序是0I，1B，2B，3P。

按照解码顺序排序：

解码顺序	0	1	2	3	4	5	6	7	8	9	10	11	12
类型	I	P	B	B	P	B	B	I	B	B	P	B	B
显示顺序	0	3	1	2	6	4	5	9	7	8	12	10	11

按照显示顺序排序：

显示顺序	0	1	2	3	4	5	6	7	8	9	10	11	12
类型	I	B	B	P	B	B	I	B	B	P	B	B	P
解码顺序	0	2	3	1	5	6	4	8	9	7	11	12	10

6.2.5　参考图像

P图像可有显示顺序上位于当前图像之前的多幅参考图像，这些参考图像称为前向参考图像。F图像可有显示顺序上位于当前图像之前的多幅参考图像，这些参考图像称为前向参考图像。B图像有一幅显示顺序位于当前图像之前的参考图像和一幅显示顺序位于当前图像之后的参考图像，这两幅参考图像分别称为前向参考图像和后向参考图像。S图像可有显示顺序位于当前图像之前的一幅参考图像，该参考图像为最近解码的G图像或GB图像。

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64个像素。对色度样本矩阵，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32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最大编码单元在图像内的部分，条带之间不应重叠。条带结构见图5。

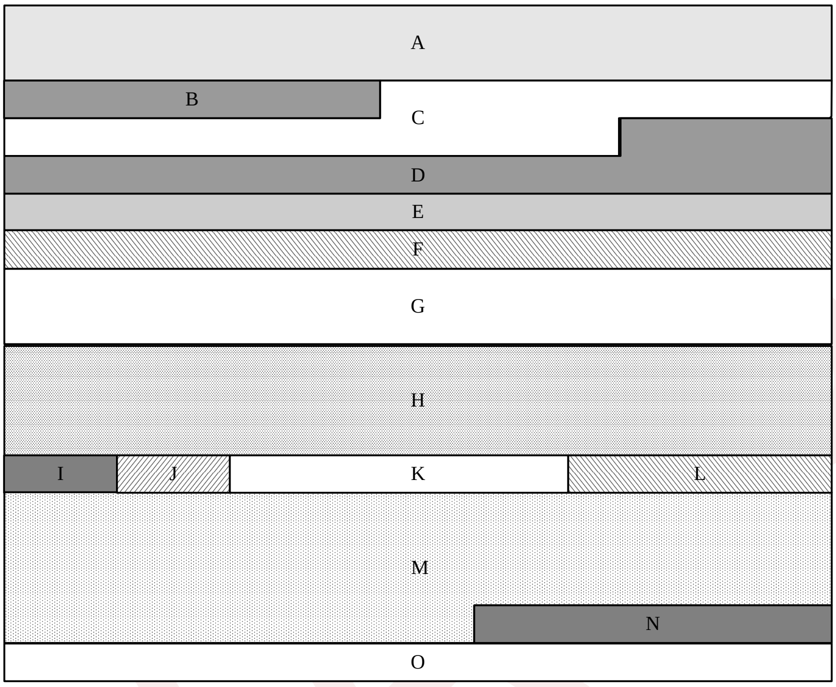


图5　条带结构

6.4 最大编码单元、编码树、编码单元和变换块

图像划分为最大编码单元，最大编码单元之间不应重叠，最大编码单元左上角的样本不应超出图像边界，最大编码单元右下角的样本可超出图像边界。

最大编码单元划分为一个或多个编码单元，由编码树决定。编码单元划分为一个或多个变换块。

变换块中的编码系数划分为按顺序排列的系数块，每个系数块是一个包含4×4编码系数的块。

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）	00～8F
保留	90～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
帧内预测图像起始码（intra_picture_start_code）	B3
保留	B4
视频扩展起始码（extension_start_code）	B5
帧间预测图像起始码（inter_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) == intra_picture_start_code)	
intra_picture_header()	
Else	
inter_picture_header()	
extension_and_user_data(1)	
picture_data()	
}while((next_bits(32)==inter_picture_start_code)||(next_bits(32)==intra_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)	

表 13（续）

视频序列定义	描述符
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)
field_coded_sequence	u(1)
horizontal_size	u(14)
vertical_size	u(14)
chroma_format	u(2)
sample_precision	u(3)
if (profile_id == 0x22) {	
encoding_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)
low_delay	u(1)
marker_bit	f(1)
temporal_id_enable_flag	u(1)
bbv_buffer_size	u(18)

表 14（续）

序列头定义	描述符
lcu_size	u(3)
weight_quant_enable_flag	u(1)
if (WeightQuantEnableFlag == 1)	
{	
load_seq_weight_quant_data_flag	u(1)
if (load_seq_weight_quant_data_flag == '1') {	
weight_quant_matrix( )	
}	
}	
scene_picture_disable_flag	u(1)
multi_hypothesis_skip_enable_flag	u(1)
dual_hypothesis_prediction_enable_flag	u(1)
weighted_skip_enable_flag	u(1)
asymmetric_motion_partitions_enable_flag	u(1)
nonsquare_quadtree_transform_enable_flag	u(1)
nonsquare_intra_prediction_enable_flag	u(1)
secondary_transform_enable_flag	u(1)
sample_adaptive_offset_enable_flag	u(1)
adaptive_loop_filter_enable_flag	u(1)
pmvr_enable_flag	u(1)
marker_bit	f(1)
num_of_rcs	u(6)
for(i=0; i<NumOfRcs; i++)	
reference_configuration_set(i)	
if (low_delay == '0')	
output_reorder_delay	u(5)
cross_slice_loopfilter_enable_flag	u(1)
reserved_bits	r(2)
next_start_code()	
}	

7.1.2.3 参考图像配置集定义

参考图像配置集定义见表15。

表15　参考图像配置集定义

参考图像配置集定义	描述符
reference_configuration_set(i) {	
refered_by_others_flag[i]	u(1)
num_of_reference_picture[i]	u(3)
for(j=0; j<NumOfReferencePicture[i]; j++)	
delta_doi_of_reference_picture[i][j]	u(6)
num_of_removed_picture[i]	u(3)
for(j=0; j<NumOfRemovedPicture[i]; j++)	
delta_doi_of_removed_picture[i][j]	u(6)
marker_bit	f(1)
}	

7.1.2.4　加权量化数据定义

自定义加权量化矩阵定义见表16。

表16　自定义加权量化矩阵定义

自定义加权量化矩阵定义	描述符
weight_quant_matrix( ) {	
for (SizeId = 0; SizeId < 2; SizeId++) {	
WQMSize = 1 << (SizeId+2)	
for (i=0; i<WQMSize; i++) {	
for (j=0; j<WQMSize; j++) {	
weight_quant_coeff	ue(v)
if (SizeId == 0)	
$WeightQuantMatrix_{4x4}$[i][j] = WeightQuantCoeff	
Else	
$WeightQuantMatrix_{8x8}$[i][j] = WeightQuantCoeff	
}	
}	
}	
}	

7.1.2.5　扩展和用户数据语法

7.1.2.5.1　扩展和用户数据定义

扩展和用户数据定义见表17。

表17　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()	
}	
}	

7.1.2.5.2　扩展数据定义

扩展数据定义见表18。

表18　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 */	
if (next_bits(4) == '0010')　　　/* 序列显示扩展 */	
sequence_display_extension()	
else if (next_bits(4) == '0011')　　/* 时域可伸缩扩展 */	
temporal_scalability_extension()	
else if (next_bits(4) == '0100')　　/* 版权扩展 */	
copyright_extension()	
else if (next_bits(4) == '1010')　　/* 目标设备显示和内容元数据扩展 */	
mastering_display_and_content_metadata_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()	

表 18（续）

扩展数据定义	描述符
else if (next_bits(4) == '0101')　　/* 高动态范围图像元数据扩展 */	
hdr_dynamic_metadata_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 if (next_bits(4) == '1100')　　/* 感兴趣区域参数扩展 */	
roi_parameters_extension()	
else {	
while (next_bits(24) != '0000 0000 0000 0000 0000 0001')	
reserved_extension_data_byte	u(8)
}	
}	
}	
}	

7.1.2.5.3　用户数据定义

用户数据定义见表19。

表19　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)
}	
}	

7.1.2.6　序列显示扩展定义

序列显示扩展定义见表20。

表20　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)

表 20（续）

序列显示扩展定义	描述符
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)
td_mode_flag	u(1)
if (td_mode_flag == '1') {	
td_packing_mode	u(8)
view_reverse_flag	u(1)
}	
next_start_code()	
}	

7.1.2.7 时域可伸缩扩展定义

时域可伸缩扩展定义见表21。

表21 时域可伸缩扩展定义

时域可伸缩扩展定义	描述符
temporal_scalability_extension() {	
extension_id	f(4)
num_of_temporal_level_minus1	u(3)
for (i=0; i<num_of_temporal_level_minus1; i++) {	
temporal_frame_rate_code[i]	u(4)
temporal_bit_rate_lower[i]	u(18)
marker_bit	f(1)
temporal_bit_rate_upper[i]	u(12)
}	
next_start_code()	
}	

7.1.2.8 版权扩展定义

版权扩展定义见表22。

表22　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()	
}	

7.1.2.9　目标设备显示和内容元数据扩展定义

目标设备显示和内容元数据扩展定义见表23。

表23　目标设备显示和内容元数据扩展定义

目标设备显示和内容元数据扩展定义	描述符
mastering_display_and_content_metadata_extension() {	
extension_id	f(4)
for (c=0; c<3; c++) {	
display_primaries_x[c]	u(16)
marker_bit	f(1)
display_primaries_y[c]	u(16)
marker_bit	f(1)
}	
white_point_x	u(16)
marker_bit	f(1)
white_point_y	u(16)
marker_bit	f(1)
max_display_mastering_luminance	u(16)
marker_bit	f(1)
min_display_mastering_luminance	u(16)

表 23（续）

目标设备显示和内容元数据扩展定义	描述符
marker_bit	f(1)
max_content_light_level	u(16)
marker_bit	f(1)
max_picture_average_light_level	u(16)
marker_bit	f(1)
reserved_bits	r(16)
next_start_code()	
}	

7.1.2.10 摄像机参数扩展定义

摄像机参数扩展定义见表24。

表24 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)

表 24（续）

摄像机参数扩展定义	描述符
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)
reserved_bits	r(16)
next_start_code()	
}	

7.1.2.11　感兴趣区域参数扩展定义

感兴趣区域参数扩展定义见表25。

表25　感兴趣区域参数扩展定义

感兴趣区域参数扩展定义	描述符
roi_parameters_extension() {	
extension_id	f(4)
current_picture_roi_num	u(8)
roiIndex = 0	
if (PictureType != 0) {	
prev_picture_roi_num	u(8)
for (i=0; i<PrevPictureROINum; i++) {	
roi_skip_run	ue(v)
if (roi_skip_run != '0') {	
for (j=0; j<roi_skip_run; j++) {	

表 25（续）

感兴趣区域参数扩展定义	描述符
skip_roi_mode[i+j]	u(1)
if (j % 22 == 0) {	
marker_bit	f(1)
if (skip_roi_mode == '1') {	
ROIInfo [roiIndex]->asisx = PrevROIInfo [i+j] ->asisx	
ROIInfo [roiIndex]->asisy = PrevROIInfo [i+j] ->asisy	
ROIInfo [roiIndex]->width = PrevROIInfo [i+j] -> width	
ROIInfo [roiIndex]->height = PrevROIInfo [i+j] -> height	
roiIndex++	
}	
}	
i += j	
marker_bit	f(1)
}	
else {	
roi_axisx_delta	se(v)
marker_bit	f(1)
roi_axisy_delta	se(v)
marker_bit	f(1)
roi_width_delta	se(v)
marker_bit	f(1)
roi_height_delta	se(v)
marker_bit	f(1)
ROIInfo [roiIndex]->asisx = PrevROIInfo [i+j] ->asisx + ROIAxisxDelta	
ROIInfo [roiIndex]->asisy = PrevROIInfo [i+j]->asisy + ROIAxisyDelta	
ROIInfo [roiIndex]->width = PrevROIInfo [i+j] -> width + ROIWidthDelta	
ROIInfo [roiIndex]->height = PrevROIInfo [i+j]-> height + ROIHeightDelta	
roiIndex++	
}	
}	
}	
for (i=roiIndex; i<current_picture_roi_num; i++) {	
roi_axisx	u(6)
marker_bit	f(1)
roi_axisy	u(6)
marker_bit	f(1)
roi_width	u(6)

表 25（续）

感兴趣区域参数扩展定义	描述符
marker_bit	f(1)
roi_height	u(6)
marker_bit	f(1)
ROIInfo [roiIndex]->asisx = roi_axisx	
ROIInfo [roiIndex]->asisy = roi_axisy	
ROIInfo [roiIndex]->width = roi_width	
ROIInfo [roiIndex]->height = roi_height	
roiIndex++	
}	
for (i=0; i<roiIndex; i++) {	
PrevROIInfo[i]->asisx = ROIInfo [i]->asisx	
PrevROIinfo[i]->asisy = ROIInfo [i]->asisy	
PrevROIinfo[i]->width = ROIInfo [i]->width	
PrevROIinfo[i]->height = ROIInfo [i]->height	
}	
next_start_code()	
}	

7.1.2.12 高动态范围图像扩展定义

高动态范围图像元数据扩展定义见表26。

表26　高动态范围图像扩展定义

高动态范围图像元数据扩展定义	描述符
hdr_dynamic_metadata_extension() {	
extension_id	f(4)
while (next_bits(24) ! '0000 0000 0000 0000 0000 0001') {	
extension_data_byte	u(8)
}	
next_start_code()	
}	

7.1.3　图像定义

7.1.3.1　帧内预测图像头定义

帧内预测图像头定义见表27。

表27 帧内预测图像头定义

帧内预测图像头定义	描述符
intra_picture_header() {	
intra_picture_start_code	f(32)
bbv_delay	u(32)
time_code_flag	u(1)
if (time_code_flag == '1')	
time_code	u(24)
if (ScenePictureDisableFlag == 0) {	
scene_picture_flag	u(1)
if (ScenePictureFlag == 1) {	
scene_picture_output_flag	u(1)
}	
}	
decode_order_index	u(8)
if (temporal_id_enable_flag == '1')	
temporal_id	u(3)
if (low_delay == '0' && (ScenePictureFlag == 0 || scene_picture_output_flag ==1))	
picture_output_delay	ue(v)
use_rcs_flag	u(1)
if (UseRcsFlag)	
rcs_index	u(5)
Else	
reference_configuration_set(NumOfRcs)	
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)
if (field_coded_sequence == '1') {	
top_field_picture_flag	u(1)
reserved_bits	r(1)
}	
fixed_picture_qp	u(1)
picture_qp	u(7)
loop_filter_disable_flag	u(1)
if (! loop_filter_disable_flag) {	
loop_filter_parameter_flag	u(1)

表 27（续）

帧内预测图像头定义	描述符
if (loop_filter_parameter_flag) {	
alpha_c_offset	se(v)
beta_offset	se(v)
}	
}	
chroma_quant_param_disable_flag	u(1)
if (chroma_quant_param_disable_flag == ‘0’) {	
chroma_quant_param_delta_cb	se(v)
chroma_quant_param_delta_cr	se(v)
}	
if (WeightQuantEnableFlag == 1) {	
pic_weight_quant_enable_flag	u(1)
if (PicWeightQuantEnableFlag) {	
pic_weight_quant_data_index	u(2)
if (pic_weight_quant_data_index == '01') {	
reserved_bits	r(1)
weight_quant_param_index	u(2)
weight_quant_model	u(2)
if (weight_quant_param_index == ‘01’) {	
for (i=0; i<6; i++) {	
weight_quant_param_delta1[i]	se(v)
}	
}	
if (weight_quant_param_index == ‘10’) {	
for (i=0; i<6; i++) {	
weight_quant_param_delta2[i]	se(v)
}	
}	
}	
else if (pic_weight_quant_data_index == '10') {	
weight_quant_matrix( )	
}	
}	
}	
if (AlfEnableFlag == 1) {	
for (compIdx=0; compIdx<3; compIdx++) {	
picture_alf_enable_flag[compIdx]	u(1)
}	

表 27（续）

帧内预测图像头定义	描述符
if (loop_filter_parameter_flag) {	
if (PictureAlfEnableFlag[0] == 1 || PictureAlfEnableFlag[1] == 1 || PictureAlfEnableFlag[2] == 1) {	
alf_parameter_set( )	
}	
}	
next_start_code()	
}	

7.1.3.2　帧间预测图像头定义

帧间预测图像头定义见表28。

表28　帧间预测图像头定义

帧间预测图像头定义	描述符
inter_picture_header() {	
inter_picture_start_code	f(32)
bbv_delay	u(32)
picture_coding_type	u(2)
if (ScenePictureDisableFlag == 0 && picture_coding_type == '01') {	
scene_pred_flag	u(1)
}	
if (ScenePictureDisableFlag == 0 && picture_coding_type != '10') {	
if (ScenePredFlag == 0) {	
scene_reference_enable_flag	u(1)
}	
}	
decode_order_index	u(8)
if (temporal_id_enable_flag == '1')	
temporal_id	u(3)
if (low_delay == '0')	
picture_output_delay	ue(v)
use_rcs_flag	u(1)
if (UseRcsFlag)	
rcs_index	u(5)
Else	
reference_configuration_set(NumOfRcs)	
if (low_delay == '1')	
bbv_check_times	ue(v)

表 28（续）

帧间预测图像头定义	描述符
progressive_frame	u(1)
if (progressive_frame == '0') {	
picture_structure	u(1)
}	
top_field_first	u(1)
repeat_first_field	u(1)
if (field_coded_sequence == '1') {	
top_field_picture_flag	u(1)
reserved_bits	r(1)
}	
fixed_picture_qp	u(1)
picture_qp	u(7)
if (! (picture_coding_type=='10' && PictureStructure==1))	
reserved_bits	r(1)
random_access_decodable_flag	u(1)
loop_filter_disable_flag	u(1)
if (! loop_filter_disable_flag) {	
loop_filter_parameter_flag	u(1)
if (loop_filter_parameter_flag) {	
alpha_c_offset	se(v)
beta_offset	se(v)
}	
}	
chroma_quant_param_disable_flag	u(1)
if (chroma_quant_param_disable_flag == ‘0’) {	
chroma_quant_param_delta_cb	se(v)
chroma_quant_param_delta_cr	se(v)
}	
if (WeightQuantEnableFlag == 1) {	
pic_weight_quant_enable_flag	u(1)
if (PicWeightQuantEnableFlag) {	
pic_weight_quant_data_index	u(2)
if (pic_weight_quant_data_index == '01') {	
reserved_bits	r(1)
weight_quant_param_index	u(2)
weight_quant_model	u(2)
if (weight_quant_param_index == ‘01’) {	

表 28（续）

帧间预测图像头定义	描述符
for (i=0; i<6; i++) {	
weight_quant_param_delta1[i]	se(v)
}	
}	
if (weight_quant_param_index == ‘10’) {	
for (i=0; i<6; i++) {	
weight_quant_param_delta2[i]	se(v)
}	
}	
}	
else if (pic_weight_quant_data_index == '10') {	
weight_quant_matrix( )	
}	
}	
}	
if (AlfEnableFlag == 1) {	
for (compIdx=0; compIdx<3; compIdx++) {	
picture_alf_enable_flag[compIdx]	u(1)
}	
if (PictureAlfEnableFlag[0] == 1 || PictureAlfEnableFlag[1] == 1 || PictureAlfEnableFlag[2] == 1) {	
alf_parameter_set( )	
}	
}	
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29。

表29 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 ++) {	
picture_centre_horizontal_offset	i(16)
marker_bit	f(1)

表 29（续）

图像显示扩展定义	描述符
picture_centre_vertical_offset	i(16)
marker_bit	f(1)
}	
next_start_code()	
}	

7.1.3.4 图像数据定义

图像数据定义见表30。

表30　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)	
}	

7.1.4 条带定义

条带定义见表31。

表31　条带定义

条带定义	描述符
slice() {	
slice_start_code	f(32)
if (vertical_size > (144 × $2^{LcuSizeInBit}$))	
slice_vertical_position_extension	u(3)
slice_horizontal_position	u(8)
if (horizontal_size > (255 × $2^{LcuSizeInBit}$))	
slice_horizontal_position_extension	u(2)
if (fixed_picture_qp == '0') {	
fixed_slice_qp	u(1)
slice_qp	u(7)
}	
if (SaoEnableFlag) {	
for (compIdx=0; compIdx<3; compIdx++) {	

表 31（续）

条带定义	描述符
slice_sao_enable_flag[compIdx]	u(1)
}	
}	
while (! byte_aligned())	
aec_byte_alignment_bit	f(1)
do {	
if (SaoEnableFlag) {	
if (SliceSaoEnableFlag[0] || SliceSaoEnableFlag[1] || SliceSaoEnableFlag[2]) {	
if (MergeFlagExist)	
sao_merge_type_index	ae(v)
if (SaoMergeMode == 'SAO_NON_MERGE') {	
for (compIdx=0; compIdx<3; compIdx++) {	
if (SliceSaoEnableFlag[compIdx]) {	
sao_mode[compIdx]	ae(v)
if (SaoMode[compIdx] == 'SAO_Interval') {	
for (j=0; j<MaxOffsetNumber; j++) {	
sao_interval_offset_abs[compIdx][j]	ae(v)
if (SaoIntervalOffsetAbs[compIdx][j])	
sao_interval_offset_sign[compIdx][j]	ae(v)
}	
sao_interval_start_pos[compIdx]	ae(v)
sao_interval_delta_pos_minus2[compIdx]	ae(v)
}	
if (SaoMode[compIdx] == 'SAO_Edge') {	
for (j=0; j<MaxOffsetNumber; j++)	
sao_edge_offset[compIdx][j]	ae(v)
sao_edge_type[compIdx]	ae(v)
}	
}	
}	
}	
}	
}	
for (compIdx=0; compIdx<3; compIdx++) {	
if (PictureAlfEnableFlag[compIdx] == 1)	

表 31（续）

条带定义	描述符
alf_lcu_enable_flag[compIdx][LcuIndex]	ae(v)
}	
coding_unit_tree(LcuSizeInBit, LcuIndex)	
aec_lcu_stuffing_bit	ae(v)
} while (! is_end_of_slice())	
next_start_code()	
}	

7.1.5 编码树定义

编码树定义见表32。

表32 编码树定义

编码树定义	描述符
coding_unit_tree(SizeInBit, PositionInPic) {	
x0 = (PositionInPic % MinCuWidth) × MiniSize	
y0 = (PositionInPic / MinCuWidth) × MiniSize	
if (x0+(1<< SizeInBit)<=MinCuWidth × MiniSize && y0+(1<<SizeInBit)<= MinCuHeight × MiniSize && SizeInBit>Log(MiniSize))	
split_flag	ae(v)
else if (SizeInBit == Log(MiniSize))	
SplitFlag = 0	
Else	
SplitFlag = 1	
if (SplitFlag) {	
x1=x0+(1<<(SizeInBit-1))	
y1=y0+(1<<(SizeInBit-1))	
coding_unit_tree(SizeInBit-1, PositionInPic)	
if (x1< MinCuWidth × MiniSize)	
coding_unit_tree(SizeInBit-1, PositionInPic +(1<<(SizeInBit- Log(MiniSize)-1)))	
if (y1< MinCuHeight × MiniSize)	
coding_unit_tree(SizeInBit-1, PositionInPic + MinCuWidth × (1<<(SizeInBit- Log(MiniSize)-1)))	
if (x1< MinCuWidth × MiniSize &&y1< MinCuHeight × MiniSize)	
coding_unit_tree(SizeInBit-1, PositionInPic+ MinCuWidth × (1<<(SizeInBit- Log(MiniSize)-1))+ (1<<(SizeInBit- Log(MiniSize)-1)))	

表 32（续）

编码树定义	描述符
}	
else {	
coding_unit(SizeInBit, PositionInPic)	
}	
}	

7.1.6 编码单元定义

编码单元定义见表33。

表33 编码单元定义

编码单元定义	描述符
coding_unit (SizeInBit, PositionInPic) {	
if (PictureType != 0)	
cu_type_index	ae(v)
if ((CuTypeIndex == 3 || CuTypeIndex == 4) && (AsymmetricMotionPartitionsEnableFlag == 1) && SizeInBit > 3)	
shape_of_partition_index	ae(v)
if (PictureType == 2 && CuTypeIndex >= 2 && CuTypeIndex <= 4))	
if (CuTypeIndex == 2 || SizeInBit > 3)	
b_pu_type_index	ae(v)
Else	
b_pu_type_min_index	ae(v)
if (PictureType == 3 && CuTypeIndex >= 2 && CuTypeIndex <= 4 && RefPicNum > 1 && DualHypothesisPredictionEnableFlag == 1)	
if (CuTypeIndex == 2 || SizeInBit > 3)	
f_pu_type_index	ae(v)
if ((CuType == 'F_Skip' || CuType == 'F_Direct') && RefPicNum > 1 && WeightedSkipEnableFlag == 1)	
weighted_skip_mode	ae(v)
if (CuType == 'B_Skip' || CuType == 'B_Direct_2N' || ((CuType == 'F_Skip' || CuType == 'F_Direct') && WeightedSkipMode == 0 && MultiHypothesisSkipEnableFlag == 1))	
cu_subtype_index	ae(v)
if (CuType != 'P_Skip' && CuType != 'B_Skip' && CuType != 'F_Skip' && CuType != 'S_Skip') {	
if (CuType == 'B_NxN') {	
for (i=0; i<4; i++)	
b_pu_type_index2[i]	ae(v)

表 33（续）

编码单元定义	描述符
}	
if (CuType == 'F_N' && RefPicNum > 1 && DualHypothesisPredictionEnableFlag == 1) {	
for (i=0; i<4; i++)	
f_pu_type_index2[i]	ae(v)
}	
if (IntraCuFlag == 1) {	
if ((SizeInBit == 3) || ((NsipEnableFlag==1) && (3<SizeInBit<6)))	
transform_split_flag	ae(v)
if (NsipEnableFlag==1 && (SizeInBit==4 || SizeInBit==5) && TransformSplitFlag==1)	
intra_pu_type_index	
for (i=0; i<NumOfIntraPredBlock; i++)	
intra_luma_pred_mode	ae(v)
if (chroma_format != '00')	
intra_chroma_pred_mode	ae(v)
}	
if ((PictureType == 1 || PictureType == 3) && RefPicNum > 1) {	
for (i = 0; i < CuMvNum; i++)	
pu_reference_index	ae(v)
}	
if ((PictureType == 3 && ((CuTypeIndex >= 2 && CuTypeIndex <= 4 && FPuTypeIndex == 0) || (CuTypeIndex == 5 && FFourPuTypeIndex2 == 0)))) {	
if (! ((SizeInBit == 3) && (CuTypeIndex == 3 || CuTypeIndex == 4)))	
dir_multi_hypothesis_mode	ae(v)
}	
for (i = 0; i < maxPredUnitOrder; i++) {	
if (FwdMVExist) {	
mv_diff_x_abs	ae(v)
if (MvDiffXAbs)	
mv_diff_x_sign	ae(v)
mv_diff_y_abs	ae(v)
if (MvDiffYAbs)	
mv_diff_y_sign	ae(v)
}	
}	
for (i = maxPredUnitOrder; i < 2×maxPredUnitOrder; i++) {	

表 33（续）

编码单元定义	描述符
if (BckMVExist) {	
mv_diff_x_abs	ae(v)
if (MvDiffXAbs)	
mv_diff_x_sign	ae(v)
mv_diff_y_abs	ae(v)
if (MvDiffYAbs)	
mv_diff_y_sign	ae(v)
}	
}	
if (IntraCuFlag == 0) {	
if (CuType != 'B_Direct_2N' && CuType != 'P_Direct' && CuType != 'F_Direct' && CuType != 'S_Direct')	
ctp_zero_flag	ae(v)
CuCtp = 0	
if (! CtpZeroFlag) {	
transform_split_flag	ae(v)
ctp_uv	ae(v)
CuCtp = (ctp_uv << (NumOfTransBlocks -2))	
if (ctp_uv != '0' || TransformSplitFlag)	
{	
for (i = 0; i < (NumOfTransBlocks - 2); i++)	
{	
ctp_y[i]	ae(v)
CuCtp += ctp_y[i] << i	
}	
}	
else	
CuCtp += ctp_y[0]	
}	
}	
else {	
CuCtp = 0	
for (i = 0; i < (NumOfTransBlocks - 2); i++)	
{	
ctp_y[i]	ae(v)
CuCtp += ctp_y[i] << i	

表 33（续）

编码单元定义	描述符
}	
ctp_uv	ae(v)
CuCtp += (ctp_uv << (NumOfTransBlocks - 2))	
}	
if (! FixedQP) {	
if (CuCtp > 0) {	
cu_qp_delta	ae(v)
PreviousDeltaQP = cu_qp_delta	
}	
else {	
PreviousDeltaQP = 0	
}	
}	
for (i = 0; i < NumOfTransBlocks; i++)	
block(i, SizeInBit, TransformSplitFlag)	
}	
}	

7.1.7　变换块定义

变换块定义见表34。

表34　变换块定义

变换块定义	描述符
block(i, SizeInBit, TransformSplitFlag) {	
for (x=0; x<M_1; x++) {	
for (y=0; y<M_2; y++)	
QuantCoeffMatrix[x][y] = 0	
}	
if (CuCtp & (1 << i)) {	
IsChroma = (i > (TransformSplitFlag * 3))	
Log2TransformSize = IsChroma ? (SizeInBit-1) : Clip3(0, 5, (SizeInBit – TransformSplitFlag))	
if (SizeInBit == 6 && TransformSplitFlag == 1 && ! IsChroma)	
Log2TransformSize = 4	
idxW = Log2TransformWidth - 2	
idxH = Log2TransformHeight - 2	
LastCGPos = 0	

表 34（续）

变换块定义	描述符
if (Log2TransformSize == 3) {	
last_cg_pos	ae(v)
LastCGPos = last_cg_pos	
}	
else if (Log2TransformSize >3) {	
last_cg0_flag	ae(v)
if (last_cg0_flag != 0) {	
last_cg_x	ae(v)
if (last_cg_x == 0) {	
if (! IsChroma && IntraModeIdx == 2)	
MaxNum = MaxCGNumX	
else	
MaxNum = MaxCGNumY	
if (MaxNum > 1)	
last_cg_y_minus1	ae(v)
LastCGY = last_cg_y_minus1 + 1	
}	
else {	
last_cg_y	ae(v)
LastCGY = last_cg_y	
}	
if (IntraModeIdx == 2 && IsChroma == 0) {	
LastCGX = LastCGY	
LastCGY = last_cg_x	
}	
else {	
LastCGX = last_cg_x	
}	
LastCGPos = ScanCGInBlk[idxW][idxH][LastCGX][LastCGY]	
}	
}	

表 34（续）

变换块定义	描述符
for(CGPos = LastCGPos; CGPos >= 0; CGPos--) {	
if((CGPos < LastCGPos))	
nonzero_cg_flag	ae(v)
if (NonZeroCGFlag != 0) {	
CGStartPos = CGPos << 4	
CGx = InvScanCGInBlk[idxW][idxH][CGPos][0]	
CGy = InvScanCGInBlk[idxW][idxH][CGPos][1]	
last_coeff_pos_x	ae(v)
last_coeff_pos_y	ae(v)
if (CGx ==0 && CGy >0 && (IsChroma || (! IsChroma && IntraModeIdx == 2))) {	
LastCoeffPosX = LastCoeffPosYtemp	
LastCoeffPosY = LastCoeffPosXtemp	
}	
else {	
LastCoeffPosX = LastCoeffPosXtemp	
LastCoeffPosY = LastCoeffPosYtemp	
}	
if (priIdx != 0) {	
LastCoeffPosX = 3 - LastCoeffPosX	
if (IsChroma || (! IsChroma && IntraModeIdx == 2))	
LastCoeffPosY = 3 - LastCoeffPosY	
}	
CoeffPosInCG = ScanCoeffInCG[LastCoeffPosX][LastCoeffPosY]	
for (x=0; x<16; x++)	
AbsLevelArray[x] = 0	
j = 0	
while (CoeffPosInCG >= 0) {	
coeff_level_minus1_band[j]	ae(v)
coeff_level_minus1_pos_in_band[j]	ae(v)
AbsLevel[j] = CoeffLevelMinus1Band[j]×32 + CoeffLevelMinus1PosInBand[j] + 1	

表 34（续）

变换块定义	描述符
AbsLevelArray[CoeffPosInCG] = AbsLevel[j]	
absSum = 0	
for (k=CoeffPosInCG+1; k <= min(CoeffPosInCG+6,15); k++)	
absSum += AbsLevelArray[k]	
if (CoeffPosInCG > 0)	
coeff_run[j]	ae(v)
CoeffPosInCG -= (coeff_run[j] + 1)	
j ++	
}	
CoeffPosInCG = ScanCoeffInCG[LastCoeffPosX][LastCoeffPosY]	
j = 0	
while (CoeffPosInCG >= 0) {	
coeff_sign[j]	ae(v)
PosxInBlk = InvScanCoeffInBlk[idxW][idxH][CGStartPos + CoeffPosInCG][0]	
PosyInBlk = InvScanCoeffInBlk[idxW][idxH][CGStartPos + CoeffPosInCG][1]	
QuantCoeffMatrix[PosxInBlk][PosyInBlk]=coeff_sign[j]?–AbsLevel[j]:AbsLevel[j]	
CoeffPosInCG -= (coeff_run[j] + 1)	
j ++	
}	
}	
}	
}	
}	

7.1.8 自适应修正滤波参数定义

自适应修正滤波参数定义见表35。

表35　自适应修正滤波参数定义

自适应修正滤波参数定义	描述符
alf_parameter_set( ) {	
if (PictureAlfEnableFlag[0] == 1) {	
alf_filter_num_minus1	ue(v)
for (i=0; i<alf_filter_num_minus1+1; i++) {	
if (i > 0 && alf_filter_num_minus1 != 15)	
alf_region_distance[i]	ue(v)
for (j=0; j<9; j++)	

表 35（续）

自适应修正滤波参数定义	描述符
alf_coeff_luma[i][j]	se(v)
}	
}	
if (PictureAlfEnableFlag[1] == 1) {	
for (j=0; j<9; j++)	
alf_coeff_chroma[0][j]	se(v)
}	
if (PictureAlfEnableFlag[2] == 1) {	
for (j=0; j<9; j++)	
alf_coeff_chroma[1][j]	se(v)
}	
}	

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36。

表36 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	时域可伸缩扩展
0100	版权扩展
0101	保留
0110	保留
0111	图像显示扩展
1000	保留
1001	保留
1010	目标设备显示和内容元数据扩展
1011	摄像机参数扩展
1100	感兴趣区域参数扩展
1101～1111	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或G图像后续的B图像可能因缺少参考图像，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。

档次标号 profile_id

8位无符号整数。表示位流符合的档次。

级别标号 level_id

8位无符号整数。表示位流符合的级别。

档次和级别见附录B。

逐行序列标志 progressive_sequence

二值变量。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’表示编码视频序列只包含逐行扫描图像，或表示编码视频序列只包含隔行扫描图像。

如果progressive_sequence的值为‘1’，相邻两个显示时刻间隔为帧周期。如果progressive_sequence的值为‘0’，相邻两个显示时刻间隔为场周期。

场图像序列标志 field_coded_sequence

二值变量。值为‘1’表示编码视频序列中的图像是场图像；值为‘0’表示编码视频序列中的图像是帧图像。如果progressive_sequence的值为‘1’，则field_coded_sequence的值应为‘0’。

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。

MinCuWidth计算如式（11）所示。

$$MinCuWidth = (horizontal\_size + MiniSize - 1) / MiniSize \tag{11}$$

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，当progressive_sequence和field_coded_sequence的值均为‘0’时，MinCuHeight计算如式（12）所示。

$$MinCuHeight = 2 \times ((vertical\_size + 2 \times MiniSize - 1) / (2 \times MiniSize)) \tag{12}$$

在其他情况下，MinCuHeight计算如式（13）所示。

$$MinCuHeight = (vertical\_size + MiniSize - 1) / MiniSize \tag{13}$$

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

MiniSize × MiniSize 为最小的编码单元的大小。MiniSize的值由档次规定，见附录B。

horizontal_size、vertical_size与图像边界的关系见图6。图6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MinCuWidth和MinCuHeight决定。例如horizontal_size的值为1920，vertical_size的值为1080，则当progressive_sequence和field_coded_sequence的值均为‘0’时，MinCuWidth × MiniSize等于1920，MinCuHeight × MiniSize等于1088；否则MinCuWidth × MiniSize等于1920，MinCuHeight × MiniSize等于1080。

图6　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37。

表37　色度格式

chroma_format的值	含义
00	保留
01	4:2:0
10	保留
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38。如果sample_precision的值为‘001’，SamplePrecision的值为8；如果sample_precision的值为‘010’，SamplePrecision的值为10。

表38　样本精度

sample_precision的值	含义
000	禁止
001	亮度和色度均为8bit精度
010	亮度和色度均为10bit精度
011～111	保留

编码样本精度　encoding_precision

3位无符号整数。规定亮度和色度样本的编码精度，见表39。如果encoding_precision的值为‘001’，BitDepth的值为8；如果encoding_precision的值为‘010’，BitDepth的值为10。如果位流中不存在encoding_precision，则BitDepth的值为8。BitDepth的值不应小于SamplePrecision的值。

表39　编码样本精度

encoding_precision的值	含义
000	禁止
001	亮度和色度均为8bit精度
010	亮度和色度均为10bit精度
011～111	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40。

表40　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。样本宽高比按式（14）计算。

$$SAR = DAR \times vertical\_size \div horizontal\_size \quad\cdots\cdots(14)$$

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，样本宽高比按式（15）计算。

$$SAR = DAR \times display\_vertical\_size \div display\_horizontal\_size \quad\cdots\cdots(15)$$

帧率代码　frame_rate_code

4位无符号整数。规定帧率，见表41。

表41　帧率代码

frame_rate_code的值	帧率
0000	禁止
0001	24000÷1001(23.976...)
0010	24
0011	25

表 41（续）

frame_rate_code的值	帧率
0100	30000÷1001(29.97...)
0101	30
0110	50
0111	60000÷1001(59.94...)
1000	60
1001	100
1010	120
1011	200
1100	240
1101	300
1110～1111	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。

如果progressive_sequence的值为‘1’，帧周期等于帧率的倒数。

如果progressive_sequence的值为‘0’，场周期等于帧率的倒数的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。BirRate按式（16）计算。

$$BitRate = (bit\_rate\_upper \ll 18) + bit\_rate\_lower \quad (16)$$

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

二值变量。值为‘1’表示视频序列不包含B图像，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’表示视频序列可包含B图像，存在图像重排序延时，位流中不包含所谓“大图像”（见附录C）。

时间层标识允许标志　temporal_id_enable_flag

二值变量。值为‘1’表示视频序列允许使用时间层标识；值为‘0’表示视频序列不使用时间层标识。

位流缓冲区尺寸　bbv_buffer_size

18位无符号整数。规定了位流参考缓冲区管理对视频序列解码的位流缓冲区尺寸（见附录C）。BBS是位流参考缓冲区管理对视频序列解码所需的位流缓冲区最小尺寸（按位计算），见式（17）。

$$BBS = 16 \times 1024 \times bbv\_buffer\_size \quad (17)$$

最大编码单元尺寸　lcu_size

3位无符号整数。表示最大编码单元的大小，LcuSizeInBit的值等于lcu_size的值。

加权量化允许标志　weight_quant_enable_flag

二值变量。值为‘1’表示视频序列允许使用加权量化；值为‘0’表示视频序列不使用加权量化。WeightQuantEnableFlag的值等于weight_quant_enable_flag的值。

加权量化矩阵加载标志　load_seq_weight_quant_data_flag

二值变量。值为‘1’表示4×4和8×8变换块的加权量化矩阵按照表16（见7.1.2.4）从序列头中加载；值为‘0’表示4×4和8×8变换块的加权量化矩阵根据附录D

确定。LoadSeqWeightQuantDataFlag的值等于load_seq_weight_quant_data_flag的值。如果位流中不存在load_seq_weight_quant_data_flag，LoadSeqWeightQuantDataFlag的值等于0。

场景图像禁用标志　scene_picture_disable_flag

二值变量。值为‘1’表示视频序列不应包括G图像、GB图像和S图像；值为‘0’表示视频序列可包含G图像、GB图像和S图像。ScenePictureDisableFlag的值等于scene_picture_disable_flag的值。

多参考跳过模式允许标志　multi_hypothesis_skip_enable_flag

二值变量。值为‘1’表示视频序列中的F图像可使用多参考跳过模式；值为‘0’表示不应使用多参考跳过模式。MultiHypothesisSkipEnableFlag的值等于multi_hypothesis_skip_enable_flag的值。

多参考图像双前向预测模式允许标志　dual_hypothesis_prediction_enable_flag

二值变量。值为‘1’表示视频序列中的F图像可使用多参考图像的双前向预测模式；值为‘0’表示不应使用多参考图像的双前向预测模式。DualHypothesisPredictionEnableFlag的值等于dual_hypothesis_prediction_enable_flag的值。

加权跳过模式允许标志　weighted_skip_enable_flag

二值变量。值为‘1’表示视频序列中的F图像可使用加权跳过模式；值为‘0’表示不应使用加权跳过模式。WeightedSkipEnableFlag的值等于weighted_skip_enable_flag的值。

非对称运动划分允许标志　asymmetric_motion_partitions_enable_flag

二值变量。值为‘1’表示可使用非对称运动划分；值为‘0’表示不应使用非对称运动划分。AsymmetricMotionPartitionsEnableFlag的值等于asymmetric_motion_partitions_enable_flag的值。

非正方形四叉树变换允许标志　nonsquare_quadtree_transform_enable_flag

二值变量。值为‘1’表示可使用非正方形变换；值为‘0’表示不应使用非正方形变换。NsqtEnableFlag的值等于nonsquare_quadtree_transform_enable_flag的值。

非正方形帧内预测允许标志　nonsquare_intra_prediction_enable_flag

二值变量。值为‘1’表示可使用非正方形帧内预测；值为‘0’表示不应使用非正方形帧内预测。NsipEnableFlag的值等于nonsquare_intra_prediction_enable_flag的值。如果NsqtEnableFlag的值为0，则NsipEnableFlag的值也应为0。

二次变换允许标志　secondary_transform_enable_flag

二值变量。值为‘1’表示可使用二次变换；值为‘0’表示不应使用二次变换。SecondaryTransformEnableFlag的值等于secondary_transform_enable_flag的值。

样值偏移补偿允许标志　sample_adaptive_offset_enable_flag

二值变量。值为‘1’表示可使用样值偏移补偿；值为‘0’表示不应使用样值偏移补偿。SaoEnableFlag的值等于sample_adaptive_offset_enable_flag的值。

自适应修正滤波允许标志　adaptive_loop_filter_enable_flag

二值变量。值为‘1’表示可使用自适应修正滤波；值为‘0’表示不应使用自适应修正滤波。AlfEnableFlag的值等于adaptive_loop_filter_enable_flag的值。

渐进运动矢量允许标志　pmvr_enable_flag

二值变量。值为‘0’表示PmvrThreshold的值为-1；值为‘1’表示PmvrThreshold的值为2。

参考图像配置集数量　num_of_rcs

6位无符号整数。表示参考图像配置集的数量。NumOfRcs的值等于num_of_rcs的值。符合本部分的位流中num_of_rcs的值应小于等于32。

图像重排序延迟　output_reorder_delay

5位无符号整数。由于图像编解码顺序与显示顺序不一致带来的重排序延迟，以解码图像为单位。由于一幅解码图像的显示时间和progressive_sequence、progressive_frame、repeat_first_field、picture_structure等语法元素的值有关，所以这段时间的绝对长度是不固定的，但是在这段时间内显示的解码图像数是固定的。low_delay值为‘0’时，OutputReorderDelay的值等于output_reorder_delay的值；low_delay值为‘1’时，OutputReorderDelay的值为0。

跨条带环路滤波 cross_slice_loopfilter_enable_flag

二值变量。值为‘1’时表示可跨越条带边界进行去块效应滤波、样本偏移补偿及自适应修正滤波；值为‘0’表示不应跨越条带边界进行去块效应滤波、样本偏移补偿及自适应修正滤波。

7.2.2.3 参考图像配置集

被参考标志 refered_by_others_flag[i]

二值变量。值为‘1’表示当前图像可用做参考图像；值为‘0’表示当前图像不应用做参考图像。i 是参考图像配置集的编号。ReferedByOthersFlag[i]的值等于 refered_by_others_flag[i]的值。如果当前图像是 GB 图像，则 refer_by_others_flag[i]的值应为‘1’。

参考图像数量 num_of_reference_picture[i]

3 位无符号整数。表示当前图像的参考图像数量。参考图像数量不应超过参考图像缓冲区的大小。NumOfRefPic[i]的值等于 num_of_reference_picture[i]的值。i 是参考图像配置集的编号。

符合本部分的位流应满足以下要求：

a) 如果当前图像的 PictureType 等于 0，则 num_of_reference_picture[i]的值应为 0；
b) 如果当前图像的 PictureType 等于 1 或 3，则 num_of_reference_picture[i]的值应大于或等于 1。
c) 如果当前图像的 PictureType 等于 2，则 num_of_reference_picture[i]的值应为 2；
d) 如果当前图像的 PictureType 等于 4，则 num_of_reference_picture[i]的值应为 1。

