ICS 35.040.40
CCS L 64

团 体 标 准

T/AI 129.4—2026

信息技术 感知无损压缩 第4部分：专业制作图像

Information technology - Perceptual lossless compression - Part 4: Professional production image

2026－01－30 发布 2026－01－30 实施

中关村视听产业技术创新联盟 发布

目　　次

前　言

本文件按照GB/T 1.1—2020《标准化工作导则　第1部分：标准化文件的结构和起草规则》的规定起草。

本文件是T/AI 129《信息技术 感知无损压缩》的第4部分。T/AI 129已经发布了以下部分：

——第1部分：图像；

——第4部分：专业制作图像。

本文件由数字音视频编解码技术标准工作组提出。

本文件由中关村视听产业技术创新联盟归口。

本文件起草单位：中央广播电视总台、北京大学、华为技术有限公司、成都索贝数码科技股份有限公司、深圳市大疆创新科技有限公司、国家广播电影电视总局广播电视规划院、上海海思技术有限公司、北京博雅睿视科技有限公司、中兴通讯股份有限公司、鹏城实验室。

本文件主要起草人：姜文波、马思伟、潘晓菲、郑萧桢、张嘉琪、赵寅、毛珏、王苫社、杨海涛、张伟民、宋泽田、林翔宇、柳鑫、韩旭、张金沙、郑建宏、黄成、杨明佳、蒋孝淳、贾川民、魏建超、李振纲、曾幸、孟现东、赵海英、崔晓冉、黄铁军、高文。

引　言

T/AI 129旨在确立智能媒体压缩的方法，拟由四部分构成。

——第1部分：图像。目的在于确立智能媒体感知无损图像压缩方法。

——第2部分：符合性。目的在于确立智能媒体感知无损图像压缩的符合性测试方法。

——第3部分：参考软件。目的在于确立智能媒体感知无损图像压缩的软件实现方法。

——第4部分：专业制作图像。目的在于确立智能媒体感知无损专业制作图像压缩方法。

本文件的发布机构请注意，声明符合本文件时，可能涉及如下28项与数字视频编解码技术相关的专利的使用。专利申请号及名称如下：

序号	专利申请号	专利名称	相关章、条
1	CN202210687517.7	视频压缩码率控制的方法及装置	9.4.2.1
2	CN202310316448.3	处理视频数据的方法、编码方法、电子设备及存储介质	9.4.3.2
3	PCT/RU2025/000052	SCANNING METHODS AND APPARATUSES FOR PROCESSING IMAGE DATA	9.4
4	PCT/RU2025/000080	Efficient compression method for significance flags coding using coefficient regrouping by frequency components of transform block within block group	9.5
5	PCT/RU2025/000048	Coding Methods and Related Apparatuses	7.2.9.5
6	CN202510122962.2	编码方法、解码方法及相关装置	6.1,9.3,A.3
7	RU2025104720	一种系数编码方法	7.1,7.2,8.3
8	CN202510241354.3	编码方法、解码方法及装置、相应的编码器和解码器	5.9,7.1,7.2, 8.3
9	RU2025104627	一种特殊模式 下系数幅值二 值化编码方法	5.9,7.1,7.2, 8.3
10	CN202510122970.7	编码方法、解码方法及相关装置	5.9,7.1,7.2,9.4
11	CN202510241362 .8	编码方法、解码方法及相关装置	7.1,7.2,8.1,9.4
12	CN202510246569 .4	编码方法、解码方法及相关装置	7.1,7.2,9.4
13	CN202510246092 .X	编码方法、解码方法及相关装置	7.2,8.3