参考图像解码顺序偏移量 delta_doi_of_reference_picture[i][j]

6 位无符号整数，取值范围应是 1～63。如果 SceneReferenceEnableFlag 等于 0，说明当前图像参考图像队列中的参考图像与当前图像解码顺序的差值；如果 SceneReferenceEnableFlag 等于 1，说明当前图像参考图像队列中不位于 RefPicNum-1 位置的参考图像与当前图像解码顺序的差值。i 是参考图像配置集的编号，j 是参考图像的编号。对同一个参考图像配置集，不同编号的参考图像解码顺序偏移量的值应各不相同。DeltaDoiOfRefPic[i][j]的值等于 delta_doi_of_reference_picture[i][j]的值。

待移出图像数量 num_of_removed_picture[i]

3 位无符号整数。说明从当前图像开始不会被参考的图像的数量。NumOfRemovedPic[i]的值等于num_of_removed_picture[i]的值。i 是参考图像配置集的编号。

待移出图像解码顺序偏移量 delta_doi_of_removed_picture[i][j]

6 位无符号整数，取值范围应是 1～63。说明从当前图像开始不会被参考的图像与当前图像解码顺序的差值。i 是参考图像配置集的编号，j 是待移出图像的编号。对同一个参考图像配置集，不同编号的待移出图像解码顺序偏移量的值应各不相同。对同一个参考图像配置集，待移出图像和参考图像应各不相同。DeltaDoiOfRemovedPic[i][j]的值等于 delta_doi_of_removed_picture[i][j]的值。

7.2.2.4 自定义加权量化矩阵

加权量化矩阵系数 weight_quant_coeff

加权量化矩阵的系数，取值范围应是1～255。WeightQuantCoeff的值等于weight_quant_coeff的值。

7.2.2.5 扩展和用户数据

7.2.2.5.1　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’。

7.2.2.5.2　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’。

7.2.2.6　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42。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42　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

二值变量。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

二值变量。值为‘1’表示位流中有colour_primaries、transfer_characteristics和matrix_coefficients；值为‘0’表示位流中没有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43。

表43　彩色三基色

colour_primaries的值	彩色三基色
0	禁止
1	基色　x　y 绿　0.300　0.600 蓝　0.150　0.060 红　0.640　0.330 白 D65　0.3127　0.3290。[a]
2	未作规定的视频 图像特性未知
3	保留
4	基色　x　y 绿　0.21　0.71 蓝　0.14　0.08 红　0.67　0.33 白 C　0.310　0.316。[b]
5	基色　x　y 绿　0.29　0.60 蓝　0.15　0.06 红　0.64　0.33 白 D65　0.313　0.329。[c]
6	基色　x　y 绿　0.310　0.595 蓝　0.155　0.070 红　0.630　0.340 白 D65　0.3127　0.3290。[d]
7	基色　x　y 绿　0.310　0.595 蓝　0.155　0.070 红　0.630　0.340 白 D65　0.3127　0.3290。[e]
8	普通胶片（彩色滤光镜，C光源） 基色　x　y 绿　0.243　0.692 (Wratten 58) 蓝　0.145　0.049 (Wratten 47) 红　0.681　0.319 (Wratten 25) 白 C　0.310　0.316。
9	基色　x　y 绿　0.170　0.797 蓝　0.131　0.046 红　0.708　0.292 白 D65　0.3127　0.3290。[f]

表 43（续）

colour_primaries的值	彩色三基色
10～255	保留

[a] GY/T 155-2000 中规定的彩色三基色。
[b] ITU-R BT.470-2 System M 中规定的彩色三基色。
[c] ITU-R BT.470-2 System B, G 中规定的彩色三基色。
[d] SMPTE 170M 中规定的彩色三基色。
[e] SMPTE 240M (1987)中规定的彩色三基色。
[f] ITU-R BT.2020 1 中规定的彩色三基色。

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44。表43中Lc是线性输入信号，与光的强度成正比。

表44　光电转移特性

transfer_characteristics的值	光电转移特性
0	禁止
1	$E' = 1.099\ Lc^{0.45} - 0.099$，$1.33 >= Lc >= 0.018$ $E' = 4.500\ Lc$，$0.018 > Lc >= -0.0045$ $E' = -\{1.099(-4Lc)^{0.45}-0.099\}/4, -0.0045 > Lc >= -0.25$。[a]
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。[b]
5	假设显示伽玛值为2.8。[c]
6	$E' = 1.099\ Lc^{0.45} - 0.099$，$1 >= Lc >= 0.018$ $E' = 4.500\ Lc$，$0.018 > Lc >= 0$。[d]
7	$E' = 1.1115\ Lc^{0.45} - 0.1115$，$Lc >= 0.0228$ $E' = 4.0\ Lc$，$0.0228 > Lc$。[e]
8	线性转移特性 即 $E' = L_c$
9	对数转移特性（范围100:1） $E' = 1.0-(\log_{10}(Lc))/2$，$1 >= Lc >= 0.01$ $E' = 0.0$，$0.01> Lc$
10	对数转移特性（范围316.22777:1） $E' = 1.0-(\log_{10}(Lc))/2.5$，$1 >= Lc >= 0.0031622777$ $E' = 0.0$，$0.0031622777> Lc$
11	$E' = 1.0993\ Lc^{0.45} - 0.0993$，$1 >= Lc >= 0.0181$ $E' = 4.500\ Lc$，$0.0181 > Lc >= 0$。[f]

表 44（续）

transfer_characteristics的值	光电转移特性
12	$E' = ((c_1 + c_2 \times Lc^n) \div (1 + c_3 \times Lc^n))^m$ $c_1 = c_3 - c_2 + 1 = 3424 \div 4096 = 0.8359375$ $c_2 = 32 \times 2413 \div 4096 = 18.8515625$ $c_3 = 32 \times 2392 \div 4096 = 18.6875$ $m = 128 \times 2523 \div 4096 = 78.84375$ $n = 0.25 \times 2610 \div 4096 = 0.1593017578125$ Lc等于1对应于显示亮度为10000cd/m^2。[g]
13	保留
14	$E' = (3 \times Lc)^{0.5}, 1/12 >= Lc >= 0$ $E' = a \times Ln(12 \times Lc - b) + c, 1 >= Lc > 1/12$ $a = 0.17883277$ $b = 0.28466892$ $c = 0.55991073$。[g]
15～255	保留

[a] GY/T 155-2000 中规定的光电转移特性。

[b] ITU-R BT.470 System M 中规定的光电转移特性。

[c] ITU-R BT.470 System B, G 中规定的光电转移特性。

[d] ITU-R BT.709, SMPTE 170M, ITU-R BT.2020 10-bit System 中规定的光电转移特性。

[e] SMPTE 240M 中规定的光电转移特性。

[f] ITU-R BT.2020 12-bit System 中规定的光电转移特性。

[g] GY/T 315—2018 中规定的光电转移特性。

如果位流中没有序列显示扩展，或者colour_description的值是‘0’，则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45。

表45　彩色信号转换矩阵

matrix_coefficients的值	彩色信号转换矩阵
0	禁止
1	$E_Y' = 0.2126\ E_R' + 0.7152\ E_G' + 0.0722\ E_B'$ $E_{CB}' = (E_B' - E_Y')/1.8556$ $E_{CR}' = (E_R' - E_Y')/1.5748$。[a]
2	未作规定的视频 图像特性未知
3	保留
4	$E'_Y = 0.59\ E'_G + 0.11\ E'_B + 0.30\ E'_R$ $E'_{CB} = -0.331\ E'_G + 0.500\ E'_B - 0.169\ E'_R$ $E'_{CR} = -0.421 E'_G - 0.079\ E'_B + 0.500\ E'_R$。[b]

表 45（续）

matrix_coefficients的值	彩色信号转换矩阵
5	$E'_Y = 0.587\ E'_G + 0.114\ E'_B + 0.299\ E'_R$ $E'_{CB} = -0.331\ E'_G + 0.500\ E'_B - 0.169\ E'_R$ $E'_{CR} = -0.419\ E'_G - 0.081\ E'_B + 0.500\ E'_R$ 。[c]
6	$E'_Y = 0.587\ E'_G + 0.114\ E'_B + 0.299\ E'_R$ $E'_{CB} = -0.331\ E'_G + 0.500\ E'_B - 0.169\ E'_R$ $E'_{CR} = -0.419\ E'_G - 0.081\ E'_B + 0.500\ E'_R$ 。[d]
7	$E'_Y = 0.701\ E'_G + 0.087\ E'_B + 0.212\ E'_R$ $E'_{CB} = -0.384\ E'_G + 0.500\ E'_B - 0.116\ E'_R$ $E'_{CR} = -0.445\ E'_G - 0.055\ E'_B + 0.500\ E'_R$ 。[e]
8	$E_Y' = 0.2627\ E_R' + 0.6780\ E_G' + 0.0593\ E_B'$ $E_{CB}' = (E_B' - E_Y')/1.8814$ $E_{CR}' = (E_R' - E_Y')/1.4746$。[f]
9	$E'_Y = (0.2627\ E_R + 0.6780\ E_G + 0.0593\ E_B)'$ $E'_{CB} = \begin{cases} \dfrac{E'_B - E'_Y}{-2N_B}, & N_B \le E'_B - E'_Y \le 0 \\ \dfrac{E'_B - E'_Y}{2P_B}, & 0 \le E'_B - E'_Y \le P_B \end{cases}$ $E'_{CR} = \begin{cases} \dfrac{E'_R - E'_Y}{-2N_R}, & N_R \le E'_R - E'_Y \le 0 \\ \dfrac{E'_R - E'_Y}{2P_R}, & 0 \le E'_R - E'_Y \le P_R \end{cases}$ 其中， $P_B = 0.7910,\ N_B = -0.9702$ $P_R = 0.4969,\ N_R = -0.8591$。[g]
10～255	保留

[a] GY/T 155-2000 中规定的彩色信号转换矩阵。

[b] FCC 中规定的彩色信号转换矩阵。

[c] ITU-R BT.470 System B, G 中规定的彩色信号转换矩阵。

[d] SMPTE 170M 中规定的彩色信号转换矩阵。

[e] SMPTE 240M 中规定的彩色信号转换矩阵。

[f] ITU-R BT.2020 非恒定亮度系统中规定的彩色信号转换矩阵。

[g] ITU-R BT.2020 恒定亮度系统中规定的彩色信号转换矩阵。

在表45中：

a) E'_Y 是值在 0 和 1 之间的模拟量；

b) E'_{CB} 和 E'_{CR} 是值在-0.5 和 0.5 之间的模拟量；

c) E'_R、E'_G 和 E'_B 是值在 0 和 1 之间的模拟量；

d) Y、Cb 和 Cr 与 E'_Y、E'_{CB} 和 E'_{CR} 的关系如下：

如果sample_range的值为‘0’：

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

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

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

如果sample_range的值为‘1’：

$Y = ((2^{BitDepth} - 1) \times E'_{Y})$

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

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

其中BitDetph是编码样本精度。例如：

BitDepth = 8，sample_range的值为‘0’时：

$Y = (219 \times E'_{Y}) + 16$

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

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

Y的取值范围是16～235，Cb和Cr的取值范围是16～240。

BitDepth = 8，sample_range的值为‘1’时：

$Y = (255 \times E'_{Y})$

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

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

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 $0 \sim 2^{BitDepth}-1$。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。

立体视频模式标志　td_mode_flag

二值变量。值为‘0’表示当前视频序列是单目视频；值为‘1’表示当前视频序列包含多个视点或深度信息。

立体视频拼接模式　td_packing_mode

8位无符号整数。规定立体视频的拼接方式，见表46。

表46　立体视频拼接模式

td_packing_mode的值	含义
0	左右拼接
1	上下拼接
其他	保留

表45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。

视点反转标志 view_reverse_flag

二值变量。值为‘0’表示视点顺序不变；值为‘1’表示视点顺序反转。

7.2.2.7 时域可伸缩扩展

视频扩展标号 extension_id

位串‘0011’。标识版权扩展。

时间层数量 num_of_temporal_level_minus1

3位无符号整数。表示视频序列包含的时间层数量。时间层数量等于视频序列中所有图像的最高时间层标识的值与最低时间层标识的值之差加1。时间层数量的值不应该超过MAX_TEMPORAL_ID的值。

时间层帧率代码 temporal_frame_rate_code[i]

4位无符号整数。表示截取到时间层标识值等于i时的帧率，帧率见表40。i表示视频序列中所有图像的最高时间层标识。

时间层比特率低位 temporal_bit_rate_lower[i]

18位无符号整数。表示截取到时间层标识值等于i时的BitRate的低18位。i表示视频序列中所有图像的最高时间层标识。

时间层比特率高位 temporal_bit_rate_upper[i]

12位无符号整数。表示截取到时间层标识值等于i时的BitRate的高12位。i表示视频序列中所有图像的最高时间层标识。

7.2.2.8 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

二值变量。值为‘1’表示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’表示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’，表示没有相关版权信息。

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

二值变量。值为‘1’表示源视频的内容是原创的；值为‘0’表示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位到第63位。

版权号2 copyright_number_2

22位无符号整数。CopyrightNumber的第22位到第43位。

版权号3 copyright_number_3

22位无符号整数。CopyrightNumber的第0位到第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。

7.2.2.9 目标显示设备和内容元数据扩展

视频扩展标号 extension_id

位串‘1010’。标识目标显示设备和内容元数据扩展。

显示设备三基色X坐标，显示设备三基色Y坐标 display_primaries_x[c]，display_primaries_y[c]

16位无符号整数。分别表示归一化后的显示设备三基色的色度x坐标和y坐标。该坐标符合ISO 11664-1中规定的CIE 1931，以0.00002为单位，范围从0到50000。c的值为0、1、2分别对应于绿、蓝、红三色。

显示设备标准白光X坐标，显示设备标准白光Y坐标 white_point_x，white_point_y

16位无符号整数。分别表示归一化后的显示设备标准白光的色度x坐标和y坐标。该坐标符合CIE 1931（见ISO 11664-1/CIE S 014-1、ISO 11664-3/CIE S 014-3和CIE S 015），以0.00002为单位，范围从0到50000。

显示设备最大显示亮度 max_display_mastering_luminance

16位无符号整数。表示显示设备的最大显示亮度。以1cd/m^{2}为单位，范围从1cd/m^{2}到65535cd/m^2。

显示设备最小显示亮度 min_display_mastering_luminance

16位无符号整数。表示显示设备的最小显示亮度。以0.0001cd/m^{2}为单位，范围从0.0001cd/m^{2}到6.5535cd/m^2。

max_display_mastering_luminance的值应大于min_display_mastering_luminance的值。

显示内容最大亮度 max_content_light_level

16位无符号整数。表示显示内容的最大亮度。以1cd/m^{2}为单位，范围从1cd/m^{2}到65535cd/m^2。

max_content_light_level的值为某一显示内容的所有显示图像的最大亮度PictureMaxLightLevel的最大值。显示图像最大亮度PictureMaxLightLevel计算如下：

a) 对显示图像有效显示区域内的所有像素依次计算像素的 R、G、B 分量的最大值 maxRGB。有效显示区域是由 display_horizontal_size 和 display_vertical_size 共同定义的矩形区域：
   1) 将像素的非线性(R’,G’,B’)值转换为线性(R,G,B)值，并校准为以 1cd/m^2 为单位的值；
   2) 由像素校准后的(R,G,B)值，计算得到像素 R、G、B 分量的最大值 maxRGB。
b) 显示图像的 PictureMaxLightLevel 等于有效显示区域内的所有像素的 maxRGB 中的最大值。

显示内容最大图像平均亮度 max_picture_average_light_level

16位无符号整数。表示显示内容的最大图像平均亮度。以1cd/m^{2}为单位，范围从1cd/m^{2}到65535cd/m^2。

max_picture_average_light_level的值为某一显示内容的所有显示图像的图像平均亮度PictureAverageLightLevel的最大值。显示图像平均亮度PictureAverageLightLevel计算如下：

a) 对显示图像有效显示区域内的所有像素依次计算像素的 R、G、B 分量的最大值 maxRGB。有效显示区域是由 display_horizontal_size 和 display_vertical_size 共同定义的矩形区域：
   3) 将像素的非线性(R’,G’,B’)值转换为线性(R,G,B)值，并校准为以 1cd/m^2 为单位的值；
   4) 由像素校准后的(R,G,B)值，计算得到像素 R、G、B 分量的最大值 maxRGB。
b) 显示图像的 PictureAverageLightLevel 等于有效显示区域内的所有像素的 maxRGB 的平均值。

7.2.2.10 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4194.303mm。

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4194.303mm。

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4194.303。

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-2147483.648mm到2147483.647mm。

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-2097152到2097151。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-2097152到2097151。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7和图8所示。

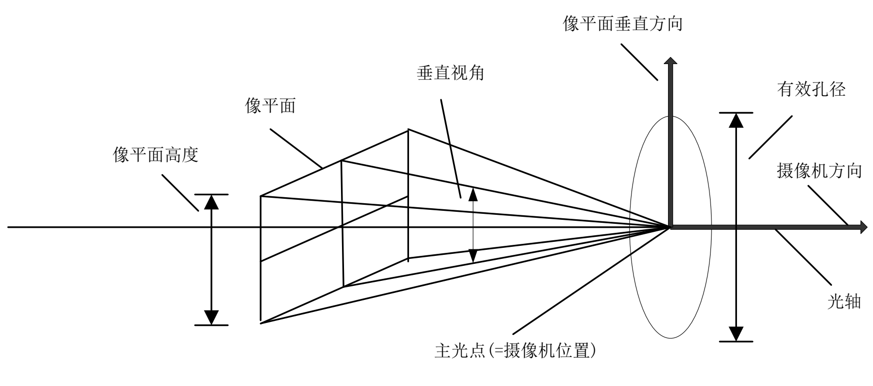


图7　摄像机原理示意图

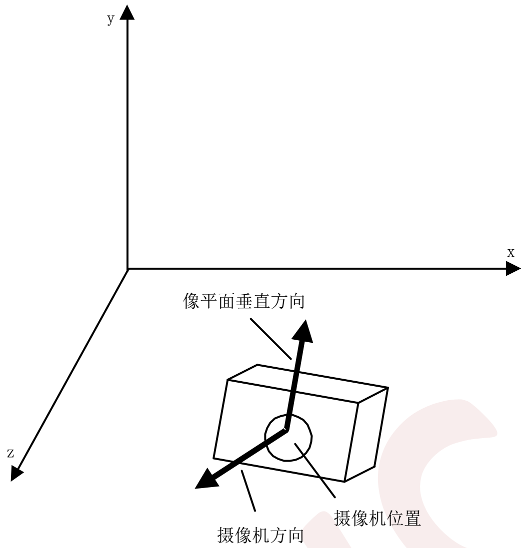


图8　摄像机坐标系统示意图

7.2.2.11　感兴趣区域参数扩展

感兴趣区域参数扩展标号　extension_id

位串‘1100’。标识ROI参数扩展。

ROIInfo数组存储了当前图像的感兴趣区域（ROI）信息，其中ROIInfo[i]表示当前图像第i个ROI的信息。ROIInfo[i]为一结构体，其中每个变量包括asisx、asisy、width和height四个参数。ROIInfo[i]->asisx表示第i个ROI的左上角区域在图像中的横坐标位置，ROIInfo[i]->asisy表示第i个ROI的左上角区域在图像中的纵坐标位置，ROIInfo[i]->width表示第i个ROI在图像中的宽度，ROIInfo[i]->width的单位为图像的列数，ROIInfo[i]->height表示第i个ROI在图像中的高度，ROIInfo[i]->height的单位为图像的行数。

PrevROIInfo数组存储了当前图像解码顺序前一幅图像的感兴趣区域（ROI）信息，其中PrevROIInfo[i]表示当前图像解码顺序前一幅图像第i个ROI的信息。PrevROIInfo[i]为一结构体，其中每个变量包括asisx、asisy、width和height四个参数。PrevROIInfo[i]->asisx表示第i个ROI的左上角区域在图像中的横坐标位置，PrevROIInfo[i]->asisy表示第i个ROI的左上角区域在图像中的纵坐标位置，PrevROIInfo[i]->width表示第i个ROI在图像中的宽度，PrevROIInfo[i]->width的单位为图像的列数，PrevROIInfo[i]->height表示第i个ROI在图像中的高度，PrevROIInfo[i]->height的单位为图像的行数。

当前图像感兴趣区域数　current_picture_roi_num

8位无符号整数。表示当前图像的ROI的个数。

前一幅图像感兴趣区域数　prev_picture_roi_num

8位无符号整数。表示当前图像按解码顺序前一幅图像的ROI的个数。PrevPictureROINum的值等于prev_picture_roi_num的值。如果位流中不存在prev_picture_roi_num，PrevPictureROINum的值为0。

跳过模式感兴趣区域数　roi_skip_run

表示ROI被标记为跳过模式的个数。当ROI被标记为跳过模式时，ROI的信息由skip_roi_mode导出。

感兴趣区域跳过模式　skip_roi_mode[i+j]

1位无符号整数。值为0表示当前图像按解码顺序前一幅图像中第i+j个ROI在当前图像中已经不存在。值为1表示当前图像按解码顺序前一幅图像中ROI参数为ROIInfo[i+j]的ROI与当前图像标号为roiIndex的ROI的参数相同。

感兴趣区域横坐标增量 roi_axisx_delta

表示一个ROI左上角在图像中的横坐标的增量。

感兴趣区域纵坐标增量 roi_axisy_delta

表示一个ROI左上角在图像中的纵坐标的增量。

感兴趣区域宽度增量 roi_width_delta

表示一个ROI在图像中的宽度的增量。

感兴趣区域高度增量 roi_height_delta

表示一个ROI在图像中的高度的增量。

roi_axisx_delta、roi_axisy_delta、roi_width_delta和roi_height_delta的单位是像素，用于获得ROIInfo[roiIndex]->asisx、ROIInfo[roiIndex]->asisy、ROIInfo[roiIndex]->width和ROIInfo[roiIndex]->height。ROIInfo[roiIndex]->width的值不应大于horizontal_size，ROIInfo[roiIndex]->height的值不应大于vertical_size，ROIInfo[roiIndex]->asisx+ROIInfo[roiIndex]->width的值不应大于horizontal_size，ROIInfo[roiIndex]->asisy+ROIInfo[roiIndex]->height的值不应大于vertical_size。

感兴趣区域横坐标 roi_axisx

表示一个ROI左上角在图像中的横坐标。

感兴趣区域纵坐标 roi_axisy

表示一个ROI左上角在图像中的纵坐标。

感兴趣区域宽度 roi_width

表示一个ROI在图像中的宽度。

感兴趣区域高度 roi_height

表示一个ROI在图像中的高度。

roi_axisx、roi_axisy、roi_width和roi_height的单位是像素，用于获得ROIInfo[roiIndex]->asisx、ROIInfo[roiIndex]->asisy、ROIInfo[roiIndex]->width和ROIInfo[roiIndex]->height。ROIInfo[roiIndex]->width的值不应大于horizontal_size，ROIInfo[roiIndex]->height的值不应大于vertical_size，ROIInfo[roiIndex]->asisx+ROIInfo[roiIndex]->width的值不应大于horizontal_size，ROIInfo[roiIndex]->asisy+ROIInfo[roiIndex]->height的值不应大于vertical_size。

7.2.2.12 高动态范围图像元数据扩展

HDR动态元数据扩展标号 extension_id

位串‘0101’。标识HDR动态元数据扩展。

视频扩展数据字节 extension_data_byte

8位无符号整数。视频扩展数据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’。

7.2.3 图像

7.2.3.1 帧内预测图像头

帧内预测图像起始码 intra_picture_start_code

位串‘0x000001B3’。标识I、G或GB图像的开始。

BBV延时 bbv_delay

32位无符号整数。

BbvDelay的值等于bbv_delay的值。如果temporal_id_enable_flag的值为‘1’，则bbv_delay的值应为BbvDelayMax。BbvDelayMax的值等于0xFFFFFFFF。

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27MHz系统时钟导出的90kHz时钟周期数来表示。如果视频序列中某一幅图像的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

二值变量。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：TimeCodeHours、TimeCodeMinutes、TimeCodeSeconds和TimeCodePictures，见表47。表中TimeCodeHours，TimeCodeMinutes，TimeCodeSeconds和TimeCodePictures字段是无符号整数。time_code描述从当前图像开始（含当前图像）的位流中第一幅图像（显示顺序）的显示时间。

表47 时间编码（time_code）

time_code 的字段	取值	单位	描述符
保留	0	--	u(1)
TimeCodeHours	0～23	小时	u(5)
TimeCodeMinutes	0～59	分	u(6)
TimeCodeSeconds	0～59	秒	u(6)
TimeCodePictures	0～63	如果帧率小于64，单位是图像；否则，单位是1/64秒	u(6)

场景图像标志 scene_picture_flag

二值变量。值为‘1’ 表示当前图像是场景图像，场景图像包括G图像或GB图像；值为‘0’表示当前图像是I图像。ScenePictureFlag的值等于scene_picture_flag的值。如果位流中不存在scene_picture_flag，则ScenePictureFlag的值为0。

场景图像输出标志 scene_picture_output_flag

二值变量。值为‘1’表示当前图像是G图像，应被输出；值为‘0’表示当前图像是GB图像，不应被输出。符合本部分的位流中每两个相邻的GB图像之间的位流所包含的图像起始码的数量不应小于120。

解码顺序索引 decode_order_index

8位无符号整数。说明当前图像的解码顺序索引值。当前图像的解码顺序索引DOI的值等于decode_order_index的值。GB图像的DOI应等于其解码顺序的前一幅图像的解码顺序索引值。

时间层标识 temporal_id

3位无符号整数。说明当前图像的时间层标识。时间层标识的取值范围是0～MAX_TEMPORAL_ID。G或GB图像的时间层标识应为0。如果位流中不存在temporal_id，则当前图像的时间层标识应为0。时间层标识为0说明是最低层。

图像输出延迟　picture_output_delay

从图像完成解码到输出需要等待的时间，以解码图像（不包括GB图像）为单位，解析过程见8.2。除GB图像外，low_delay的值为‘0’时，PictureOutputDelay的值等于picture_output_delay的值。low_delay的值为‘1’时，PictureOutputDelay的值为0。picture_output_delay的值应小于64。当前图像的显示顺序索引POI的值等于DOI + PictureOutputDelay - OutputReorderDelay。GB图像没有显示顺序索引。

参考图像配置集标志　use_rcs_flag

二值变量。值为‘1’表示位流中存在rcs_index；值为‘0’表示位流中存在新的参考图像配置集，其编号为NumOfRcs。UseRcsFlag的值等于use_rcs_flag的值。

参考图像配置集索引　rcs_index

5位无符号整数。rcs_index的值应小于NumOfRcs。RcsIndex的值等于rcs_index的值。如果位流中不存在rcs_index，则RcsIndex的值应为NumOfRcs。RcsIndex的值是当前图像所使用的参考图像配置集的编号。

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。

BbvCheckTimes大于0表示当前图像是一个大图像（见附录C）。

逐行帧标志　progressive_frame

二值变量。值为‘0’表示当前图像所在的帧的两场是隔行场，这两场之间存在一个场时间间隔；值为‘1’表示当前图像所在的帧的两场实际上来自同一时刻。

如果progressive_sequence的值为‘1’，则progressive_frame的值应为‘1’。如果field_coded_sequence的值为‘1’时，则progressive_frame的值应为‘0’。

图像编码结构标志　picture_structure

二值变量。picture_structure的值为‘0’表示当前帧的两场的编码数据依次出现；值为‘1’表示当前帧的两场的编码数据交融出现。

如果progressive_sequence的值为‘1’，则picture_structure的值也应为‘1’。如果field_coded_sequence的值为‘0’，则picture_structure的值应为‘1’；如果field_coded_sequence的值为‘1’，则picture_structure的值应为‘0’。PictureStructure的值等于picture_structure的值。如果位流中不存在picture_structure，则PictureStructure的值应为1。

顶场在先　top_field_first

二值变量。其含义由progressive_sequence、progressive_frame、picture_structure和repeat_first_field决定。如果field_coded_sequence的值为‘1’，则top_field_first的值应为‘1’。

a) 如果progressive_sequence和field_coded_sequence的值均是‘0’，top_field_first说明解码场的输出显示顺序。

   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；如果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一起说明显示当前帧的次数（1次、2次或3次）。

5) 如果 repeat_first_field 的值是‘0’，top_field_first 的值也应是‘0’，显示当前帧一次；
6) 如果 top_field_first 的值是‘0’，repeat_first_field 的值是‘1’，显示当前帧两次；
7) 如果 top_field_first 和 repeat_first_field 的值都是‘1’，显示当前帧三次。

c) 如果 progressive_sequence、field_coded_sequence 和 progressive_frame 的值都是‘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

二值变量。如果progressive_frame的值是‘0’，则repeat_first_field的值应为‘0’。

a) 如果 progressive_sequence 和 progressive_frame 的值都是‘0’，则 repeat_first_field 的值也应是‘0’，显示两个场，第一场后面跟着第二场。如果 field_coded_sequence 的值是‘0’，则由 top_field_first 决定是第一场是顶场还是底场。
b) 如果 progressive_sequence 的值是‘0’，progressive_frame 的值是‘1’，那么：
   8) 如果 repeat_first_field 的值是‘0’，显示两个场，第一场（由 top_field_first 决定是顶场还是底场），后面跟着第二场。
   9) 如果 repeat_first_field 的值是‘1’，显示三个场，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。

顶场场图像标志 top_field_picture_flag

二值变量。值为‘1’表示当前图像是顶场图像；值为‘0’表示当前图像是底场图像。

固定图像量化因子 fixed_picture_qp

二值变量。值为‘1’表示在该幅图像内量化因子不变；值为‘0’表示在该帧图像内量化因子可变。

图像量化因子 picture_qp

7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+8×(BitDepth–8))。

去块滤波禁用标志 loop_filter_disable_flag

二值变量。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。

去块滤波参数标志 loop_filter_parameter_flag

二值变量。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的值等于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的值等于0。

色度量化参数禁用标志 chroma_quant_param_disable_flag

二值变量。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9.5.2。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_quant_param_cb和chroma_quant_param_cr的值均为‘0’。

图像加权量化允许标志 pic_weight_quant_enable_flag

二值变量。值为'1'表示当前图像可使用加权量化；值为'0'表示当前图像不应使用加权量化。PicWeightQuantEnableFlag的值等于pic_weight_quant_enable_flag。如果当前图像的图像头位流中不存在pic_weight_quant_enable_flag，则PicWeightQuantEnableFlag的值应为0。

图像加权量化数据加载索引 pic_weight_quant_data_index

2位无符号整数。值为'00'表示当前图像的4×4和8×8变换块的加权量化矩阵根据序列头确定；值为'01'表示当前图像的4×4和8×8变换块的加权量化矩阵根据当前图像头中加载的加权量化参数导出；值为'10'表示当前图像的4×4和8×8变换块的加权量化矩阵从当前图像头中直接加载；值为'11'保留。

加权量化参数索引 weight_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 weight_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。如果当前图像头的位流中存在weight_quant_model，则WeightQuantModel的值等于weight_quant_model的值。

加权量化参数增量1 weight_quant_param_delta1[i]

加权量化参数增量2 weight_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9.2.5。如果当前图像头的位流中不存在weight_quant_param_delta1[i]或weight_quant_param_delta2[i]，则weight_quant_param_delta1[i]或weight_quant_param_delta2[i]的值为0。

图像自适应修正滤波允许标志 picture_alf_enable_flag[compIdx]

二值变量。值为'1'表示当前图像的第compIdx个分量可使用自适应修正滤波；值为'0'则表示当前图像的第compIdx个分量不应使用自适应修正滤波。PictureAlfEnableFlag[compIdx]的值等于picture_alf_enable_flag[compIdx]的值。其中compIdx等于0表示亮度分量，compIdx等于1表示Cb分量，compIdx等于2表示Cr分量。

7.2.3.2 帧间预测图像头

帧间预测图像起始码 inter_picture_start_code

位串'0x000001B6'。标识F、P、S或B图像的开始。

图像编码方式 picture_coding_type

2位无符号整数。

场景预测标志 scene_pred_flag

二值变量。值为'1'表示当前图像是S图像；值为'0'表示当前图像是P图像。ScenePredFlag的值等于scene_pred_flag的值。如果位流中不存在scene_pred_flag，ScenePredFlag的值为0。

picture_coding_type与ScenePredFlag共同规定帧间预测图像的类型，见表48。

表48 帧间预测图像的类型

picture_coding_type的值	ScenePredFlag	帧间预测图像的类型
00	-	禁止
01	0	P图像
01	1	S图像
10	-	B图像
11	-	F图像

场景图像预测模式允许标志　scene_reference_enable_flag

二值变量。值为‘1’表示应将图像从场景图像缓冲区移入参考图像队列；值为‘0’表示不应将图像从场景图像缓冲区移入参考图像队列。SceneReferenceEnableFlag的值等于scene_reference_enable_flag的值，如果位流中不存在scene_reference_enable_flag，SceneReferenceEnableFlag值为0。

如果SceneReferenceEnableFlag的值为1，场景图像缓冲区中应存在G或GB图像。

随机访问正确解码标志　random_access_decodable_flag

二值变量。值为‘1’表示当前图像只参考解码顺序在当前图像对应的序列头之后且random_access_decodable_flag的值为‘1’的图像；值为‘0’表示当前图像不一定只参考解码顺序在当前图像对应的序列头之后且random_access_decodable_flag的值为‘1’的图像。其中，对应的序列头指的是码流中在当前图像之前最近的一个序列头。RandomAccessDecodableFlag的值等于random_access_decodable_flag的值。如果当前图像的图像头中不存在random_access_decodable_flag，则当前图像的RandomAccessDecodableFlag的值应为1。

如果当前图像的RandomAccessDecodableFlag的值为0，则在其对应的序列头发生随机访问时，当前图像可能无法正确解码。

序列头后的第一个解码图像应是I、G或GB图像。如果序列头后的第一个解码图像是GB图像，则该序列头后的第二个解码图像应是S图像。解码顺序在该I、G、S图像之后，显示顺序在该I、G或S图像之前的图像称为该序列头对应的前置图像，只有前置图像的 RandomAccessDecodableFlag的值可以为0，其余情况下，图像的RandomAccessDecodableFlag的值应为1。并且对于同一个序列头对应的前置图像，RandomAccessDecodableFlag的值为0的图像的显示顺序应在RandomAccessDecodableFlag的值为1的图像的显示顺序之前。

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。

图像中心水平偏移　picture_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。

图像中心垂直偏移　picture_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。

如果一幅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：丢失图像的中心偏移量的值与前一非丢失图像的值相同。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。

图像中心偏移量参数见图9。

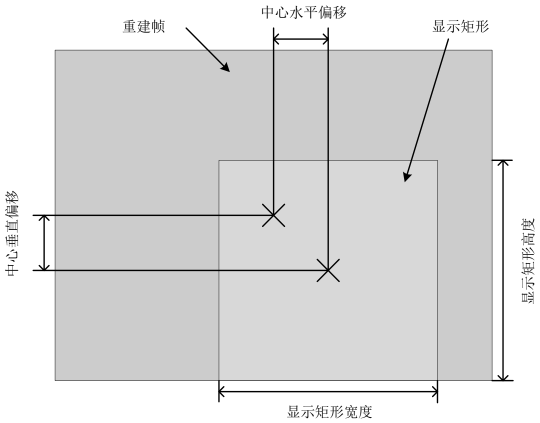


图9　图像中心偏移量参数

注1：显示矩形的尺寸可能大于重建图像。

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

注3：图 9 中，picture_centre_horizontal_offset 和 picture_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‘0000 0000 0000 0000 0000 0001’。后8位为slice_vertical_position，其范围是0x00～0x8F。

条带垂直位置　slice_vertical_position

8位无符号整数，给出条带的第一个最大编码单元在图像中的垂直位置LcuRow，以最大编码单元为单位。slice_vertical_position的值的范围应为0x00～0x8F。

如果图像的vertical_size大于$144\times2^{LcuSizeInBit}$，LcuRow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$144\times2^{LcuSizeInBit}$，位流中不应出现slice_vertical_position_extension。

LcuRow按下面的方法计算：

```
if ( vertical_size > 144×2^LcuSizeInBit )
    LcuRow = ( slice_vertical_position_extension << 7 ) + slice_vertical_position
else
    LcuRow = slice_vertical_position
```

条带水平位置 slice_horizontal_position

8位无符号整数，给出条带的第一个最大编码单元在图像中的水平位置LcuColumn，以最大编码单元为单位。slice_horizontal_position的值的范围应为0x00～0xFF。

如果图像的horizontal_size 大于$255\times2^{LcuSizeInBit}$，LcuColumn由slice_horizontal_position和slice_horizontal_position_extension决定。

条带水平位置扩展 slice_horizontal_position_extension

2 位无符号整数，如果图像的horizontal_size小于或等于$255\times2^{LcuSizeInBit}$，位流中不应出现slice_horizontal_position_extension。

LcuColumn按下面的方法计算：

```
if ( horizontal_size > 255×2^LcuSizeInBit )
    LcuColumn = ( slice_horizontal_position_extension << 8 ) + slice_horizontal_position
else
    LcuColumn = slice_horizontal_position
```

固定条带量化因子　fixed_slice_qp

二值变量。值为‘1’表示在该条带内量化因子不变；值为‘0’表示在该条带内量化因子可变。

条带量化因子　slice_qp

7位无符号整数。给出条带的量化因子，取值范围是0～(63+8×(BitDepth−8))。SliceQp的值等于slice_qp，如果slice_qp不存在，SliceQp的值等于picture_qp。

条带样值偏移补偿允许标志　slice_sao_enable_flag[compIdx]

二值变量。SliceSaoEnableFlag[compIdx]的值等于slice_sao_enable_flag[compIdx]的值。SliceSaoEnableFlag[compIdx]的值为1表示在该条带内第compIdx个分量可使用样值偏移补偿；值为0则表示在该条带内第compIdx个分量不应使用样值偏移补偿。其中compIdx等于0表示亮度分量，compIdx等于1表示Cb分量，compIdx等于2表示Cr分量。如果位流中不存在slice_sao_enable_flag[compIdx]，则SliceSaoEnableFlag[compIdx]的值为0。

高级熵编码字节对齐填充位　aec_byte_alignment_bit

填充位。值应为‘1’。

样值偏移补偿合并方式索引　sao_merge_type_index

样值偏移补偿合并方式的索引值。解析过程见8.3。SaoMergeTypeIndex的值等于sao_merge_type_index的值。如果当前样本的样值偏移补偿单元E的左边样值偏移补偿单元L存在，且与E对应的最大编码单元和L对应的最大编码单元处于同一条带，则SaoMergeLeftAvai的值为1；否则SaoMergeLeftAvai的值为0。如果当前样本的样值偏移补偿单元E的上边样值偏移补偿单元U存在，且与E

对应的最大编码单元和U对应的最大编码单元处于同一条带，则SaoMergeUpAvai的值为1；否则SaoMergeUpAvai的值为0。

样值偏移补偿模式　sao_mode[compIdx]

样值偏移补偿模式的索引值。解析过程见8.3。其中compIdx等于0表示亮度分量，compIdx等于1表示Cb分量，compIdx等于2表示Cr分量。由sao_mode的值查表49得到SaoMode和MaxOffsetNumber的值。

表49　样值偏移补偿模式和最大偏移数

sao_mode的值	SaoMode的值	MaxOffsetNumber的值	偏移补偿模式
0	SAO_Off	0	关闭模式
1	SAO_Interval	4	区间模式
2	SAO_Edge	4	边缘模式

样值偏移补偿区间模式偏移值绝对值　sao_interval_offset_abs[compIdx][j]

区间模式偏移值的绝对值。解析过程见8.3。SaoIntervalOffsetAbs[compIdx][j]的值等于sao_interval_offset_abs[compIdx][j]的值。sao_interval_offset_abs[compIdx][j]的取值范围应为0～7。

样值偏移补偿区间模式偏移值符号值　sao_interval_offset_sign[compIdx][j]

区间模式偏移值的符号位。解析过程见8.3。如果sao_interval_offset_sign[compIdx][j]的值为'0'，则SaoIntervalOffsetSign[compIdx][j]的值等于1；否则SaoIntervalOffsetSign[compIdx][j]的值等于-1。如果位流中不存在sao_interval_offset_sign[compIdx][j]，则SaoIntervalOffsetSign[compIdx][j]的值为0。

样值偏移补偿区间模式起始偏移子区间位置　sao_interval_start_pos[compIdx]

区间模式起始偏移子区间的位置。解析过程见8.3。SaoIntervalStartPos[compIdx]的值等于sao_interval_start_pos[compIdx]的值。sao_interval_start_pos[compIdx]的取值范围应为0～31。

样值偏移补偿区间模式起始偏移子区间位置差　sao_interval_delta_pos_minus2[compIdx]

区间模式起始偏移子区间的位置差。解析过程见8.3。SaoIntervalDeltaPosMinus2[compIdx]的值等于sao_interval_delta_pos_minus2[compIdx]的值。sao_interval_delta_pos_minus2[compIdx]的取值范围应为0～14。

样值偏移补偿边缘模式偏移值　sao_edge_offset[compIdx][j]

边缘模式的偏移值。解析过程见8.3。SaoEdgeOffset[compIdx][j]的值等于sao_edge_offset[compIdx][j]的值。sao_edge_offset[compIdx][0]的取值范围应为-1～6，sao_edge_offset[compIdx][1]的取值范围应为0～1，sao_edge_offset[compIdx][2]的取值范围应为-1～0，sao_edge_offset[compIdx][3]的取值范围应为-6～1。

样值偏移补偿边缘模式类型　sao_edge_type[compIdx]

边缘模式的类型。解析过程见8.3。SaoEdgeType[compIdx]的值等于sao_edge_type[compIdx]的值。

最大编码单元自适应修正滤波允许标志　alf_lcu_enable_flag[compIdx][LcuIndex]

二值变量。值为'1'表示第LcuIndex个最大编码单元的分量compIdx样本应使用自适应修正滤波；值为'0'表示第LcuIndex个最大编码单元的分量compIdx样本不应使用自适应修正滤波。其中compIdx等于0表示亮度分量，compIdx等于1表示Cb分量，compIdx等于2表示Cr分量。AlfLCUEnableFlag[compIdx][LcuIndex]的值等于alf_lcu_enable_flag[compIdx][LcuIndex]的值。

熵编码最大编码单元填充位　aec_lcu_stuffing_bit

填充位。条带的最后一个最大编码单元的单元aec_lcu_stuffing_bit的值应为‘1’，解析过程见8.3。

7.2.5 编码树

拆分标志 split_flag

二值变量。值为‘1’表示应使用划分过程进行拆分；值为‘0’表示不应拆分。SplitFlag的值等于split_flag的值。split_flag的解析过程见8.3。

7.2.6 编码单元

编码单元类型索引 cu_type_index

编码单元类型的索引值。解析过程见8.3。CuTypeIndex的值等于cu_type_index的值。如果位流中不存在cu_type_index，CuTypeIndex的值为0。

如果PictureType的值为4，则CuTypeIndex的值应小于或等于2。

如果满足以下条件之一，IntraCuFlag的值为1；否则IntraCuFlag的值为0：

a) PictureType 的值为 0；

b) PictureType 的值为 4 且 CuTypeIndex 的值为 2；

c) CuTypeIndex 的值为 6。

预测单元形状索引 shape_of_partition_index

预测单元形状的索引值。解析过程见8.3。ShapeOfPartitionIndex的值等于shape_of_partition_index的值。如果位流中不存在shape_of_partition_index，ShapeOfPartitionIndex的值为0。

预测单元类型索引 b_pu_type_index

B图像的编码单元被划分为一个或两个预测单元时，预测单元类型的索引值。解析过程见8.3。解码过程见9.5.3。

简化编码单元预测单元类型索引 b_pu_type_min_index

B图像的编码单元被划分为两个预测单元时，简化编码单元（即CuTypeIndex等于3或4，且SizeInBit等于3的编码单元）预测单元类型的索引值。解析过程见8.3。解码过程见9.5.3。

如果当前编码单元是简化编码单元，BPuTypeIndex的值等于b_pu_type_min_index的值；否则BPuTypeIndex的值等于b_pu_type_index的值。如果位流中不存在b_pu_type_index和b_pu_type_min_index，则BPuTypeIndex的值为0。

F图像预测单元类型索引 f_pu_type_index

F图像的编码单元被划分为一个或两个预测单元时，预测单元类型的索引值。解析过程见8.3。解码过程见9.5.3。FPuTypeIndex的值等于f_pu_type_index的值。如果位流中不存在f_pu_type_index，FPuTypeIndex的值为0。

编码块加权跳过模式 weighted_skip_mode

编码块加权跳过模式的类型。解析过程见8.3。解码过程见9.5.3。WeightedSkipMode的值等于weighted_skip_mode的值。如果位流中不存在weighted_skip_mode，WeightedSkipMode的值为0。

编码单元子类型索引 cu_subtype_index

编码单元子类型的索引值。解析过程见8.3。解码过程见9.5.3。CuSubtypeIdx的值等于cu_subtype_index的值。如果位流中不存在cu_subtype_index，CuSubtypeIdx的值等于0。

预测单元类型第二索引 b_pu_type_index2[i]

B图像预测单元类型的索引值。解析过程见8.3。解码过程见9.5.3。BPuTypeIndex2[i]的值等于b_pu_type_index2[i]的值。

F图像预测单元类型第二索引 f_pu_type_index2[i]

F图像预测单元类型的索引值。解析过程见8.3。解码过程见9.5.3。FPuTypeIndex2[i]的值等于f_pu_type_index2[i]的值。如果位流中不存在f_pu_type_index2[i]，FPuTypeIndex2[i]的值为0。如果当前编码单元的所有预测单元的FPuTypeIndex2[i]的值全为0，则FFourPuTypeIndex2的值为0，否则FFourPuTypeIndex2的值为1。

帧内预测单元类型索引 intra_pu_type_index

帧内预测单元类型的索引值。解析过程见8.3。解码过程见9.5.3。IntraPuTypeIndex的值等于intra_pu_type_index的值。如果位流中不存在intra_pu_type_index，则IntraPuTypeIndex的值等于2。

帧内亮度预测模式 intra_luma_pred_mode

用于确定亮度块的帧内预测模式。解析过程见8.3。解码过程见9.5.6。intra_luma_pred_mode的取值范围为0～32。

帧内色度预测模式 intra_chroma_pred_mode

用于确定4:2:0格式下编码块中PredBlockOrder为NumOfIntraPredBlock、NumOfIntraPredBlock+1的两个色度块的帧内预测模式。解析过程见8.3。解码过程见9.5.6。

预测单元参考索引 pu_reference_index

用于确定预测单元的参考索引。解析过程见8.3。解码过程见9.5.7。PuRefIdx的值等于pu_reference_index的值。如果位流中不存在pu_reference_index，则PuRefIdx的值为0。

方向性多假设预测模式 dir_multi_hypothesis_mode

用于确定亮度块方向性多假设预测的模式。解析过程见8.3。解码过程见9.5.3。DirMultiHypothesisMode的值等于dir_multi_hypothesis_mode的值。如果位流中不存在dir_multi_hypothesis_mode，DirMultiHypothesisMode的值为0。

运动矢量水平分量差绝对值 mv_diff_x_abs

运动矢量垂直分量差绝对值 mv_diff_y_abs

运动矢量差值的绝对值。MvDiffXAbs等于mv_diff_x_abs的值，MvDiffYAbs等于mv_diff_y_abs的值。如果当前图像是B图像，先解码全部前向运动矢量，然后解码全部后向运动矢量。解析过程见8.3。

运动矢量水平分量差符号值 mv_diff_x_sign

运动矢量垂直分量差符号值 mv_diff_y_sign

运动矢量差值的符号位。MvDiffXSign的值等于mv_diff_x_sign的值，MvDiffYSign的值等于mv_diff_y_sign。如果位流中不存在mv_diff_x_sign或mv_diff_y_sign，则MvDiffXSign或MvDiffYSign的值为0。如果MvDiffXSign的值为0，MvDiffX等于MvDiffXAbs；如果MvDiffXSign的值为1，MvDiffX等于-MvDiffXAbs。如果MvDiffYSign的值为0，MvDiffY等于MvDiffYAbs；如果MvDiffYSign的值为1，MvDiffY等于-MvDiffYAbs。MvDiffX和MvDiffY的取值范围为-32768～32767（按亮度像素样本为-8192～8191.75）。

变换块系数标志 ctp_zero_flag

二值变量。解析过程见8.3。CtpZeroFlag的值等于ctp_zero_flag的值。如果位流中不存在ctp_zero_flag，则CtpZeroFlag的值为0。

变换块拆分标志 transform_split_flag

二值变量。解析过程见8.3。TransformSplitFlag的值等于transform_split_flag的值。如果位流中不存在transform_split_flag，TransformSplitFlag的值为0。

色度变换块编码模板 ctp_uv

ctp_uv确定色度变换块中是否包含编码数据。解析过程见8.3。

亮度变换块编码模板 ctp_y[i]

ctp_y[i]确定亮度变换块i（亮度变换块的编号i见9.5.5）是否包含编码数据。如果transform_split_flag的值为‘0’且ctp_uv的值为0，则ctp_y[0]的值应为1。解析过程见8.3。

编码单元量化参数增量　cu_qp_delta

给出当前单元的单元量化参数相对预测量化参数的增量。解析过程见8.3，解码过程见9.5.2。cuQpDelta的值等于cu_qp_delta的值。cuQpDelta的取值范围应是(-32-4×(BitDepth-8))～(32+4×(BitDepth-8))。

7.2.7　变换块

变量M_1、M_2、Log2TransformWidth和Log2TransformHeight的定义见9.5.5。扫描表ScanCGInBlk，ScanCoeffInCG和InvScanCoeffInBlk的定义见附录E。变量IntraModeIdx的定义见8.3.3.2.14。

尾系数块位置　last_cg_pos

用于确定最后一个非零系数块在变换块中的位置，解析过程见8.3。

尾系数块位置零标志　last_cg0_flag

二值变量。用于确定最后一个非零系数块在变换块中的位置，解析过程见8.3。

尾系数块横坐标　last_cg_x

用于确定最后一个非零系数块在变换块中的水平位置，解析过程见8.3。

横坐标为零尾系数块纵坐标　last_cg_y_minus1

用于确定最后一个非零系数块在变换块中的垂直位置，解析过程见8.3。如果位流中不存在last_cg_y_minus1，则last_cg_y_minus1的值为0。

横坐标非零尾系数块纵坐标　last_cg_y

用于确定最后一个非零系数块在变换块中的垂直位置，解析过程见8.3。

非零系数块标志　nonzero_cg_flag

二值变量。值为‘1’表示当前系数块包含非零量化系数；值为‘0’表示当前系数块是全零块。解析过程见8.3。NonZeroCGFlag的值等于nonzero_cg_flag的值。如果位流中不存在nonzero_cg_flag，NonZeroCGFlag的值为1。

尾系数横坐标　last_coeff_pos_x

用于确定在非零系数块中最后一个非零量化系数的水平位置，解析过程见8.3。

尾系数纵坐标　last_ coeff_pos_y

用于确定在非零系数块中最后一个非零量化系数的垂直位置，解析过程见8.3。

系数幅值段　coeff_level_minus1_band

用于确定非零量化系数幅度所在的段，CoeffLevelMinus1Band的值等于coeff_level_minus1_band的值，解析过程见8.3。

系数幅值段位置　coeff_level_minus1_pos_in_band

用于确定非零量化系数幅度在所在的段中的位置，CoeffLevelMinus1PosInBand的值等于coeff_level_minus1_pos_in_band的值，解析过程见8.3。

系数游程　coeff_run

用于确定游程的长度，解析过程见8.3。

系数符号　coeff_sign

用于确定非零量化系数的正负性，解析过程见8.3。

7.2.8　自适应修正滤波参数

图像亮度分量自适应修正滤波器个数　alf_filter_num_minus1

alf_filter_num_minus1 的 值 加 1 表 示 当 前 图 像 亮 度 分 量 自 适 应 修 正 滤 波 器 的 个 数 。alf_filter_num_minus1的取值范围应为0～15。

图像亮度分量相邻自适应修正滤波区域个数　alf_region_distance[i]

alf_region_distance[i]表示亮度分量第i个自适应修正滤波区域基本单元起始标号与第i-1个自适应修正滤波区域基本单元起始标号间的差值。alf_region_distance[i]的取值范围应为1～15。如果位流中不存在alf_region_distance[i]，如果i等于0，则alf_region_distance[i]的值为0，如果i不等于0且alf_filter_num_minus1的值为15，则alf_region_distance[i]的值为1。符合标准的位流应满足alf_region_distance[i]（i=0～alf_filter_num_minus1）之和应小于等于15。

图像亮度分量自适应修正滤波器系数　alf_coeff_luma[i][j]

alf_coeff_luma[i][j]表示亮度分量第i个自适应修正滤波器的第j个系数。AlfCoeffLuma[i][j]的值等于alf_coeff_luma[i][j]的值。从符合本部分的位流中解码得到的AlfCoeffLuma[i][j](j=0～7)的取值范围应为-64～63，AlfCoeffLuma[i][8]的取值范围应为-1088～1071。

图像色度分量自适应修正滤波器系数　alf_coeff_chroma[0][j]，alf_coeff_chroma[1][j]

alf_coeff_chroma[0][j]表示Cb分量第j个自适应修正滤波器的系数，alf_coeff_chroma[1][j] 表示Cr分量第j个自适应修正滤波器的系数。AlfCoeffChroma[0][j]的值等于alf_coeff_chroma[0][j]的值，AlfCoeffChroma[1][j]的值等于alf_coeff_chroma[1][j]的值。AlfCoeffChroma[i][j](i=0～1，j=0～7)的取值范围应为-64～63，AlfCoeffChroma[i][8]（i=0～1）的取值范围应为-1088～1071。

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)
```

$$\text{CodeNum} = 2^{\text{leadingZeroBits} + k} - 2^k + \text{read\_bits}(\text{leadingZeroBits} + k)$$

表50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50　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
	……	……

表 50（续）

阶数	码字结构	CodeNum取值范围
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)和 se(v))的解析过程

ue(v)和se(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：

h) ue(v)：语法元素的值等于 CodeNum；

i) se(v)：根据表 51 给出的有符号指数哥伦布码的映射关系求语法元素的值。

表51 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	$(-1)^{k+1}\times\mathrm{Ceil}(k\div 2)$

8.3 ae(v)的解析过程

8.3.1 概述

对条带进行解析前，首先初始化所有二元符号模型和熵编码解码器，见8.3.2。然后从位流中依次解析得到二元符号串，见8.3.3。最后根据二元符号串得到ae(v)描述的语法元素的值，见8.3.4。

8.3.2 初始化

8.3.2.1 初始化二元符号模型

解码器应保存全部二元符号模型，每个二元符号模型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。

8.3.2.2 初始化熵编码解码器

rS1、rT1、bFlag、cFlag、valueS、boundS、valueT和valueD是用于熵编码解码器的变量。boundS是一个大于0的整数。valueD的值为0或1，bFlag的值为0或1，cFlag的值为0或1。valueS和boundS的位宽是大于或等于Log(boundS+1)的最小整数，rS1的位宽是大于或等于Log(boundS+2)的最小整数，rT1的位宽是8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。如果boundS的值为254，则valueS和rS1的位宽是8位。

注：valueS记录了预先连续读进多少位才会使valueT的最高位为1。这一功能可用于快速读取位流及并行解码。在极端的状况下，valueS的值有可能超过16位可表示的范围。boundS限制了连续读进‘0’的个数，从而对valueS的位宽进行合理的控制。

valueS、valueT和valueD的初始化过程用伪代码描述如下：

```
valueS = 0
valueT = read_bits(9)
valueD = 1
```

8.3.3 二元符号串解析

8.3.3.1 概述

解析二元符号串的步骤如下：

a) 设二元符号的索引号 binIdx 的值为-1，二元符号串为空。
b) binIdx 的值加 1，然后进行以下操作：
   1) 如果当前二元符号是以下二元符号之一，置 BypassFlag 的值为 1：
      ——sao_mode 中 binIdx 不为 0 的二元符号；
      ——sao_interval_offset_abs 中 binIdx 不为 0 的二元符号；
      ——sao_interval_offset_sign 的二元符号；
      ——sao_edge_offset 的二元符号；
      ——sao_interval_start_pos 的二元符号；
      ——sao_interval_delta_pos_minus2 的二元符号；
      ——sao_edge_type 的二元符号；
      ——mv_diff_x_sign 或 mv_diff_y_sign 的二元符号；
      ——mv_diff_x_abs 或 mv_diff_y_abs 中 binIdx 大于或等于 3 的二元符号；
      ——coeff_sign 的二元符号；
      ——dir_multi_hypothesis_mode 中已解析的二元符号串的二元符号为‘10’且 binIdx 等于 2 的二元符号；
      ——dir_multi_hypothesis_mode 中 binIdx 大于 2 的二元符号；
      ——当 CoeffLevelMinus1Band 的值为 1 时，coeff_level_minus1_pos_in_band 的二元符号。

2) 否则，如果当前二元符号是以下二元符号之一，置 BypassFlag 的值为 0 且 StuffingBitFlag 的值为 1：

——aec_lcu_stuffing_bit 的二元符号；

——当 SizeInBit 不等于 Log(MiniSize)时，cu_type_index 中 binIdx 等于 5 的二元符号；

——coeff_level_minus1_band 的二元符号。

3) 否则，置 BypassFlag 和 StuffingBitFlag 的值为 0，根据 binIdx 得到每个二元符号对应的唯一的 ctxIdx，并根据 ctxIdx 导出二元符号模型 ctx（见 8.3.3.2）。

c) 解析当前二元符号（见 8.3.3.3）；

d) 将由步骤 c 得到的二元符号加入二元符号串的尾部，得到更新的二元符号串；

e) 将由步骤 d 得到的二元符号串与 8.3.4 中对应的表格进行比较。如果该二元符号串与表格中某个二元符号串相匹配，则完成二元符号串的解析；否则回到步骤 b，继续解析下一个二元符号。

8.3.3.2 导出二元符号模型

8.3.3.2.1 导出二元符号模型

二元符号模型ctx等于ctxArray[ctxIdx]，其中ctxArray是保存二元符号模型的数组，ctxIdx是数组的索引值。语法元素的每个二元符号的ctxIdx等于ctxIdxInc加ctxIdxStart。各语法元素对应的ctxIdxStart及每个二元符号对应的ctxIdxInc见表52。

表52 语法元素对应的 ctxIdxStart 和 ctxIdxInc

语法元素	ctxIdxInc	ctxIdxStart	ctx的数量
sao_merge_type_index	binIdx+SaoMergeLeftAvai+SaoMergeUpAvai-1	0	3
sao_mode	0	3	1
sao_interval_offset_abs	0	4	1
alf_lcu_enable_flag	0	5	1
split_flag	SizeInBit-4	6	3
cu_type_index	binIdx	9	5
shape_of_partition_index	binIdx	14	2
b_pu_type_index	见8.3.3.2.2	16	15
b_pu_type_min_index	binIdx	31	2
f_pu_type_index	binIdx+(cu_type_index!=2)	33	3
weighted_skip_mode	Min(binIdx,2)	36	3
cu_subtype_index	binIdx	39	4
b_pu_type_index2	见8.3.3.2.3	43	4
f_pu_type_index2	0	47	1
transform_split_flag	见8.3.3.2.4	48	3
intra_pu_type_index	0	51	1
intra_luma_pred_mode	见8.3.3.2.5	52	7
intra_chroma_pred_mode	见8.3.3.2.6	59	3

表 52（续）

语法元素	ctxIdxInc	ctxIdxStart	ctx的数量
pu_reference_index	见8.3.3.2.7	62	3
dir_multi_hypothesis_mode	见8.3.3.2.8	65	12
mv_diff_x_abs	见8.3.3.2.9	77	3
mv_diff_y_abs	见8.3.3.2.9	80	3
ctp_zero_flag	0	83	1
ctp_uv	Min(binIdx,1)+(IntraCuFlag<<1)	84	4
ctp_y[i]	见8.3.3.2.10	88	4
cu_qp_delta	见8.3.3.2.11	92	4
last_cg_pos	见8.3.3.2.12	96	6
last_cg0_flag	IsChroma	102	2
last_cg_x	IsChroma	104	2
last_cg_y_minus1 或 last_cg_y	IsChroma	106	2
nonzero_cg_flag	见8.3.3.2.13	108	3
last_coeff_pos_x	见8.3.3.2.14	111	30
last_coeff_pos_y	见8.3.3.2.14	141	30
CoeffLevelMinus1Band 为 0 时 coeff_level_minus1_pos_in_band	见8.3.3.2.15	171	40
coeff_run	见8.3.3.2.16	211	57

8.3.3.2.2 确定 b_pu_type_index 的 ctxIdxInc

根据以下方法确定b_pu_type_index的ctxIdxInc：

a) 如果 cu_type_index 的值等于 2，则 ctxIdxInc 等于 binIdx；

b) 如果 cu_type_index 的值不等于 2，则根据该语法元素当前已解析的二元符号串和 binIdx 查表 53 得到 ctxIdxInc。

表53 前缀二元符号串与 ctxIdxInc 的关系

binIdx	当前已解析的二元符号串	ctxIdxInc 的值
0	空	3
1	0	4
2	00	5
1	1	6
2	01	6
3	000	6
3	001	6
2	10	7
3	100	8

表 53（续）

binIdx	当前已解析的二元符号串	ctxIdxInc 的值
3	010	9
4	0100	10
4	0010	11
4	0000	13
5	00100	12
5	00000	14

8.3.3.2.3　确定 b_pu_type_index2 的 ctxIdxInc

根据以下方法确定b_pu_type_index2的ctxIdxInc：

a)　如果 binIdx 等于 0，则 ctxIdxInc 等于 0；

b)　否则，如果 binIdx 等于 1 且二元符号串的第一个二元符号为‘0’，则 ctxIdxInc 等于 1；

c)　否则，如果 binIdx 等于 1，则 ctxIdxInc 等于 2；

d)　否则，ctxIdxInc 等于 3。

8.3.3.2.4　确定 transform_split_flag 的 ctxIdxInc

根据以下方法确定transform_split_flag的ctxIdxInc：

a)　如果 IntraCuFlag 等于 0，则 ctxIdxInc 等于 0；

b)　否则，如果 SizeInBit 的值不等于 4 或 5，则 ctxIdxInc 等于 1；

c)　否则，ctxIdxInc 等于 2。

8.3.3.2.5　确定 intra_luma_pred_mode 的 ctxIdxInc

根据以下方法确定intra_luma_pred_mode的ctxIdxInc：

根据该语法元素 binIdx 和当前已解析的二元符号串和 binIdx，查表 54 得到 ctxIdxInc。表 54 中，xi(i=1～4)的值为‘0’或‘1’。

表54　前缀二元符号串与 ctxIdxInc 的关系

binIdx	当前已解析的二元符号串	ctxIdxInc的值
0	空	0
1	0	1
2	0 x_1	2
3	0 x_1 x_2	3
4	0 x_1 x_2 x_3	4
5	0 x_1 x_2 x_3 x_4	5
1	1	6

8.3.3.2.6　确定 intra_chroma_pred_mode 的 ctxIdxInc

根据以下方法确定intra_chroma_pred_mode的ctxIdxInc：

a)　如果 binIdx 等于 0，则：

ctxIdxInc = a

其中 a 由以下方法确定（当前预测块 E 与其左边预测块 A 的关系见 9.5.4）：

1) 如果当前预测块 E 的左边预测块 A“存在”，并且预测块 A 的预测模式不是 Intra_Chroma_DM，则 a 等于 1；
2) 否则，a 等于 0。

b) 否则，ctxIdxInc 等于 2。

8.3.3.2.7 确定 pu_reference_index 的 ctxIdxInc

根据以下方法确定pu_reference_index的ctxIdxInc：

a) 如果 binIdx 等于 0，则 ctxIdxInc 等于 0；
b) 否则，如果 binIdx 等于 1，则 ctxIdxInc 等于 1；
c) 否则，ctxIdxInc 等于 2。

8.3.3.2.8 确定 dir_multi_hypothesis_mode 的 ctxIdxInc

根据以下方法确定dir_multi_hypothesis_mode的ctxIdxInc：

根据该语法元素当前已解析的二元符号串和 binIdx 查表 55 得到 ctxIdxInc，并计算：

ctxIdxInc = ctxIdxInc + (SizeInBit – Log(MiniSize))×3

表55 前缀二元符号串与 ctxIdxInc 的关系

binIdx	当前已解析的二元符号串	ctxIdxInc 的值
0	空	0
1	1	1
2	11	2

8.3.3.2.9 确定 mv_diff_x_abs 或 mv_diff_y_abs 的 ctxIdxInc

根据以下方法确定mv_diff_x_abs或mv_diff_y_abs的ctxIdxInc：

a) 如果 binIdx 等于 0，则 ctxIdxInc 等于 0；
b) 否则，如果 binIdx 等于 1，则 ctxIdxInc 等于 1；
c) 否则，如果 binIdx 等于 2，则 ctxIdxInc 等于 2；
d) 否则，BypassFlag 的值为 1。

8.3.3.2.10 确定 ctp_y[i]的 ctxIdxInc

根据以下方法确定ctp_y[i]的ctxIdxInc：

ctxIdxInc = a + 2×b

其中 a 和 b 由以下方法确定（当前变换块 E 与其左边变换块 A、上边变换块 B 的关系见 9.5.4）：

a) 如果当前变换块 E 的左边变换块 A（或上边变换块 B）“存在”，并且变换块 A（或变换块 B）中包含非零系数，则 a（或 b）等于 1；
b) 否则，a（或 b）等于 0。

8.3.3.2.11 确定 cu_qp_delta 的 ctxIdxInc

根据以下方法确定cu_qp_delta的ctxIdxInc：

a) 如果 binIdx 和 PreviousDeltaQP 均等于 0，则 ctxIdxInc 等于 0；
b) 否则，如果 binIdx 等于 0 且 PreviousDeltaQP 不等于 0，则 ctxIdxInc 等于 1；
c) 否则，如果 binIdx 等于 1，则 ctxIdxInc 等于 2；
d) 否则，ctxIdxInc 等于 3。

8.3.3.2.12 确定 last_cg_pos 的 ctxIdxInc

根据以下方法确定last_cg_pos的ctxIdxInc：

a) 如果 IsChroma 等于 0，则：

ctxIdxInc=Min(binIdx,2)

b) 否则：

ctxIdxInc=Min(binIdx,2)+3

8.3.3.2.13 确定 nonzero_cg_flag 的 ctxIdxInc

根据以下方法确定nonzero_cg_flag的ctxIdxInc：

a) 如果 IsChroma 等于 0，则：

ctxIdxInc=Min(CGPos,1)

b) 否则，ctxIdxInc 等于 2。

8.3.3.2.14 确定 last_coeff_pos_x 或 last_coeff_pos_y 的 ctxIdxInc

根据以下方法确定last_coeff_pos_x或last_coeff_pos_y的ctxIdxInc：

a) 如果 Log2TransformSize 等于 2，则：
   3) 如果 IsChroma 等于 0，则：
   ctxIdxInc=Min(binIdx,1)+2×(IntraModeIdx==2)+20
   4) 否则：
   ctxIdxInc=Min(binIdx,1)+28
b) 否则：
   5) 计算 isLastCG：

```
if (CGPos == LastCGPos)
    isLastCG = 1
else
    isLastCG = 0
```

   6) 如果 IsChroma 等于 0，则：

```
if ((CGx + CGy) == 0)
    regionIdx = 0
else if ((CGx × CGy) == 0)
    regionIdx = 1
else
    regionIdx = 2
if (regionIdx > 1)
    ctxIdxInc = 10×isLastCG + Min(binIdx,1)
else
    ctxIdxInc = 10×isLastCG + (4×(regionIdx>0) + 2×(IntraModeIdx==2) + 2) +
Min(binIdx,1)
```

7) 否则：

ctxIdxInc = Min(binIdx, 1)+ 2×isLastCG + 24

其中，如果IntraCuFlag等于0或IsChroma等于1，则IntraModeIdx等于2；否则IntraModeIdx由IntraLumaPredMode查表56得到。

表56 IntraModeIdx 与 IntraLumaPredMode 的关系

IntraLumaPredMode的值	IntraModeIdx的值
0	2
1	2
2	2
3	1
4	1
5	2
6	2
7	2
8	0
9	0
10	0
11	0
12	0
13	0
14	0
15	0
16	0
17	2
18	2
19	2
20	1
21	1
22	1
23	1
24	1
25	1
26	1
27	1
28	1
29	2
30	2
31	2
32	0

8.3.3.2.15 确定 CoeffLevelMinus1Band 为 0 时 coeff_level_minus1_pos_in_band 的 ctxIdxInc

根据以下方法确定coeff_level_minus1_pos_in_band的ctxIdxInc：

```
if ((CGPos == 0) && (CoeffPosInCG < 3))
    regionIdx = 0
else
    regionIdx = 1
pairsInCGIdx = Min((pairsInCG+1)/2,2)
ctxIdxInc = 10×regionIdx + Min(priIdx,pairsInCGIdx+2)+ (5×pairsInCGIdx)/2 +
(IsChroma ? 20 : 0)
```

其中pairsInCG是当前系数块内已解码非零系数幅值的个数。对每个系数块进行解码前，pairsInCG应初始化为0。priIdx由lMax查表57得到。表57中lMax是当前M×N变换块已解码系数的最大幅值。对每个M×N变换块进行解码前，lMax应初始化为0。

表57 priIdx 与 lMax 的关系

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

8.3.3.2.16 确定 coeff_run 的 ctxIdxInc

根据以下方法确定coeff_run的ctxIdxInc：

a) 首先，根据 InvScanCoeffInCG 、 IntraModeIdx 、 CoeffPosInCG 和 binIdx 得到 regionRunLumaIdx：

```
if (IsChroma) {
    if ((CoeffPosInCG – binIdx) < 6)
        regionRunLumaIdx = 0
    else
        regionRunLumaIdx = 1
}
else if (IntraModeIdx == 2) {
    if ((CoeffPosInCG – binIdx) < 4)
        regionRunLumaIdx = 0
    else if ((CoeffPosInCG – binIdx) < 11)
        regionRunLumaIdx = 1
    else
        regionRunLumaIdx = 2
}
else {
    Cy = InvScanCoeffInCG[CoeffPosInCG-1-binIdx][1]
```

```
        regionRunLumaIdx = (Cy + 1) / 2
    }
```

b) 然后，根据 IsChroma、absSum、CoeffPosInCG、CGPos、regionRunLumaIdx 和 binIdx 得到 ctxIdxInc：

```
    if (CGPos == 0){
        if (CoeffPosInCG-binIdx == 1)
            ctxIdxInc = Min(2,(absSum+AbsLevel)/2)
        else
            ctxIdxInc = 3×(1+regionRunLumaIdx) + Min(2,(absSum+AbsLevel)/2)
    }
    else {
        ctxIdxInc = 3×(IsChroma ? (3+regionRunLumaIdx) : (4+regionRunLumaIdx))+
    Min(2,(absSum+AbsLevel)/2)
    }
    ctxIdxInc= ctxIdxInc + 3×(Log2TransformSize==2 ? 0 : (IsChroma ? 3 : 4))+
    (IsChroma ? 33 : 0)
```

8.3.3.3 二元符号解析

8.3.3.3.1 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 8.3.3.3.4）；
   2) 否则，如果 StuffingBitFlag 的值为 1，则执行 decode_aec_stuffing_bit 过程（见 8.3.3.3.3）；
   3) 否则，执行 decode_decision 过程（见 8.3.3.3.2）。
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’。

8.3.3.3.2 decode_decision

decode_decision过程的输入是bFlag、cFlag、rS1、rT1、valueS、valueT、valueD以及上下文模型ctx。decode_decision过程的输出是二元符号值binVal。令cFlag等于1，decode_decision过程用伪代码描述如下：

```
decode_decision( )
{
    predMps = ctx->mps
    lgPmps =  ctx->lgPmps >> 2

    if (valueD || (bFlag == 1 && rS1 == boundS ) ) {
        rS1 = 0
        valueS = 0
        while ( valueT < 0x100 && valueS < boundS ) {
            valueS++
            valueT = (valueT << 1 ) | read_bits(1)
```

```
    }
    if ( valueT < 0x100 )
        bFlag = 1
    else
        bFlag = 0
    valueT = valueT & 0xFF
}

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

if( rS2 > valueS || (rS2 == valueS && valueT >= rT2) && bFlag == 0 ) {
    binVal = ! predMps
    if ( sFlag == 0 )
        tRlps = lgPmps
    else
        tRlps = rT1 + lgPmps
    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)
    }
    rT1 = tRlps & 0xFF
    valueD = 1
}
else {
    binVal = predMps
    rS1 = rS2
    rT1 = rT2
    valueD = 0
}
```

```
        if ( cFlag )
            ctx = update_ctx(binVal, ctx)
        return (binVal)
    }
```

8.3.3.3.3 decode_aec_stuffing_bit

decode_aec_stuffing_bit过程的输入是bFlag、cFlag、rS1、rS2、valueS、valueT和valueD。decode_aec_stuffing_bit过程的输出是二元符号值binVal。令cFlag等于0，ctx->lgPmps等于4，ctx->mps等于0，带入decode_decision过程实现decode_aec_stuffing_bit过程。

注：decode_aec_stuffing_bit过程也可参考附录F的描述方式实现。

8.3.3.3.4 decode_bypass

decode_bypass过程的输入是bFlag、cFlag、rS1、rS2、valueS、valueT和valueD。decode_bypass过程的输出是二元符号值binVal。令cFlag等于0，ctx->lgPmps等于1024，ctx->mps等于0，带入decode_decision过程实现decode_bypass过程。

注：decode_bypass过程也可参考附录F的描述方式实现。

8.3.3.3.5 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)
}
```

8.3.4 反二值化方法

8.3.4.1 概述

本条定义语法元素的反二值化方法，见表58。

表58 语法元素的反二值化方法

语法元素	反二值化方法
sao_merge_type_index	见 8.3.4.5
sao_mode	见 8.3.4.2，maxVal=2，sao_mode 的值等于 synElVal
sao_interval_offset_abs	见 8.3.4.2，maxVal=7，sao_interval_offset_abs 的值等于 synElVal
sao_interval_offset_sign	见 8.3.4.4，sao_interval_offset_sign 的值等于 synElVal
sao_interval_start_pos	见 8.3.4.6
sao_interval_delta_pos_minus2	见 8.3.4.7
sao_edge_offset[compIdx][j]（j = 0～3）	见 8.3.4.8
sao_edge_type	见 8.3.4.9
alf_lcu_enable_flag	见 8.3.4.4，alf_lcu_enable_flag 的值等于 synElVal
aec_lcu_stuffing_bit	见 8.3.4.4，aec_lcu_stuffing_bit 的值等于 synElVal
split_flag	见 8.3.4.4，split_flag 的值等于 synElVal
cu_type_index	见 8.3.4.10
shape_of_partition_index	见 8.3.4.2，maxVal=2，shape_of_partition_index 的值等于 synElVal
b_pu_type_index	见 8.3.4.11
b_pu_type_min_index	见 8.3.4.12
f_pu_type_index	见 8.3.4.13
weighted_skip_mode	见 8.3.4.2，maxVal= RefPicNum-1，weighted_skip_mode 的值等于 synElVal
cu_subtype_index	见 8.3.4.2，maxVal=4，cu_subtype_index 的值等于 synElVal
b_pu_type_index2	见 8.3.4.14

表 58（续）

语法元素	反二值化方法
f_pu_type_index2	见 8.3.4.4，f_pu_type_index2 的值等于 synElVal
transform_split_flag	见 8.3.4.4，transform_split_flag 的值等于 synElVal
intra_pu_type_index	见 8.3.4.15
intra_luma_pred_mode	见 8.3.4.16
intra_chroma_pred_mode	见 8.3.4.2，maxVal=4，intra_chroma_pred_mode 的值等于 synElVal。如果 isRedundant 的值为 1，intra_chroma_pred_mode 的值不应为 4。
pu_reference_index	见 8.3.4.2，maxVal= RefPicNum-1，pu_reference_index 的值等于 synElVal
dir_multi_hypothesis_mode	见 8.3.4.17
mv_diff_x_abs	见 8.3.4.18
mv_diff_x_sign	见 8.3.4.4，mv_diff_x_sign 的值等于 synElVal
mv_diff_y_abs	见 8.3.4.18
mv_diff_y_sign	见 8.3.4.4，mv_diff_y_sign 的值等于 synElVal
ctp_zero_flag	见 8.3.4.4，ctp_zero_flag 的值等于 synElVal
ctp_uv	见 8.3.4.19
ctp_y[i]	见 8.3.4.4
cu_qp_delta	见 8.3.4.20
last_cg_pos	见 8.3.4.2，maxVal=3，last_cg_pos 的值等于 synElVal
last_cg0_flag	见 8.3.4.4，last_cg0_flag 的值等于 snyElVal
last_cg_x	见 8.3.4.2，maxVal 的值由 8.3.4.22 确定，last_cg_x 的值等于 synElVal
last_cg_y_minus1	见 8.3.4.2，maxVal 的值由 8.3.4.22 确定，last_cg_y_minus1 的值等于 synElVal
last_cg_y	见 8.3.4.2，maxVal 的值由 8.3.4.22 确定，last_cg_y 的值等于 synElVal
nonzero_cg_flag	见 8.3.4.4，nonzero_cg_flag 的值等于 synElVal
last_coeff_pos_x	见 8.3.4.2，maxVal=3，LastCoeffPosXtemp 的值等于 synElVal
last_coeff_pos_y	见 8.3.4.2，maxVal=3，LastCoeffPosYtemp 的值等于 synElVal
coeff_level_minus1_band	见 8.3.4.4，coeff_level_minus1_band 的值等于 synElVal
当 CoeffLevelMinus1Band 为 0 时 coeff_level_minus1_pos_in_band	见 8.3.4.2，maxVal=31，coeff_level_minus1_pos_in_band 的值等于 synElVal
当 CoeffLevelMinus1Band 为 1 时 coeff_level_minus1_pos_in_band	见 8.3.4.21，coeff_level_minus1_pos_in_band 的值等于 synElVal
coeff_sign	见 8.3.4.4，coeff_sign 的值等于 synElVal
coeff_run	见 8.3.4.2，maxVal=CoeffPosInCG，coeff_run 的值等于 synElVal

8.3.4.2 采用截断一元码的反二值化方法

由二元符号串根据表59得到synElVal的值。

表59　synElVal 与二元符号串的关系（截断一元码）

synElVal	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	
…	0	0	0	0	0	0	…	
maxVal-1	0	0	0	0	0	0	…	1
maxVal	0	0	0	0	0	0	…	0
binIdx	0	1	2	3	4	5	…	maxVal-1

8.3.4.3　采用一元码的反二值化方法

由二元符号串根据表60得到synElVal的值。

表60　synElVal 的值与二元符号串的关系（一元码）

synElVal	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	0	1	2	3	4	5

8.3.4.4　采用标记位的反二值化方法

由二元符号串根据表61得到synElVal的值。

表61　synElVal 的值与二元符号串的关系（标记位）

synElVal	二元符号串
0	0
1	1
binIdx	0

8.3.4.5　sao_merge_type_index 的反二值化方法

由SaoMergeLeftAvai的值、SaoMergeUpAvai的值和二元符号串查表62得到sao_merge_type_index的值。

表62　sao_merge_type_index 的值与二元符号串的关系

SaoMergeLeftAvai的值	SaoMergeUpAvai的值	sao_merge_type_index的值	二元符号串	
1	0	0	0	
		1	1	
0	1	0	0	
		1	1	
1	1	0	0	0
		1	1	
		2	0	1
binIdx			0	1

8.3.4.6　sao_interval_start_pos 的反二值化方法

由二元符号串查表63得到sao_interval_start_pos的值。

表63　sao_interval_start_pos 的值与二元符号串的关系

sao_interval_start_pos的值	二元符号串				
0	0	0	0	0	0
1	1	0	0	0	0
2	0	1	0	0	0
3	1	1	0	0	0
4	0	0	1	0	0
…	…	…	…	…	…
28	0	0	1	1	1
29	1	0	1	1	1
30	0	1	1	1	1
31	1	1	1	1	1
binIdx	0	1	2	3	4

8.3.4.7　sao_interval_delta_pos_minus2 的反二值化方法

由二元符号串查表64得到sao_interval_delta_pos_minus2的值。

表64　sao_interval_delta_pos_minus2 的值与二元符号串的关系

sao_interval_delta_pos_minus2的值	二元符号串							
0	1	0						
1	1	1						

表 64（续）

sao_interval_delta_pos_minus2的值	二元符号串							
2	0	1	0	0				
3	0	1	0	1				
4	0	1	1	0				
5	0	1	1	1				
6	0	0	1	0	0	0		
7	0	0	1	0	0	1		
8	0	0	1	0	1	0		
9	0	0	1	0	1	1		
10	0	0	1	1	0	0		
11	0	0	1	1	0	1		
12	0	0	1	1	1	0		
13	0	0	1	1	1	1		
14	0	0	0					
binIdx	0	1	2	3	4	5	6	7

8.3.4.8 sao_edge_offset[compIdx][j]的反二值化方法

由二元符号串查表65得到sao_edge_offset[compIdx][0]或sao_edge_offset[compIdx][3]的值；由二元符号串查表66得到sao_edge_offset[compIdx][1]或sao_edge_offset[compIdx][2]的值。

表65 sao_edge_offset[compIdx][0]和 sao_edge_offset[compIdx][3]的值与二元符号串的关系

sao_edge_offset[compIdx][0]	sao_edge_offset[compIdx][3]	二元符号串						
1	-1	1						
0	0	0	1					
2	-2	0	0	1				
-1	1	0	0	0	1			
3	-3	0	0	0	0	1		
4	-4	0	0	0	0	0	1	
5	-5	0	0	0	0	0	0	1
6	-6	0	0	0	0	0	0	0
binIdx		0	1	2	3	4	5	6

表66　sao_edge_offset[compIdx][1]和 sao_edge_offset[compIdx][2]的值与二元符号串的关系

sao_edge_offset[compIdx][1]	sao_edge_offset[compIdx][2]	二元符号串
0	0	1
1	-1	0
binIdx		0

8.3.4.9　sao_edge_type 的反二值化方法

由二元符号串查表 67 得到 sao_interval_start_pos 的值。

表67　sao_edge_type 的值与二元符号串的关系

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

8.3.4.10　cu_type_index 的反二值化方法

如果 PictureType 的值为 4，由二元符号串查表 68 得到 cu_type_index 的值。否则，如果 SizeInBit 的值大于 3，由二元符号串查表 69 得到 cu_type_index 的值；如果 SizeInBit 的值等于 3，由二元符号串查表 70 得到 cu_type_index 的值。

表68　cu_type_index 的值与二元符号串的关系（PictureType 的值为 4）

cu_type_index 的值	二元符号串	
0	1	
1	0	1
2	0	0
binIdx	0	1

表69　cu_type_index 的值与二元符号串的关系（PictureType 的值不为 4 且 SizeInBit 大于 3）

cu_type_index的值	二元符号串					
0	1					
1	0	1				
2	0	0	1			
3	0	0	0	1		

表 69（续）

cu_type_index的值	二元符号串					
4	0	0	0	0	1	
5	0	0	0	0	0	1
6	0	0	0	0	0	0
binIdx	0	1	2	3	4	5

表70　cu_type_index 的值与二元符号串的关系（PictureType 的值不为 4 且 SizeInBit 等于 3）

cu_type_index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	
6	0	0	0	0	0	
binIdx	0	1	2	3	4	5

8.3.4.11　b_pu_type_index 的反二值化方法

如果 cu_type_index 的值等于 2，由二元符号串查表 71 得到 b_pu_type_index 的值；否则，由二元符号串查表 72 得到 b_pu_type_index 的值。

表71　b_pu_type_index 的值与二元符号串的关系（cu_type_index 等于 2）

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

表72　b_pu_type_index 的值与二元符号串的关系（cu_type_index 不等于 2）

b_pu_type_index的值	二元符号串					
0	1	1				
1	1	0	1			
2	1	0	0	1		
3	1	0	0	0		
4	0	1	1			
5	0	1	0	1		

表 72（续）

b_pu_type_index的值	二元符号串					
6	0	1	0	0	1	
7	0	1	0	0	0	
8	0	0	1	1		
9	0	0	1	0	1	
10	0	0	1	0	0	1
11	0	0	1	0	0	0
12	0	0	0	1		
13	0	0	0	0	1	
14	0	0	0	0	0	1
15	0	0	0	0	0	0
binIdx	0	1	2	3	4	5

8.3.4.12 b_pu_type_min_index 的反二值化方法

由二元符号串查表 73 得到 b_pu_type_min_index 的值。

表73 b_pu_type_min_index 的值与二元符号串的关系

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

8.3.4.13 f_pu_type_index 的反二值化方法

如果 cu_type_index 的值等于 2，由二元符号串查表 74 得到 f_pu_type_index 的值；否则，由二元符号串查表 75 得到 f_pu_type_index 的值。

表74 f_pu_type_index 的值与二元符号串的关系（cu_type_index 等于 2）

f_pu_type_index的值	二元符号串
0	0
1	1
binIdx	0

表75　f_pu_type_index 的值与二元符号串的关系（cu_type_index 不等于 2）

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

8.3.4.14　b_pu_type_index2 的反二值化方法

由二元符号串查表 76 得到 b_pu_type_index2 的值。

表76　b_pu_type_index2 的值与二元符号串的关系

b_pu_type_index2的值	二元符号串		
0	0	0	
1	0	1	
2	1	0	
3	1	1	0
4	1	1	1
binIdx	0	1	2

8.3.4.15　intra_pu_type_index 的反二值化方法

由二元符号串查表 77 得到 intra_pu_type_index 的值。

表77　intra_pu_type_index 的值与二元符号串的关系

intra_pu_type_index的值	二元符号串
0	1
1	0
binIdx	0

8.3.4.16　intra_luma_pred_mode 的反二值化方法

由二元符号串查表 78 得到 intra_luma_pred_mode 的值。

表78　intra_luma_pred_mode 的值与二元符号串的关系

intra_luma_pred_mode的值	二元符号串						
0	1	0					
1	1	1					

表 78（续）

intra_luma_pred_mode的值	二元符号串						
2	0	0	0	0	0	0	
3	0	0	0	0	0	1	
4	0	0	0	0	1	0	
…							
32	0	1	1	1	1	0	
binIdx	0	1	2	3	4	5	6

8.3.4.17 dir_multi_hypothesis_mode 的反二值化方法

由二元符号串查表 79 得到 dir_multi_hypothesis_mode 的值。

表79 dir_multi_hypothesis_mode 的值与二元符号串的关系

dir_multi_hypothesis_mode的值	二元符号串				
0	0				
3	1	0	0		
4	1	0	1		
7	1	1	0	0	
8	1	1	0	1	
1	1	1	1	0	0
2	1	1	1	0	1
5	1	1	1	1	0
6	1	1	1	1	1
binIdx	0	1	2	3	4

8.3.4.18 mv_diff_x_abs 和 mv_diff_y_abs 的反二值化方法

由二元符号串查表 80 得到 synElVal 的值。表 80 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 49）；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。

mv_diff_x_abs 的值或者 mv_diff_y_abs 的值等于 synElVal。

表80 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					

表 80（续）

synElVal的值	二元符号串										
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	0	1	2	3	4	5	6	7	8	9	10

8.3.4.19　ctp_uv 的反二值化方法

由二元符号串查表 81 得到 ctp_uv 的值。

表81　ctp_uv 的值与二元符号串的关系

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

8.3.4.20　cu_qp_delta 的反二值化方法

变量 synElVal 的反二值化方法为一元码（见 8.3.4.3）。cu_qp_delta 的值由以下方法获得：

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

8.3.4.21　当 CoeffLevelMinus1Band 为 1 时 coeff_level_minus1_pos_in_band 的反二值化方法

当 CoeffLevelMinus1Band 为 1 时，coeff_level_minus1_pos_in_band 的反二值化方法为 0 阶指数哥伦布码（见表 49）。

8.3.4.22　last_cg_x，last_cg_y 和 last_cg_y_minus1 反二值化方法中 maxVal 的确定

last_cg_x，last_cg_y 和 last_cg_y_minus1 反二值化方法中的 maxVal 由 IsChroma 和 IntraModeIdx 查表 82 得到。

表82　last_cg_x，last_cg_y 和 last_cg_y_minus1 反二值化方法中 maxVal 值

IsChroma	IntraModeIdx	last_cg_x 对应的 maxVal	last_cg_y 对应的 maxVal	last_cg_y_minus1 对应的 maxVal
0	2	MaxCGNumY	MaxCGNumX	MaxCGNumX-1
0	不是 2	MaxCGNumX	MaxCGNumY	MaxCGNumY-1
1	-	MaxCGNumX	MaxCGNumY	MaxCGNumY-1

设当前变换块尺寸为 $M_1 \times M_2$，表 82 中 MaxCGNumX=(M_1>>2)-1，MaxCGNumY=(M_2>>2)-1。

9　解码过程

9.1　序列解码

序列解码过程如下：

a)　第一步，将 InitialFlag 的值初始化为 0。

b)　第二步，解码序列头：

　1)　步骤 2.1，如果 InitialFlag 的值为 1，直接执行步骤 2.2；否则：

　　——将 DOIPrev 的值初始化为 0；

　　——清空解码图像缓冲区，包括参考图像缓冲区和场景图像缓冲区；

　　——将 InitialFlag 的值置为 1。

　2)　步骤 2.2，根据序列头中得到的 WeightQuantEnableFlag 和 LoadSeqWeightQuantDataFlag 的值，按照附录 D 或表 16（见 7.1.2.4）确定 4x4 加权量化矩阵 WeightQuantMatrix4x4 和 8x8 加权量化矩阵 WeightQuantMatrix8x8；

c)　第三步，依次解码图像，直到遇到序列起始码或序列结束码或视频编辑码。

d)　第四步，

　1)　如果遇到序列起始码，继续执行第二步；

　2)　否则，如果遇到视频序列结束码或者视频编辑码，则按照 POI 从小到大的顺序依次输出参考图像缓冲区中每幅“未输出”图像，直到所有图像都已输出。

9.2　图像解码

9.2.1　概述

图像的解码过程如下：

a)　解码图像头（见 9.2.2）。

b)　如果当前图像是 F、P、S 或 B 图像，构建参考图像队列（见 9.2.3）。

c)　解码当前图像的各个条带（见 9.3），得到补偿后样本。

d)　如果 loop_filter_disable_flag 的值为‘0’，对补偿后样本进行去块效应滤波操作（见 9.10），得到滤波后样本；如果 loop_filter_disable_flag 的值为‘1’，则将补偿后样本直接作为滤波后样本。

e) 如果 sample_adaptive_offset_enable_flag 的值为‘1’，对滤波后样本进行样值偏移补偿操作（见 9.11），得到偏移后样本；如果 sample_adaptive_offset_enable_flag 的值为‘0’，则将滤波后样本直接作为偏移后样本。
f) 如果 adaptive_loop_filter_enable_flag 的值为‘1’，对偏移后样本进行自适应修正滤波操作（见 9.12），得到重建样本；如果 adaptive_loop_filter_enable_flag 的值为‘0’，则将偏移后样本直接作为重建样本。
g) 重建样本构成解码图像；更新解码图像缓冲区并输出解码图像（见 9.2.4）。
h) 如果 ReferedByOthersFlag[RcsIndex]等于 1，则对每个像素按 9.13 进行运动信息存储。