14	CN202510246563.7	编码方法、解码方法及相关装置	7.1,7.2,8.1
15	CN202510121319.8	解码方法、编码方法及装置、相应的编码器和解码器	9.4
16	CN202510123846.2	解码方法、编码方法及装置、相应的编码器和解码器	9.4,9.2,9.3
17	CN202510247510.7	解码方法、编码方法及装置、相应的编码器和解码器	7.2,9.4,9.3
18	CN202510239187.9	编码方法、解码方法及装置、相应的编码器和解码器	6.1,7.2,9.4,9.5
19	CN202510242244.9	编码方法、解 码方法及装置、相应的编码器和解码器	6.1,9.4,9.5
20	CN202510239268.9	编码方法、解码方法及相关装置	7.1,7.2,8.1
21	CN202510123226.9	解码方法、编码方法及装置、相应的编码器和解码器	9.7,9.4
22	CN202510121637.4	解码方法、编 码方法及装置、相应的编码器和解码器	9.4
23	CN202510120989.8	编码方法、解码方法及相关装置	9.4,9.5,9.6
24	CN202510121218.0	编码方法、解码方法及相关装置	9.4,9.5,9.6
25	CN202510121229.9	编码方法、解码方法及相关装置	5.9,6.1,7.1,7.2,9.5,9.6
26	CN202510240720.3	编码方法、解码方法及相关装置	7.1,7.2
27	CN202510892586.5	编码方法、解码方法及相关装置	C.2,C.3
28	CN202510941310.1	编码方法、解码方法及相关装置	B.2,3,4

本文件的发布机构对上述专利的真实性、有效性和范围无任何立场。

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

联 系 人：黄铁军（数字音视频编解码技术标准工作组秘书长）
通讯地址：北京大学理科2号楼2641室
邮政编码：100871
电子邮件：tjhuang@pku.edu.cn
电话：+8610-62756172
传真：+8610-62751638

网址：http://www.avs.org.cn

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

信息技术 感知无损压缩 第4部分：专业制作图像

1 范围

本文件规定了面向制作域场景的低复杂度视频编码的码流结构与解码过程。

本文件适用于制作域的视频采集、编辑制作、存储、传输等。

2 规范性引用文件

下列文件中的内容通过文中的规范性引用而构成本文件必不可少的条款。其中，注日期的引用文件，仅该日期对应的版本适用于本文件；不注日期的引用文件，其最新版本（包括所有的修改单）适用于本文件。

GB/T 33475.2—2024 信息技术 高效多媒体编码 第2部分:视频

GB/T 46269.1—2025 高动态范围（HDR）视频技术 第1部分:元数据及适配

ITU-T H.273 用于视频信号类型识别的独立于编码的代码点/Coding-independent code points for video signal type identification