9.2.2 图像头解码

图像头解码过程如下：

a) 如果当前图像起始码是 0x000001B3，则 PictureType 等于 0：
   1) 如果 ScenePictureFlag 等于 0，则当前图像是 I 图像；
   2) 如果 ScenePictureFlag 等于 1：
      ——如果 scene_picture_output_flag 等于‘1’，则当前图像是 G 图像；
      ——如果 scene_picture_output_flag 等于‘0’，则当前图像是 GB 图像。
b) 如果当前图像起始码是 0x000001B6：
   1) 如果 picture_coding_type 等于‘01’：
      ——如果 ScenePredFlag 等于 0，则 PictureType 等于 1，当前图像是 P 图像；
      ——如果 ScenePredFlag 等于 1，则 PictureType 等于 4，当前图像是 S 图像。
   2) 如果 picture_coding_type 等于‘10’，则 PictureType 等于 2，当前图像是 B 图像；
   3) 如果 picture_coding_type 等于‘11’，则 PictureType 等于 3，当前图像是 F 图像。
c) 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。
d) 如果 DOI 小于 DOIPrev，则参考图像缓冲区中的所有图像的解码顺序索引和显示顺序索引都减去 256，DOIPrev 等于 DOI。
e) 如果当前图像不是 GB 图像，则 POI 等于 DOI+PictureOutputDelay-OutputReorderDelay。参考图像缓冲区中的图像的显示顺序索引值与当前图像的显示顺序索引值的差值的绝对值应小于 128。
f) 如果当前图像为 I、G 或 GB 图像，当前图像的参考图像数量 RefPicNum 为 0；如果当前图像为 S 图像，当前图像的参考图像数量 RefPicNum 为 1；如果当前图像不为 I、G、GB 和 S 图像，初始化 RefPicNum 为 NumOfRefPic[RcsIndex]。初始化 RemovedPicNum 为 NumOfRemovedPic[RcsIndex]。
g) 如果 PicWeightQuantEnableFlag 的值等于 1，从当前图像头的位流中导出的加权量化矩阵。
   1) 第一步，按照 9.2.5.2 确定 4×4 和 8×8 变换块的加权量化矩阵；
   2) 第二步，根据第一步确定的 8×8 加权量化矩阵，按照 9.2.5.3 确定 M_1×M_2变换块（M_1或者 M_2大于 8）的加权量化矩阵。

9.2.3 参考图像队列

构建参考图像队列的方法如下：

a) 如果当前图像是 S 图像，则参考图像队列只有 1 幅参考图像，将场景图像缓冲区中的图像移入参考图像队列 RefPicNum-1 的位置；
b) 否则：

1) 初始化 j 为 0；

2) 在参考图像缓冲区中进行 RefPicNum 次查找，每次查找解码顺序索引等于 DOI-DeltaDoiOfRefPic[RcsIndex][j]且已标记为“被参考”的图像，并将该图像移入参考图像队列位置 j，并令 j 等于 j+1，被参考图像的时间层标识的值应小于或等于当前解码图像的时间层标识的值;定义当前图像对应的序列头为解码顺序在当前图像之前最近的一个序列头，如果当前图像满足以下条件，则当前图像应能找到 RefPicNum 个参考图像：

——当前图像对应的序列头后的第一个解码图像为 I 图像，并且当前图像的显示顺序在这个 I 图像之后；

——当前图像对应的序列头后的第一个解码图像为 G 图像，并且当前图像的显示顺序在这个 G 图像之后；

——当前图像对应的序列头后的第一个解码图像为 GB 图像，该序列头后的第二个解码图像是 S 图像，并且当前图像的显示顺序在这个 S 图像之后。

3) 如果当前图像为 P 或 F 图像，并且 SceneReferenceEnableFlag 等于 1，则将参考图像队列 RefPicNum-1 位置的图像替换为场景图像缓冲区中的图像。

9.2.4 解码图像缓冲区

符合本部分的编码视频序列应保证参考图像缓冲区中的图像和场景图像缓冲区中的图像总数在解码过程中始终不超过解码图像缓冲区大小（由级别规定，见附录B）。

解码完一幅图像后，对参考图像缓冲区和场景图像缓冲区依次进行以下操作：

a) 如果当前解码图像不是 GB 图像，执行以下步骤输出图像：

1) 查找“可输出”图像。“可输出”图像包括参考图像缓冲区中被标记为“未输出”且解码顺序索引与图像输出延迟之和小于等于 DOI 的图像。如果 PictureOutputDelay 的值为 0，则“可输出”图像还包括当前解码图像。

2) 如果存在“可输出”图像，则输出“可输出”图像中显示顺序索引最小的图像，并把该图像标记为“已输出”图像。

b) 移出图像：

1) 如果当前解码图像是 G 或者 GB 图像，则移出场景图像缓冲区中的 G 和 GB 图像；

2) 在参考图像缓冲区中查找解码顺序索引为 DOI-DeltaDoiOfRemovedPic[RcsIndex][j]的图像，并将找到的图像标记为“不被参考”图像，其中 j 遍历 0 到 NumOfRemovedPicture[RcsIndex]-1，被标记为“不被参考”的图像的时间层标识的值应大于或等于当前解码图像的时间层标识的值；

3) 移出参考图像缓冲区中被标记为“不被参考”且“已输出”的图像。

c) 移入图像：

1) 如果当前解码图像是 GB 图像，则将其移入场景图像缓冲区；

2) 如果当前解码图像是 G 图像，则先将其移入场景图像缓冲区，然后再进行以下操作：

——如果 ReferedByOthersFlag[RcsIndex]等于 1 或 PictureOutputDelay 大于 0，则将其移入参考图像缓冲区；

——如果 ReferedByOthersFlag[RcsIndex]等于 1，则将其标记为“被参考”图像；

——如果 PictureOutputDelay 大于 0，则将其标记为“未输出”图像。

3) 如果当前解码图像不是 G 或 GB 图像，则：

——如果 ReferedByOthersFlag[RcsIndex]等于 1 或 PictureOutputDelay 大于 0，则将其移入参考图像缓冲区；

——如果 ReferedByOthersFlag[RcsIndex]等于 1，则将其标记为“被参考”图像；

——如果 PictureOutputDelay 大于 0，则将其标记为“未输出”图像。

9.2.5 确定加权量化矩阵

9.2.5.1 概述

本条定义确定当前图像的加权量化矩阵的方法。

如果WeightQuantEnableFlag的值为0或PicWeightQuantEnableFlag的值为0，则加权量化矩阵WeightQuantMatrix$_{8x8}$的元素WeightQuantMatrix$_{8x8}$[i][j]（i,j=0～7）的值均初始化为64；加权量化矩阵WeightQuantMatrix$_{4x4}$的元素WeightQuantMatrix$_{4x4}$[i][j]（i,j=0～3）的值均初始化为64；加权量化矩阵WeightQuantMatrix$_{M1xM2}$的元素WeightQuantMatrix$_{M1xM2}$[i][j]（i,j=0～(1 << SizeInBit) − 1）的值均初始化为64，其中M_1或者M_2大于8；WqmShift初始化为2。

否则，如果WeightQuantEnableFlag的值为1且PicWeightQuantEnableFlag的值为1，先根据9.2.5.2确定4×4和8×8加权量化矩阵，再根据9.2.5.3映射导出$M_1 \times M_2$加权量化矩阵，其中M_1或者M_2大于8；WqmShift初始化为2。

9.2.5.2 确定 4x4 和 8x8 加权量化矩阵

如果pic_weight_quant_data_index的值为‘00’，则根据LoadSeqWeightQuantDataFlag的值，按照附录D或表16（见7.1.2.4）确定4×4加权量化矩阵WeightQuantMatrix$_{4x4}$和和8×8加权量化矩阵WeightQuantMatrix$_{8x8}$。

否则，如果pic_weight_quant_data_index的值为‘10’，则根据表16（见7.1.2.4）确定4×4加权量化矩阵WeightQuantMatrix$_{4x4}$和8×8加权量化矩阵WeightQuantMatrix$_{8x8}$。

否则，如果pic_weight_quant_data_index的值为‘01’，则按照以下步骤确定4×4和8×8加权量化矩阵：

a) 第一步，确定当前图像的 wqP（wqP 的元素的取值范围应是 1～255）：
   1) 如果 weight_quant_param_index 的值为‘00’，则 wqP[i]=WeightQuantParamDefault[i]（i=0～5），其中，加权量化参数 WeightQuantParamDefault[*i*]＝{64，49，53，58，58，64}（i=0～5）。
   2) 如果 weight_quant_param_index 的值为‘01’，从 weight_quant_param_delta1[i]解析得到 wqPDelta1[i]。wqP[i]=wqPDelta1[i] + WeightQuantParamBase1[i]（i=0～5），其中，加权量化参数 WeightQuantParamBase1[i]＝{67，71，71，80，80，106}（i=0～5）。
   3) 如果 weight_quant_param_index 的值为‘10’，从 weight_quant_param_delta2[i]解析得到 wqPDelta2[i]。wqP[i]=wqPDelta2[i] + WeightQuantParamBase2[i]（i=0～5），其中，加权量化参数 WeightQuantParamBase2[i]＝{64，49，53，58，58，64}（i=0～5）。

b) 第二步，根据 WeightQuantModel 确定 8×8 加权量化矩阵 wqM8x8：
   1) 如果 WeightQuantModel 的值为 0，则 wqM8x8 如下：

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

   2) 如果 WeightQuantModel 的值为 1，则 wqM8x8 如下：

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

3) 如果 WeightQuantModel 的值为 2，则 wqM8x8 如下：

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

c) 第三步，根据 WeightQuantModel 确定 4×4 加权量化矩阵 wqM4x4：

1) 如果 WeightQuantModel 的值为 0，则 wqM4x4 如下：

$$wqM4x4=\begin{bmatrix} wqP[0] & wqP[4] & wqP[3] & wqP[5] \\ wqP[4] & wqP[2] & wqP[1] & wqP[5] \\ wqP[3] & wqP[1] & wqP[1] & wqP[5] \\ wqP[5] & wqP[5] & wqP[5] & wqP[5] \end{bmatrix}$$

2) 如果 WeightQuantModel 的值为 1，则 wqM4x4 如下:

$$wqM4x4=\begin{bmatrix} wqP[0] & wqP[4] & wqP[4] & wqP[5] \\ wqP[3] & wqP[2] & wqP[2] & wqP[5] \\ wqP[3] & wqP[2] & wqP[1] & wqP[5] \\ wqP5] & wqP[5] & wqP[5] & wqP[5] \end{bmatrix}$$

3) 如果 WeightQuantModel 的值为 2，则 wqM4x4 如下:

$$wqM4x4=\begin{bmatrix} wqP[0] & wqP[4] & wqP[3] & wqP[5] \\ wqP[4] & wqP[3] & wqP[2] & wqP[5] \\ wqP[3] & wqP[2] & wqP[1] & wqP[5] \\ wqP5] & wqP[5] & wqP[5] & wqP[5] \end{bmatrix}$$

8×8 变换块的加权量化矩阵 $WeightQuantMatrix_{8x8}$ 等于 wqM8x8，4×4 变换块的加权量化矩阵 $WeightQuantMatrix_{4x4}$ 等于 wqM4x4。

9.2.5.3 $M_1 \times M_2$ 加权量化矩阵的导出方法

如果 M_1 或者 M_2 中至少有一个大于8，其中，$M_1 \times M_2$ 的取值为4×16、16×4、8×32、32×8、16×16或32×32，按照如下方式确定 $M_1 \times M_2$ 变换块所使用的加权量化矩阵 $WeightQuantMatrix_{M1xM2}$：

a) 根据 9.2.5.2 确定 8×8 变换块所使用的加权量化矩阵 $WeightQuantMatrix_{8x8}$；

b) 如果 M_1 或者 M_2 等于 16，确定 $M_1 \times M_2$ 变换块所使用的加权量化矩阵 $WeightQuantMatrix_{M1xM2}$ 为：
$WeightQuantMatrix_{M1xM2}[x][y]=WeightQuantMatrix_{8x8}[x>>1][y>>1]$, x=0～$M_1$-1, y=0～$M_2$-1；
如果 M_1 或者 M_2 等于 32，确定 $M_1 \times M_2$ 变换块所使用的加权量化矩阵 $WeightQuantMatrix_{M1xM2}$ 为：
$WeightQuantMatrix_{M1xM2}[x][y]=WeightQuantMatrix_{8x8}[x>>2][y>>2]$, x=0～$M_1$-1, y=0～$M_2$-1。

9.3 条带解码

条带解码过程如下：

a) LcuIndex 等于 $LcuRow \times MinCuWidth \times 2^{LcuSizeInBit-Log(MiniSize)} + LcuColumn \times 2^{LcuSizeInBit-Log(MiniSize)}$；

b) 如果 fixed_picture_qp 等于‘0’，预测量化参数 PreviousQP 等于 SliceQp，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；

c) 依次解码各个最大解码单元，见 9.4。

9.4 最大编码单元解码

按照光栅扫描顺序依次解码最大解码单元，其解码过程如下：

a) 解码当前最大编码单元的编码树；

b) 完成当前最大编码单元的解码后，按以下过程更新 LcuIndex：

```
if (((LcuIndex % MinCuWidth) + (1<<(LcuSizeInBit−Log(MiniSize)))) >= MinCuWidth) {
    if (MinCuWidth % (1 << (LcuSizeInBit - Log(MiniSize))) == 0)
        t = 1 << (LcuSizeInBit - Log(MiniSize))
    else
        t = MinCuWidth % (1 << (LcuSizeInBit - Log(MiniSize)))
    LcuIndex = LcuIndex + MinCuWidth×((1<<(LcuSizeInBit-Log(MiniSize)))-1) + t
}
else
    LcuIndex = LcuIndex + (1 << (LcuSizeInBit - Log(MiniSize)))
```

9.5 编码单元解码

9.5.1 概述

编码单元解码分为预测样本解码和残差样本解码。预测样本解码包括：确定编码单元类型和预测类型（见9.5.3），对于预测类型为帧内的编码单元分别导出其所包含的所有帧内预测块的帧内预测模式（见9.5.6）并进行帧内预测（见9.7），对于预测类型为帧间的编码单元分别导出其所包含的所有帧间预测单元的运动信息（见9.5.7和9.5.8）并进行帧间预测（见9.8）。残差样本解码包括：确定量化参数（见9.5.2），确定编码单元划分为变换块的方式（见9.5.5），依次解码各个变换块（见9.6）。完成预测样本解码和残差样本解码后，进行预测补偿得到补偿后样本（见9.9）。

9.5.2 确定量化参数

本条确定量化参数QP_X（X为Y、Cb或Cr）。

如果当前编码单元的cu_qp_delta不在位流中，当前编码单元的量化参数CurrentQP等于PreviousQP。否则，当前编码单元的量化参数CurrentQP等于PreviousQP加上cuQpDelta。CurrentQP的取值范围应是0～(63+8×(BitDepth−8))。其中，PreviousQP的值等于当前编码单元的左边编码单元A的量化参数QP_Y。如果当前编码单元的左边编码单元A“不可用”或fixed_picture_qp等于‘1’，则PreviousQP的值等于SliceQp。

亮度的量化参数QP_Y等于其所在编码单元的CurrentQP。

先按以下方法计算xCb、xCr（xCb、xCr的取值范围应是-16～63）：

$$xCb=CurrentQP-8\times(BitDepth-8)+chroma\_quant\_param\_delta\_cb$$

$$xCr=CurrentQP-8\times(BitDepth-8)+chroma\_quant\_param\_delta\_cr$$

然后分别以xCb和xCr为索引查表82得到QP'_{cb}和QP'_{cr}。Cb、Cr色度块的量化参数QP_{cb}和QP_{cr}按以下方法得到：

$$QP_{cb}=Clip3(0,63+8\times(BitDepth-8),QP'_{cb}+8\times(BitDepth-8))$$

$$QP_{cr}=Clip3(0,63+8\times(BitDepth-8),QP'_{cr}+8\times(BitDepth-8))$$

本条确定的亮度量化参数QP_Y的取值范围应是0～(63+8×(BitDepth-8))。详见表83.

表83　色度量化参数与 xCb、xCr 的映射关系

xCb、xCr的值	QP'_{cb}和QP'_{cr}的值
<43	xCb、xCr
43	42
44	43
45	43
46	44
47	44
48	45
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

9.5.3　编码单元类型和相关信息

如果当前编码单元的IntraCuFlag为1，根据TransformSplitFlag和IntraPuTypeIndex查表84确定编码单元的编码单元类型（CuType）、预测划分方式、帧内亮度预测块数（NumOfIntraPredBlock）和预测块预测模式（PbPredMode）。

表84　IntraCUFlag 值为 1 时的编码单元类型和相关信息

TransformSplitFlag的值	IntraPuTypeIndex的值	编码单元类型	预测划分方式	帧内亮度预测块数	预测块预测模式
0	-	I_2N	I_NO_SPLIT	1	由9.5.6导出

表 84（续）

TransformSplitFlag的值	IntraPuTypeIndex的值	编码单元类型	预测划分方式	帧内亮度预测块数	预测块预测模式
1	2	I_N	I_CROSS_SPLIT	4	由9.5.6导出
1	0	I_nNxN	I_HOR	4	由9.5.6导出
1	1	I_NxnN	I_VER	4	由9.5.6导出

如果当前图像是P图像，根据CuTypeIndex、ShapeOfPartitionIndex和SizeInBit查表85确定编码单元的编码单元类型（CuType）、编码单元运动矢量数（CuMvNum）、预测划分方式和预测单元预测模式（PuPredMode）。

表85　P 图像的编码单元类型和相关信息

CuTypeIndex 的值	ShapeOfPartitionIndex 的值	SizeInBit 的值	编码单元类型	编码单元运动矢量数	预测划分方式	预测单元预测模式			
						PredUnitOrder 的值为 0	PredUnitOrder 的值为 1	PredUnitOrder 的值为 2	PredUnitOrder 的值为 3
0	-	>3	P_Skip	0	CROSS_SPLIT	单前向	单前向	单前向	单前向
0	-	3	P_Skip	0	NO_SPLIT	单前向	-	-	-
1	-	>3	P_Direct	0	CROSS_SPLIT	单前向	单前向	单前向	单前向
1	-	3	P_Direct	0	NO_SPLIT	单前向	-	-	-
2	-	-	P_2N	1	NO_SPLIT	单前向	-	-	-
3	0	-	P_2N_H	2	HOR_SYM	单前向	单前向	-	-
3	1	-	P_2N_HU	2	HOR_UP	单前向	单前向	-	-
3	2	-	P_2N_HD	2	HOR_DOWN	单前向	单前向	-	-
4	0	-	P_2N_V	2	VER_SYM	单前向	单前向	-	-
4	1	-	P_2N_VL	2	VER_LEFT	单前向	单前向	-	-
4	2	-	P_2N_VR	2	VER_RIGHT	单前向	单前向	-	-
5	-	-	P_N	4	CROSS_SPLIT	单前向	单前向	单前向	单前向
6	-	-	由表 84 得到	0	由表 84 得到	-			

如果当前图像是S图像，根据CuTypeIndex查表86确定编码单元的编码单元类型（CuType）、预测划分模式和预测单元预测模式（PuPredMode）。

表86 S 图像编码单元类型和相关信息

CuTypeIndex 的值	编码单元类型	预测划分方式	预测单元预测模式
0	S_Skip	NO_SPLIT	单前向
1	S_Direct	NO_SPLIT	单前向
2	由表 84 得到		-

如果当前图像是B图像，根据CuTypeIndex和ShapeOfPartitionIndex查表87确定编码单元的编码单元类型（CuType）、预测划分方式和预测单元预测模式（PuPredMode），其中：

a) 如果 CuTypeIndex 等于 0 或 1，根据编码单元类型、CuSubtypeIdx 和 SizeInBit 查表 88 确定编码单元的编码单元子类型（CuSubtype）、预测划分方式、预测单元的预测单元运动矢量数（PuMvNum）和预测单元预测模式（PuPredMode）；
b) 如果 CuTypeIndex 等于 2，根据 BPuTypeIndex 查表 89 确定各预测单元的预测单元运动矢量数（PuMvNum）和预测单元预测模式（PuPredMode）；
c) 如果 CuTypeIndex 等于 3 或 4，根据 BPuTypeIndex、SizeInBit 查表 90 确定各预测单元的预测单元运动矢量数（PuMvNum）和预测单元预测模式（PuPredMode）；
d) 如果 CuTypeIndex 等于 5，根据各预测单元的 BPuTypeIndex2 查表 91 确定各预测单元的预测单元类型（PuType）、预测单元运动矢量数（PuMvNum）和预测单元预测模式（PuPredMode）。

表87 B 图像的编码单元类型和相关信息

<table>
<tr><th>CuTypeIndex 的值</th><th>ShapeOfPartitionIndex 的值</th><th>编码单元类型</th><th>预测划分方式</th><th>预测单元预测模式</th></tr>
<tr><td>0</td><td>-</td><td>B_Skip</td><td>由表 87 得到</td><td>由表 87 得到</td></tr>
<tr><td>1</td><td>-</td><td>B_Direct_2N</td><td>由表 87 得到</td><td>由表 87 得到</td></tr>
<tr><td>2</td><td>-</td><td>B_2N</td><td>NO_SPLIT</td><td>由表 88 得到</td></tr>
<tr><td>3</td><td>0</td><td>B_2N_H</td><td>HOR_SYM</td><td rowspan="6">由表 89 得到</td></tr>
<tr><td>3</td><td>1</td><td>B_2N_HU</td><td>HOR_UP</td></tr>
<tr><td>3</td><td>2</td><td>B_2N_HD</td><td>HOR_DOWN</td></tr>
<tr><td>4</td><td>0</td><td>B_2N_V</td><td>VER_SYM</td></tr>
<tr><td>4</td><td>1</td><td>B_2N_VL</td><td>VER_LEFT</td></tr>
<tr><td>4</td><td>2</td><td>B_2N_VR</td><td>VER_RIGHT</td></tr>
<tr><td>5</td><td>-</td><td>B_NxN</td><td>CROSS_SPLIT</td><td>由表 90 得到</td></tr>
<tr><td>6</td><td>-</td><td colspan="2">由表 84 得到</td><td>-</td></tr>
</table>

表88　B 图像的编码单元子类型和相关信息

编码单元类型	CuSubtypeIdx 的值	SizeInBit 的值	编码单元子类型	预测划分方式	预测单元预测模式	预测单元运动矢量数
B_Skip	0	>3	B_Skip_Sym	CROSS_SPLIT	对称	0
	0	3	B_Skip_Sym	NO_SPLIT	对称	0
	1	-	B_Skip_Spatial_Bi	NO_SPLIT	双向	0
	2	-	B_Skip_Spatial_Bck	NO_SPLIT	后向	0
	3	-	B_Skip_Spatial_Sym	NO_SPLIT	对称	0
	4	-	B_Skip_Spatial_Fwd	NO_SPLIT	单前向	0
B_Direct_2N	0	>3	B_Direct_2N_Sym	CROSS_SPLIT	对称	0
	0	3	B_Direct_2N_Sym	NO_SPLIT	对称	0
	1	-	B_Direct_2N_Spatial_Bi	NO_SPLIT	双向	0
	2	-	B_Direct_2N_Spatial_Bck	NO_SPLIT	后向	0
	3	-	B_Direct_2N_Spatial_Sym	NO_SPLIT	对称	0
	4	-	B_Direct_2N_Spatial_Fwd	NO_SPLIT	单前向	0

表89　B 图像 B_2N 编码单元各预测单元的预测模式和相关信息

BPuTypeIndex的值	预测单元运动矢量数	预测单元预测模式
0	1	单前向
1	1	后向
2	1	对称
3	2	双向

表90　B 图像二分编码单元各预测单元的预测模式和相关信息

BPuTypeIndex 的值	SizeInBit的值	预测单元运动矢量数		预测单元预测模式	
		PredUnitOrder 的值为0	PredUnitOrder 的值为1	PredUnitOrder 的值为 0	PredUnitOrder 的值为 1
0	>3	2	2	双向	双向
0	3	1	1	单前向	单前向
1	>3	2	1	双向	后向
1	3	1	1	单前向	后向
2	>3	2	1	双向	单前向
2	3	1	1	后向	后向
3	>3	2	1	双向	对称
3	3	1	1	后向	单前向
4	>3	1	1	后向	后向
5	>3	1	2	后向	双向

表 90（续）

BPuTypeIndex的值	SizeInBit的值	预测单元运动矢量数		预测单元预测模式	
		PredUnitOrder的值为0	PredUnitOrder的值为1	PredUnitOrder的值为 0	PredUnitOrder的值为 1
6	>3	1	1	后向	单前向
7	>3	1	1	后向	对称
8	>3	1	1	单前向	单前向
9	>3	1	2	单前向	双向
10	>3	1	1	单前向	后向
11	>3	1	1	单前向	对称
12	>3	1	1	对称	对称
13	>3	1	2	对称	双向
14	>3	1	1	对称	后向
15	>3	1	1	对称	单前向

表91　B 图像 B_NxN 编码单元各预测单元的预测单元类型和相关信息

BPuTypeIndex2的值	预测单元类型	预测单元运动矢量数	预测单元预测模式
0	PU_Direct_NxN	0	对称
1	PU_Fwd_NxN	1	单前向
2	PU_Bck_NxN	1	后向
3	PU_Sym_NxN	1	对称
4	PU_Bi_NxN	2	双向

如果当前图像是F图像，根据CuTypeIndex和ShapeOfPartitionIndex查表92确定编码单元的编码单元类型（CuType）、编码单元运动矢量数（CuMvNum）、预测划分方式和预测单元预测模式（PuPredMode），其中：

a) 如果 CuTypeIndex 等于 0 或 1，根据编码单元类型、WeightedSkipMode、CuSubtypeIdx 和 SizeInBit 查表 93 确定编码单元的编码单元子类型（CuSubtype）、预测划分方式、预测单元的预测运动矢量数和预测单元预测模式（PuPredMode）；
b) 如果 CuTypeIndex 等于 2，根据 FPuTypeIndex 和 DirMultiHypothesisMode 查表 94 确定各预测单元的预测单元运动矢量数（PuMvNum）和预测单元预测模式(PuPredMode）；
c) 如果 CuTypeIndex 等于 3 或 4，根据 FPuTypeIndex 和 DirMultiHypothesisMode 查表 95 确定各预测单元的预测单元运动矢量数（PuMvNum）和预测单元预测模式（PuPredMode）；
d) 如果 CuTypeIndex 等于 5，根据 DirMultiHypothesisMode 和各预测单元的 FPuTypeIndex2 查表 96 确定各预测单元的预测单元运动矢量数（PuMvNum）和预测单元预测模式（PuPredMode）。

表92　F 图像的编码单元类型和相关信息

CuTypeIndex 的值	ShapeOfPartitionIndex 的值	编码单元类型	预测划分方式	编码单元运动矢量数	预测单元预测模式
0	-	F_Skip	由表 93 得到	0	由表 93 得到
1	-	F_Direct	由表 93 得到	0	
2	-	F_2N	NO_SPLIT	1	由表 93 得到
3	0	F_H	HOR_SYM	2	由表 95 得到
3	1	F_HU	HOR_UP	2	
3	2	F_HD	HOR_DOWN	2	
4	0	F_V	VER_SYM	2	
4	1	F_VL	VER_LEFT	2	
4	2	F_VR	VER_RIGHT	2	
5	-	F_N	CROSS_SPLIT	4	由表 96 得到
6	-	由表 84 得到		0	由表 84 得到

表93　F 图像的编码单元子类型和相关信息

编码单元类型	WeightedSkipMode的值	CuSubtypeIdx的值	SizeInBit的值	编码单元子类型	预测划分方式	预测单元运动矢量数	预测单元预测模式
F_Skip	非零	-	>3	F_Weighted_Skip	CROSS_SPLIT	0	双前向
	非零	-	3	F_Weighted_Skip	NO_SPLIT	0	双前向
	0	0	>3	F_Skip_Temporal	CROSS_SPLIT	0	单前向
	0	0	3	F_Skip_Temporal	NO_SPLIT	0	单前向
	0	1	-	F_Skip_Spatial_dual_fst	NO_SPLIT	0	双前向
	0	2	-	F_Skip_Spatial_dual_snd	NO_SPLIT	0	双前向
	0	3	-	F_Skip_Spatial_single_fst	NO_SPLIT	0	单前向
	0	4	-	F_Skip_Spatial_single_snd	NO_SPLIT	0	单前向
F_Direct	非零	-	>3	F_Weighted_Direct	CROSS_SPLIT	0	双前向
	非零	-	3	F_Weighted_Direct	NO_SPLIT	0	双前向
	0	0	>3	F_Direct_Temporal	CROSS_SPLIT	0	单前向
	0	0	3	F_Direct_Temporal	NO_SPLIT	0	单前向
	0	1	-	F_Direct_Spatial_dual_fst	NO_SPLIT	0	双前向
	0	2	-	F_Direct_Spatial_dual_snd	NO_SPLIT	0	双前向
	0	3	-	F_Direct_Spatial_single_fst	NO_SPLIT	0	单前向
	0	4	-	F_Direct_Spatial_single_snd	NO_SPLIT	0	单前向

表94 F 图像 F_2N 编码单元各预测单元的预测模式和相关信息

FPuTypeIndex的值	DirMultiHypothesisMode的值	预测单元运动矢量数	预测单元预测模式
0	非零	1	双前向
0	0	1	单前向
1	-	1	双前向

表95 F 图像二分编码单元各预测单元的预测模式和相关信息

FPuTypeIndex的值	SizeInBit的值	DirMultiHypothesisMode的值	预测单元运动矢量数		预测单元预测模式	
			PredUnitOrder 的值为 0	PredUnitOrder 的值为 1	PredUnitOrder 的值为 0	PredUnitOrder 的值为 1
0	>3	非零	1	1	双前向	双前向
0	>3	0	1	1	单前向	单前向
-	3	-	1	1	单前向	单前向
1	>3	-	1	1	单前向	双前向
2	>3	-	1	1	双前向	单前向
3	>3	-	1	1	双前向	双前向

表96 F 图像 F_N 编码单元各预测单元的预测单元类型和相关信息

DirMultiHypothesisMode的值	FPuTypeIndex2的值	预测单元运动矢量数	预测单元预测模式
0	0	1	单前向
非零	0	1	双前向
-	1	1	双前向

表85、表88、表89、表90、表91、表92、表93、表94、表95和表96中的编码单元运动矢量数（CuMvNum）表示当前编码单元在位流中的运动矢量数，预测单元运动矢量数（PuMvNum）表示预测单元在位流中的运动矢量数。一个编码单元的编码单元运动矢量数（CuMvNum）等于其所包含的预测单元的预测单元运动矢量数（PuMvNum）之和。

表84、表85、表86、表87、表88、表89和表93中的预测划分方式表示编码单元划分为帧内预测块或帧间预测单元的方式：

a） 如果预测划分方式为‘I_NO_SPLIT’，1 个 2N×2N 的编码单元划分为 1 个 2N×2N 的亮度预测块、1 个 N×N 的 Cb 预测块和 1 个 N×N 的 Cr 预测块（见图 10）；
b） 如果预测划分方式为‘I_CROSS_SPLIT’，1 个 2N×2N 的编码单元划分为 4 个 N×N 的亮度预测块、1 个 N×N 的 Cb 预测块和 1 个 N×N 的 Cr 预测块（见图 11）；
c） 如果预测划分方式为‘I_HOR’，1 个 2N×2N 的编码单元划分为 4 个 2N×0.5N 的亮度预测块、1 个 N×N 的 Cb 预测块和 1 个 N×N 的 Cr 预测块（见图 12）；
d） 如果预测划分方式为‘I_VER’，1 个 2N×2N 的编码单元划分为 4 个 0.5N×2N 的亮度预测块、1 个 N×N 的 Cb 预测块和 1 个 N×N 的 Cr 预测块（见图 13）；

e) 如果预测划分类型为‘NO_SPLIT’，1 个 2N×2N 的编码单元划分为 1 个 2N×2N 的帧间预测单元（见图 14）；
f) 如果预测划分类型为‘HOR_SYM’，1 个 2N×2N 的编码单元划分为 2 个 2N×N 的帧间预测单元（见图 15）；
g) 如果预测划分类型为‘VER_SYM’，1 个 2N×2N 的编码单元划分为 2 个 N×2N 的帧间预测单元（见图 16）；
h) 如果预测划分类型为‘CROSS_SPLIT’，1 个 2N×2N 的编码单元划分为 4 个 N×N 的帧间预测单元（见图 17）；
i) 如果预测划分类型为‘HOR_UP’，1 个 2N×2N 的编码单元划分为 1 个 2N×0.5N 的帧间预测单元和 1 个 2N×1.5N 的帧间预测单元（见图 18）；
j) 如果预测划分类型为‘HOR_DOWN’，1 个 2N×2N 的编码单元划分为 1 个 2N×1.5N 的帧间预测单元和 1 个 2N×0.5N 的帧间预测单元（见图 19）；
k) 如果预测划分类型为‘VER_LEFT’，1 个 2N×2N 的编码单元划分为 1 个 0.5N×2N 的帧间预测单元和 1 个 1.5N×2N 的帧间预测单元（见图 20）；
l) 如果预测划分类型为‘VER_RIGHT’，1 个 2N×2N 的编码单元划分为 1 个 1.5N×2N 的帧间预测单元和 1 个 0.5N×2N 的帧间预测单元（见图 21）。

图10～图13中矩形里的数字表示各预测块的PredBlockOrder。图14～图21中矩形里的数字表示各帧间预测单元的PredUnitOrder。

如果编码单元的编码单元类型为“I_2N”、“I_N”、“I_nNxN”或“I_NxnN”，则其预测类型为帧内，其所包含的预测块的预测类型为帧内；否则该编码单元的编码单元的预测类型为帧间，其所包含的预测单元和预测块的预测类型为帧间。

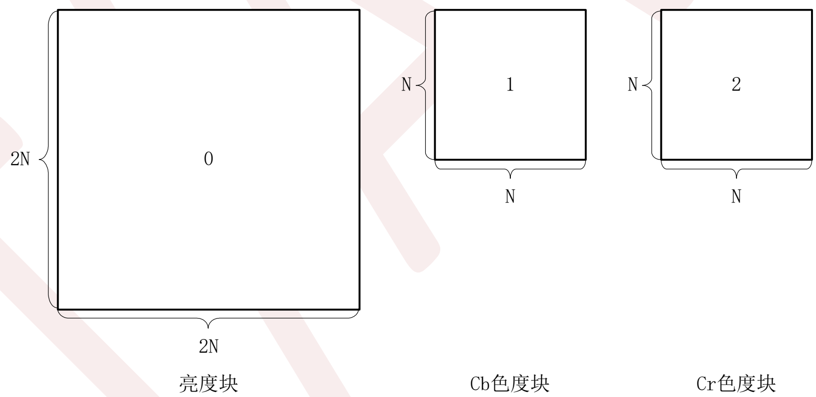


图10　编码单元划分为帧内预测块（预测划分方式‘I_NO_SPLIT’）

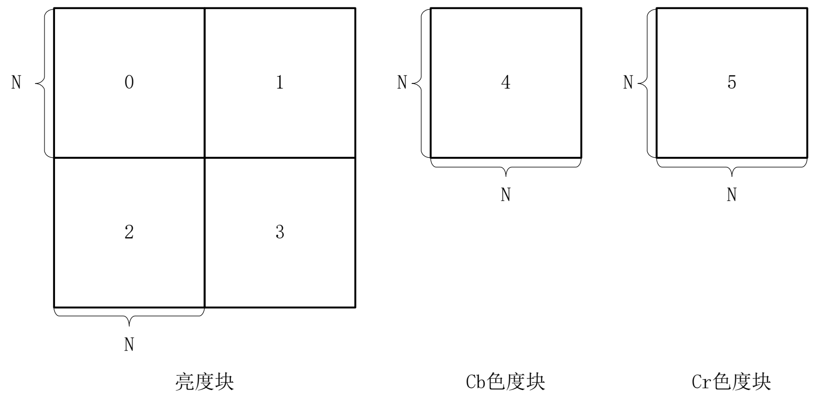


图11　编码单元划分为帧内预测块（预测划分方式‘I_CROSS_SPLIT’）

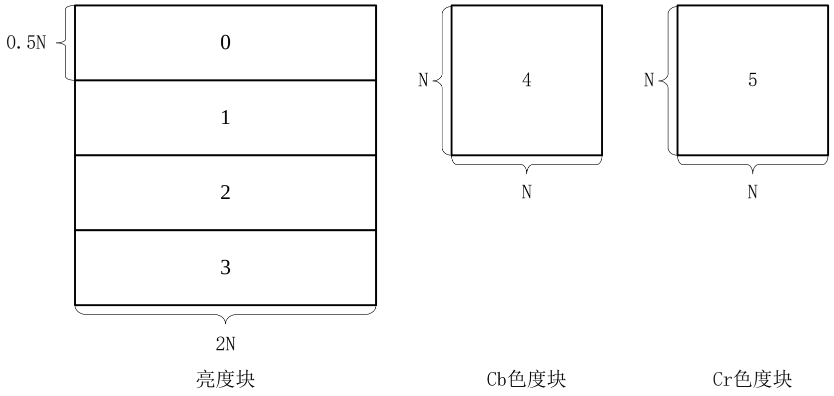


图12　编码单元划分为帧内预测块（预测划分方式‘I_HOR’）

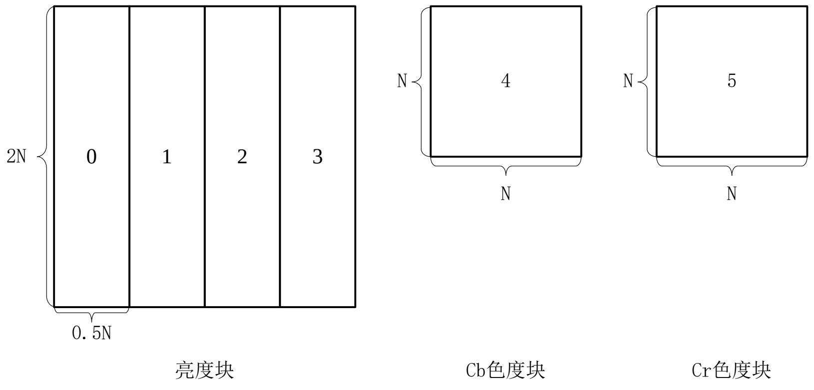


图13　编码单元划分为帧内预测块（预测划分方式‘I_VER’）

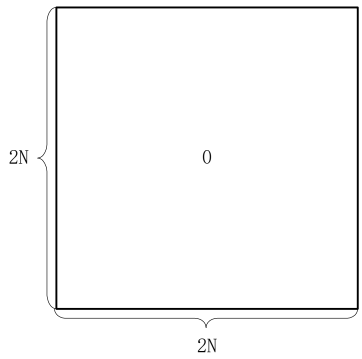


图14　编码单元划分为帧间预测单元（划分类型‘NO_SPLIT’）

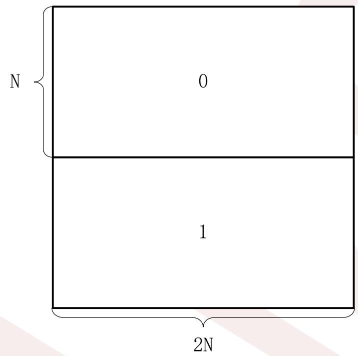


图15　编码单元划分为帧间预测单元（划分类型‘HOR_SYM’）

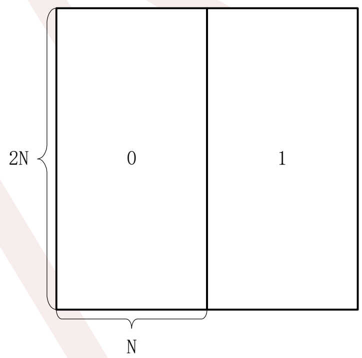


图16　编码单元划分为帧间预测单元（划分类型‘VER_SYM’）

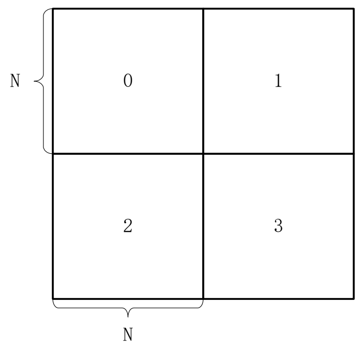


图17　编码单元划分为帧间预测单元（划分类型‘CROSS_SPLIT’）

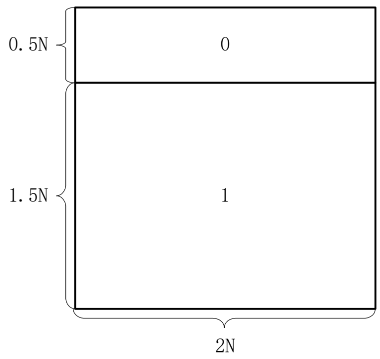


图18　编码单元划分为帧间预测单元（划分类型‘HOR_UP’）

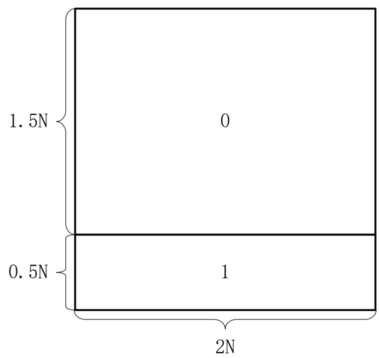


图19　编码单元划分为帧间预测单元（划分类型‘HOR_DOWN’）

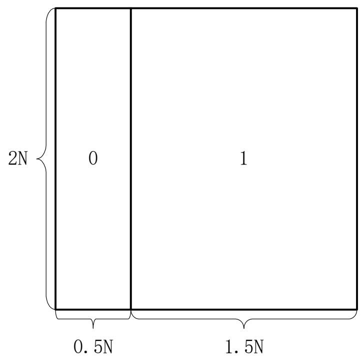


图20　编码单元划分为帧间预测单元（划分类型‘VER_LEFT’）

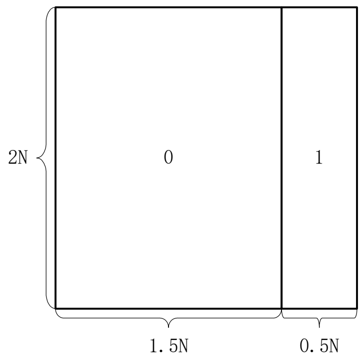


图21　编码单元划分为帧间预测单元（划分类型‘VER_RIGHT’）

maxPredUnitOrder的值是当前帧间预测单元的PredUnitOrder的最大值加1。

9.5.4 相邻块

块E的相邻块A是样本(x_0-1, y_0)所在的块，块E的相邻块B是样本(x_0, y_0-1)所在的块，块E的相邻块C是样本(x_1+1, y_0-1)所在的块，块E的相邻块D是样本(x_0-1, y_0-1)所在的块，块E的相邻块F是样本（x_0-1，y_1）所在的块，块E的相邻块G是样本（x_1, y_0-1）所在的块。其中(x_0, y_0)是块E左上角样本在图像中的坐标，(x_1, y_0)是块E右上角样本在图像中的坐标，(x_0, y_1)是块E左下角样本在图像中的坐标。块E和它的相邻块A、B、C和D的空间位置关系见图22。

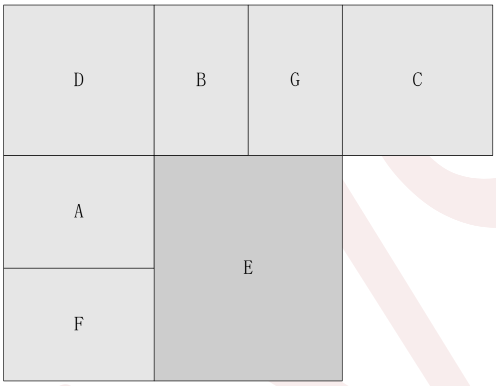


图22　块 E 和相邻块的空间位置关系

相邻块X（X为A、B、C、D、F或G）“存在”指该块应在图像内并且该块应与块E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”。

9.5.5 确定编码单元划分为变换块的方式

如果同时满足以下条件，则TransformSplitDirection的值为1：

a) NsqtEnableFlag 和 TransformSplitFlag 的值均为 1；

b) SizeInBit 大于 3；

c) 当前编码单元预测划分方式为‘HOR_SYM’、‘HOR_UP’或‘HOR_DOWN’。

或者，如果同时满足以下条件，则变量 TransformSplitDirection 的值为 1：

a) NsipEnableFlag 和 TransformSplitFlag 的值均为 1；

b) SizeInBit 的值等于 4 或 5；

c) 当前编码单元预测划分方式为‘I_HOR’。

否则，如果同时满足以下条件，则TransformSplitDirection的值为2：

a) NsqtEnableFlag 和 TransformSplitFlag 的值均为 1；

b) SizeInBit 大于 3；

c) 当前编码单元预测划分方式为‘VER_SYM’、‘VER_LEFT’或‘VER_RIGHT’。

或者，如果同时满足以下条件，则变量 TransformSplitDirection 的值为 2：

a) NsipEnableFlag 和 TransformSplitFlag 的值均为 1；

b) SizeInBit 的值等于 4 或 5；

c) 当前编码单元预测划分方式为‘I_VER’。

否则，TransformSplitDirection的值为0。

当前编码单元划分为变换块的方式如下：

a) 如果 TransformSplitFlag 的值为 0，则当前编码单元划分为 3 个变换块（1 个正方形亮度变换块和 2 个正方形色度变换块，见图 23），NumOfTransBlocks 的值为 3；
b) 否则，如果 TransformSplitFlag 的值为 1，并且 TransformSplitDirection 的值为 0，则当前编码单元划分为 6 个变换块（4 个正方形亮度变换块和 2 个正方形色度变换块，见图 24），NumOfTransBlocks 的值为 6；
c) 否则，如果 TransformSplitFlag 和 TransformSplitDirection 的值均为 1，则当前编码单元划分为 6 个变换块（4 个非正方形亮度变换块和 2 个正方形色度变换块，见图 25），NumOfTransBlocks 的值为 6；
d) 否则，如果 TransformSplitFlag 的值为 1，并且 TransformSplitDirection 的值为 2，当前编码单元划分为 6 个变换块（4 个非正方形亮度变换块和 2 个正方形色度变换块，见图 26），NumOfTransBlocks 的值为 6。

图 23～图 26 中数字为编码单元中变换块的顺序号。CuCtp 表示变换块顺序号为 0 到 NumOfTransBlocks-1的块是否包含非零变换系数。CuCtp的第n位（n等于0的位是最低有效位）等于‘0’表示顺序号为n的变换块没有非零系数，等于‘1’表示该变换块至少有一个非零系数。如果CuCtp的第n位等于‘1’，则按以下步骤解码对应的$M_1 \times M_2$变换块：

a) 第一步，令 M 的值等于 1 << (SizeInBit-TransformSplitFlag)，令变量 UpSampleEnableFlag 的值为 0。
b) 第二步，如果变换块是正方形的亮度变换块，M1=M2=M。否则，如果变换块是非正方形的亮度变换块，TransformSplitDirection 等于 1 时，M1=2M，M2=0.5M；TransformSplitDirection 等于 2 时，M1=0.5M，M2=2M。否则，如果变换块是色度块，M1=M2=1<<(SizeInBit-1)。
c) 第三步，如果 M1 和 M2 都不等于 64，跳至第四步，否则 M1=M1>>1，M2=M2>>1，令 UpSampleEnableFlag 的值等于 1。
d) 第四步，令 Log2TransformWidth 的值为 Log(M1)，令 Log2TransformHeight 的值为 Log(M2)。

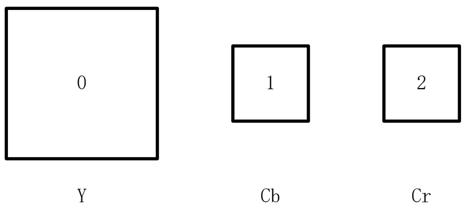


图23 编码单元划分为 1 个正方形亮度变换块和 2 个正方形色度变换块（4:2:0 格式）

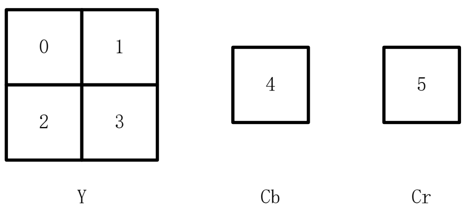


图24 编码单元划分为 4 个正方形亮度变换块和 2 个正方形色度变换块（4:2:0 格式）

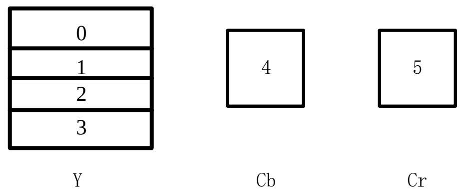


图25　编码单元划分为 4 个非正方形亮度变换块和 2 个正方形色度变换块（4:2:0 格式）

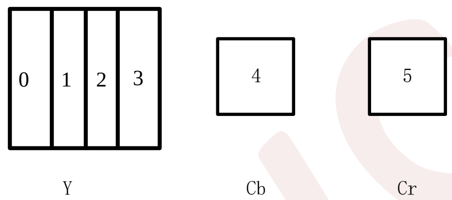


图26　编码单元划分为 4 个非正方形亮度变换块和 2 个正方形色度变换块（4:2:0 格式）

9.5.6　帧内预测模式

当前编码单元的每一个预测块用以下方法确定其帧内预测模式：

a)　如果当前预测块 E 是亮度块：

1)　计算当前预测块预测模式的预测值：

——如果左边预测块 A“存在”并且是帧内预测块，则将左边预测块的 IntraLumaPredMode 赋值给 intraPredModeA；否则 intraPredModeA 等于 0。

——如果上边预测块 B“存在”并且是帧内预测块，则将上边预测块的 IntraLumaPredMode 赋值给 intraPredModeB；否则 intraPredModeB 等于 0。

——如果 intraPredModeA 不等于 intraPredModeB，则 predIntraPredMode0 等于 Min(intraPredModeA, intraPredModeB)，predIntraPredMode1 等于 Max(intraPredModeA, intraPredModeB)；否则：

- 如果 intraPredModeA 等于 0，则 predIntraPredMode0 等于 0，predIntraPredMode1 等于 2。
- 如果 intraPredModeA 不等于 0，则 predIntraPredMode0 等于 0，predIntraPredMode1 等于 intraPredModeA。

2)　如果 intra_luma_pred_mode 的值为 0，则 IntraLumaPredMode 等于 predIntraPredMode0；否则，如果 intra_luma_pred_mode 的值为 1，则 IntraLumaPredMode 等于 predIntraPredMode1；否则：

——如果 intra_luma_pred_mode 减 2 的值小于 predIntraPredMode0，则 IntraLumaPredMode 等于 intra_luma_pred_mode 减 2；

——否则，如果 intra_luma_pred_mode 减 1 的值大于 predIntraPredMode0 并且小于 predIntraPredMode1，则 IntraLumaPredMode 等于 intra_luma_pred_mode 减 1；

——否则，IntraLumaPredMode 等于 intra_luma_pred_mode。

b)　如果当前预测块 E 是色度块：

1) 如果当前编码单元中 PredUnitOrder 的值为 0 的预测块的亮度预测模式 IntraLumaPredMode 等于 0、2、12 或 24，则 isRedundant 等于 1；否则 isRedundant 等于 0。
2) 如果 isRedundant 等于 0，IntraChromaPredMode 等于 intra_chroma_pred_mode；否则，依次执行以下操作：
——如果 IntraLumaPredMode 等于 0，则 predIntraChromaPredMode 等于 1；如果 IntraLumaPredMode 等于 2，则 predIntraChromaPredMode 等于 4；如果 IntraLumaPredMode 等于 12，则 predIntraChromaPredMode 等于 3；如果 IntraLumaPredMode 等于 24，则 predIntraChromaPredMode 等于 2。
——如果 intra_chroma_pred_mode 等于 0，则 IntraChromaPredMode 等于 0；否则，如果 intra_chroma_pred_mode 的值小于 predIntraChromaPredMode，则 IntraChromaPredMode 等于 intra_chroma_pred_mode；否则 IntraChromaPredMode 等于 intra_chroma_pred_mode 加 1。

c) 根据 IntraLumaPredMode 的值，查表 97 得到亮度预测块的帧内预测模式。根据 IntraChromaPredMode 的值，查表 98 得到色度预测块的帧内预测模式。

表97　亮度预测块帧内预测模式

IntraLumaPredMode的值	帧内预测模式
0	Intra_Luma_DC
1	Intra_Luma_Plane
2	Intra_Luma_Bilinear
3～11	Intra_Luma_Angular
12	Intra_Luma_Vertical
13～23	Intra_Luma_Angular
24	Intra_Luma_Horizontal
25～32	Intra_Luma_Angular

表98　色度预测块帧内预测模式

IntraChromaPredMode的值	帧内预测模式
0	Intra_Chroma_DM
1	Intra_Chroma_DC
2	Intra_Chroma_Horizontal
3	Intra_Chroma_Vertical
4	Intra_Chroma_Bilinear

亮度预测块帧内预测模式见图27。

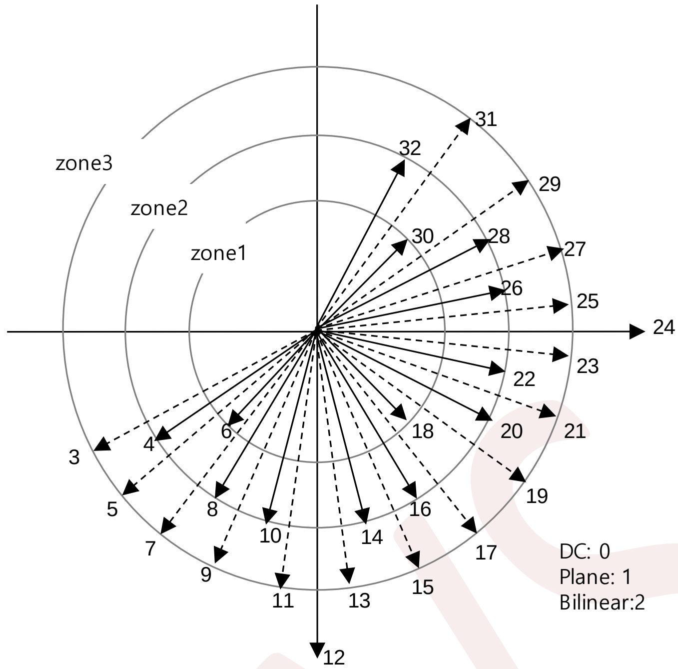


图27　亮度预测块帧内预测模式

9.5.7　参考图像选择

本条导出当前预测单元的参考索引值。当前预测单元的参考索引值表示当前预测单元进行解码所用的参考图像在参考图像队列中的编号。

如果当前图像不是B图像，且当前预测单元所在的编码单元的编码单元类型不是P_Skip、P_Direct、F_Skip或F_Direct，且当前预测单元的预测模式为单前向，则当前预测单元的前向参考索引值等于PuRefIdx的值。如果当前图像是B图像，且当前预测单元的预测模式为单前向，则当前预测单元的前向参考索引值等于1。

如果当前预测单元所在的编码单元编码单元类型为P_Skip或P_Direct，或编码单元子类型为F_Skip_Temporal或F_Direct_Temporal，则当前预测单元的前向参考索引值等于0。

如果当前预测单元的预测单元预测模式为后向，则当前预测单元的后向参考索引值等于0。

如果当前预测单元的预测单元预测模式为对称或双向，则当前预测单元的前向参考索引值等于1，后向参考索引值等于0。

如果同时满足以下条件，则当前预测单元的第一参考索引值等于0，第二参考索引值等于WeightedSkipMode：

a)　当前预测单元所在编码单元的编码单元类型为 F_Skip 或 F_Direct；

b)　当前预测单元的 WeightedSkipMode 不等于 0。

如果同时满足以下条件，则当前预测单元的第一参考索引值和第二参考索引值按照9.5.8.4.2方式导出：

a)　当前预测单元所在编码单元的编码单元类型为 F_Skip 或 F_Direct；

b)　当前预测单元的 WeightedSkipMode 等于 0；

c)　当前预测单元所在编码单元的编码单元子类型不为‘F_Skip_Temporal’或‘F_Direct_Temporal’。

如果同时满足以下条件，则当前预测单元的前向参考索引值等于PuRefIdx：

a) 当前预测单元所在编码单元的编码单元类型不为 F_Skip 或 F_Direct；
b) 当前预测单元的预测单元预测模式为双前向预测；
c) 当前预测单元所在编码单元的 DirMultiHypothesisMode 不等于 0。

如果同时满足以下条件，第一参考索引值等于PuRefIdx的值，第二参考索引值为0～(RefPicNum-1)中不等于PuRefIdx的最小值：

a) 当前预测单元所在编码单元的编码单元类型不为 F_Skip 或 F_Direct；
b) 当前预测单元的预测单元预测模式为双前向预测；
c) 当前预测单元所在编码单元的 DirMultiHypothesisMode 等于 0。

9.5.8 运动矢量

9.5.8.1 概述

如果同时满足以下条件，则当前预测单元的FwdMVExist的值为1，否则FwdMVExist的值为0：

a) 当前预测单元所在编码单元的编码单元类型不是‘P_Skip’、‘P_Direct’、‘S_Skip’、‘S_Direct’、‘B_Skip’、‘B_Direct_2N’、‘F_Skip’或‘F_Direct’；
b) 当前预测单元的预测模式为单前向、双前向、对称或双向；
c) 当前预测单元的预测类型不是‘PU_Direct_NxN’。

如果同时满足以下条件，则当前预测单元的BckMVExist的值为1，否则BckMVExist的值为0：

a) 当前预测单元所在编码单元的编码单元类型不是‘B_Skip’或‘B_Direct_2N’；
b) 当前预测单元的预测模式为后向或双向；
c) 当前预测单元的预测类型不是‘PU_Direct_NxN’。

运动矢量基本单位为1/4样本。

如果当前预测单元满足以下条件之一，则按照9.5.8.4定义的方法导出运动矢量：

a) 当前预测单元所在编码单元的编码单元类型是‘P_Skip’、‘P_Direct’、‘S_Skip’、‘S_Direct’、‘B_Skip’、‘B_Direct_2N’、‘F_Skip’或‘F_Direct’。
b) 当前预测单元的预测单元类型为‘PU_Direct_NxN’。

如果以上条件均不满足，则：

a) 如果当前预测单元的 FwdMvExist 为 1，则先按照 9.5.8.2 定义的方法得到运动矢量预测值，然后按照 9.5.8.3 定义的方法解码得到前向运动矢量或第一运动矢量；
b) 如果当前预测单元的 BckMvExist 为 1，则先按照 9.5.8.2 定义的方法得到运动矢量预测值，然后按照 9.5.8.3 定义的方法解码后向运动矢量。

9.5.8.2 运动矢量预测

图像的距离索引DistanceIndex等于POI乘2。

当前预测单元（属于当前图像）和它的运动矢量所指向的参考单元（属于参考图像）之间的距离BlockDistance计算如下：如果参考单元所在的参考图像是参考图像队列RefPicNum-1位置的图像，且当前图像的SceneReferenceEnableFlag值为1，则BlockDistance等于1。否则，

a) 如果参考单元在当前预测单元之前（显示顺序），BlockDistance 等于当前预测单元所在图像的 DistanceIndex 减去参考单元所在图像的 DistanceIndex 的差加上 512 的和模 512。
b) 如果参考单元在当前预测单元之后（显示顺序），BlockDistance 等于参考单元所在图像的 DistanceIndex 减去当前预测单元所在图像的 DistanceIndex 的差加上 512 的和模 512。

当前预测单元的亮度预测块E的相邻亮度预测块A、B、C、D各块所在的预测单元中与当前待预测运动矢量同类型的运动矢量记为mvA、mvB、mvC、mvD，对应的BlockDistance记为BlockDistanceA、 BlockDistanceB、BlockDistanceC、BlockDistanceD，对应的参考索引记为referenceIndexA、referenceIndexB、referenceIndexC、referenceIndexD。

依次执行以下操作：

a) 如果相邻亮度预测块 X（X 为 A、B 或 D）和其所在的预测单元满足以下条件之一，则 mvX 等于零矢量，BlockDistanceX 等于 1，referenceIndexX 等于-1；否则 mvX 等于相邻亮度预测块 X 所在的预测单元中与当前待预测运动矢量同类型的运动矢量，BlockDistanceX 等于 mvX 对应的 BlockDistance，referenceIndexX 等于 mvX 对应的 referenceIndex。
   1) 相邻亮度预测块 X“不可用”；
   2) 相邻亮度预测块 X 采用帧内预测模式；
   3) 相邻亮度预测块 X 所在的预测单元不存在与当前待预测运动矢量同类型的运动矢量；
   4) 当前图像的 SceneReferenceEnableFlag 值为 1，且当前待预测运动矢量和 mvX 中只有一个指向参考图像队列 RefPicNum-1 位置的参考图像。

b) 对相邻亮度预测块 C 执行以下操作：
   1) 如果相邻亮度预测块 C“不可用”,则 mvC 等于 mvD,BlockDistanceC 等于 BlockDistanceD，referenceIndexC 等于 referenceIndexD;
   2) 否则,如果相邻亮度预测块 C 和其所在的预测单元满足以下条件之一,则 mvC 等于零矢量，BlockDistanceC 等于 1，referenceIndexC 等于-1:
      ——相邻亮度预测块 C 采用帧内预测模式;
      ——相邻亮度预测块 C 所在的预测单元不存在与当前待预测运动矢量同类型的运动矢量;
      ——当前图像的 SceneReferenceEnableFlag 值为 1，且当前待预测运动矢量和 mvC 中只有一个指向参考图像队列 RefPicNum-1 位置的参考图像。
   3) 否则，mvC 等于相邻亮度预测块 C 所在的预测单元中与当前待预测运动矢量同类型的运动矢量，BlockDistanceC 等于 mvC 对应的 BlockDistance，referenceIndexC 等于 mvC 对应的 referenceIndex。

与当前待预测运动矢量同类型的运动矢量如下：

a) 当前图像是 P 图像，如果待预测的运动矢量是前向运动矢量，则与当前待预测运动矢量同类型的运动矢量是前向运动矢量;
b) 当前图像是 B 图像，如果待预测的运动矢量是前向运动矢量，则与当前待预测运动矢量同类型的运动矢量是前向运动矢量；如果待预测的运动矢量是后向运动矢量，则与当前待预测运动矢量同类型的运动矢量是后向运动矢量;
c) 当前图像是 F 图像，如果当前待预测的运动矢量是第一运动矢量，则与当前待预测运动矢量同类型的运动矢量是第一运动矢量或前向运动矢量；如果当前待预测的运动矢量是前向运动矢量，则与当前待预测运动矢量同类型的运动矢量是第一运动矢量或前向运动矢量。

当前待预测运动矢量预测值MVEPred计算过程如下:

a) 第一步，如果 referenceIndexX（X 为 A、B 或 C）不等于-1，则根据 BlockDistanceX 和 BlockDistanceE 对 mvX(mvX_x,mvX_y)进行缩放得到 MVX(MVX_x,MVX_y)；否则 MVX 为零矢量。然后进行第二步。其中 BlockDistanceE 是当前预测单元对应的 BlockDistance。

MVA_x = Clip3(-32768, 32767, Sign(mvA_x) × ((Abs(mvA_x) × BlockDistanceE × (16384 / BlockDistanceA) + 8192) >> 14))

MVA_y = Clip3(-32768, 32767, Sign(mvA_y+delta1) × ((Abs(mvA_y+delta1) × BlockDistanceE × (16384 / BlockDistanceA) + 8192) >> 14) - delta2)

MVB_x = Clip3(-32768, 32767, Sign(mvB_x) × ((Abs(mvB_x) × BlockDistanceE × (16384 / BlockDistanceB) + 8192) >> 14))

$$MVB\_y = Clip3(-32768, 32767, Sign(mvB\_y+delta1) \times ((Abs(mvB\_y+delta1) \times BlockDistanceE \times (16384 / BlockDistanceB) + 8192) >> 14) - delta2)$$

$$MVC\_x = Clip3(-32768, 32767, Sign(mvC\_x) \times ((Abs(mvC\_x) \times BlockDistanceE \times (16384 / BlockDistanceC) + 8192) >> 14))$$

$$MVC\_y = Clip3(-32768, 32767, Sign(mvC\_y+delta1) \times ((Abs(mvC\_y+delta1) \times BlockDistanceE \times (16384 / BlockDistanceC) + 8192) >> 14) - delta2)$$

其中,delta1 和 delta2 按照以下方式导出:如果 field_coded_sequence 的值为‘0’,则 delta1 和 delta2 的值均为 0;否则:

1) 如果 mvX 所在预测单元所在的场为顶场图像，并且 mvX（X 为 A、B 或 C）指向的场为底场图像，则 delta1 等于 2;
2) 如果 mvX 所在预测单元所在的场为底场图像，并且 mvX（X 为 A、B 或 C）指向的场为顶场图像，则 delta1 等于-2;
3) 如果 mvX 所在预测单元所在的场为顶场图像，并且 mvX（X 为 A、B 或 C）指向的场为顶场图像，则 delta1 等于 0;
4) 如果 mvX 所在预测单元所在的场为底场图像，并且 mvX（X 为 A、B 或 C）指向的场为底场图像，则 delta1 等于 0;
5) 如果 MVX 所在预测单元所在的场为顶场图像，并且 MVX（X 为 A、B 或 C）指向的场为底场图像，则 delta2 等于 2;
6) 如果 MVX 所在预测单元所在的场为底场图像，并且 MVX（X 为 A、B 或 C）指向的场为顶场图像，则 delta2 等于-2;
7) 如果 MVX 所在预测单元所在的场为顶场图像，并且 MVX（X 为 A、B 或 C）指向的场为顶场图像，则 delta2 等于 0;
8) 如果 MVX 所在预测单元在的场为底场图像，并且 MVX（X 为 A、B 或 C）指向的场为底场图像，则 delta2 等于 0。

b) 第二步，如果 referenceIndexA、referenceIndexB、referenceIndexC 三者中只有一个 referenceIndexX 不为-1，那么 MVEPred 等于 MVX（X 为 A、B 或 C），计算过程结束；否则执行第三步。
c) 第三步，如果当前亮度预测块 E 所在编码单元的预测划分方式是‘HOR_SYM’、‘HOR_UP’、‘HOR_DOWN’、‘VER_SYM’、‘VER_LEFT’和‘VER_RIGHT’，计算过程如下；否则执行第四步。
   1) ‘VER_SYM’、‘VER_LEFT’和‘VER_RIGHT’:
      ——E 为当前编码单元左边预测单元的亮度预测块：如果 referenceIndexA 等于 referenceIndexE，MVEPred 等于 MVA，计算过程结束；否则执行第四步。
      ——E 为当前编码单元右边预测单元的亮度预测块：如果 referenceIndexC 等于 referenceIndexE，MVEPred 等于 MVC，计算过程结束；否则执行第四步。
   2) ‘HOR_SYM’、‘HOR_UP’和‘HOR_DOWN’:
      ——E 为当前编码单元上边预测单元的亮度预测块：如果 referenceIndexB 等于 referenceIndexE，MVEPred 等于 MVB，计算过程结束；否则执行第四步。
      ——E 为当前编码单元下边预测单元的亮度预测块：如果 referenceIndexA 等于 referenceIndexE，MVEPred 等于 MVA，计算过程结束；否则执行第四步。

   其中，referenceIndexE 是当前待预测运动矢量的参考索引值。
d) 第四步，计算 MVEPred 的 x 和 y 运动矢量分量:
   MVEPred_x 的计算如下:

1) 如果 MVA_x < 0 且 MVB_x > 0 且 MVC_x > 0，或 MVA_x > 0 且 MVB_x < 0 且 MVC_x < 0，则 MVEPred_x = (MVB_x + MVC_x)/2；
2) 否则，如果 MVB_x < 0 且 MVA_x > 0 且 MVC_x > 0，或 MVB_x > 0 且 MVA_x < 0 且 MVC_x < 0，则 MVEPred_x = (MVA_x + MVC_x)/2；
3) 否则，如果 MVC_x < 0 且 MVA_x > 0 且 MVB_x > 0，或 MVC_x > 0 且 MVA_x < 0 且 MVB_x < 0，则 MVEPred_x = (MVA_x + MVB_x)/2；
4) 否则，如果 Abs(MVA_x-MVB_x)小于或等于 Abs(MVB_x-MVC_x)，且 Abs(MVA_x-MVB_x)小于或等于 Abs(MVC_x-MVA_x)，则 MVEPred_x = (MVA_x + MVB_x)/2；
5) 否则，如果 Abs(MVB_x-MVC_x)小于或等于 Abs(MVA_x-MVB_x)，且 Abs(MVB_x-MVC_x)小于或等于 Abs(MVC_x-MVA_x)，则 MVEPred_x = (MVB_x + MVC_x)/2；
6) 否则，MVEPred_x = (MVA_x + MVC_x)/2。

MVEPred_y 的计算如下：

1) 如果 MVA_y < 0 且 MVB_y > 0 且 MVC_y > 0，或 MVA_y > 0 且 MVB_y < 0 且 MVC_y < 0，则 MVEPred_y = (MVB_y + MVC_y)/2；
2) 否则，如果 MVB_y < 0 且 MVA_y > 0 且 MVC_y > 0，或 MVB_y > 0 且 MVA_y < 0 且 MVC_y < 0，则 MVEPred_y = (MVA_y + MVC_y)/2；
3) 否则，如果 MVC_y < 0 且 MVA_y > 0 且 MVB_y > 0，或 MVC_y > 0 且 MVA_y < 0 且 MVB_y < 0，则 MVEPred_y = (MVA_y + MVB_y)/2；
4) 否则，如果 Abs(MVA_y-MVB_y)小于或等于 Abs(MVB_y-MVC_y)，且 Abs(MVA_y-MVB_y)小于或等于 Abs(MVC_y-MVA_y)，则 MVEPred_y = (MVA_y + MVB_y)/2；
5) 否则，如果 Abs(MVB_y-MVC_y)小于或等于 Abs(MVA_y-MVB_y)，且 Abs(MVB_y-MVC_y)小于或等于 Abs(MVC_y-MVA_y)，则 MVEPred_y = (MVB_y + MVC_y)/2；
6) 否则，MVEPred_y = (MVA_y + MVC_y)/2。

9.5.8.3 运动矢量解码

如果PmvrThreshold等于-1，mvE等于MVEPred加上由MvDiffX和MvDiffY解码得到的运动矢量增量的和：

mvE_x = Clip3(-32768, 32767, MvDiffX + MVEPred_x)
mvE_y = Clip3(-32768, 32767, MvDiffY + MVEPred_y)

否则，按以下方法得到mvE：

a) 如果 Abs(MvDiffX-ctrd_x)大于 PmvrThreshold：

mvE_x = Clip3(-32768, 32767, MVEPred_x + MvDiffX×2 - ctrd_x - Sign(MvDiffX-ctrd_x)×PmvrThreshold)
mvE_y = Clip3(-32768, 32767, MVEPred_y + MvDiffY×2 + ctrd_y)

b) 否则，如果 Abs(MvDiffY-ctrd_y)大于 PmvrThreshold：

mvE_x = Clip3(-32768, 32767, MVEPred_x + MvDiffX×2 + ctrd_x)
mvE_y = Clip3(-32768, 32767, MVEPred_y + MvDiffY×2 - ctrd_y - Sign(MvDiffY-ctrd_y)×PmvrThreshold)

c) 否则：

mvE_x = Clip3(-32768, 32767, MvDiffX + MVEPred_x)
mvE_y = Clip3(-32768, 32767, MvDiffY + MVEPred_y)

其中ctrd_x和ctrd_y计算如下：

ctrd_x = ((MVEPred_x >> 1) << 1) - MVEPred_x

$$ctrd\_y = ((MVEPred\_y >> 1) << 1) - MVEPred\_y$$

得到mvE后，继续进行以下操作：

a) 如果当前预测单元的预测模式是单前向，mvE 是当前预测单元的前向运动矢量（记作 MvE）；
b) 如果当前预测单元的预测模式是后向，mvE 是当前预测单元的后向运动矢量（记作 MvE）；
c) 如果当前预测单元的预测模式是双向，mvE 是当前预测单元的前向运动矢量（记作 MvE0）或后向运动矢量（记作 MvE1）；
d) 如果当前预测单元的预测模式是对称，mvE 是当前预测单元的前向运动矢量（记作 MvE1），然后根据以下步骤得到当前预测单元的后向运动矢量（记作 MvE2）：
   1) 计算 BlockDistance1 和 BlockDistance2：
      $$BlockDistance1 = (DistanceIndexCur - DistanceIndex1 + 512) \% 512$$
      $$BlockDistance2 = (DistanceIndex2 - DistanceIndexCur + 512) \% 512$$
      其中 DistanceIndexCur 是当前预测单元所在图像的距离索引，DistanceIndex1 是前向参考图像的距离索引，DistanceIndex2 是后向参考图像的距离索引。
   2) 计算 MvE2：
      $$MvE2\_x = Clip3(-32768, 32767, -((MvE1\_x \times BlockDistance2 \times (16384 / BlockDistance1) + 8192) >> 14))$$
      $$MvE2\_y = Clip3(-32768, 32767, -(((MvE1\_y+delta1) \times BlockDistance2 \times (16384 / BlockDistance1) + 8192) >> 14) - delta2)$$
      其中delta1、delta2的导出方式见9.5.8.2， 此时9.5.8.2中的mvX为MvE1，MVX为MvE2。
e) 如果当前预测单元的预测模式是双前向且 DirMultiHypothesisMode 等于零，mvE 是当前预测单元的第一运动矢量（记作 MvE1），然后根据以下步骤得到当前预测单元的第二运动矢量（记作 MvE2）：
   1) 计算 BlockDistance1 和 BlockDistance2：
      ——如果当前预测单元的第一运动矢量指向参考图像队列 RefPicNum-1 位置的图像，且当前图像的 SceneReferenceEnableFlag 等于 1，则 BlockDistance1 等于 1；否则 BlockDistance1 = (DistanceIndexCur - DistanceIndex1+ 512) % 512。
      ——如果当前预测单元的第二运动矢量指向参考图像队列 RefPicNum-1 位置的图像，且当前图像的 SceneReferenceEnableFlag 等于 1，则 BlockDistance2 等于 1。否则 BlockDistance2 = (DistanceIndexCur - DistanceIndex2 + 512) % 512。
      其中 DistanceIndexCur 是当前预测单元所在图像的距离索引，DistanceIndex1 是第一参考图像的距离索引，DistanceIndex2 是第二参考图像的距离索引。
   2) 计算 MvE2：
      $$MvE2\_x = Clip3(-32768, 32767, (MvE1\_x \times BlockDistance2 \times (16384 / BlockDistance1) + 8192) >> 14)$$
      $$MvE2\_y = Clip3(-32768, 32767, (((MvE1\_y+delta1) \times BlockDistance2 \times (16384 / BlockDistance1) + 8192) >> 14) - delta2)$$
      其中delta1、delta2的导出方式见9.5.8.2，此时9.5.8.2中的mvX为MvE1， MVX为MvE2。
f) 如果当前预测单元的预测模式是双前向且 DirMultiHypothesisMode 不等于 0,mvE 是当前预测单元的前向运动矢量，继续按以下方法得到当前预测单元运动矢量甲（记作 MvE0）和运动矢量乙（记作 MvE1），并根据 MvE0 和 MvE1 在帧间预测过程中导出预测样本：
   $$MvE0\_x = Clip3(-32768, 32767, mvE\_x - dx)$$
   $$MvE0\_y = Clip3(-32768, 32767, mvE\_y - dy)$$
   $$MvE1\_x = Clip3(-32768, 32767, mvE\_x + dx)$$

MvE1_y = Clip3(-32768, 32767, mvE_y + dy)

其中 dx、dy 的值由 DirMultiHypothesis 查表 99 得到。

表99 DirMultiHypothesisMode 与 dx、dy 的关系

DirMultiHypothesis的值	dx的值	dy的值
1	1	0
2	0	1
3	1	-1
4	1	1
5	2	0
6	0	2
7	2	-2
8	2	2

9.5.8.4 导出运动矢量

9.5.8.4.1 概述

如果当前预测单元所在的编码单元的编码单元类型为‘P_Skip’、‘P_Direct’、‘F_Skip’或‘F_Direct’则按照9.5.8.4.2定义的方法导出运动矢量。

如果当前预测单元所在的编码单元的编码单元类型为‘S_Skip’、‘S_Direct’，则当前预测单元的前向运动矢量MvE是零矢量。

如果当前预测单元所在的编码单元的编码单元类型为‘B_Skip’或‘B_Direct_2N’，或者当前预测单元类型为‘PU_Direct_NxN’，则按照9.5.8.4.3定义的方法导出运动矢量。9.5.8.4.2和9.5.8.4.3中使用的运动信息存储单元的定义见9.13。

9.5.8.4.2 运动矢量导出方法 1

首先，如果编码单元类型为‘P_Skip’或‘P_Direct’，或编码单元子类型为‘F_Skip_Temporal’、‘F_Direct_Temporal’、‘F_Weighted_Skip’或‘F_Weighted_Direct’，首先计算PicCur到PicCol的BlockDistance（记作BlockDistanceCurCol）和PicCol到PicColRef的BlockDistance（记作BlockDistanceColRef）：

a) 如果当前图像的 SceneReferenceEnableFlag 值为 1 且 RefPicNum 值为 1，则 BlockDistanceCurCol 和 BlockDistanceColRef 均为 1；
b) 否则，如果 MvSuCol 中存储的参考图像索引值为-1，则 BlockDistanceColRef 等于 1，BlockDistanceCurCol = (DistanceIndexCur - DistanceIndexCol + 512) % 512；
c) 否则，BlockDistanceCurCol = (DistanceIndexCur - DistanceIndexCol + 512) % 512，BlockDistanceColRef = (DistanceIndexCol - DistanceIndexColRef + 512) % 512。

其中PicCur是当前预测单元所在的图像，DistanceIndexCur是PicCur的距离索引，PicCol是参考图像队列中标记为0的参考图像，DistanceIndexCol是PicCol的距离索引，MvSuCol是PicCol中与当前预测单元左上角亮度样本位置对应的亮度样本所在的运动信息存储单元，DistanceIndexColRef是MvSuCol的前向或第一参考图像PicColRef的距离索引；然后按以下方法导出当前预测单元的运动矢量：

a) 如果 MvSuCol 中存储的参考图像索引值为-1，则 mvE 是零矢量，继续执行步骤 c）。
b) 如果步骤 a）的条件不满足，按以下方法导出 mvE(mvE_x, mvE_y)：

$mvE\_x = Clip3(-32768, 32767, (BlockDistanceCurCol \times mvcol\_x \times (16384 / BlockDistanceColRef) + 8192) >> 14)$

$mvE\_y = Clip3(-32768, 32767, ((BlockDistanceCurCol \times (mvcol\_y+delta1) \times (16384 / BlockDistanceColRef) + 8192) >> 14) -delta2)$

其中，mvcol(mvcol_x, mvcol_y)是 MvSuCol 的前向运动矢量或第一运动矢量，delta1、delta2 的导出方式见 9.5.8.2，此时 9.5.8.2 中的 mvX 为 mvcol， MVX 为 mvE。

c) 如果编码单元类型是‘P_Skip’或‘P_Direct’，或编码单元子类型是‘F_Skip_Temporal’或‘F_Direct_Temporal’，则 mvE 是当前预测单元的前向运动矢量 MvE；否则根据上述方法 a）和 b）导出 mvE 后，再设参考图像队列中编号为 WeightedSkipMode 的参考图像的距离索引为 DistanceIndexI，当前预测单元所在图像到该参考图像的 BlockDistance 为 BlockDistanceI，如果 WeightedSkipMode 等于 RefPicNum-1 且当前图像的 SceneReferenceEnableFlag 值为 1，则 BlockDistanceI 等于 1；否则 BlockDistanceI = (DistanceIndexCur - DistanceIndexI + 512) % 512。如果 WeightedSkipMode 的值为 0，则当前预测单元的前向运动矢量 MvE 等于 mvE；否则，当前预测单元的第一运动矢量 MvE0 等于 mvE，第二运动矢量 MvE1(MvE1_x,MvE1_y)计算如下：

$MvE1\_x = Clip3(-32768, 32767, (MvE0\_x \times BlockDistanceI \times (16384/BlockDistanceCurCol) + 8192) >> 14)$

$MvE1\_y = Clip3(-32768, 32767, (((MvE0\_y+delta1) \times BlockDistanceI \times (16384/BlockDistanceCurCol) + 8192) >> 14)-delta2)$

其中 delta1、delta2 的导出方式见 9.5.8.2， 此时 9.5.8.2 中的 mvX 为 MvE0， MVX 为 MvE1。

其次，如果编码单元类型不是‘P_Skip’或‘P_Direct’，且编码单元子类型不是‘F_Skip_Temporal’、‘F_Direct_Temporal’ 、‘F_Weighted_Skip’和‘F_Weighted_Direct’，按以下方法导出运动矢量：

d) 如果当前预测单元所在编码单元的编码单元子类型为‘F_Skip_Spatial_dual_fst’或‘F_Direct_Spatial_dual_fst’,则按以下方式导出第一运动矢量 MvE0 和第二运动矢量 MvE1，并且同时导出第一参考索引值和第二参考索引值：

1) 如果当前预测单元的亮度预测块的相邻亮度预测块 F、G、C、A、B、D 六者中为双前向预测块的个数大于或等于 1，则按 F、G、C、A、B、D 的顺序依次扫描相邻亮度预测块得到第一个扫描到的双前向预测块,将该预测块所在预测单元的第一运动矢量和第一参考索引值分别作为当前预测单元的第一运动矢量 MvE0 和第一参考索引值，将该预测块所在预测单元的第二运动矢量和第二参考索引值分别作为当前预测单元的第二运动矢量 MvE1 和第二参考索引值；

2) 否则，如果当前预测单元的亮度预测块的相邻亮度预测块 F、G、C、A、B、D 六者中为单前向预测块的个数大于或等于 2，则按 F、G、C、A、B、D 的顺序依次扫描相邻亮度预测块得到第一个扫描到的单前向预测块，将该单前向预测块作为一号单前向预测块，按 D、B、A、C、G、F 的顺序依次扫描得到第一个扫描到的单前向预测块，将该单前向预测块作为二号单前向预测块,将一号单前向预测块所在预测单元的前向运动矢量和前向参考索引值分别作为当前预测单元的第一运动矢量 MvE0 和第一参考索引值，将二号单前向预测块所在预测单元的前向运动矢量和前向参考索引值分别作为当前预测单元的第二运动矢量 MvE1 和第二参考索引值；

3) 否则，当前预测单元的第一运动矢量 MvE0 和第二运动矢量 MvE1 均为零矢量，当前预测单元的第一参考索引值和第二参考索引值均为 0。

e) 否则，如果当前预测单元所在编码单元的编码单元子类型为‘F_Skip_Spatial_dual_snd’或‘F_Direct_Spatial_dual_snd’，按以下方式导出第一运动矢量 MvE0 和第二运动矢量 MvE1：
   1) 如果当前预测单元的亮度预测块的相邻亮度预测块 F、G、C、A、B、D 六者中为双前向预测块的个数大于或等于 2，则按 D、B、A、C、G、F 的顺序依次扫描相邻亮度预测块得到第一个扫描到的双前向预测块，将该预测块所在预测单元的第一运动矢量和第一参考索引值分别作为当前预测单元的第一运动矢量 MvE0 和第一参考索引值，将该预测块所在预测单元的第二运动矢量和第二参考索引值分别作为当前预测单元的第二运动矢量 MvE1 和第二参考索引值；
   2) 否则，如果当前预测单元的亮度预测块的相邻亮度预测块 F、G、C、A、B、D 六者中为双前向预测块的个数等于 1，并且为单前向预测块的个数大于或等于 1，则按 F、G、C、A、B、D 的顺序依次扫描相邻亮度预测块得到第一个扫描到的单前向预测块，将该单前向预测块作为一号单前向预测块，按 D、B、A、C、G、F 的顺序依次扫描得到第一个扫描到的单前向预测块，将该单前向预测块作为二号单前向预测块，将一号单前向预测块所在预测单元的前向运动矢量和前向参考索引值分别作为当前预测单元的第一运动矢量 MvE0 和第一参考索引值，将二号单前向预测块所在预测单元的前向运动矢量和前向参考索引值分别作为当前预测单元的第二运动矢量 MvE1 和第二参考索引值；
   3) 否则，当前预测单元的第一运动矢量 MvE0 和第二运动矢量 MvE1 均为零矢量，当前预测单元的第一参考索引值和第二参考索引值均为 0。
f) 否则，如果当前预测单元所在编码单元的编码单元子类型为‘F_Skip_Spatial_single_fst’或‘F_Direct_Spatial_single_fst’，按以下方式导出前向运动矢量 MvE：
   1) 如果当前预测单元的亮度预测块的相邻亮度预测块 F、G、C、A、B、D 六者中为单前向预测块的个数大于或等于 1，则按 F、G、C、A、B、D 的顺序依次扫描相邻亮度预测块得到第一个扫描到的单前向预测块，将该预测块所在预测单元的前向运动矢量和前向参考索引值分别作为当前预测单元的前向运动矢量 MvE 和前向参考索引值；
   2) 否则，如果当前预测单元的亮度预测块的相邻亮度预测块 F、G、C、A、B、D 六者中为双前向预测块的个数大于或等于 1，则按 D、B、A、C、G、F 的顺序依次扫描相邻亮度预测块得到第一个扫描到的双前向预测块，将该双前向预测块所在预测单元的第一运动矢量和第一参考索引值分别作为当前预测单元的前向运动矢量 MvE 和前向参考索引值；
   3) 否则，当前预测单元的前向运动矢量 MvE 为零矢量，前向参考参考索引值为 0。
g) 否则，如果当前预测单元所在编码单元的编码单元子类型为‘F_Skip_Spatial_single_snd’或‘F_Direct_Spatial_single_snd’，按以下方式导出前向运动矢量 MvE：
   1) 如果当前预测单元的亮度预测块的相邻亮度预测块 F、G、C、A、B、D 六者中为单前向预测块的个数大于或等于 2，则按 D、B、A、C、G、F 的顺序依次扫描相邻亮度预测块得到第一个扫描到的单前向预测块，将该预测块所在预测单元的前向运动矢量和前向参考索引值分别作为当前预测单元的前向运动矢量 MvE 和前向参考索引值；
   2) 否则，如果当前预测单元的亮度预测块的相邻亮度预测块 F、G、C、A、B、D 六者中为双前向预测块的个数大于或等于 1，则按 D、B、A、C、G、F 的顺序依次扫描相邻亮度预测块得到第一个双前向预测块，将该双前向预测块所在预测单元的第二运动矢量和第二参考索引值分别作为当前预测单元的前向运动矢量 MvE 和前向参考索引值；
   3) 否则，当前预测单元的前向运动矢量 MvE 为零矢量，前向参考参考索引值为 0。

其中，双前向预测块是指其所在预测单元预测模式为双前向，且其所在编码单元的DirMultiHypothesisMode等于0的亮度预测块；单前向预测块是指其所在预测单元预测模式为单前向，或其所在编码单元的DirMultiHypothesisMode不等于0的亮度预测块。

9.5.8.4.3 运动矢量导出方法 2

首先，如果编码单元子类型是‘B_Skip_Sym’或‘B_Direct_2N_Sym’，或预测单元的预测单元类型是‘PU_Direct_NxN’，按以下步骤导出前向运动矢量MvE0和后向运动矢量MvE1：

a) 第一步，
   1) 如果后向参考图像中与当前预测单元的左上角亮度样本位置对应的亮度样本所在的运动信息存储单元存储的参考帧索引为-1，则当前预测单元的前后向参考图像均为缺省参考图像，即前向的参考图像为参考索引值等于 1 的图像，后向的参考图像为参考索引值等于 0 的图像。以当前预测单元所在编码单元的尺寸和位置作为当前预测单元的尺寸和位置，然后将根据 9.5.8.2 得到的前向运动矢量预测值和后向运动矢量预测值分别作为当前预测单元的前向运动矢量 MvE0 和后向运动矢量 MvE1，结束运动矢量导出过程。
   2) 否则，
      ——当前预测单元的前后向参考图像均为缺省参考图像，即前向的参考图像为参考索引值等于 1 的图像，后向的参考图像为参考索引值等于 0 的图像。其前后向参考图像的距离索引分别记为 DistanceIndexFw 和 DistanceIndexBw；当前预测单元的前后向 BlockDistance 分别记为 BlockDistanceFw 和 BlockDistanceBw。
      ——在后向参考图像中与当前预测单元的左上角亮度样本位置对应的亮度样本所在的运动信息存储单元的前向运动矢量记为 mvRef(mvRef_x,mvRef_y)，该运动信息存储单元所在图像的距离索引记为 DistanceIndexCol，该运动矢量指向的参考单元所在图像的距离索引记为 DistanceIndexRef。

b) 第二步，

$$BlockDistanceRef = (DistanceIndexCol - DistanceIndexRef + 512)\ \%\ 512$$

$$BlockDistanceFw = (DistanceIndexCur - DistanceIndexFw + 512)\ \%\ 512$$

$$BlockDistanceBw = (DistanceIndexBw - DistanceIndexCur + 512)\ \%\ 512$$

   其中 DistanceIndexCur 是当前预测单元所在图像的距离索引。

c) 第三步，
   1) 计算当前预测单元的前向运动矢量 mvE0(mvE0_x, mvE0_y)：
      ——如果 mvRef_x 小于 0：
      ——$MvE0\_x = Clip3(-32768, 32767, -(((16384 / BlockDistanceRef) \times (1 - mvRef\_x \times BlockDistanceFw) - 1) >> 14))$
      ——如果 mvRef_x 大于或等于 0：
      ——$mvE0\_x = Clip3(-32768, 32767, ((16384 / BlockDistanceRef) \times (1 + mvRef\_x \times BlockDistanceFw) - 1) >> 14)$
      ——如果 mvRef_y+delta1 小于 0：
      ——$MvE0\_y = Clip3(-32768, 32767, -(((16384 / BlockDistanceRef) \times (1 - (mvRef\_y + delta1) \times BlockDistanceFw) - 1) >> 14) - delta2)$
      ——如果 mvRef_y+delta1 大于或等于 0：
      ——$mvE0\_y = Clip3(-32768, 32767, (((16384 / BlockDistanceRef) \times (1 + (mvRef\_y+delta1) \times BlockDistanceFw) - 1) >> 14) - delta2)$
      ——其中 delta1、delta2 的导出方法见 9.5.8.2， 此时 9.5.8.2 中的 mvX 为 mvRef，MVX 为 mvE0。
   2) 计算当前预测单元的后向运动矢量 mvE1(mvE1_x, mvE1_y)：
      ——如果 mvRef_x 小于 0：