3 术语和定义

GB/T 33475.2—2024界定的以及下列术语和定义适用于本文件。

3.1

子图 subpicture

图像中W×H的矩形区域，由图像在水平（和垂直）划分得到。

注：子图的解码过程之间相互独立。

3.2

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。

3.3

小波系数 wavelet coefficient

子图中的像素经一次小波水平变换和一次小波垂直变换产生的数值。

3.4

子带 band

子图经一次小波水平变换和一次小波垂直变换后得到的一个小波系数矩阵。

3.5

样本 sample

构成矩阵的基本元素。

3.6

样值 sample value

样本的幅值。

3.7

编码单元 coding unit

包括一个宽为M高为N的亮度样值矩阵和对应的色度样值矩阵，由子带划分得到。

3.8

宏块 macro block

编码单元的三个样值矩阵（亮度和两个色度）中的一个矩阵。

3.9

块 block

一个宽为M高为N的样值矩阵。

3.10

变换系数 transform coefficient

离散正弦或离散余弦变换域上的一个标量。

3.11

变换块 transform block

一个宽为M高为N的变换系数矩阵，由宏块对应的变换系数构成。

3.12

反变换 inverse transform

将变换系数转换成小波系数的过程。

3.13

反量化 dequantization

对量化系数缩放后得到变换系数或小波系数的过程。

3.14

量化系数 quantization coefficient

反量化前变换系数或小波系数的值。

3.15

残差块 residual block

一个M×N的残差矩阵，由宏块对应的残差构成。

3.16

重建样本 reconstructed sample

由解码器根据码流解码得到并构成解码图像的样本。

3.17

预测单元 prediction unit

由一个亮度预测块和对应的色度预测块组成。

3.18

帧内编码　intra coding

使用帧内预测对编码单元或子带进行编码。

3.19

帧内预测　intra prediction

在相同解码LL子带中使用先前解码的样值生成当前样本预测值的过程。

3.20

运动矢量　motion vector

用于帧间预测的二维矢量，由当前LL子带指向参考子图，其值为当前块和参考块之间的坐标偏移量。

3.21

帧间预测　inter prediction

使用先前解码LL子带生成当前LL子带样本预测值的过程。

3.22

语法元素　syntax element

码流中的数据单元解析后的结果。

3.23

参考子图　reference subpicture

解码过程中用于后续LL子带帧间预测的低分辨率图。

4 缩略语

下列缩略语适用于本文件。

BAC　二进制算术编码（Binary Arithmetic Coding）
CICP 编码无关码点（Coding-Independent Code Points）
LSB　最低有效位（Least Significant Bit）
MSB　最高有效位（Most Significant Bit）
MB　宏块（Macro Block）
MV　运动矢量（Motion Vector）
QP　量化参数（Quantization Parameter）
TB　变换块（Transform Block）
VLC　变长编码（Variable Length Coding）

5 约定

5.1 概述

本文件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

算术运算符说明见表1。

表1 算术运算符说明

算术运算符	说明
+	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
$\sum_{i=a}^{b} f(i)$	自变量i取由a到b（含b）的所有整数值时，函数 $f(i)$ 的累加和
a % b	模运算，a除以b的余数，其中a与b都是正整数

5.3 逻辑运算符

逻辑运算符说明见表2。

表2 逻辑运算符说明

逻辑运算符	说明
a && b	a和b之间的与逻辑运算
a || b	a和b之间的或逻辑运算
!	逻辑非运算
express ? a : b	如果表达式*express*的结果为真或不为0，则使用a进行赋值；否则，使用b进行赋值

5.4 关系运算符

关系运算符说明见表3。

表3 关系运算符说明

关系运算符	说明
>	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

位运算符说明见表4。

表4 位运算符说明

位运算符	说明
&	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

赋值运算说明见表5。

表5 赋值运算说明

赋值运算	说明
=	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（4）。

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

式中：

x ——自变量x。

$$\mathrm{clip}(i,j,x) = \begin{cases} i; x < i \\ j; x > j \\ x; i \le x \le j \end{cases} \quad \cdots\cdots (2)$$

式中：

x ——自变量x；

i ——下界；

j ——上界。

$$\min = \begin{cases} x; & x <= y \\ y; & x > y \end{cases} \quad \cdots\cdots (3)$$

式中：

x ——自变量x；

y ——自变量y。

$$\max = \begin{cases} x; & x >= y \\ y; & x < y \end{cases} \quad \cdots\cdots (4)$$

式中：

x ——自变量x；

y ——自变量y。

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 码流语法、解析过程和解码过程的描述方法

5.9.1 码流语法的描述方法

码流语法描述方法类似C语言。码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文件的码流语法的超集，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，表示从码流中读一个语法元素数值，并且码流指针前进指向语法元素之后的下一个位置。语法表中，语法元素解析可能涉及到一个或多个码流指针，通过表7中的“码流指针ID”列来指示语法元素对应的码流指针ID，例如，码流指针ID的取值为1对应于1号码流指针。

表7 语法描述的伪代码示例

伪代码	码流指针ID	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	1	ce(v)
conditioning statement		
/*花括号括起来的语句组是复合语句，在功能上视作单个语句。*/		
{		
statement		
…		
}		
/*“while”语句测试condition是否为TRUE，如果为TRUE，则重复执行循环体，直到condition不为TRUE。*/		
while (condition)		
statement		

表 7 语法描述的伪代码示例（续）

伪代码	码流指针ID	描述符
/*“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		
/*“break”语句用于do-while、while和for循环体中，可使当前循环体立即终止循环。*/		
break		

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码流指针，这个指针指向码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。

5.9.2.2 byte_aligned(n)

如果n号码流指针的当前位置是字节对齐的，返回TRUE，否则返回FALSE。

5.9.2.3 read_bits(n)