——mvE1_x = Clip3(-32768, 32767, ((16384 / BlockDistanceRef) × (1 - mvRef_x × BlockDistanceBw) - 1) >> 14)

——如果 mvRef_x 大于或等于 0：

——mvE1_x = Clip3(-32768, 32767, -(((16384 / BlockDistanceRef) × (1 + mvRef_x × BlockDistanceBw) - 1) >> 14))

——如果 mvRef_y+delta1 小于 0：

——mvE1_y = Clip3(-32768, 32767, (((16384 / BlockDistanceRef) × (1 - (mvRef_y+delta1) × BlockDistanceBw) - 1) >> 14) - delta2)

——如果 mvRef_y+delta1 大于或等于 0：

——mvE1_y = Clip3(-32768, 32767, -(((16384 / BlockDistanceRef) × (1 + (mvRef_y+delta1) × BlockDistanceBw) - 1) >> 14) - delta2)

——其中 delta1、delta2 的导出方法见 9.5.8.2， 此时 9.5.8.2 中的 mvX 为 mvRef，MVX 为 mvE1。

其次，如果编码单元子类型不是‘B_Skip_Sym’或‘B_Direct_2N_Sym’，并且预测单元类型不是‘PU_Direct_NxN’，按以下方法导出运动矢量：

a) 如果当前预测单元的预测模式是双向，按以下方法导出前向运动矢量 MvE0 和后向运动矢量 MvE1：
   1) 如果当前预测单元的亮度预测块的相邻亮度预测块 F、G、C、A、B、D 六者中为双向预测块的个数大于或等于 1，则按 F、G、C、A、B、D 的顺序依次扫描相邻亮度预测块得到第一个扫描到的双向预测块,将该预测块所在预测单元的前向运动矢量和后向运动矢量分别作为当前预测单元的前向运动矢量 MvE0 和后向运动矢量 MvE1；
   2) 否则，如果当前预测单元的亮度预测块的相邻亮度预测块 F、G、C、A、B、D 六者中为单前向预测块的个数大于或等于 1，且为后向预测块的个数大于或等于 1，则按 F、G、C、A、B、D 的顺序依次扫描相邻亮度预测块得到第一个扫描到的单前向预测块和第一个扫描到的后向预测块,将单前向预测块所在预测单元的前向运动矢量和后向预测块所在预测单元的后向运动矢量分别作为当前预测单元的前向运动矢量 MvE0 和后向运动矢量 MvE1；
   3) 否则，当前预测单元的前向运动矢量 MvE0 和后向运动矢量 MvE1 均为零矢量。
b) 否则，如果当前预测单元的预测模式是对称，按以下方法导出前向运动矢量 MvE0 和后向运动矢量 MvE1：
   1) 如果当前预测单元的亮度预测块的相邻亮度预测块 F、G、C、A、B、D 六者中为对称预测块的个数大于或等于 1，则按 F、G、C、A、B、D 的顺序依次扫描相邻亮度预测块得到第一个扫描到的对称预测块,将该预测块所在预测单元的前向运动矢量和后向运动矢量分别作为当前预测单元的前向运动矢量 MvE0 和后向运动矢量 MvE1；
   2) 否则，如果当前预测单元的亮度预测块的相邻亮度预测块 F、G、C、A、B、D 六者中为双向预测块的个数大于或等于 2，则按 D、B、A、C、G、F 的顺序依次扫描相邻亮度预测块得到第一个扫描到的双向预测块,将该预测块所在预测单元的前向运动矢量和后向运动矢量分别作为当前预测单元的前向运动矢量 MvE0 和后向运动矢量 MvE1；
   3) 否则，如果当前预测单元的亮度预测块的相邻亮度预测块 F、G、C、A、B、D 六者中为后向预测块的个数大于或等于 1，则按 F、G、C、A、B、D 的顺序依次扫描相邻亮度预测块得到第一个扫描到的后向预测块,将该预测块所在预测单元的后向运动矢量作为当前预测单元的后向运动矢量 MvE1，并将此后向运动矢量取反后作为当前预测单元的前向运动矢量 MvE0；

4) 否则，如果当前预测单元的亮度预测块的相邻亮度预测块 F、G、C、A、B、D 六者中为单前向预测块的个数大于或等于 1，则按 F、G、C、A、B、D 的顺序依次扫描相邻亮度预测块得到第一个扫描到的单前向预测块，将该预测块所在预测单元的前向运动矢量作为当前预测单元的前向运动矢量 MvE0，并将此前向运动矢量取反后作为当前预测单元的后向运动矢量 MvE1；

5) 否则，当前预测单元的前向运动矢量 MvE0 和后向运动矢量 MvE1 均为零矢量。

c) 否则，如果当前预测单元的预测模式是后向，按以下方法导出后向运动矢量 MvE：

1) 如果当前预测单元的亮度预测块的相邻亮度预测块 F、G、C、A、B、D 六者中为后向预测块的个数大于或等于 1，则按 F、G、C、A、B、D 的顺序依次扫描相邻亮度预测块得到第一个扫描到的后向预测块，将该预测块所在预测单元的后向运动矢量作为当前预测单元的后向运动矢量 MvE；

2) 否则，如果当前预测单元的亮度预测块的相邻亮度预测块 F、G、C、A、B、D 六者中为双向预测块的个数大于或等于 1，则按 D、B、A、C、G、F 的顺序依次扫描相邻亮度预测块得到第一个扫描到的双向预测块，将该预测块所在预测单元的后向运动矢量作为当前预测单元的后向运动矢量 MvE；

3) 否则，当前预测单元的后向运动矢量 MvE 为零矢量。

d) 否则，如果当前预测单元的预测模式是单前向，按以下方法导出前向运动矢量 MvE：

1) 如果当前预测单元的亮度预测块的相邻亮度预测块 F、G、C、A、B、D 六者中为单前向预测块的个数大于或等于 1，则按 F、G、C、A、B、D 的顺序依次扫描相邻亮度预测块得到第一个扫描到的单前向预测块，将该预测块所在预测单元的前向运动矢量作为当前预测单元的前向运动矢量 MvE；

2) 否则，如果当前预测单元的亮度预测块的相邻亮度预测块 F、G、C、A、B、D 六者中为双向预测块的个数大于或等于 1，则按 D、B、A、C、G、F 的顺序依次扫描相邻亮度预测块得到第一个扫描到的双向预测块，将该预测块所在预测单元的前向运动矢量作为当前预测单元的前向运动矢量 MvE；

3) 否则，当前预测单元的前向运动矢量 MvE 为零矢量。

其中，对称预测块是指其所在预测单元预测模式为对称的亮度预测块，双向预测块是指其所在预测单元预测模式为双向的亮度预测块，单前向预测块是指其所在预测单元预测模式为单前向的亮度预测块，后向预测块是指其所在预测单元预测模式为后向的亮度预测块。

9.6 变换块解码

9.6.1 概述

本条定义 $M_1 \times M_2$ 变换块的解码过程。变换块的量化系数经过反量化(见9.6.3)和反变换(见9.6.4)，得到残差样本矩阵。

9.6.2 反量化

本条定义将 $M_1 \times M_2$ 二维量化系数矩阵QuantCoeffMatrix转换为 $M_1 \times M_2$ 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QuantCoeffMatrix的元素的取值范围应为 -2^{15}～$2^{15}-1$。

当IntraModeIdx的值为1且IsChroma为0时，如果编码单元类型为‘I_2N’或‘I_N’，对QuantCoeffMatrix进行转置操作（即交换QuantCoeffMatrix[i][j]和QuantCoeffMatrix[j][i]的值，其中 $i=0$～M_1-1, $j=0$～M_2-1）。

二维变换系数矩阵CoeffMatrix由下式得到：

$$\text{CoeffMatrix}[i][j] = \text{Clip3}(-32768,\ 32767,\ ((((\text{QuantCoeffMatrix}[i][j] \times \text{WeightQuantMatrix}_{M1xM2}[i][j]) >> \text{WqmShift}) \times (\text{DequantTable}(QP_X))>>4) + 2^{((\text{ShiftTable}(QPx)+\text{shift1})-1)}) >> (\text{ShiftTable}(QP_X) + \text{shift1}))),\ i=0\sim M_1-1, j=0\sim M_2-1$$

其中shift1等于m + BitDepth – 14（BitDepth是编码样本精度。如果UpSampleEnable的值为1，$m=Log(M_1 \times M_2)/2+1$；否则$m=Log(M_1 \times M_2)/2$）。量化参数$QP_X$（X为Y、Cb或Cr）与DequantTable和ShiftTable的关系见表100。

表100　QP_X与DequantTable和ShiftTable的关系

QP_X的值	DequantTable(QP_X)的值	ShiftTable(QP_X)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
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

表 100（续）

QPX 的值	DequantTable(QPX)的值	ShiftTable(QPX)的值
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
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
64	32768	6
65	36061	6
66	38968	6

表 100（续）

QPX 的值	DequantTable(QPX)的值	ShiftTable(QPX)的值
67	42495	6
68	46341	6
69	50535	6
70	55437	6
71	60424	6
72	32932	5
73	35734	5
74	38968	5
75	42495	5
76	46177	5
77	50535	5
78	55109	5
79	59933	5

9.6.3 反变换

本条定义将 $M_1 \times M_2$ 变换系数矩阵CoeffMatrix转换为残差样本矩阵ResidueMatrix的过程，步骤如下：

a) 如果当前变换块是亮度帧内预测残差块，M_1 或 M_2 的值大于 4 且 SecondaryTransformEnableFlag 的值等于 1，对 CoeffMatrix 执行以下操作：

1) 由变换系数矩阵得到 4×4 矩阵 C：

$$c_{ij} = \mathrm{coeff}_{ij}\ ,\quad i=0\sim3, j=0\sim3$$

其中 c_{ij} 是矩阵 C 的元素，$coeff_{ij}$ 是变换系数矩阵的元素。

2) 如果 IntraLumaPredMode 的值为 0～2 或 13～32，且 9.7.2 中坐标为(x0-1, y0+j-1)（j=1～N）的参考样本“可用”，则：

$$c_{ij} = \mathrm{Clip3}(-32768,\ 32767,\ (p_{ij} + 2^6) >> 7),\quad i=0\sim3, j=0\sim3$$

其中 p_{ij} 是 4×4 矩阵 P 的元素。矩阵 P 的计算如下：

$$P = C \times S_4$$

其中 S_4 是 4×4 反变换矩阵。

3) 如果 IntraLumaPredMode 的值为 0～23，且 9.7.2 中坐标为(x0+i-1, y0-1)（i=1～M）的参考样本“可用”，则：

$$c_{ij} = \mathrm{Clip3}(-32768,\ 32767,\ (q_{ij} + 2^6) >> 7),\quad i=0\sim3, j=0\sim3$$

其中 q_{ij} 是 4×4 矩阵 Q 的元素。矩阵 Q 的计算如下：

$$Q = S_4^{\mathrm{T}} \times C$$

其中 S_4^{T} 是 S_4 的转置。

4) 根据矩阵 C 修改变换系数矩阵的元素的值：

$$\mathrm{coeff}_{ij} = c_{ij}\ ,\quad i=0\sim3, j=0\sim3$$

b) 对变换系数矩阵进行如下垂直反变换，得到矩阵 K：

1) 如果当前变换块是亮度帧内预测残差块，M_1 和 M_2 的值均等于 4 且 SecondaryTransformEnableFlag 的值等于 1，则：

k_{ij} = Clip3(-32768, 32767, (v_{ij} + 2^4)>>5)，i=0～M_1-1, j=0～M_2-1

其中 v_{ij}是矩阵 V 的元素，k_{ij}是矩阵 K 的元素。矩阵 V 的计算如下：

$$V = D_4^T \times \text{CoeffMatrix}$$

其中 D_4^T 是反变换矩阵 D_4 的转置矩阵。

2) 否则：

k_{ij} = Clip3(-32768, 32767, (v_{ij} + 2^4)>>5)，i=0～M_1-1, j=0～M_2-1

其中 v_{ij}是矩阵 V 的元素，k_{ij}是矩阵 K 的元素。矩阵 V 的计算如下：

$$V = T_{M2}^T \times \text{CoeffMatrix}$$

其中 T_{M2}^T 是 T_{M2} 的转置矩阵，T_{M2} 是 $M_2 \times M_2$ 反变换矩阵。

c) 对矩阵 K 进行如下水平反变换，得到矩阵 H：

1) 如果当前变换块是亮度帧内预测残差块，M_1 和 M_2 的值均等于 4 且 SecondaryTransformEnableFlag 的值等于 1，则：

h_{ij} = Clip3(-MaxValue-1, MaxValue, (w_{ij} + $2^{\text{shift1}-1}$)>>shift1)，i=0～M_1-1, j=0～M_2-1

其中 $w_{i,j}$是矩阵 W 的元素，h_{ij}是矩阵 H 的元素。MaxValue 的值为(1 << BitDepth) - 1，shift1 等于 22 - BitDepth。矩阵 W 的计算如下：

$$W = K \times D_4$$

其中 D_4 是反变换矩阵。

2) 否则：

h_{ij} = Clip3(-MaxValue-1, MaxValue, (w_{ij} + $2^{\text{shift1}-1}$)>>shift1)　i=0～M_1-1, j=0～M_2-1

其中 w_{ij}是矩阵 W 的元素，h_{ij}是矩阵 H 的元素。如果 UpSampleEnableFlag 的值为 1，则 MaxValue 的值为(1 << (BitDepth + 1)) - 1， shift1 等于 19 – BitDepth；否则 MaxValue 的值为(1 << BitDepth) - 1，shift1 等于 20 - BitDepth。矩阵 W 的计算如下：

$$W = K \times T_{M1}$$

其中 T_{M1} 是 $M_1 \times M_1$ 反变换矩阵。

d) 如果 UpSampleEnableFlag 的值为 0，则把矩阵 H 直接作为残差样值矩阵 ResidueMatrix，结束反变换操作；否则 ResidueMatrix 的元素 rij 计算如下：

$r_{2i,j} = g_{i,j}$　　i=0～M_1-1, j=0～2M_2-1

$r_{2i+1,j}$ = ($r_{2i,j}$ + $r_{2i+2,j}$) >> 1　　i=0～M_1-2, j=0～2M_2-1

$r_{i,j} = r_{i-1,j}$　　i=2M_1-1, j=0～2M_2-1

其中 $M_1 \times 2M_2$ 的矩阵 G 的元素 g_{ij} 计算如下：

$g_{i,2j} = h_{i,j}$ >> 1　　i=0～M_1-1, j=0～M_2-1

$g_{i,2j+1}$ = ($g_{i,2j}$ + $g_{i,2j+2}$) >> 1　　i=0～M_1-1, j=0～M_2-2

$g_{i,j} = g_{i,j-1}$　　i=0～M_1-1, j=2M_2-1

反变换矩阵如下：

$$S_4 = \begin{bmatrix} 123 & -35 & -8 & -3 \\ -32 & -120 & 30 & 10 \\ 14 & 25 & 123 & -22 \\ 8 & 13 & 19 & 126 \end{bmatrix}$$

$$D_4=\begin{bmatrix}34 & 58 & 72 & 81\\ 77 & 69 & -7 & -75\\ 79 & -33 & -75 & 58\\ 55 & -84 & 73 & -28\end{bmatrix}$$

$$T_4=\begin{bmatrix}32 & 32 & 32 & 32\\ 42 & 17 & -17 & \text{-}42\\ 32 & -32 & -32 & 32\\ 17 & -42 & 42 & -17\end{bmatrix}$$

$$T_8=\begin{bmatrix}32 & 32 & 32 & 32 & 32 & 32 & 32 & 32\\ 44 & 38 & 25 & 9 & -9 & -25 & -38 & -44\\ 42 & 17 & -17 & -42 & -42 & -17 & 17 & 42\\ 38 & -9 & -44 & -25 & 25 & 44 & 9 & -38\\ 32 & -32 & -32 & 32 & 32 & -32 & -32 & 32\\ 25 & -44 & 9 & 38 & -38 & -9 & 44 & -25\\ 17 & -42 & 42 & -17 & -17 & 42 & -42 & 17\\ 9 & -25 & 38 & -44 & 44 & -38 & 25 & -9\end{bmatrix}$$

$$T_{16}=\begin{bmatrix}32 & 32 & 32 & 32 & 32 & 32 & 32 & 32 & 32 & 32 & 32 & 32 & 32 & 32 & 32 & 32\\ 45 & 43 & 40 & 35 & 29 & 21 & 13 & 4 & -4 & -13 & -21 & -29 & -35 & -40 & -43 & -45\\ 44 & 38 & 25 & 9 & -9 & -25 & -38 & -44 & -44 & -38 & -25 & -9 & 9 & 25 & 38 & 44\\ 43 & 29 & 4 & -21 & -40 & -45 & -35 & -13 & 13 & 35 & 45 & 40 & 21 & -4 & -29 & -43\\ 42 & 17 & -17 & -42 & -42 & -17 & 17 & 42 & 42 & 17 & -17 & -42 & -42 & -17 & 17 & 42\\ 40 & 4 & -35 & -43 & -13 & 29 & 45 & 21 & -21 & -45 & -29 & 13 & 43 & 35 & -4 & -40\\ 38 & -9 & -44 & -25 & 25 & 44 & 9 & -38 & -38 & 9 & 44 & 25 & -25 & -44 & -9 & 38\\ 35 & -21 & -43 & 4 & 45 & 13 & -40 & -29 & 29 & 40 & -13 & -45 & -4 & 43 & 21 & -35\\ 32 & -32 & -32 & 32 & 32 & -32 & -32 & 32 & 32 & -32 & -32 & 32 & 32 & -32 & -32 & 32\\ 29 & -40 & -13 & 45 & -4 & -43 & 21 & 35 & -35 & -21 & 43 & 4 & -45 & 13 & 40 & -29\\ 25 & -44 & 9 & 38 & -38 & -9 & 44 & -25 & -25 & 44 & -9 & -38 & 38 & 9 & -44 & 25\\ 21 & -45 & 29 & 13 & -43 & 35 & 4 & -40 & 40 & -4 & -35 & 43 & -13 & -29 & 45 & -21\\ 17 & -42 & 42 & -17 & -17 & 42 & -42 & 17 & 17 & -42 & 42 & -17 & -17 & 42 & -42 & 17\\ 13 & -35 & 45 & -40 & 21 & 4 & -29 & 43 & -43 & 29 & -4 & -21 & 40 & -45 & 35 & -13\\ 9 & -25 & 38 & -44 & 44 & -38 & 25 & -9 & -9 & 25 & -38 & 44 & -44 & 38 & -25 & 9\\ 4 & -13 & 21 & -29 & 35 & -40 & 43 & -45 & 45 & -43 & 40 & -35 & 29 & -21 & 13 & -4\end{bmatrix}$$

$$T_{32}=\begin{bmatrix}T_{L16\times 32} & T_{R16\times 32}\end{bmatrix}$$

$$
T_{L16\times32}=\begin{bmatrix}
32 & 32 & 32 & 32 & 32 & 32 & 32 & 32 & 32 & 32 & 32 & 32 & 32 & 32 & 32 & 32 \\
45 & 45 & 44 & 43 & 41 & 39 & 36 & 34 & 30 & 27 & 23 & 19 & 15 & 11 & 7 & 2 \\
45 & 43 & 40 & 35 & 29 & 21 & 13 & 4 & -4 & -13 & -21 & -29 & -35 & -40 & -43 & -45 \\
45 & 41 & 34 & 23 & 11 & -2 & -15 & -27 & -36 & -43 & -45 & -44 & -39 & -30 & -19 & -7 \\
44 & 38 & 25 & 9 & -9 & -25 & -38 & -44 & -44 & -38 & -25 & -9 & 9 & 25 & 38 & 44 \\
44 & 34 & 15 & -7 & -27 & -41 & -45 & -39 & -23 & -2 & 19 & 36 & 45 & 43 & 30 & 11 \\
43 & 29 & 4 & -21 & -40 & -45 & -35 & -13 & 13 & 35 & 45 & 40 & 21 & -4 & -29 & -43 \\
43 & 23 & -7 & -34 & -45 & -36 & -11 & 19 & 41 & 44 & 27 & -2 & -30 & -45 & -39 & -15 \\
42 & 17 & -17 & -42 & -42 & -17 & 17 & 42 & 42 & 17 & -17 & -42 & -42 & -17 & 17 & 42 \\
41 & 11 & -27 & -45 & -30 & 7 & 39 & 43 & 15 & -23 & -45 & -34 & 2 & 36 & 44 & 19 \\
40 & 4 & -35 & -43 & -13 & 29 & 45 & 21 & -21 & -45 & -29 & 13 & 43 & 35 & -4 & -40 \\
39 & -2 & -41 & -36 & 7 & 43 & 34 & -11 & -44 & -30 & 15 & 45 & 27 & -19 & -45 & -23 \\
38 & -9 & -44 & -25 & 25 & 44 & 9 & -38 & -38 & 9 & 44 & 25 & -25 & -44 & -9 & 38 \\
36 & -15 & -45 & -11 & 39 & 34 & -19 & -45 & -7 & 41 & 30 & -23 & -44 & -2 & 43 & 27 \\
35 & -21 & -43 & 4 & 45 & 13 & -40 & -29 & 29 & 40 & -13 & -45 & -4 & 43 & 21 & -35 \\
34 & -27 & -39 & 19 & 43 & -11 & -45 & 2 & 45 & 7 & -44 & -15 & 41 & 23 & -36 & -30 \\
32 & -32 & -32 & 32 & 32 & -32 & -32 & 32 & 32 & -32 & -32 & 32 & 32 & -32 & -32 & 32 \\
30 & -36 & -23 & 41 & 15 & -44 & -7 & 45 & -2 & -45 & 11 & 43 & -19 & -39 & 27 & 34 \\
29 & -40 & -13 & 45 & -4 & -43 & 21 & 35 & -35 & -21 & 43 & 4 & -45 & 13 & 40 & -29 \\
27 & -43 & -2 & 44 & -23 & -30 & 41 & 7 & -45 & 19 & 34 & -39 & -11 & 45 & -15 & -36 \\
25 & -44 & 9 & 38 & -38 & -9 & 44 & -25 & -25 & 44 & -9 & -38 & 38 & 9 & -44 & 25 \\
23 & -45 & 19 & 27 & -45 & 15 & 30 & -44 & 11 & 34 & -43 & 7 & 36 & -41 & 2 & 39 \\
21 & -45 & 29 & 13 & -43 & 35 & 4 & -40 & 40 & -4 & -35 & 43 & -13 & -29 & 45 & -21 \\
19 & -44 & 36 & -2 & -34 & 45 & -23 & -15 & 43 & -39 & 7 & 30 & -45 & 27 & 11 & -41 \\
17 & -42 & 42 & -17 & -17 & 42 & -42 & 17 & 17 & -42 & 42 & -17 & -17 & 42 & -42 & 17 \\
15 & -39 & 45 & -30 & 2 & 27 & -44 & 41 & -19 & -11 & 36 & -45 & 34 & -7 & -23 & 43 \\
13 & -35 & 45 & -40 & 21 & 4 & -29 & 43 & -43 & 29 & -4 & -21 & 40 & -45 & 35 & -13 \\
11 & -30 & 43 & -45 & 36 & -19 & -2 & 23 & -39 & 45 & -41 & 27 & -7 & -15 & 34 & -44 \\
9 & -25 & 38 & -44 & 44 & -38 & 25 & -9 & -9 & 25 & -38 & 44 & -44 & 38 & -25 & 9 \\
7 & -19 & 30 & -39 & 44 & -45 & 43 & -36 & 27 & -15 & 2 & 11 & -23 & 34 & -41 & 45 \\
4 & -13 & 21 & -29 & 35 & -40 & 43 & -45 & 45 & -43 & 40 & -35 & 29 & -21 & 13 & -4 \\
2 & -7 & 11 & -15 & 19 & -23 & 27 & -30 & 34 & -36 & 39 & -41 & 43 & -44 & 45 & -45
\end{bmatrix}
$$

$$
T_{R16\times32}=\begin{bmatrix}
32 & 32 & 32 & 32 & 32 & 32 & 32 & 32 & 32 & 32 & 32 & 32 & 32 & 32 & 32 & 32 \\
-2 & -7 & -11 & -15 & -19 & -23 & -27 & -30 & -34 & -36 & -39 & -41 & -43 & -44 & -45 & -45 \\
-45 & -43 & -40 & -35 & -29 & -21 & -13 & -4 & 4 & 13 & 21 & 29 & 35 & 40 & 43 & 45 \\
7 & 19 & 30 & 39 & 44 & 45 & 43 & 36 & 27 & 15 & 2 & -11 & -23 & -34 & -41 & -45 \\
44 & 38 & 25 & 9 & -9 & -25 & -38 & -44 & -44 & -38 & -25 & -9 & 9 & 25 & 38 & 44 \\
-11 & -30 & -43 & -45 & -36 & -19 & 2 & 23 & 39 & 45 & 41 & 27 & 7 & -15 & -34 & -44 \\
-43 & -29 & -4 & 21 & 40 & 45 & 35 & 13 & -13 & -35 & -45 & -40 & -21 & 4 & 29 & 43 \\
15 & 39 & 45 & 30 & 2 & -27 & -44 & -41 & -19 & 11 & 36 & 45 & 34 & 7 & -23 & -43 \\
42 & 17 & -17 & -42 & -42 & -17 & 17 & 42 & 42 & 17 & -17 & -42 & -42 & -17 & 17 & 42 \\
-19 & -44 & -36 & -2 & 34 & 45 & 23 & -15 & -43 & -39 & -7 & 30 & 45 & 27 & -11 & -41 \\
-40 & -4 & 35 & 43 & 13 & -29 & -45 & -21 & 21 & 45 & 29 & -13 & -43 & -35 & 4 & 40 \\
23 & 45 & 19 & -27 & -45 & -15 & 30 & 44 & 11 & -34 & -43 & -7 & 36 & 41 & 2 & -39 \\
38 & -9 & -44 & -25 & 25 & 44 & 9 & -38 & -38 & 9 & 44 & 25 & -25 & -44 & -9 & 38 \\
-27 & -43 & 2 & 44 & 23 & -30 & -41 & 7 & 45 & 19 & -34 & -39 & 11 & 45 & 15 & -36 \\
-35 & 21 & 43 & -4 & -45 & -13 & 40 & 29 & -29 & -40 & 13 & 45 & 4 & -43 & -21 & 35 \\
30 & 36 & -23 & -41 & 15 & 44 & -7 & -45 & -2 & 45 & 11 & -43 & -19 & 39 & 27 & -34 \\
32 & -32 & -32 & 32 & 32 & -32 & -32 & 32 & 32 & -32 & -32 & 32 & 32 & -32 & -32 & 32 \\
-34 & -27 & 39 & 19 & -43 & -11 & 45 & 2 & -45 & 7 & 44 & -15 & -41 & 23 & 36 & -30 \\
-29 & 40 & 13 & -45 & 4 & 43 & -21 & -35 & 35 & 21 & -43 & -4 & 45 & -13 & -40 & 29 \\
36 & 15 & -45 & 11 & 39 & -34 & -19 & 45 & -7 & -41 & 30 & 23 & -44 & 2 & 43 & -27 \\
25 & -44 & 9 & 38 & -38 & -9 & 44 & -25 & -25 & 44 & -9 & -38 & 38 & 9 & -44 & 25 \\
-39 & -2 & 41 & -36 & -7 & 43 & -34 & -11 & 44 & -30 & -15 & 45 & -27 & -19 & 45 & -23 \\
-21 & 45 & -29 & -13 & 43 & -35 & -4 & 40 & -40 & 4 & 35 & -43 & 13 & 29 & -45 & 21 \\
41 & -11 & -27 & 45 & -30 & -7 & 39 & -43 & 15 & 23 & -45 & 34 & 2 & -36 & 44 & -19 \\
17 & -42 & 42 & -17 & -17 & 42 & -42 & 17 & 17 & -42 & 42 & -17 & -17 & 42 & -42 & 17 \\
-43 & 23 & 7 & -34 & 45 & -36 & 11 & 19 & -41 & 44 & -27 & -2 & 30 & -45 & 39 & -15 \\
-13 & 35 & -45 & 40 & -21 & -4 & 29 & -43 & 43 & -29 & 4 & 21 & -40 & 45 & -35 & 13 \\
44 & -34 & 15 & 7 & -27 & 41 & -45 & 39 & -23 & 2 & 19 & -36 & 45 & -43 & 30 & -11 \\
9 & -25 & 38 & -44 & 44 & -38 & 25 & -9 & -9 & 25 & -38 & 44 & -44 & 38 & -25 & 9 \\
-45 & 41 & -34 & 23 & -11 & -2 & 15 & -27 & 36 & -43 & 45 & -44 & 39 & -30 & 19 & -7 \\
-4 & 13 & -21 & 29 & -35 & 40 & -43 & 45 & -45 & 43 & -40 & 35 & -29 & 21 & -13 & 4 \\
45 & -45 & 44 & -43 & 41 & -39 & 36 & -34 & 30 & -27 & 23 & -19 & 15 & -11 & 7 & -2
\end{bmatrix}
$$

9.7 帧内预测

9.7.1 概述

本条定义M×N亮度块和L×L色度块的帧内预测过程。应分别使用9.7.2和9.7.3定义的方法获得当前块的亮度和色度参考样本，并分别使用9.7.4和9.7.5定义的方法进行亮度和色度帧内预测，从而分别得到M×N亮度块和L×L色度块的预测样本矩阵predMatrix。

对于亮度块，当前块上边的参考样本记为r[i]，左边的参考样本记为c[j]，其中r[0]等于c[0]，如果i大于2M，则r[i]=r[2M]，如果j大于2N，则c[j]=c[2N]。

对于色度块，当前块上边的参考样本记为row[i]，左边的参考样本记为col[i]，其中row[0]等于col[0]，如果i大于2L，则row[i]=row[2L]，col[i]=col[2L]。

9.7.2 亮度参考样本的获得

用I表示当前块所在的图像的补偿后样本的亮度样值矩阵。

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) 初始化 r[i]、c[j]为 $2^{BitDepth-1}$（i=0～2M, j=0～2N；BitDepth 是编码样本精度）；

b) 如果坐标为(x0+i-1, y0-1)（i=1～M）的样本均“可用”，则 r[i]等于 I[x0+i-1][y0-1]，r[i]“可用”；否则 r[i]“不可用”；
c) 如果坐标为(x0+i-1, y0-1)（i=M+1～2M）的样本均“可用”，则 r[i]等于 I[x0+i-1][y0-1]，r[i]“可用”；否则 r[i]等于 r[M]，r[i]是否“可用”由 r[M]是否“可用”决定；
d) 如果坐标为(x0-1, y0+j-1)（j=1～N）的样本均“可用”，则 c[j]等于 I[x0-1][y0+j-1]，c[j]“可用”；否则 c[j]“不可用”；
e) 如果坐标为(x0-1, y0+j-1)（j=N+1～2N）的样本均“可用”，则 c[j]等于 I[x0-1][y0+j-1]，c[j]“可用”；否则 c[j]等于 c[N]，c[j]是否“可用”由 c[N]是否“可用”决定；
f) 如果坐标为(x0-1, y0-1)的样本“可用”，则 r[0]等于 I[x0-1][y0-1]，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”。

9.7.3 色度参考样本的获得

用I表示当前块所在的图像的补偿后样本的色度样值矩阵。

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) 初始化 row[i]、col[i]为 $2^{BitDepth-1}$（i=0～2L；BitDepth 是编码样本精度）；
b) 如果坐标为(x0+i-1, y0-1)（i=1～L）的样本均“可用”，则 row[i]等于 I[x0+i-1][y0-1]，row[i]“可用”；否则 row[i]“不可用”；
c) 如果坐标为(x0+i-1, y0-1)（i=L+1～2L）的样本均“可用”，则 row[i]等于 I[x0+i-1][y0-1]，row[i]“可用”；否则 row[i]等于 row[L]，row[i]是否“可用”由 row[L]是否“可用”决定；
d) 如果坐标为(x0-1, y0+i-1)（i=1～L）的样本均“可用”，则 col[i]等于 I[x0-1][y0+i-1]，col[i]“可用”；否则 col[i]“不可用”；
e) 如果坐标为(x0-1, y0+i-1)（i=L+1～2L）的样本均“可用”，则 col[i]等于 I[x0-1][y0+i-1]，col[i]“可用”；否则 col[i]等于 c[L]，col[i]是否“可用”由 col[L]是否“可用”决定；
f) 如果坐标为(x0-1, y0-1)的样本“可用”，则 row[0]等于 I[x0-1][y0-1]，row[0]“可用”；否则
   1) 如果 row[1]“可用”并且 col[1]“不可用”，则 row[0]等于 row[1]，row[0]“可用”；否则
   2) 如果 col[1]“可用”并且 row[1]“不可用”，则 row[0]等于 col[1]，row[0]“可用”；否则
   3) 如果 row[1]“可用”并且 col[1]“可用”，则 row[0]等于 row[1]，row[0]“可用”；否则 row[0]“不可用”。

9.7.4 亮度预测块帧内预测

根据IntraLumaPredMode决定亮度块帧内预测方法。

a) IntraLumaPredMode 等于 12（Intra_Luma_Vertical 预测）
   predMatrix[x][y] = r[x+1]（x=0～M-1,y=0～N-1）
b) IntraLumaPredMode 等于 24（Intra_Luma_Horizontal 预测）
   predMatrix[x][y] = c[y+1]（x=0～M-1,y=0～N-1）
c) IntraLumaPredMode 等于 0（Intra_Luma_DC 预测）
   1) 如果 r[i]、c[j]（i=1～M,j=1～N）均“可用”，则

predMatrix[x][y]= $((\sum_{i=1}^{M} r[i] + \sum_{j=1}^{N} c[j] + ((M+N) >> 1)) \times (512/(M+N))) >> 9$ (x=0～M-1, y=0～N-1)；

2) 否则，如果 r[i]（i=1～M）“可用”，且 c[j]（j=1～N）“不可用”，则

predMatrix[x][y] = $(\sum_{i=1}^{M} r[i] + (M >> 1)) >> \mathrm{Log}(M)$ （x=0～M-1, y=0～N-1）；

3) 否则，如果 c[j]（j=1～N）“可用”，且 r[i]（i=1～M）“不可用”，则

predMatrix[x][y] = $(\sum_{j=1}^{N} c[j] + (N >> 1)) >> \mathrm{Log}(N)$ （x=0～M-1, y=0～N-1）；

4) 否则，predMatrix[x][y] = 2BitDepth-1（x=0～M-1, y=0～N-1；BitDepth 是编码样本精度）。

d) IntraLumaPredMode 等于 1（Intra_Luma_Plane 预测）

predMatrix[x][y]= Clip1((ia+(x-((M>>1)-1))×ib+(y-((N>>1)-1))×ic+16)>>5) （ x=0 ～M-1, y=0～N-1）。

其中，ia = (r[M]+c[N]) << 4,

ib = ((ih<<5) × imh + (1<<(ish-1))) >> ish,

ic = ((iv<<5) × imv + (1<<(isv-1))) >> isv,

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

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

imh = ibMult[Log(M)-2],

ish = ibShift[Log(M)-2],

imv = ibMult[Log(N)-2],

isv = ibShift[Log(N)-2],

$$ih = \sum_{i=0}^{(M>>1)-1} (i+1) \times (r[(M>>1)+1+i] - r[(M>>1)-1-i]),$$

$$iv = \sum_{i=0}^{(N>>1)-1} (i+1) \times (c[(N>>1)+1+i] - c[(N>>1)-1-i])。$$

e) IntraLumaPredMode 等于 2（Intra_Luma_Bilinear 预测）

predMatrix[x][y]=Clip1(((((ia-c[y+1])×(x+1))<<Log(N))+(((ib-r[x+1])×(y+1))<<Log(M))+((r[x+1]+c[y+1])<<(Log(M)+Log(N)))+((ic<<1)-ia-ib)×x×y+(1<<(Log(M)+Log(N))))>>(Log(M)+Log(N)+1))

其中，ia = r[M]，ib = c[N]，如果 M 等于 N，则 ic=(ia+ib+1)>>1；否则 ic=(((ia<<Log(M))+(ib<<Log(N)))×13+(1<<(ishift+5)))>>(ishift+6)，其中 ishift= Log(Min(M,N))。

f) IntraLumaPredMode 不等于 0、1、2、12、24（Intra_Luma_Angular 预测）

初始化 r[-1]=c[1]，r[-2]=c[2]，c[-1]=r[1]，c[-2]=r[2]。根据 IntraLumaPredMode 的值查表 101 得到 dx、dy、xyAxis、xySign、imx、isx、imy 和 isy。其中：

dx = dirDx[IntraLumaPredMode]

dy = dirDy[IntraLumaPredMode]
xyAxis = dirXYFlag[IntraLumaPredMode]
xySign = dirXYSign[IntraLumaPredMode]
imx = divDxy[IntraLumaPredMode][0]
isx = divDxy[IntraLumaPredMode][1]
imy = divDyx[IntraLumaPredMode][0]
isy = divDyx[IntraLumaPredMode][1]

然后依次执行以下步骤：

1) 如果 xySign小于0，则
——如果 xyAxis 等于 0（参考上边像素），offset = (((y+1)×imx×32)>>isx)-(((y+1)×imx)>>isx)×32，iX=x+(((y+1)×imx)>>isx)，iY=-1；
——否则，xyAxis 等于 1（参考左边像素），offset = (((x+1)×imy×32)>>isy)-(((x+1)×imy)>>isy)×32，iX=-1，iY=y+(((x+1)×imy)>>isy)。

2) 如果xySign大于0，则offsetx=(((y+1)×imx×32)>>isx)-(((y+1)×imx)>>isx)×32，iXx=x-(((y+1)×imx)>>isx)，offsety=(((x+1)×imy×32)>>isy)-(((x+1)×imy)>>isy)×32，iYy=y-(((x+1)×imy)>>isy)：
——如果 iYy 小于或等于-1（参考上边像素），则 offset=offsetx，iX=iXx，iY=-1；
——否则（参考左边像素），offset=offsety，iX=-1，iY=iYy。

3) 如果iY等于-1（参考上边像素）：
——如果(dx×dy)小于 0，则 iXn=iX+1，iXnP2=iX+2，iXnN1=iX-1；
——否则，iXn=iX-1，iXnP2=iX-2，iXnN1=iX+1
——predMatrix[x][y] = (r[iXnN1+1]×(32-offset)+r[iX+1]×(64-offset)+ r[iXn+1]×(32+offset)+r[iXnP2+1]×offset+64)>>7（x=0～M-1,y=0～N-1）

4) 否则，如果 iX等于-1（参考左边像素）：
——如果(dx×dy)小于 0，iYn=iY+1，iYnP2=iY+2，iYnN1=iY-1；
——否则，iYn=iY-1，iYnP2=iY-2，iYnN1=iY+1
——predMatrix[x][y] = (c[iYnN1+1]×(32-offset)+c[iY+1]×(64-offset)+ c[iYn+1]×(32+offset)+c[iYnP2+1]×offset+64)>>7（x=0～M-1,y=0～N-1）

表101　角度预测模式的方向

IntraLumaPredMode 的值	dirDx[IntraLumaPredMode] 的值	dirDy[IntraLumaPredMode] 的值	dirXYFlag[IntraLumaPredMode] 的值	dirXYSign[IntraLumaPredMode] 的值	divDxy[IntraLumaPredMode] 的值	divDyx[IntraLumaPredMode] 的值
0	-	-	-	-	-	-
1	-	-	-	-	-	-
2	-	-	-	-	-	-
3	11	-4	0	-1	{11,2}	{93,8}
4	2	-1	0	-1	{2,0}	{1,1}
5	11	-8	0	-1	{11,3}	{93,7}
6	1	-1	0	-1	{1,0}	{1,0}
7	8	-11	0	-1	{93,7}	{11,3}

表 101（续）

IntraLumaPredMode 的值	dirDx[IntraLumaPredMode]的值	dirDy[IntraLumaPredMode]的值	dirXYFlag[IntraLumaPredMode]的值	dirXYSign[IntraLumaPredMode]的值	divDxy[IntraLumaPredMode]的值	divDyx[IntraLumaPredMode]的值
8	1	−2	0	−1	{1,1}	{2,0}
9	4	−11	0	−1	{93,8}	{11,2}
10	1	−4	0	−1	{1,2}	{4,0}
11	1	−8	0	−1	{1,3}	{8,0}
12	–	–	–	–	–	–
13	1	8	0	1	{1,3}	{8,0}
14	1	4	0	1	{1,2}	{4,0}
15	4	11	0	1	{93,8}	{11,2}
16	1	2	0	1	{1,1}	{2,0}
17	8	11	0	1	{93,7}	{11,3}
18	1	1	0	1	{1,0}	{1,0}
19	11	8	0	1	{11,3}	{93,7}
20	2	1	0	1	{2,0}	{1,1}
21	11	4	0	1	{11,2}	{93,8}
22	4	1	0	1	{4,0}	{1,2}
23	8	1	0	1	{8,0}	{1,3}
24	–	–	–	–	–	–
25	8	−1	1	−1	{8,0}	{1,3}
26	4	−1	1	−1	{4,0}	{1,2}
27	11	−4	1	−1	{11,2}	{93,8}
28	2	−1	1	−1	{2,0}	{1,1}
29	11	−8	1	−1	{11,3}	{93,7}
30	1	−1	1	−1	{1,0}	{1,0}
31	8	−11	1	−1	{93,7}	{11,3}
32	1	−2	1	−1	{1,1}	{2,0}

9.7.5　色度预测块帧内预测

根据IntraChromaPredMode决定色度块帧内预测方法。

a)　IntraChromaPredMode 等于 0（Intra_Chroma_Angular 预测）
   1)　如果当前编码单元中 PredBlockOrder 的值为 0 的预测块的 IntraLumaPredMode 等于 12，则 IntraChromaPredMode 的值为 3，转到步骤 d)。
   2)　如果当前编码单元中 PredBlockOrder 的值为 0 的预测块的 IntraLumaPredMode 等于 24，则 IntraChromaPredMode 的值为 2，转到步骤 c)。
   3)　如果当前编码单元中 PredBlockOrder 的值为 0 的预测块的 IntraLumaPredMode 等于 0，则 IntraChromaPredMode 的值为 1，转到步骤 b)。
   4)　如果当前编码单元中 PredBlockOrder 的值为 0 的预测块的 IntraLumaPredMode 等于 2，则 IntraChromaPredMode 的值为 4，转到步骤 e)。

5）如果当前编码单元中 PredBlockOrder 的值为 0 的预测块的 IntraLumaPredMode 等于 1，则 predMatrix[x][y] = Clip1((ia + (x-((L>>1)-1))×ib + (y-((L>>1)-1))×ic + 16) >> 5)（x,y=0～L-1）。

其中，ia = (row[L]+col[L])<<4，

ib = ((ih<<5)×im +(1<<(is-1)))>>is，

ic = ((iv<<5)×im +(1<<(is-1)))>>is，

ibMult[5]={13, 17, 5, 11, 23}，

ibShift[5]={7, 10, 11, 15, 19}，

im = ibMult[Log(L)-2]，

is = ibShift[Log(L)-2]，

$$ih = \sum_{i=0}^{(L>>1)-1}(i+1)\times(\mathrm{row}\,[(L>>1)+1+i]-\mathrm{row}\,[(L>>1)-1-i])$$，

$$iv = \sum_{i=0}^{(L>>1)-1}(i+1)\times(col[(L>>1)+1+i]-col[(L>>1)-1-i])$$。

6）如果当前编码单元中 PredBlockOrder 的值为 0 的预测块的 IntraLumaPredMode 不等于 0、1、2、12、24

初始化 row[-1]=col[1]，row[-2]=col[2]，col[-1]=row[1]，col[-2]=row[2]。根据 IntraLumaPredMode 的值查表 100 得到 dx、dy、xyAxis、xySign、imx、isx、imy 和 isy。其中：

dx = dirDx[IntraLumaPredMode]

dy = dirDy[IntraLumaPredMode]

xyAxis = dirXYFlag[IntraLumaPredMode]

xySign = dirXYsign[IntraLumaPredMode]

imx = divDxy[IntraLumaPredMode][0]

isx = divDxy[IntraLumaPredMode][1]

imy = divDyx[IntraLumaPredMode][0]

isy = divDyx[IntraLumaPredMode][1]

然后依次执行以下步骤：

——如果 xySign 小于 0，则：

- 如果 xyAxis 等于 0（参考上边像素），offset=(((y+1)×imx×32)>>isx)-((((y+1)×imx)>>isx)×32)，iX=x+(((y+1)×imx)>>isx)，iY=-1；
- 否则，xyAxis 等于 1（参考左边像素），offset=(((x+1)×imy×32)>>isy)-((((x+1)×imy)>>isy)×32)，iX=-1，iY=y+(((x+1)×imy)>>isy)。

——如果 xySign 大于 0，则 offsetx=(((y+1)×imx×32)>>isx)-((((y+1)×imx)>>isx)×32)，iXx=x-(((y+1)×imx)>>isx)，offsety=(((x+1)×imy×32)>>isy)-((((x+1)×imy)>>isy)×32)，iYy=y-(((x+1)×imy)>>isy)：

- 如果 iYy 小于或等于-1（参考上边像素），offset=offsetx，iX=iXx，iY=-1；
- 否则（参考左边像素），offset=offsety，iX=-1，iY=iYy。

——如果 iY 等于-1（参考上边像素），依次执行以下操作：

- 如果(dx×dy)小于 0，则 iXn=iX+1，iXnP2=iX+2，iXnN1=iX-1。

- 否则，iXn=iX-1，iXnP2=iX-2，iXnN1=iX+1。
- predMatrix[x][y]=(row[iXnN1+1]×(32-offset)+row[iX+1]×(64-offset)+row[iXn+1]×(32+offset)+row[iXnP2+1]×offset+64)>>7（x,y=0～L-1）。

——否则，如果 iX 等于-1（参考左边像素），依次执行以下操作：

- 如果(dx×dy)小于 0，则 iYn=iY+1，iYnP2=iY+2，iYnN1=iY-1。
- 否则，iYn=iY-1，iYnP2=iY-2，iYnN1=iY+1。
- predMatrix[x][y]=(col[iYnN1+1]×(32-offset)+col[iY+1]×(64-offset)+col[iYn+1]×(32+offset)+col[iYnP2+1]×offset+64)>>7（x,y=0～L-1）。

b) IntraChromaPredMode 等于 1（Intra_Chroma_DC 预测）

1) 如果 row[i]、col[i]（i=1～L）均“可用”,则

predMatrix[x][y]= $(\sum_{i=1}^{L}(\text{row}\,[i]+col[i])+L)>>\text{Log}(2L)$（x,y=0～L-1）；

2) 否则，如果 row[i]（i=1～L）“可用”，且 col[i]（i=1～L）“不可用”，则

predMatrix[x][y] = $(\sum_{i=1}^{L}row[i]+(L>>1))>>\text{Log}(L)$（x,y=0～L-1）；

3) 否则，如果 col[i]（i=1～L）“可用”，且 row[i]（i=1～L）“不可用”，则

predMatrix[x][y] = $(\sum_{i=1}^{L}col[i]+(L>>1))>>\text{Log}(L)$（x,y=0～L-1）；

4) 否则，predMatrix[x][y] = 2BitDepth-1（x,y=0～L-1；BitDepth 是编码样本精度）。

c) IntraChromaPredMode 等于 2（Intra_Chroma_Horizontal 预测）
predMatrix[x][y] = col[y+1]（x,y=0～L-1）。

d) IntraChromaPredMode 等于 3（Intra_Chroma_Vertical 预测）
predMatrix[x][y] = row[x+1]（x,y=0～L-1）。

e) IntraChromaPredMode 等于 4（Intra_Chroma_Bilinear 预测）
predMatrix[x][y]=Clip1(((((ia-col[y+1])×(x+1))<<Log(L))+(((ib-row[x+1])×(y+1))<<Log(L))+((row[x+1]+col[y+1])<<(Log(L)+Log(L)))+((ic<<1)-ia-ib)×x×y+(1<<(Log(L)+Log(L))))>>(Log(L)+Log(L)+1))
其中，ia=row[L]，ib=col[L]，ic=(ia+ib+1)>>1。

9.8 帧间预测

9.8.1 预测样本导出

本条定义用于帧间预测的预测样本矩阵的导出过程。所导出的预测样本矩阵的水平尺寸和垂直尺寸与当前预测单元中对应分量预测块的水平尺寸和垂直尺寸一致。

记当前预测单元亮度预测块左上角样本在当前图像的亮度样本矩阵中的位置为(xE,yE)。

如果当前预测单元的预测模式为单前向，则亮度前向预测样本矩阵predMatrixFw中的元素predMatrixFw[x][y]的值为前向参考图像的四分之一精度亮度样本矩阵中位置为(((xE+x)<<2)+MvE_x,(yE+y)<<2)+MvE_y)))的样本值，色度前向预测样本矩阵predMatrixFw中的元素predMatrixFw[x][y]的值为前向参考图像的八分之一精度色度样本矩阵中位置为

(((xE+2×x)<<2)+MvC_x,(yE+2×y)<<2)+MvC_y)))的样本值。其中MvE_x和MvE_y分别为当前预测单元前向运动矢量MvE的水平和垂直分量，MvC_x等于MvE_x，MvC_y等于MvE_y-delta3。

如果当前预测单元的预测模式为后向，则亮度后向预测样本矩阵predMatrixBw中的元素predMatrixBw[x][y]的值为后向参考图像的四分之一精度亮度样本矩阵中位置为(((xE+x)<<2)+MvE_x,(yE+y)<<2)+MvE_y)))的样本值，色度后向预测样本矩阵predMatrixBw中的元素predMatrixBw[x][y]的值为后向参考图像的八分之一精度色度样本矩阵中位置为(((xE+2×x)<<2)+MvC_x,(yE+2×y)<<2)+MvC_y)))的样本值。其中MvE_x和MvE_y分别为当前预测单元后向运动矢量MvE的水平和垂直分量，MvC_x等于MvE_x，MvC_y等于MvE_y-delta3。

如果当前预测单元的预测模式为对称或双向，则亮度前向预测样本矩阵predMatrixFw中的元素predMatrixFw[x][y]的值为前向参考图像的四分之一精度亮度样本矩阵中位置为(((xE+x)<<2)+MvE0_x,(yE+y)<<2)+MvE0_y)))的样本值,色度前向预测样本矩阵predMatrixFw中的元素predMatrixFw[x][y]的值为前向参考图像的八分之一精度色度样本矩阵中位置为(((xE+2×x)<<2)+MvC0_x,(yE+2×y)<<2)+MvC0_y)))的样本值，亮度后向预测样本矩阵predMatrixBw中的元素predMatrixBw[x][y]的值为后向参考图像的四分之一精度亮度样本矩阵中位置为(((xE+x)<<2)+MvE1_x,(yE+y)<<2)+MvE1_y)))的样本值,色度后向预测样本矩阵predMatrixBw中的元素predMatrixBw[x][y]的值为后向参考图像的八分之一精度色度样本矩阵中位置为(((xE+2×x)<<2)+MvC1_x,(yE+2×y)<<2)+MvC1_y)))的样本值。其中MvE0_x和MvE0_y分别为当前预测单元前向运动矢量MvE0的水平和垂直分量,MvE1_x和MvE1_y分别为当前预测单元后向运动矢量MvE1的水平和垂直分量，MvC0_x等于MvE0_x，MvC0_y等于MvE0_y-delta3，MvC1_x等于MvE1_x，MvC1_y等于MvE1_y-delta3。

如果当前预测单元的预测模式为双前向且DirMultiHypothesisMode等于0，则亮度第一预测样本矩阵predMatrix1中的元素predMatrix1[x][y]的值为第一参考图像的四分之一精度亮度样本矩阵中位置为(((xE+x)<<2)+MvE0_x,(yE+y)<<2)+MvE0_y)))的样本值，色度第一预测样本矩阵predMatrix1中的元素predMatrix1[x][y]的值为第一参考图像的八分之一精度色度样本矩阵中位置为(((xE+2×x)<<2)+MvC0_x,(yE+2×y)<<2)+MvC0_y)))的样本值，亮度第二预测样本矩阵predMatrix2中的元素predMatrix2[x][y]的值为第二参考图像的四分之一精度亮度样本矩阵中位置为(((xE+x)<<2)+MvE1_x,(yE+y)<<2)+MvE1_y)))的样本值，色度第二预测样本矩阵predMatrix2中的元素predMatrix2[x][y]的值为第二参考图像的八分之一精度色度样本矩阵中位置为(((xE+2×x)<<2)+MvC1_x,(yE+2×y)<<2)+MvC1_y)))的样本值。其中，MvE0_x和MvE0_y分别是当前预测单元第一运动矢量MvE0的水平和垂直分量,MvE1_x和MvE1_y分别是当前预测单元第二运动矢量MvE1的水平和垂直分量，MvC0_x等于MvE0_x，MvC0_y等于MvE0_y-delta3，MvC1_x等于MvE1_x，MvC1_y等于MvE1_y-delta3。

如果当前预测单元的预测模式为双前向且DirMultiHypothesisMode不等于0，则亮度第一预测样本矩阵predMatrix1中的元素predMatrix1[x][y]的值为前向参考图像的四分之一精度亮度样本矩阵中位置为(((xE+x)<<2)+MvE0_x,(yE+y)<<2)+MvE0_y)))的样本值，色度第一预测样本矩阵predMatrix1中的元素predMatrix1[x][y]的值为前向参考图像的八分之一精度色度样本矩阵中位置为(((xE+2×x)<<2)+MvC0_x,(yE+2×y)<<2)+MvC0_y)))的样本值，亮度第二预测样本矩阵predMatrix2中的元素predMatrix2[x][y]的值为前向参考图像的四分之一精度亮度样本矩阵中位置为(((xE+x)<<2)+MvE1_x,(yE+y)<<2)+MvE1_y)))的样本值，色度第二预测样本矩阵predMatrix2中的元素predMatrix2[x][y]的值为前向参考图像的八分之一精度色度样本矩阵中位置为(((xE+2×x)<<2)+MvC1_x,(yE+2×y)<<2)+MvC1_y)))的样本值。其中，MvE0_x和MvE0_y分别是当前预测单元运动矢量甲MvE0的水平和垂直分量,MvE1_x和MvE1_y分别是当前预测单元运动矢量乙MvE1的水平和

垂直分量，MvC0_x等于MvE0_x，MvC0_y等于MvE0_y-delta3，MvC1_x等于MvE1_x，MvC1_y等于MvE1_y-delta3。

其中delta3的值按照如下规则确定：delta3的值初始化为0；如果field_coded_sequence的值为‘1’且当前块所在的场为顶场图像同时mvE指向的场为底场图像，则delta3的值为2；如果field_coded_sequence的值为‘1’且当前块所在的场为底场图像同时mvE指向的场为顶场图像，则delta3的值为-2。

其中参考图像的亮度四分之一精度样本矩阵和色度八分之一精度样本矩阵中各个位置的元素值通过9.8.3和9.8.2定义的插值方法得到。参考图像外的整数样本应使用该图像内距离该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。

9.8.2 亮度样本插值过程

图28给出了参考图像亮度样本矩阵中整数样本、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。这些样本位置与其在四分之一精度的亮度样本矩阵中的坐标(f_x,f_y)的对应关系见表102，其中xFracL等于f_x&3，yFracl等于f_y&3。

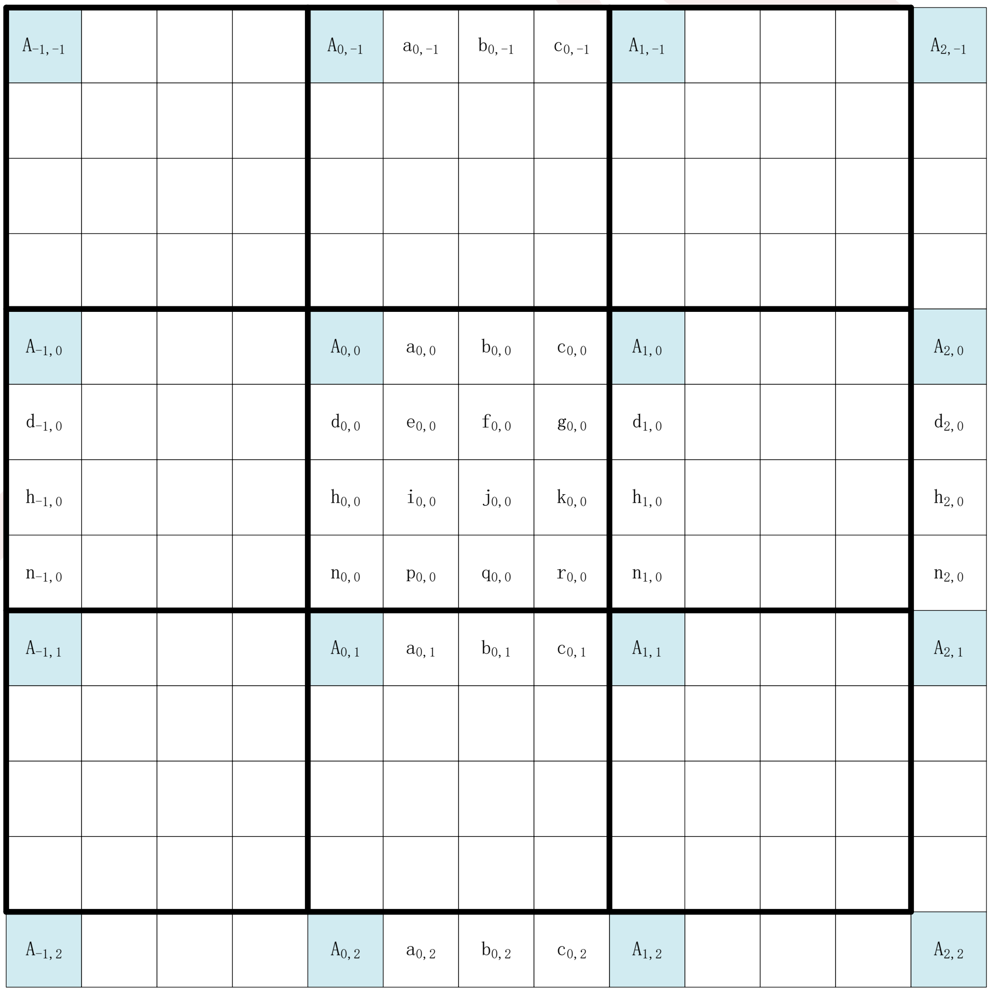


图28 整数样本、1/2 样本和 1/4 样本的位置

表102　预测样本矩阵元素

xFracL的值	yFracL的值	图30中的样本位置
0	0	$A_{0,0}$
0	1	$d_{0,0}$
0	2	$h_{0,0}$
0	3	$n_{0,0}$
1	0	$a_{0,0}$
1	1	$e_{0,0}$
1	2	$i_{0,0}$
1	3	$p_{0,0}$
2	0	$b_{0,0}$
2	1	$f_{0,0}$
2	2	$j_{0,0}$
2	3	$q_{0,0}$
3	0	$c_{0,0}$
3	1	$g_{0,0}$
3	2	$k_{0,0}$
3	3	$r_{0,0}$

样本位置 $a_{0,0}$，$b_{0,0}$, 及 $c_{0,0}$ 的预测值由水平方向距离插值点最近的 8 个整数值滤波得到，预测值获取方式如下：

$a_{0,0} = \text{Clip1}((-A_{-3,0} + 4\times A_{-2,0} - 10\times A_{-1,0} + 57\times A_{0,0} + 19\times A_{1,0} - 7\times A_{2,0} + 3\times A_{3,0} - A_{4,0} + 32) >> 6)$

$b_{0,0} = \text{Clip1}((-A_{-3,0} + 4\times A_{-2,0} - 11\times A_{-1,0} + 40\times A_{0,0} + 40\times A_{1,0} - 11\times A_{2,0} + 4\times A_{3,0} - A_{4,0} + 32) >> 6)$

$c_{0,0} = \text{Clip1}((-A_{-3,0} + 3\times A_{-2,0} - 7\times A_{-1,0} + 19\times A_{0,0} + 57\times A_{1,0} - 10\times A_{2,0} + 4\times A_{3,0} - A_{4,0} + 32) >> 6$

样本位置$d_{0,0}$，$h_{0,0}$，及 $n_{0,0}$ 的预测值由垂直方向距离插值点最近的8个整数值滤波得到，预测值获取方式如下：

$d_{0,0} = \text{Clip1}((-A_{0,-3} + 4\times A_{0,-2} - 10\times A_{0,-1} + 57\times A_{0,0} + 19\times A_{0,1} - 7\times A_{0,2} + 3\times A_{0,3} - A_{4,0} + 32) >> 6)$

$h_{0,0} = \text{Clip1}((-A_{0,-3} + 4\times A_{0,-2} - 11\times A_{0,-1} + 40\times A_{0,0} + 40\times A_{0,1} - 11\times A_{0,2} + 4\times A_{0,3} - A_{0,4} + 32) >> 6)$

$n_{0,0} = \text{Clip1}((-A_{0,-3} + 3\times A_{0,-2} - 7\times A_{0,-1} + 19\times A_{0,0} + 57\times A_{0,1} - 10\times A_{0,2} + 4\times A_{0,3} - A_{0,4} + 32) >> 6)$

样本位置 $e_{0,0}$，$i_{0,0}$，$p_{0,0}$，$f_{0,0}$，$j_{0,0}$，$q_{0,0}$，$g_{0,0}$，$k_{0,0}$ 及 $r_{0,0}$ 的预测值获取方式如下：

$e_{0,0}= \text{Clip1}((-a_{0,-3}' + 4\times a_{0,-2}' - 10\times a_{0,-1}' + 57\times a_{0,0}' + 19\times a_{0,1}' - 7\times a_{0,2}' + 3\times a_{0,3}' - a_{0,4}' + (1<<19\text{-BitDepth})) >> (20\text{-BitDepth}))$

$i_{0,0} = \text{Clip1}((-a_{0,-3}' + 4\times a_{0,-2}' - 11\times a_{0,-1}' + 40\times a_{0,0}' + 40\times a_{0,1}' - 11\times a_{0,2}' + 4\times a_{0,3}' - a_{0,4}' + (1<<19\text{-BitDepth})) >> (20\text{-BitDepth}))$

$p_{0,0} = \text{Clip1}((-a_{0,-3}' + 3\times a_{0,-2}' - 7\times a_{0,-1}' + 19\times a_{0,0}' + 57\times a_{0,1}' - 10\times a_{0,2}' + 4\times a_{0,3}' - a_{0,4}' + (1<<19\text{-BitDepth})) >> (20\text{-BitDepth}))$

$f_{0,0}= \text{Clip1}((-b_{0,-3}' + 4\times b_{0,-2}' - 10\times b_{0,-1}' + 57\times b_{0,0}' + 19\times b_{0,1}' - 7\times b_{0,2}' + 3\times b_{0,3}' - b_{0,4}' + (1<<19\text{-BitDepth})) >> (20\text{-BitDepth}))$

$j_{0,0}=$
$\mathrm{Clip1}((-b_{0,-3}' + 4\times b_{0,-2}' - 11\times b_{0,-1}' + 40\times b_{0,0}' + 40\times b_{0,1}' - 11\times b_{0,2}' + 4\times b_{0,3}' - b_{0,4}' + (1<<19\text{-BitDepth})) >> (20\text{-BitDepth}))$

$q_{0,0} =$
$\mathrm{Clip1}((-b_{0,-3}' + 3\times b_{0,-2}' - 7\times b_{0,-1}' + 19\times b_{0,0}' + 57\times b_{0,1}' - 10\times b_{0,2}' + 4\times b_{0,3}' - b_{0,4}' + (1<<19\text{-BitDepth})) >> (20\text{-BitDepth}))$

$g_{0,0} = \mathrm{Clip1}((-c_{0,-3}' + 4\times c_{0,-2}' - 10\times c_{0,-1}' + 57\times c_{0,0}' + 19\times c_{0,1}' - 7\times c_{0,2}' + 3\times c_{0,3}' - c_{0,4}' + (1<<19\text{-BitDepth})) >> (20\text{-BitDepth}))$

$k_{0,0}=$
$\mathrm{Clip1}((-c_{0,-3}' + 4\times c_{0,-2}' - 11\times c_{0,-1}' + 40\times c_{0,0}' + 40\times c_{0,1}' - 11\times c_{0,2}' + 4\times c_{0,3}' - c_{0,4}' + (1<<19\text{-BitDepth})) >> (20\text{-BitDepth}))$

$r_{0,0}= \mathrm{Clip1}((-c_{0,-3}' + 3\times c_{0,-2}' - 7\times c_{0,-1}' + 19\times c_{0,0}' + 57\times c_{0,1}' - 10\times c_{0,2}' + 4\times c_{0,3}' - c_{0,4}' + (1<<19\text{-BitDepth})) >> (20\text{-BitDepth}))$

其中：

$a_{0,i}' = (-A_{-3,i} + 4\times A_{-2,i} - 10\times A_{-1,i} + 57\times A_{0,i} + 19\times A_{1,i} - 7\times A_{2,i} + 3\times A_{3,i} - A_{4,i} + ((1<<\text{BitDepth-8})>>1)) >> (\text{BitDepth-8})$

$b_{0,i}' = (-A_{-3,i} + 4\times A_{-2,i} - 11\times A_{-1,i} + 40\times A_{0,i} + 40\times A_{1,i} - 11\times A_{2,i} + 4\times A_{3,i} - A_{4,i} + ((1<<\text{BitDepth-8})>>1)) >> (\text{BitDepth-8})$

$c_{0,i}' = (-A_{-3,i} + 3\times A_{-2,i} - 7\times A_{-1,i} + 19\times A_{0,i} + 57\times A_{1,i} - 10\times A_{2,i} + 4\times A_{3,i} - A_{4,i} + ((1<<\text{BitDepth-8})>>1)) >> (\text{BitDepth-8})$

9.8.3 色度样本插值过程

参考图像色度样本矩阵中的样本位置见图29，A，B，C，D为相邻整像素样本，dx与dy为整像素样本A周边分像素样本a(dx,dy)与A的水平和垂直距离，dx等于fx&7，dy等于fy&7，其中（fx，fy）为该分像素样本在八分之一精度的色度样本矩阵中的坐标。整像素$A_{x,y}$和周边的63个分像素样本$a_{x,y}$(dx,dy)的具体位置见图30。

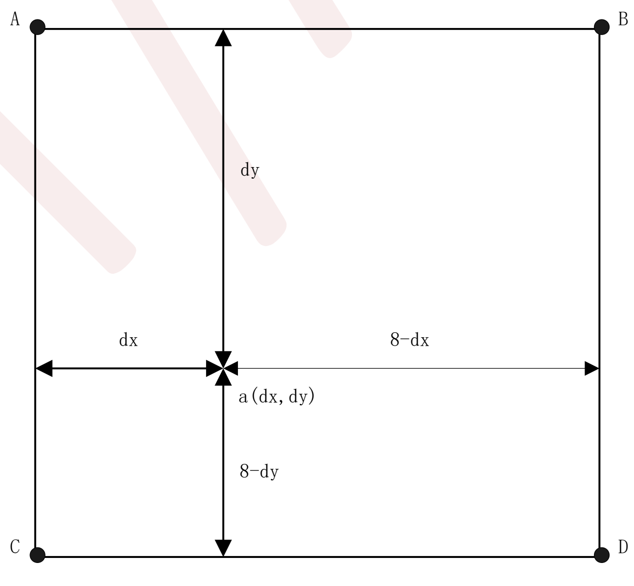


图29 色度插值

$A_{x,y}$	$a_{x,y}(1,0)$	$a_{x,y}(2,0)$	$a_{x,y}(3,0)$	$a_{x,y}(4,0)$	$a_{x,y}(5,0)$	$a_{x,y}(6,0)$	$a_{x,y}(7,0)$
$a_{x,y}(0,1)$	$a_{x,y}(1,1)$	$a_{x,y}(2,1)$	$a_{x,y}(3,1)$	$a_{x,y}(4,1)$	$a_{x,y}(5,1)$	$a_{x,y}(6,1)$	$a_{x,y}(7,1)$
$a_{x,y}(0,2)$	$a_{x,y}(1,2)$	$a_{x,y}(2,2)$	$a_{x,y}(3,2)$	$a_{x,y}(4,2)$	$a_{x,y}(5,2)$	$a_{x,y}(6,2)$	$a_{x,y}(7,2)$
$a_{x,y}(0,3)$	$a_{x,y}(1,3)$	$a_{x,y}(2,3)$	$a_{x,y}(3,3)$	$a_{x,y}(4,3)$	$a_{x,y}(5,3)$	$a_{x,y}(6,3)$	$a_{x,y}(7,3)$
$a_{x,y}(0,4)$	$a_{x,y}(1,4)$	$a_{x,y}(2,4)$	$a_{x,y}(3,4)$	$a_{x,y}(4,4)$	$a_{x,y}(5,4)$	$a_{x,y}(6,4)$	$a_{x,y}(7,4)$
$a_{x,y}(0,5)$	$a_{x,y}(1,5)$	$a_{x,y}(2,5)$	$a_{x,y}(3,5)$	$a_{x,y}(4,5)$	$a_{x,y}(5,5)$	$a_{x,y}(6,5)$	$a_{x,y}(7,5)$
$a_{x,y}(0,6)$	$a_{x,y}(1,6)$	$a_{x,y}(2,6)$	$a_{x,y}(3,6)$	$a_{x,y}(4,6)$	$a_{x,y}(5,6)$	$a_{x,y}(6,6)$	$a_{x,y}(7,6)$
$a_{x,y}(0,7)$	$a_{x,y}(1,7)$	$a_{x,y}(2,7)$	$a_{x,y}(3,7)$	$a_{x,y}(4,7)$	$a_{x,y}(5,7)$	$a_{x,y}(6,7)$	$a_{x,y}(7,7)$

图30　色度分像素点的标记

色度滤波系数见表103。

表103　色度滤波系数

数组标识	滤波器系数
C[0]	{ 0, 64, 0, 0 }
C[1]	{ -4, 62, 6, 0 }
C[2]	{ -6, 56, 15, -1 }
C[3]	{ -5, 47, 25, -3 }
C[4]	{ -4, 36, 36, -4 }
C[5]	{ -3, 25, 47, -5 }
C[6]	{ -1, 15, 56, -6 }
C[7]	{ 0, 6, 62, -4 }

对于dx等于0或dy等于0的分像素点，可直接用色度整像素插值得到，对于dx不等于0且dy不等于0的点，使用整像素行(dy等于0)上的分像素进行计算。

所有色度分像素通过以下公式计算：

a) 如果 dx 等于 0：

$a_{x,y}(0,dy) = Clip3(0, (1 << BitDepth)-1, (C[dy][0]\times A_{x,y-1} + C[dy][1]\times A_{x,y} + C[dy][2]\times A_{x,y+1} + C[dy][3]\times A_{x,y+2} + 32) >> 6)$

b) 否则，如果 dy 等于 0：

$a_{x,y}(dx,0) = Clip3(0, (1 << BitDepth)-1, (C[dx][0]\times A_{x-1,y} + C[dx][1]\times A_{x,y} + C[dx][2]\times A_{x+1,y} + C[dx][3]\times A_{x+2,y} + 32) >> 6)$

c) 否则：

$a_{x,y}(dx,dy) = Clip3(0, (1 << BitDepth)-1, (C[dy][0]\times a'_{x,y-1}(dx,0) + C[dy][1]\times a'_{x,y}(dx,0) + C[dy][2]\times a'_{x,y+1}(dx,0) + C[dy][3]\times a'_{x,y+2}(dx,0) + (1<<(19-BitDepth))) >> (20 - BitDepth))$

其中，$a'_{x,y}(dx,0)$是整像素行上的分像素的临时值，定义为：

$a'_{x,y}(dx,0) = (C[dx][0]\times A_{x-1,y} + C[dx][1]\times A_{x,y} + C[dx][2]\times A_{x+1,y} + C[dx][3]\times A_{x+2,y} + ((1<<(BitDepth-8))>>1)) >> (BitDepth - 8)$

9.9 预测补偿

如果当前预测单元的预测模式是帧内，补偿后样本矩阵CompMatrix计算如下：

CompMatrix[x][y] = Clip1(predMatrix[x][y] + ResidueMatrix[x][y])

如果当前预测单元的预测模式是对称或双向，补偿后样本矩阵CompMatrix计算如下：

CompMatrix[x][y] = Clip1(((predMatrixFw[x][y] + predMatrixBw[x][y]+1) >> 1) + ResidueMatrix[x][y])

如果当前预测单元的预测模式是单前向，补偿后样本矩阵CompMatrix计算如下：

CompMatrix[x][y] = Clip1(predMatrixFw[x][y] + ResidueMatrix[x][y])

如果当前预测单元的预测模式是后向，补偿后样本矩阵CompMatrix计算如下：

CompMatrix[x][y] = Clip1(predMatrixBw[x][y] + ResidueMatrix[x][y])

如果当前预测单元的预测模式是双前向，补偿后样本矩阵CompMatrix计算如下：

CompMatrix[x][y] = Clip1(((predMatrixFw1[x][y] + predMatrixFw2[x][y] + 1) >> 1) + ResidueMatrix[x][y])

其中predMatrix是帧内预测的预测样本矩阵，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，predMatrixFw1[x][y]是双前向预测的第一预测样本矩阵，predMatrixFw2[x][y]是双前向预测的第二预测样本矩阵。x的取值范围是[((LcuIndex % MinCUWidth) << LcuSizeInBit) ～ ((LcuIndex % MinCUWidth) << LcuSizeInBit) + (1 << LcuSizeInBit) – 1]；y的取值范围是[((LcuIndex / MinCUHeight) << LcuSizeInBit) ～ ((LcuIndex / MinCUHeight) << LcuSizeInBit) + (1 << LcuSizeInBit) – 1]。

9.10 去块效应滤波

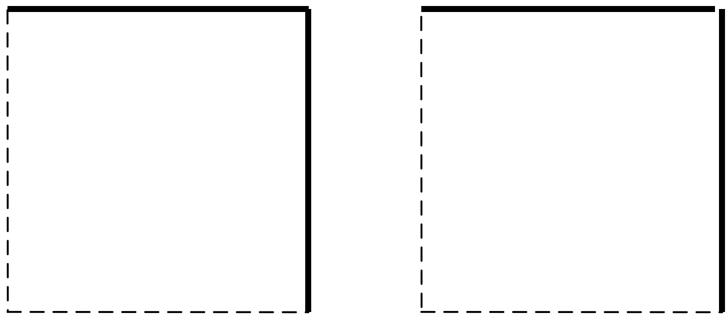
9.10.1 概述

对亮度和色度分别做去块效应滤波。去块效应滤波的单位是滤波块，按照光栅扫描顺序依次处理每个滤波块。亮度滤波块的尺寸是8×8，色度滤波块的尺寸是8×8。每个滤波块包括一条垂直边界和一条水平边界，如图31中粗线所示。亮度滤波块的每条边界的长度为8个亮度样本，平均分为8段。色度滤波块的每条边界的长度是8个色度样本，平均分为8段。

对每个滤波块，首先滤波垂直边界，然后滤波水平边界。

对每条边界，首先根据9.10.2判断该边界是否需要滤波。如果需要滤波，则根据9.10.3计算该条边界的每段边界的滤波强度，然后根据该段边界的滤波强度进行去块效应滤波（见9.10.4、9.10.5、9.10.6和9.10.7）；否则，将补偿后样本的值直接作为滤波后样本的值。

当前边界两侧的样值可能在以前的去块效应滤波过程中已经被修改，当前边界的滤波的输入为这些可能被修改的样值。当前滤波块的垂直边界的滤波过程中修改的样值作为水平边界滤波过程的输入。



注：实线为滤波块的垂直边界和水平边界，虚线为下一滤波块待滤波的边界。

图31　滤波单元中需要处理的边界示意

9.10.2　是否跳过去块效应滤波的判断

满足以下条件之一的边界不需要滤波：

a)　如果待滤波边界是图像边界，则该边界不需要滤波；

b)　如果待滤波边界是条带边界且 cross_slice_loopfilter_enable_flag 的值为‘0’，则该边界不需要滤波；

c)　如果待滤波边界既不是编码单元的边界，又不是预测单元的边界也不是变换块的边界，则该边界不需要滤波；

d)　如果待滤波边界在编码单元内部，且该编码单元的编码单元类型为“P_Skip”、“P_Direct”、“B_Skip”、“B_Direct_2N”、“F_Skip”或“F_Direct”，且不是变换块的边界，则该边界不需要滤波。

9.10.3　边界滤波强度的推导过程

图32表示某一段滤波块边界（用黑色粗线表示），其两侧的6个样本分别记为p_0、p_1、p_2和q_0、q_1、q_2。

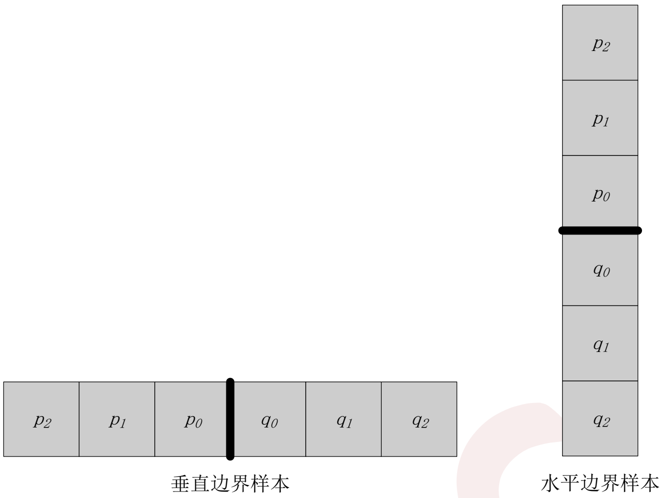


图32　滤波块边界样本

a）　如果当前图像是 P 图像或 S 图像，满足以下所有条件时该段边界的滤波强度 Bs 为 0：
   1）　p_0 和 q_0 所在的编码单元的残差样值均为 0；
   2）　p_0 和 q_0 所在的编码单元的预测类型不是帧内，且 p_0 和 q_0 所在的预测单元的前向参考索引相同并且前向运动矢量的各分量的差均小于一个整像素点。
b）　如果当前图像是 F 图像，满足以下所有条件时该段边界的滤波强度 Bs 为 0：
   1）　p_0 和 q_0 所在的编码单元的残差样值均为 0；
   2）　p_0 和 q_0 所在的编码单元的预测类型不是帧内，并且 p_0 和 q_0 所在的预测单元的前向参考索引/第一参考索引相同且前向运动矢量/第一运动矢量的各分量的差均小于一个整像素点。
c）　否则，计算边界滤波强度的过程如下：
   1）　计算 p_0 和 q_0 所在的编码单元的平均量化参数 QP_{av}。如果是亮度样本，应使用亮度块的量化参数；如果是色度样本，应使用色度块的量化参数。设 p_0 所在编码单元的量化参数为 QP_p，q_0 所在编码单元的量化参数为 QP_q，则平均量化参数按下式计算：

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

   2）　计算索引 IndexA 和 IndexB：

$$IndexA = Clip3(0, 63, QP_{av} - 8 \times (BitDepth - 8) + AlphaCOffset)$$

$$IndexB = Clip3(0, 63, QP_{av} - 8 \times (BitDepth - 8) + BetaOffset)$$

   3）　根据索引 IndexA 和 IndexB 与阈值α和β间的对应关系，由表 104 得到α'、β'的取值，根据 IndexA 查表得到α'，根据 IndexB 查表得到β'。再根据 bitDepth 按以下方法得到α、β的值。

$$\alpha = \alpha' << (BitDepth - 8)$$

$$\beta = \beta' << (BitDepth - 8)$$

表104　IndexA 和 IndexB 与块边界阈值α'和β'与的关系

索引 IndexA 或 IndexB	α′	β′	索引 IndexA 或 IndexB	α′	β′	索引 IndexA 或 IndexB	α′	β′	索引 IndexA 或 IndexB	α′	β′
0	0	0	16	4	2	32	22	6	48	46	15
1	0	0	17	4	2	33	24	7	49	48	16

表 104（续）

索引 IndexA 或 IndexB	α′	β′	索引 IndexA 或 IndexB	α′	β′	索引 IndexA 或 IndexB	α′	β′	索引 IndexA 或 IndexB	α′	β′
2	0	0	18	5	3	34	26	7	50	50	17
3	0	0	19	5	3	35	28	7	51	52	18
4	0	0	20	6	3	36	30	8	52	53	19
5	0	0	21	7	3	37	33	8	53	54	20
6	1	1	22	8	4	38	33	8	54	55	21
7	1	1	23	9	4	39	35	9	55	56	22
8	1	1	24	10	4	40	35	9	56	57	23
9	1	1	25	11	4	41	36	10	57	58	23
10	1	1	26	12	5	42	37	10	58	59	24
11	2	1	27	13	5	43	37	11	59	60	24
12	2	1	28	15	5	44	39	11	60	61	25
13	2	2	29	16	5	45	39	12	61	62	25
14	3	2	30	18	6	46	42	13	62	63	26
15	3	2	31	20	6	47	44	14	63	64	27

4) 如果 $Abs(p_0-q_0)$ 小于α并且 $Abs(p_0-q_0)$ 大于 1，按以下方法计算 Bs；否则 Bs 等于 0。

——第一步，将 fL 和 fR 的值均置为 0，

```
if ( Abs(p0 - p1) < β )
    fL += 2
if ( Abs(p0 - p2) < β )
    fL ++
if ( Abs(q0 - q1) < β )
    fR += 2
if ( Abs(q0 - q2) < β )
    fR ++
fS = fL + fR
```

——第二步，根据 fS 的值得到 Bs：

- 当 fS 等于 6 时，如果 p_0 等于 p_1 并且 q_0 等于 q_1，则 Bs 等于 4；否则 Bs 等于 3。
- 当 fS 等于 5 时，如果 p0 等于 p1 并且 q0 等于 q1，则 Bs 等于 3；否则 Bs 等于 2。
- 当 fS 等于 4 时，如果 fL 等于 2，则 Bs 等于 2；否则 Bs 等于 1。
- 当 fS 等于 3 时，并且 Abs(p1–q1)小于β，则 Bs 等于 1；否则 Bs 等于 0。
- 当 fS 为其它值时，Bs 等于 0。

——第三步，如果第二步得到的 Bs 不等于 0，按以下方法修正 Bs：

- 滤波的边界是色度块边界，Bs 减 1。

9.10.4 Bs 等于 4 时的边界滤波过程

边界滤波强度Bs的值为4时，对p_0、p_1、p_2和q_0、q_1、q_2滤波的计算过程如下(P_0、P_1、P_2和Q_0、Q_1、Q_2是滤波后的值)：

$$P_0 = (p_0 + ((p_0 + p_2) << 3) + p_2 + (q_0 << 3) + (q_2 << 2) + (q_2 << 1) + 16) >> 5$$

$P_1 = ((p_0 \ll 3) - p_0 + (p_2 \ll 2) + (p_2 \ll 1) + q_0 + (q_0 \ll 1) + 8) \gg 4$
$P_2 = ((p_0 \ll 2) + p_2 + (p_2 \ll 1) + q_0 + 4) \gg 3$
$Q_0 = (q_0 + ((q_0 + q_2) \ll 3) + q_2 + (p_0 \ll 3) + (p_2 \ll 2) + (p_2 \ll 1) + 16) \gg 5$
$Q_1 = ((q_0 \ll 3) - q_0 + (q_2 \ll 2) + (q_2 \ll 1) + p_0 + (p_0 \ll 1) + 8) \gg 4$
$Q_2 = ((q_0 \ll 2) + q_2 + (q_2 \ll 1) + p_0 + 4) \gg 3$

9.10.5 Bs 等于 3 时的边界滤波过程

边界滤波强度Bs的值为3时，对p_0、p_1和q_0、q_1滤波的计算过程如下(P_0、P_1和Q_0、Q_1是滤波后的值)：

$P_0 = (p_2 + (p_1 \ll 2) + (p_0 \ll 2) + (p_0 \ll 1) + (q_0 \ll 2) + q_1 + 8) \gg 4$
$Q_0 = (p_1 + (p_0 \ll 2) + (q_0 \ll 2) + (q_0 \ll 1) + (q_1 \ll 2) + q_2 + 8) \gg 4$
$P_1 = ((p_2 \ll 1) + p_2 + (p_1 \ll 3) + (p_0 \ll 2) + q_0 + 8) \gg 4$
$Q_1 = ((q_2 \ll 1) + q_2 + (q_1 \ll 3) + (q_0 \ll 2) + p_0 + 8) \gg 4$

9.10.6 Bs 等于 2 时的边界滤波过程

边界滤波强度Bs的值为2时，对p_0和q_0滤波的计算过程如下(P_0和Q_0是滤波后的值)：

$P_0 = ((p_1 \ll 1) + p_1 + (p_0 \ll 3) + (p_0 \ll 1) + (q_0 \ll 1) + q_0 + 8) \gg 4$
$Q_0 = ((p_0 \ll 1) + p_0 + (q_0 \ll 3) + (q_0 \ll 1) + (q_1 \ll 1) + q_1 + 8) \gg 4$

9.10.7 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是滤波后的值)：

$P_0 = ((p_0 \ll 1) + p_0 + q_0 + 2) \gg 2$
$Q_0 = ((q_0 \ll 1) + q_0 + p_0 + 2) \gg 2$

9.11 样值偏移补偿

9.11.1 概述

样值偏移补偿的处理步骤如下：

a) 如果 SliceSaoEnableFlag[compIdx]的值为 0，则将滤波后样本对应分量的值直接作为偏移后该样本分量的值；
b) 否则，根据 9.11.2 导出样值偏移补偿单元，根据 9.11.3 导出与当前样值偏移补偿单元对应的样值偏移补偿信息，然后按照 9.11.4 对当前样值偏移补偿单元内的各个样本的各分量进行操作，得到偏移后样本值。

9.11.2 导出样值偏移补偿单元

先确定当前最大编码单元，然后根据当前最大编码单元按下列步骤得到与其对应的当前样值偏移补偿单元:

a) 将当前最大编码单元 C 所在亮度和色度样本区域各向左移四个样本单位后再各向上移四个样本单位，得到区域 E1；
b) 如果区域 E1 超出当前图像边界，则将超出部分去除，得到区域 E2；否则令 E2 等于 E1；
c) 如果当前最大编码单元 C 包含图像最右列的样本且不包含图像最下行的样本，则将区域 E2 的右边界向右扩展至图像的右边界，得到区域 E3；
d) 否则，如果当前最大编码单元 C 包含图像最下行的样本且不包含图像最右列的样本，则将区域 E2 的下边界向下扩展至图像的边界，得到区域 E3；

e) 否则，如果当前最大编码单元 C 同时包含图像最右列的样本和最下行的样本，则将区域 E2 的右边界向右扩展至图像的右边界后再将新区域的下边界向下扩展至图像的下边界，得到区域 E3；
f) 否则，令 E3 等于 E2；
g) 将区域 E3 作为当前样值偏移补偿单元。

9.11.3 导出样值偏移补偿信息

本条定义了样值偏移补偿单元中各分量的样值偏移补偿操作所需信息的导出方法。这些信息包括：样值偏移补偿模式SaoMode[compIdx]、样值偏移补偿的偏移值SaoOffset[compIdx][j]、样值偏移补偿边缘模式类型SaoEdgeType[compIdx]、样值偏移补偿区间模式的起始偏移子区间位置SaoIntervalStartPos[compIdx]和样值偏移补偿区间模式的起始偏移子区间的位置差值SaoIntervalDeltaPosMinus2[compIdx]。

如果SaoMergeLeftAvai或SaoMergeUpAvai的值为1，则MergeFlagExist的值为1；否则MergeFlagExist的值为0。

根据SaoMergeLeftAvai、SaoMergeUpAvai和SaoMergeTypeIndex的值查表105得到样值偏移补偿合并模式SaoMergeMode。

表105　样值偏移补偿模式

SaoMergeLeftAvai的值	SaoMergeUpAvai的值	SaoMergeTypeIndex的值	样值偏移补偿合并模式（SaoMergeMode）
0	0	-	SAO_NON_MERGE
1	0	0	SAO_NON_MERGE
		1	SAO_MERGE_LEFT
0	1	0	SAO_NON_MERGE
		1	SAO_MERGE_UP
1	1	0	SAO_NON_MERGE
		1	SAO_MERGE_LEFT
		2	SAO_MERGE_UP

SaoMode[compIdx]的导出方法如下：如果SaoMergeMode的值为‘SAO_NON_MERGE’，从码流中解析得到SaoMode[compIdx]的值；如果SaoMergeMode的值为‘SAO_MERGE_LEFT’，SaoMode[compIdx]的值等于左边样值偏移补偿单元的SaoMode[compIdx]的值；如果SaoMergeMode值为‘SAO_MERGE_UP’,SaoMode[compIdx]的值等于上边样值偏移补偿单元的SaoMode[compIdx]的值。

a) 如果SaoMode[compIdx]的值为‘SAO_Interval’，先按以下方法导出SaoOffset[compIdx][j]（j=0～3）、SaoIntervalStartPos[compIdx] 和SaoIntervalDeltaPosMinus2[compIdx]，然后根据9.11.3.1进行样值偏移补偿操作：
   1) 如果SaoMergeMode的值为‘SAO_NON_MERGE’，从码流中解析得到SaoIntervalOffsetAbs[compIdx][j]（j=0～3）、SaoIntervalOffsetSign[compIdx][j]（j=0～3）、SaoIntervalStartPos[compIdx] 和SaoIntervalDeltaPosMinus2[compIdx]，计算得到SaoOffset[compIdx][j] = SaoIntervalOffsetAbs[compIdx][j] × SaoIntervalOffsetSign[compIdx][j]（j=0～3）；

2） 否则，如果 SaoMergeMode 的值为‘SAO_MERGE_LEFT’，SaoOffset[compIdx][j]（j=0～3）的值等于左边样值偏移补偿单元的 SaoOffset[compIdx][j]（j=0～3）的值，SaoIntervalStartPos[compIdx]和 SaoIntervalDeltaPosMinus2[compIdx]的值等于左边样值偏移补偿单元的 SaoIntervalStartPos[compIdx] 和 SaoIntervalDeltaPosMinus2[compIdx]的值；

3） 否则，SaoOffset[compIdx][j]（j=0～3）的值等于上边样值偏移补偿单元的 SaoOffset[compIdx][j]（j=0～3）的值，SaoIntervalStartPos[compIdx] 和 SaoIntervalDeltaPosMinus2[compIdx] 的值等于上边样值偏移补偿单元的 SaoIntervalStartPos[compIdx]和 SaoIntervalDeltaPosMinus2[compIdx]的值。

b） 否则，如果 SaoMode[compIdx]的值为‘SAO_Edge’，先按以下方法导出 SaoOffset[compIdx][j]（j=0～3）和 SaoEdgeType[compIdx]，然后根据 9.11.3.2 进行样值偏移补偿操作：

1） 如果 SaoMergeMode 的值为‘SAO_NON_MERGE’，从码流中解析得到 SaoEdgeOffset[compIdx][j]（j=0～3）和 SaoEdgeType[compIdx]的值，计算得到 SaoOffset[compIdx][j]= SaoEdgeOffset[compIdx][j]；

2） 否则，如果 SaoMergeMode 的值为‘SAO_MERGE_LEFT’，SaoOffset[compIdx][j]（j=0～3）和 SaoEdgeType[compIdx]的值等于左边样值偏移补偿单元的 SaoOffset[compIdx][j]（j=0～3）和 SaoEdgeType[compIdx]的值；

3） 否则，SaoOffset[compIdx][j]（j=0～3）和 SaoEdgeType[compIdx]的值等于上边样值偏移补偿单元的 SaoOffset[compIdx][j]（j=0～3）和 SaoEdgeType[compIdx]的值。

9.11.4 样值偏移补偿操作

9.11.4.1 SAO_Interval 模式的操作

如果样值偏移补偿单元的SaoMode[compIdx]为‘SAO_Interval’，进行以下操作：

a） 第一步，根据滤波后样本的 compIdx 分量值查表 106 得到该分量对应的 saoOffset[compIdx]。

表106　区间模式的偏移值

样值	偏移值（saoOffset[compIdx]）
SaoIntervalOffsetPos[compIdx][0] << shift1 ～ (SaoIntervalOffsetPos[compIdx][0] << shift1) + ((1 << shift1) - 1)	SaoOffset[compIdx][0]
SaoIntervalOffsetPos[compIdx][1] << shift1 ～ (SaoIntervalOffsetPos[compIdx][1] << shift1) + ((1 << shift1) - 1)	SaoOffset[compIdx][1]
SaoIntervalOffsetPos[compIdx][2] << shift1 ～ (SaoIntervalOffsetPos[compIdx][2] << shift1) + ((1 << shift1) - 1)	SaoOffset[compIdx][2]
SaoIntervalOffsetPos[compIdx][3] << shift1 ～ (SaoIntervalOffsetPos[compIdx][3] << shift1) + ((1 << shift1) - 1)	SaoOffset[compIdx][3]
其他	0

表105中：

shift1 = BitDepth - 5

SaoIntervalOffsetPos[compIdx][0] = SaoIntervalStartPos[compIdx]

SaoIntervalOffsetPos[compIdx][1] = (SaoIntervalStartPos[compIdx] + 1) % 32

SaoIntervalOffsetPos[compIdx][2]=(SaoIntervalStartPos[compIdx] + SaoIntervalDeltaPosMinus2[compIdx] + 2) % 32

SaoIntervalOffsetPos[compIdx][3]=(SaoIntervalStartPos[compIdx] + SaoIntervalDeltaPosMinus2[compIdx] + 3) % 32

b) 第二步，当前样本的偏移后样本 compIdx 分量值 y[compIdx]=Clip1(x[compIdx]+saoOffset[compIdx])，其中 x[compIdx]是该样本滤波后样本的 compIdx 分量的值。

9.11.4.2 SAO_Edge 模式的操作

如果样值偏移补偿单元的SaoMode[compIdx]为‘SAO_Edge’，进行以下操作：

a) 第一步，根据 SaoEdgeType[compIdx]的值确定当前滤波后样本 c 的相邻滤波后样本 a 和 b（见图 33）。如果满足以下条件之一，则当前样本的偏移后样本 compIdx 分量值 y[compIdx]=x[compIdx]，其中 x[compIdx]是该样本滤波后样本的 compIdx 分量值，结束本过程；否则，继续执行以下步骤。
   1) a 或 b 不与 c 处在同一条带内，且 cross_slice_loopfilter_enable_flag 的值为‘0’；
   2) a 或 b 不在当前图像内。

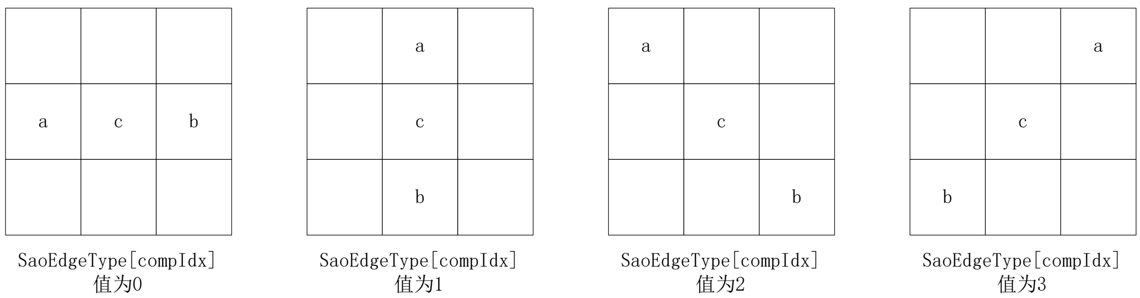


图33　当前滤波后样本和相邻滤波后样本的关系

b) 第二步，根据表 107 利用当前样本 c 的滤波后 compIdx 分量值 xc 与相邻样本 a 和 b 的滤波后样本 compIdx 分量值 xa 和 xb 的关系确定当前样本 compIdx 分量的 saoOffset[compIdx]。

表107　边缘模式的偏移值

样本分量值的关系	偏移值（saoOffset[compIdx]）
$x_c < x_a$ && $x_c < x_b$	SaoOffset[compIdx][0]
($x_c < x_a$ && $x_c == x_b$) || ($x_c == x_a$ && $x_c < x_b$)	SaoOffset[compIdx][1]
($x_c > x_a$ && $x_c == x_b$) || ($x_c == x_a$ && $x_c > x_b$)	SaoOffset[compIdx][2]
$x_c > x_a$ && $x_c > x_b$	SaoOffset[compIdx][3]
其他	0

c) 第三步，当前样本的偏移后样本 compIdx 分量值 y[compIdx]=Clip1(x[compIdx] + saoOffset[compIdx])，其中 x[compIdx]是该样本滤波后样本的 compIdx 分量值。

9.11.4.3 SAO_Off 模式的操作

如果样值偏移补偿单元的SaoMode[compIdx]为‘SAO_Off’，将当前样本滤波后样本compIdx分量的值直接作为该样本偏移后样本compIdx分量值。

9.12 自适应修正滤波

9.12.1 概述

如果PictureAlfEnableFlag[compIdx]的值为0，将偏移后样本分量的值直接作为对应重建样本分量的值，否则，对相应的偏移后样本分量进行自适应修正滤波，其中compIdx等于0表示亮度分量，等于1表示Cb分量，等于2表示Cr分量。

自适应修正滤波的单位是由最大编码单元导出的自适应修正滤波单元，按照光栅扫描顺序依次处理。首先根据9.12.2解码各分量的自适应修正滤波系数，然后根据9.12.3导出自适应修正滤波单元，根据9.12.4确定当前自适应修正滤波单元亮度分量的自适应修正滤波系数索引，最后根据9.12.5对自适应修正滤波单元的亮度和色度分量进行自适应修正滤波，得到重建样本。

9.12.2 自适应修正滤波系数解码

自适应修正滤波系数的解码过程如下：

a) 从位流中解析得到亮度样本的第 i 组滤波系数 AlfCoeffLuma[i][j]（i=0～alf_filter_num_minus1，j=0～7）。对系数 AlfCoeffLuma[i][8]（即图 34 的系数 C8）做以下处理：

$$AlfCoeffLuma[i][8] += 64 - \sum_{j=0}^{7} 2 \times AlfCoeffLuma[i][j]$$

其中 AlfCoeffLuma[i][j]（j=0～7）的位宽是 7 位，取值范围是-64～63；经上述处理后 AlfCoeffLuma[i][8]的取值范围是 0～127。

b) 根据 alf_region_distance[i]（i>1）得到亮度分量自适应修正滤波系数索引数组（记作 alfCoeffIdxTab[16]）：

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

c) 从位流中解析得到色度样本的滤波系数 AlfCoeffChroma[0][j]和 AlfCoeffChroma[1][j]（j=0～7）。对系数 AlfCoeffChroma[0][8]和 AlfCoeffChroma[1][8]（即图 34 的系数 C8）分别做以下处理：

$$AlfCoeffChroma[i][8] += 64 - \sum_{j=0}^{7} 2 \times AlfCoeffChroma[i][j], i = 0,1$$

其中 AlfCoeffChroma[i][j]（j=0～7）的位宽是 7 位，取值范围是-64～63；经上述处理后AlfCoeffChroma[i][8]的取值范围是 0～127。

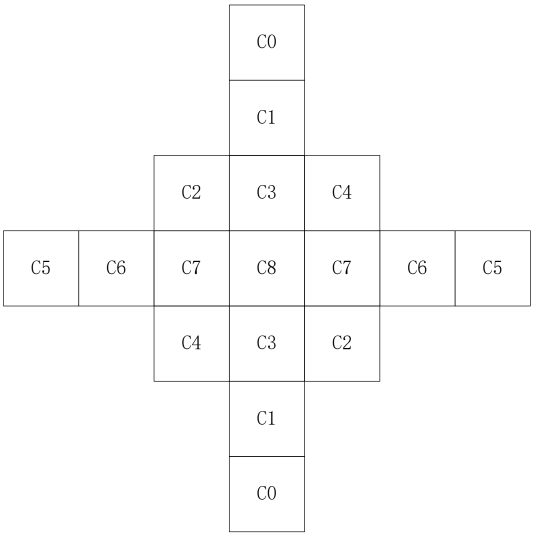


图34　自适应修正滤波系数

9.12.3　导出自适应修正滤波单元

根据当前最大编码单元按下列步骤导出自适应修正滤波单元（见图35）：

a) 将当前最大编码单元 C 所在样本区域超出图像边界的部分删除，得到样本区域 D。

b) 如果区域 D 的下边界所在的样本不属于图像的下边界，将亮度分量和色度分量样本区域 D 的下边界向上收缩四行，得到区域 E1；否则，令 E1 等于 D。区域 D 的最后一行样本为区域的下边界。

c) 如果区域 E1 的上边界所在的样本属于图像的上边界，或者属于条带边界并且 cross_slice_loopfilter_enable_flag 的值为‘0’，令 E2 等于 E1；否则，将亮度分量和色度分量样本区域 E1 的上边界向上扩展四行，得到区域 E2。区域 E1 的第一行样本为区域的上边界。

d) 将区域 E2 作为当前自适应修正滤波单元。图像的第一行样本为图像的上边界，最后一行样本为图像的下边界。

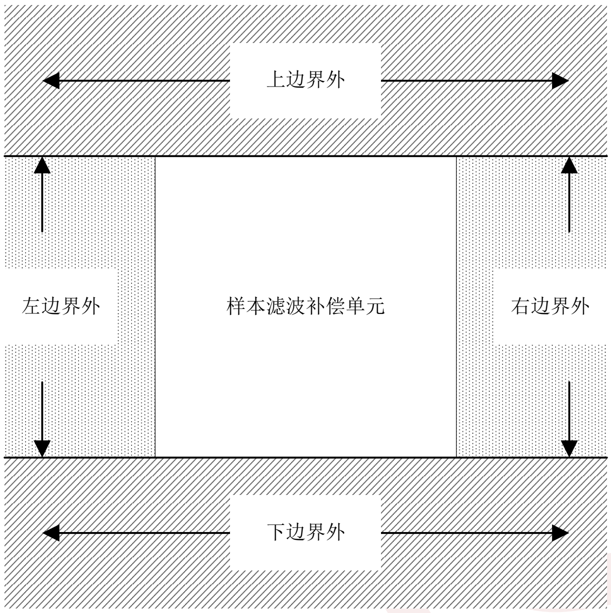


图35　自适应修正滤波单元

9.12.4　确定亮度分量自适应修正滤波单元自适应修正滤波系数索引

根据以下方法计算当前亮度分量自适应修正滤波单元的自适应修正滤波系数索引（记作filterIdx）：

```
xInterval = ((((horizontal_size + (1 << LcuSizeInBit) - 1) >> LcuSizeInBit) + 1) >> 2) << LcuSizeInBit
yInterval = ((((vertical_size + (1 << LcuSizeInBit) - 1) >> LcuSizeInBit) + 1) >> 2) << LcuSizeInBit
if (xInterval == 0  && yInterval == 0)
    index = 15
else if (xInterval == 0)
    index = Min(3, y/yInterval) × 4 + 3
else if (yInterval == 0)
    index = Min(3, x/xInterval) + 12
else
    index = Min(3, y/yInterval) × 4 + Min(3, x/xInterval)
filterIdx = alfCoeffIdxTab[regionTable[index]]
```

其中regionTable[16]={0,1,4,5,15,2,3,6,14,11,10,7,13,12, 9,8}，(x,y)是导出当前自适应修正滤波单元的最大编码单元左上角样本在图像中的坐标。

9.12.5　自适应修正滤波操作

如果AlfLCUEnableFlag[compIdx][LcuIndex]等于1，则对compIdx分量进行自适应修正滤波，否则不进行自适应修正滤波。如果自适应修正滤波过程中用到的样本为自适应修正滤波单元内的样本，则直接使用该样本进行滤波；否则，按照如下方式进行滤波：

a)　如果该样本在图像边界外，或在条带边界外且 cross_slice_loopfilter_enable_flag 的值为‘0’，则使用自适应修正滤波单元内距离该样本最近的样本代替该样本进行滤波；

b) 否则，如果该样本在自适应修正滤波单元上边界或下边界外，则使用自适应修正滤波单元内距离该样本最近的样本代替该样本进行滤波；

c) 否则，直接使用该样本进行滤波。

自适应修正滤波单元亮度分量的自适应修正滤波操作如下：

$$ptmp = AlfCoeffLuma[filterIdx][8] \times p(x, y) + \sum_{j=0}^{7} AlfCoeffLuma[filterIdx][j] \times (p(x - Hor[j], y - Ver[j]) + p(x + Hor[j], y + Ver[j]))$$

ptmp = (ptmp + 32) >> 6

p′(x,y) = Clip3(0, (1<<BitDepth)-1, ptmp)

其中，p(x,y)为偏移后样本，p’(x,y)为重建样本，Hor[j]和 Ver[j] （j=0～7）见表 106。

自适应修正滤波单元色度分量的自适应修正滤波操作如下：

$$ptmp = AlfCoeffChroma[i][8] \times p(x, y) + \sum_{j=0}^{7} AlfCoeffChroma[i][j] \times (p(x - Hor[j], y - Ver[j]) + p(x + Hor[j], y + Ver[j]))$$

ptmp = (ptmp + 32) >> 6

p′(x,y) = Clip3(0, (1<<BitDepth)-1, ptmp)

其中，p(x,y)为偏移后样本，p’(x,y)为重建样本，Hor[j]和 Ver[j]（j=0～7）见表 108。

表108 样本补偿滤波坐标偏移值

j的值	Hor[j]的值	Ver[j]的值
0	0	3
1	0	2
2	1	1
3	0	1
4	1	-1
5	3	0
6	2	0
7	1	0

9.13 运动信息存储

首先，确定样本(x,y)所对应的运动信息存储单元的左上角像素点位置(x_0,y_0)、宽度nWidth和高度nHeight：

x_0 = (x >> 4) << 4

y_0 = (y >> 4) << 4

$$nWidth = \begin{cases} MinCuWidth \times MiniSize - x_0, & x_0 + 16 \geq MinCuWidth \times MiniSize \\ 16, & x_0 + 16 < MinCuWidth \times MiniSize \end{cases}$$

$$nHeight = \begin{cases} MinCuHeight \times MiniSize - y_0, & y_0 + 16 \geq MinCuHeight \times MiniSize \\ 16, & y_0 + 16 < MinCuHeight \times MiniSize \end{cases}$$

其次，确定将运动信息存储到样本(x,y)对应的运动信息存储单元的方法如下：

a) 如果(x_0+nWidth/2,y_0+nHeight/2)亮度样本所在编码单元的预测类型为帧内，或(x_0+nWidth/2,y_0+nHeight/2)亮度样本所在预测单元的预测模式是后向预测，或(x_0+nWidth/2,y_0+nHeight/2)亮度样本所在预测单元的前向参考索引或第一参考索引等于RefPicNum-1且当前图像的SceneReferenceEnableFlag值为1，或(x_0+nWidth/2,y_0+nHeight/2)亮度样本所在图像为S图像且该亮度样本所在预测单元的前向参考索引等于RefPicNum-1，则将该运动信息存储单元的参考帧索引值设为-1；
b) 否则，如果(x_0+nWidth/2,y_0+nHeight/2)亮度样本所在预测单元的预测模式是双前向且DirMultiHypothesisMode 等于零，则将该运动信息存储单元的运动矢量和参考索引设为(x_0+nWidth/2,y_0+nHeight/2)亮度样本所在预测单元的第一运动矢量和第一参考帧索引；
c) 否则，则将该运动信息存储单元的运动矢量和参考索引设为(x_0+nWidth/2,y_0+nHeight/2)亮度样本所在预测单元的前向运动矢量和前向参考帧索引。

附 录 A
（规范性附录）
防止伪起始码方法

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、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）
高效音视频编码类和级

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。符合某个档次规定的解码器应完全支持该档次定义的子集。级是在某一档次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
0x12	基准图像类（Main picture profile）
0x20	基准8位类（Main profile）
0x22	基准10位类（Main-10bit 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。

基准图像类的位流应满足以下条件：

a)　profile_id 的值应为 0x12；
b)　progressive_sequence 的值应为‘1’时；
c)　chroma_format 的值应为‘01’；
d)　sample_precision 的值应为‘001’或‘010’；
e)　temporal_id_enable_flag 的值应为‘0’；
f)　scene_picture_disable_flag 的值应为‘1’；
g)　编码图像的起始码应为 intra_picture_start_code；
h)　lcu_size 的值取值范围应为 4～6；
i)　MiniSize 的值应为 8；
j)　B.3.2 规定的级的限制；

k) 支持的级包括：2.0.15、2.0.30、2.0.60、4.0.30、4.0.60、6.0.30、6.2.30、6.0.60、6.2.60、6.0.120、6.2.120、8.0.30、8.2.30、8.0.60、8.2.60、8.0.120、8.2.120、10.0.30、10.2.30、10.0.60、10.2.60、10.0.120 和 10.2.120。

基准8位类的位流应满足以下条件：

a) profile_id 的值应为 0x20；
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) chroma_format 的值应为‘01’；
d) sample_precision 的值应为‘001’；
e) 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；
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 progressive_frame 的值均应相同；
g) lcu_size 的值取值范围应为 4～6；
h) MiniSize 的值应为 8；
i) MAX_TEMPORAL_ID 的值应等于 7；
j) B.3.2 规定的级的限制；
k) 支持的级包括：2.0.15、2.0.30、2.0.60、4.0.30、4.0.60、6.0.30、6.2.30、6.0.60、6.2.60、6.0.120、6.2.120、8.0.30、8.2.30、8.0.60、8.2.60、8.0.120、8.2.120、10.0.30、10.2.30、10.0.60、10.2.60、10.0.120 和 10.2.120。

基准10位类的位流应满足以下条件：

a) profile_id 的值应为 0x22；
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) chroma_format 的值应为‘01’；
d) sample_precision 的值应为‘001’或‘010’；
e) 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；
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 progressive_frame 的值均应相同；
g) lcu_size 的值取值范围应为 4～6；
h) MiniSize 的值应为 8；
i) B.3.2 规定的级的限制；
j) 支持的级包括：2.0.15、2.0.30、2.0.60、4.0.30、4.0.60、6.0.30、6.2.30、6.0.60、6.2.60、6.0.120、6.2.120、8.0.30、8.2.30、8.0.60、8.2.60、8.0.120、8.2.120、10.0.30、10.2.30、10.0.60、10.2.60、10.0.120 和 10.2.120。

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	2.0.15
0x12	2.0.30
0x14	2.0.60
0x20	4.0.30
0x22	4.0.60
0x40	6.0.30
0x42	6.2.30
0x44	6.0.60
0x46	6.2.60
0x48	6.0.120
0x4A	6.2.120
0x50	8.0.30
0x52	8.2.30
0x54	8.0.60
0x56	8.2.60
0x58	8.0.120
0x5A	8.2.120
0x60	10.0.30
0x62	10.2.30
0x64	10.0.60
0x66	10.2.60
0x68	10.0.120
0x6A	10.2.120
其他	保留

B.3.2　与类无关的级的限制

对于所有类，每个编码单元编码后最大二进制位数的限制见表B.3， 其中LcuSizeInBit是最大编码单元的尺寸，BitDepth是样本点精度。

表B.3　编码单元编码后最大二进制位数

图像格式	编码单元编码后最大二进制位数
4:2:0	$2^{(\mathrm{LcuSizeInBit}-1)} + 2^{(\mathrm{LcuSizeInBit}\times 2)} \times \mathrm{BitDepth} \times 1.5$

级2.0.15、2.0.30和2.0.60的参数限制见表B.4。

表B.4　级 2.0.15、2.0.30 和 2.0.60 的参数限制

参数	级		
	2.0.15	2.0.30	2.0.60
每行最大样本数	352	352	352
每幅图像最大行数	288	288	288
每秒最大图像数	15	30	60
每帧最大条带数	16	16	16
亮度样本速率	1 520 640	3 041 280	6 082 560
最大比特率（位每秒）	1 500 000	2 000 000	2 500 000
BBV缓冲区大小（位）	1 507 328	2 015 232	2 506 752
解码图像缓冲区大小（帧）	15	15	15

级4.0.30和4.0.60的参数限制见表B.5。

表B.5　级 4.0.30 和 4.0.60 的参数限制

参数	级	
	4.0.30	4.0.60
每行最大样本数	720	720
每幅图像最大行数	576	576
每秒最大图像数	30	60
每帧最大条带数	32	32
亮度样本速率	10 368 000	24 883 200
最大比特率（位每秒）	10 000 000	20 000 000
BBV缓冲区大小（位）	10 010 624	20 004 864
解码图像缓冲区大小（帧）	15	15

级6.0.30和6.2.30的参数限制见表B.6。

表B.6　级 6.0.30 和 6.2.30 的参数限制

参数	级	
	6.0.30	6.2.30
每行最大样本数	2 048	2 048
每幅图像最大行数	1 152	1 152
每秒最大图像数	30	30
每帧最大条带数	64	64
亮度样本速率	66 846 720	66 846 720
最大比特率（位每秒）	25 000 000	100 000 000
BBV缓冲区大小（位）	25 001 984	100 007 936
解码图像缓冲区大小（帧）	Min(13369344/(MinCuWidth × MiniSize × MinCuHeight × MiniSize),16)-1	

级6.0.60和6.2.60的参数限制见表B.7。

表B.7　级 6.0.60 和 6.2.60 的参数限制

参数	级	
	6.0.60	6.2.60
每行最大样本数	2 048	2 048
每幅图像最大行数	1 152	1 152
每秒最大图像数	60	60
每帧最大条带数	64	64
亮度样本速率	133 693 440	133 693 440
最大比特率（位每秒）	40 000 000	160 000 000
BBV缓冲区大小（位）	40 009 728	160 006 144
解码图像缓冲区大小（帧）	Min(13369344/(MinCuWidth × MiniSize × MinCuHeight × MiniSize),16)-1	

级6.0.120和6.2.120的参数限制见表B.8。

表B.8　级 6.0.120 和 6.2.120 的参数限制

参数	级	
	6.0.120	6.2.120
每行最大样本数	2 048	2 048
每幅图像最大行数	1 152	1 152
每秒最大图像数	120	120
每帧最大条带数	64	64
亮度样本速率	267 386 880	267 386 880
最大比特率（位每秒）	60 000 000	240 000 000
BBV缓冲区大小（位）	60 014 592	240 009 216
解码图像缓冲区大小（帧）	Min(13369344/(MinCuWidth × MiniSize × MinCuHeight × MiniSize),16)-1	

级8.0.30和8.2.30的参数限制见表B.9。

表B.9　级 8.0.30 和 8.2.30 的参数限制

参数	级	
	8.0.30	8.2.30
每行最大样本数	4 096	4 096
每幅图像最大行数	2 304	2 304
每秒最大图像数	30	30
每帧最大条带数	128	128
亮度样本速率	283 115 520	283 115 520

表 B.9（续）

参数	级别	
	8.0.30	8.2.30
最大比特率（位每秒）	40 000 000	160 000 000
BBV缓冲区大小（位）	40 009 728	160 006 144
解码图像缓冲区大小（帧）	Min(56623104/(MinCuWidth × MiniSize × MinCuHeight × MiniSize),16)-1	

级8.0.60和8.2.60的参数限制见表B.10。

表B.10　级 8.0.60 和 8.2.60 的参数限制

参数	级	
	8.0.60	8.2.60
每行最大样本数	4 096	4 096
每幅图像最大行数	2 304	2 304
每秒最大图像数	60	60
每帧最大条带数	128	128
亮度样本速率	566 231 040	566 231 040
最大比特率（位每秒）	60 000 000	240 000 000
BBV缓冲区大小（位）	60 014 592	240 009 216
解码图像缓冲区大小（帧）	Min(56623104/(MinCuWidth × MiniSize × MinCuHeight × MiniSize),16)-1	

级8.0.120和8.2.120的参数限制见表B.11。

表B.11　级 8.0.120 和 8.2.120 的参数限制

参数	级	
	8.0.120	8.2.120
每行最大样本数	4 096	4 096
每幅图像最大行数	2 304	2 304
每秒最大图像数	120	120
每帧最大条带数	128	128
亮度样本速率	1 132 462 080	1 132 462 080
最大比特率（位每秒）	120 000 000	480 000 000
BBV缓冲区大小（位）	120 012 800	480 002 048
解码图像缓冲区大小（帧）	Min(56623104/(MinCuWidth × MiniSize × MinCuHeight × MiniSize),16)-1	

级10.0.30和10.2.30的参数限制见表B.12。

表B.12 级 10.0.30 和 10.2.30 的参数限制

参数	级	
	10.0.30	10.2.30
每行最大样本数	8 192	8 192
每幅图像最大行数	4 608	4 608
每秒最大图像数	30	30
每帧最大条带数	256	256
亮度样本速率	1 069 547 520	1 069 547 520
最大比特率（位每秒）	60 000 000	240 000 000
BBV缓冲区大小（位）	60 014 592	240 009 216
解码图像缓冲区大小（帧）	Min(213909504/(MinCuWidth × MiniSize × MinCuHeight × MiniSize),16)-1	

级10.0.60和10.2.60的参数限制见表B.13。

表B.13 级 10.0.60 和 10.2.60 的参数限制

参数	级	
	10.0.60	10.2.60
每行最大样本数	8 192	8 192
每幅图像最大行数	4 608	4 608
每秒最大图像数	60	60
每帧最大条带数	256	256
亮度样本速率	2 139 095 040	2 139 095 040
最大比特率（位每秒）	120 000 000	480 000 000
BBV缓冲区大小（位）	120 012 800	480 002 048
解码图像缓冲区大小（帧）	Min(213909504/(MinCuWidth × MiniSize × MinCuHeight × MiniSize),16)-1	

级10.0.120和10.2.120的参数限制见表B.14。

表B.14 级 10.0.120 和 10.2.120 的参数限制

参数	级	
	10.0.120	10.2.120
每行最大样本数	8 192	8 192
每幅图像最大行数	4 608	4 608
每秒最大图像数	120	120
每帧最大条带数	256	256
亮度样本速率	4 278 190 080	4 278 190 080
最大比特率（位每秒）	240 000 000	800 000 000
BBV缓冲区大小（位）	240 009 216	800 014 336
解码图像缓冲区大小（帧）	Min(213909504/(MinCuWidth × MiniSize × MinCuHeight × MiniSize),16)-1	

某一级下BBV缓冲区大小是16384位的倍数。

与表B.4到表B.10有关的语法元素包括：horizontal_size、vertical_size、frame_rate_code、bbv_buffer_size。

一幅图像中二元符号的最大个数应小于或等于：

当前图像的二进制位数×1.8 + (horizontal_size × vertical_size × (BitDepth + (BitDepth>>1))))>>5

其中，一幅图像的二元符号包括decode_decision、decode_aec_stuffing_bit、decode_bypass过程解码的二元符号。

附　录　C
（规范性附录）
位流参考缓冲区管理

C.1　概述

本附录定义了位流参考缓冲区管理（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’，编码数据应按C.3.2.1定义的方式移出BBV缓冲区；如果low_delay的值为‘1’，编码数据应按C.3.2.2定义的方式移出BBV缓冲区。符合本部分的位流在每幅图像的解码时刻均不应导致BBV缓冲区下溢。

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：

$$R_{max} = \mathrm{BitRate} \times 400 \qquad (C.1)$$

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幅图像图像的编码数据$f(n)$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 $b(n)$，$b(n)$与本幅图像起始码之间不应有其他图像的数据。

b） 本幅图像的所有编码数据。

c） 跟在本幅图像头后的图像显示扩展。

d） 跟在本幅图像后的填充数据、视频序列结束码。

如果BbvDelay的值不等于BbvDelayMax，第n幅图像进入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幅图像的起始码后第1个位到第n+1幅图像的起始码后第1个位之间所有的位数；

$\tau(n)$ ——第n幅图像的BbvDelay的值，单位为秒（s）；

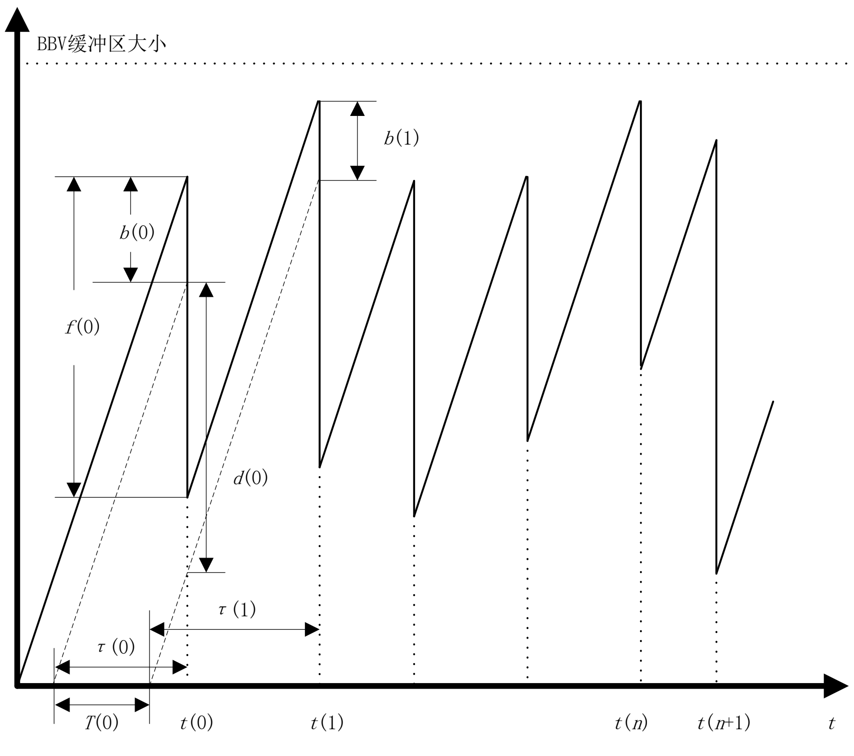
$t(n)$ ——第n幅图像的编码数据从BBV缓冲区移出的时间，单位为秒（s）；

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

$R(n)$ ——第n幅图像进入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幅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，序列头后开始的若干幅图像缺少对应的输出图像。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

如果一幅图像的编码数据后第1个起始码是视频序列结束码，此时无法确定该幅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幅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，每一幅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。输入速率由式（D.2）确定。

所有位流的$R(n)$均不应大于$Rmax$。

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。

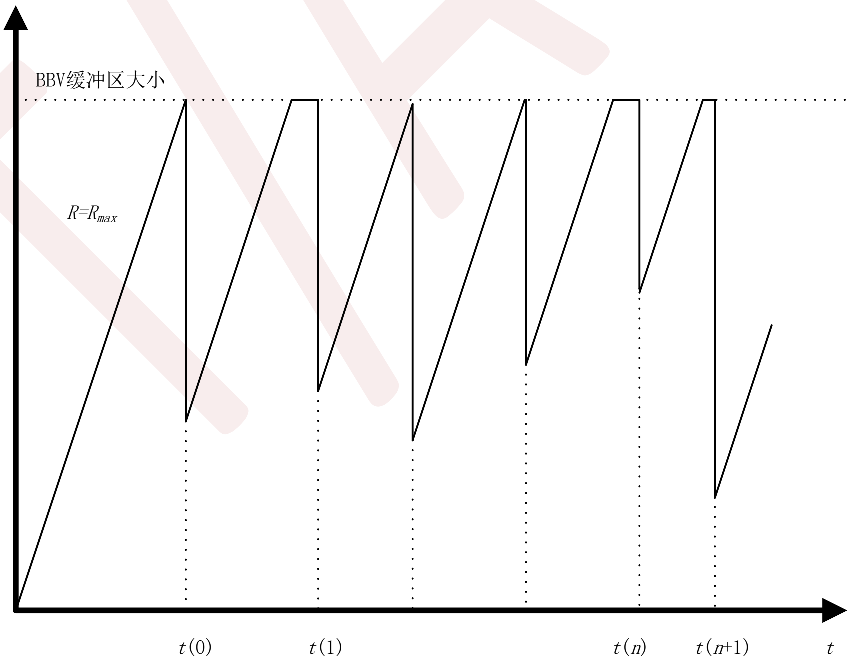
C.3.1.2 方法二

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

a) 如果 BBV 缓冲区没有充满，数据以速率 R_{max}输入缓冲区；

b) 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幅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 概述

第n幅图像的编码数据移出BBV缓冲区的时间为第n幅图像的解码时刻。视频序列的第1幅图像的解码时刻t(0)等于序列第1幅图像的图像起始码进入BBV缓冲区的时刻加上τ(0)。第n幅图像的解码时刻t(n)和编码数据的移出BBV缓冲区的方式根据C.3.2.2和C.3.2.3分别确定。

C.3.2.2 非低延迟

本条定义low_delay的值为‘0’时第n幅图像的解码时刻t(n)和编码数据的移出BBV缓冲区的方式。

第n幅图像的解码时刻t（n）为前一幅图像的解码时刻t(n-1)加上一个检测时间间隔delta(n)。检测时间间隔在C.4定义。

在每幅图像的解码时刻$t(n)$,BBV缓冲区充满度应小于BBS。并且BBV缓冲区充满度$B(n)$应大于等于$f(n)$，否则发生下溢，符合本标准的码流不应发生下溢。

在每幅图像的解码时刻$t(n)$瞬时移出该幅图像的编码数据并解码。

C.3.2.3 低延迟

本条定义low_delay的值为‘1’时第n幅图像的解码时刻t(n)和编码数据的移出BBV缓冲区的方式。

第n幅图像的解码时刻t（n）为前一幅图像的解码时刻t(n-1)加上一个检测时间间隔delta(n)再加上BbvCheckTimes*图像周期。检测时间间隔在C.4定义。如果field_coded_sequence的值为‘1’，则图像周期为帧率倒数的0.5倍，如果field_coded_sequence的值为‘0’，则图像周期为帧率的倒数。

在每幅图像的解码时刻$t(n)$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BbvCheckTimes大于0，当前解码图像定义为大图像。一个视频序列的最后一幅图像不应是大图像。

C.4 缓冲区检测时间间隔

记帧率的倒数为T。

如果第$n-1$幅图像是GB图像，BBV缓冲区检测时间间隔delta(n)等于0；否则，BBV缓冲区检测时间间隔delta(n)由解码完第$n-1$幅图像后输出的图像（见9.2.4）确定。

当 progressive_sequence 的值为‘1’且 field_coded_sequence 的值为‘0’时：

a) 如果该输出图像的 repeat_first_field 的值为‘0’，则 delta(n)等于 T;

b) 如果该输出图像的 repeat_first_field 的值为‘1’且 top_field_first 的值为‘0’,则 delta(n)等于 2T;

c) 如果该输出图像的 repeat_first_field 的值为‘1’且 top_field_first 的值为‘1’,则 delta(n)等于 3T。

当progressive_sequence的值为‘0’且field_coded_sequence的值为‘0’时：

a) 如果该输出图像的 repeat_first_field 的值为‘0’，则 delta(n)等于 T;

b) 如果该输出图像的 repeat_first_field 的值为‘1’，则 delta(n)等于 1.5T。

当progressive_sequence的值为‘0’ 且field_coded_sequence的值为‘1’时，delta(n)等于0.5T。

附 录 D
（规范性附录）
加权量化矩阵

本附录定义序列头位流中加载的4×4和8×8变换块的默认加权量化矩阵$WeightQuantMatrix_{4x4}$和$WeightQuantMatrix_{8x8}$。

$$WeightQuantMatrix_{4x4} = \begin{bmatrix} 64 & 64 & 64 & 68 \\ 64 & 64 & 68 & 72 \\ 64 & 68 & 76 & 80 \\ 72 & 76 & 84 & 96 \end{bmatrix}$$

$$WeightQuantMatrix_{8x8} = \begin{bmatrix} 64 & 64 & 64 & 64 & 68 & 68 & 72 & 76 \\ 64 & 64 & 64 & 68 & 72 & 76 & 84 & 92 \\ 64 & 64 & 68 & 72 & 76 & 80 & 88 & 100 \\ 64 & 72 & 76 & 80 & 84 & 92 & 100 & 112 \\ 68 & 72 & 80 & 84 & 92 & 104 & 112 & 128 \\ 76 & 80 & 84 & 92 & 104 & 116 & 132 & 152 \\ 96 & 100 & 104 & 116 & 124 & 140 & 164 & 188 \\ 104 & 108 & 116 & 128 & 152 & 172 & 192 & 216 \end{bmatrix}$$

附　录　E
（规范性附录）
扫描表

本附录定义扫描表ScanCGInBlk，ScanCoeffInCG、InvScanCoeffInCG、InvScanCGInBlk和InvScanCoeffInBlk。

扫描表ScanCGInBlk[i][j][x][y]的定义见表E.1～表E.7。

表E.1　扫描表 ScanCGInBlk[1][1][x][y]

y 的值	x 的值	
	0	1
0	0	1
1	2	3

表E.2　扫描表 ScanCGInBlk[2][0][x][y]

y 的值	x 值			
	0	1	2	3
0	0	1	2	3

表E.3　扫描表 ScanCGInBlk[0][2][x][y]

y 的值	x 的值
	0
0	0
1	1
2	2
3	3

表E.4　扫描表 ScanCGInBlk[2][2][x][y]

y 的值	x 的值			
	0	1	2	3
0	0	1	5	6
1	2	4	7	12
2	3	8	11	13
3	9	10	14	15

表E.5 扫描表 ScanCGInBlk[3][1][x][y]

y 的值	x 的值							
	0	1	2	3	4	5	6	7
0	0	1	4	5	8	9	12	13
1	2	3	6	7	10	11	14	15

表E.6 扫描表 ScanCGInBlk[1][3][x][y]

y 的值	x 的值	
	0	1
0	0	1
1	2	4
2	3	5
3	6	8
4	7	9
5	10	12
6	11	13
7	14	15

表E.7 扫描表 ScanCGInBlk[3][3][x][y]

y 的值	x 的值							
	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

扫描表 ScanCoeffInCG[x][y]的定义见表 E.8。

表E.8 扫描表 ScanCoeffInCG[x][y]

y 的值	x 的值			
	0	1	2	3
0	0	1	5	6
1	2	4	7	12
2	3	8	11	13
3	9	10	14	15

生成扫描表 InvScanCoeffInCG[i][j]、InvScanCGInBlk[idxW][idxH][i][j] 和 InvScanCoeffInBlk[idxW][idxH][i][j] 的过程如下（其中 idxW 等于 Log(W)－2，idxH 等于 Log(H)-2，W×H 的取值为 4×4、4×16、16×4、8×8、8×32、32×8、16×16 或 32×32）：

```
for(x=0; x<W; x++)
{
    for(y=0; y<H; y++)
    {
        cgx = x >> 2
        cgy = y >> 2
        posx = x  -  (cgx << 2)
        posy = y  -  (cgy << 2)
        if (W == 4 && H == 4) {
            scanpos = ScanCoeffInCG[posx][posy]
            InvScanCoeffInCG[scanpos][0] = posx
            InvScanCoeffInCG[scanpos][1] = posy
        }
        else {
            scanpos = (ScanCGInBlk[idxW][idxH][cgx][cgy]<<4)+ScanCoeffInCG[posx][posy]
            cgscanpos = ScanCGInBlk[idxW][idxH][cgx][cgy]
            InvScanCGInBlk[idxW][idxH][cgscanpos][0] = cgx
            InvScanCGInBlk[idxW][idxH][cgscanpos][1] = cgy
        }
        InvScanCoeffInBlk[idxW][idxH][scanpos][0] = x
        InvScanCoeffInBlk[idxW][idxH][scanpos][1] = y
    }
}
```

附 录 F
（资料性附录）
高级熵编码器解码器参考描述方法

本附录描述了高级熵编码解码器的参考实现方法，包括decode_aec_stuffing_bit和decode_bypass过程的参考描述方法。

decode_aec_stuffing_bit过程的输入是valueD、bFlag、rS1、rT1、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
   if (valueD || (bFlag == 1 && rS1 == boundS ) ) {
      rS1 = 0
      valueS = 0
      while ( valueT < 0x100 && valueS < boundS ) {
         valueS++
         valueT = (valueT << 1 ) | read_bits(1)
      }
      if ( valueT < 0x100 )
         bFlag = 1
      else
         bFlag = 0
      valueT = valueT & 0xFF
   }
   if ( rT1 ) {
      rS2 = rS1
      rT2 = rT1 - 1
   }
   else {
      rS2 = rS1 + 1
      rT2 = 255
   }
   if ( rS2 > valueS || (rS2 == valueS && valueT >= rT2) && bFlag == 0) {
      binVal = ! predMps
      if ( rS2 == valueS )
         valueT = valueT  -  rT2
      else
         valueT = 256 + ((valueT << 1) | read_bits(1))  -  rT2
      valueT = (valueT << 8 ) | read_bits(8)
      rT1 = 0
```

```
            valueD = 1
        }
        else {
            binVal = predMps
            rS1 = rS2
            rT1 = rT2
            valueD = 0
         }
        return (binVal)
    }
```

decode_bypass过程的输入是valueD、bFlag、rS1、rT1、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

        if (valueD || (bFlag == 1 && rS1 == boundS ) ) {
            rS1 = 0
            valueD = 1
            valueT = (valueT << 1 ) | read_bits(1)
            if ( valueT >=(256 + rT1) ) {
                binVal = ! predMps
                valueT -=(256 + rT1)
             }
             else
            binVal = predMps
        }
        else {
            rS2= rS1+1
            if( rS2 > valueS || (rS2 == valueS && valueT >= rT1) && bFlag == 0 ) {
            binVal = ! predMps
            if ( rS2 == valueS )
               valueT = valueT - rT1
            else
               valueT = 256 + ((valueT << 1 ) | read_bits(1)) - rT1
            rS1 = 0
            valueD = 1
         }
         else {
             binVal = predMps
             rS1= rS2
             valueD = 0
```

```
        }
    }
    return (binVal)
}
```