返回码流的随后n个二进制位，MSB在前，同时码流指针前移n个二进制位。如果n等于0，则返回0，码流指针不前移。

函数也用于解析过程和解码过程的描述。

5.9.2.4 set_bitstream_pointer1(n)

将1号码流指针从当前帧码流对应的picture_header()语法结构体之后的位置前进n个字节。

5.9.2.5 set_bitstream_pointer2(n)

将2号码流指针从当前帧码流对应的picture_header()语法结构体之后的位置前进n个字节。

5.9.2.6 set_bitstream_pointer3(n)

将3号码流指针从当前帧码流对应的picture_header()语法结构体之后的位置前进n个字节。

5.9.2.7 set_bitstream_pointer4(n)

将4号码流指针从当前帧码流对应的picture_header()语法结构体之后的位置前进n个字节。

5.9.2.8 set_bitstream_pointer5(n)

将5号码流指针从当前帧码流对应的picture_header()语法结构体之后的位置前进n个字节。

5.9.3 描述符

描述符表示不同语法元素的解析过程，见表8。

表8 描述符

描述符	说明
ae(v)	二元语法元素，用基于上下文的BAC。解析过程在8.1中定义
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
ce(v)	变长编码的语法元素。解析过程在8.3中定义
b(n)	一个任意取值的n个二进制位。解析过程由函数read_bits(n)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
r(n)	连续n个“0”。解析过程由函数read_bits(n)的返回值规定
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

5.9.4 保留、禁止和标记位

本文件定义的码流语法中，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文件的扩展。这些值不应出现在符合本文件的码流中。

“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文件的码流中。

码流中的“保留位”（reserved_bits）表明保留了一些语法单元用于将来对本文件的扩展，解码处理应忽略这些位。

6 编码码流的结构

6.1 概述

视频序列是码流的最高层语法结构，整体码流结构见图1。视频序列由第一个序列头开始，每个序列头后面跟着序列数据，序列数据包括N个编码图像，N是8位无符号整数。序列的第一帧为帧内编码图像，后续帧允许为帧间编码图像或帧内编码图像，帧间编码图像必须参考第一帧。每幅图像包含图像头和图像数据。编码图像在码流中按码流顺序排列，码流顺序应与解码顺序相同。解码顺序与显示顺序相同。图像数据包含一个或多个子图。子图数据包含子图头信息、低频子带数据和高频子带数据。当含alpha通道时，子图数据还包括alpha通道数据。低频子带数据包含低频子带BAC数据和低频子带VLC数据。高频子带数据包含高频子带BAC数据和高频子带VLC数据。

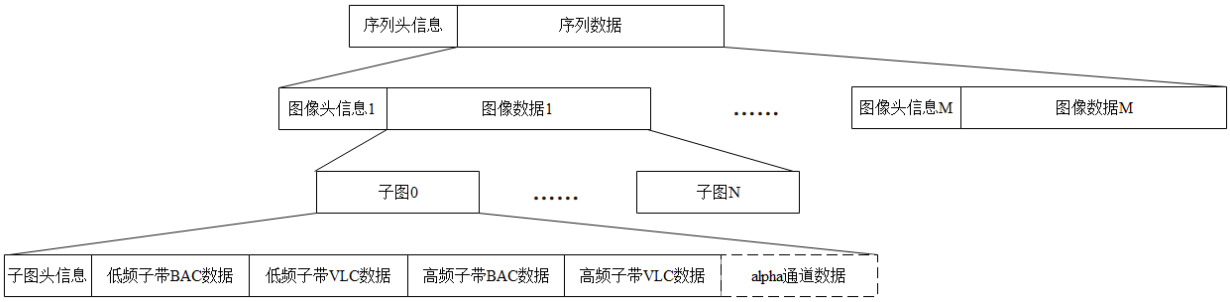

图1 序列码流结构示意图

6.2 序列

本文件支持两种序列：逐行序列和隔行序列。对于隔行序列，序列中的相邻二幅图像分别为顶场和底场，序列中的每一幅图像的帧类型均为I帧。

6.3 图像

6.3.1 概述

一幅图像的编码数据由图像头开始。图像数据包含一个或多个子图数据。

图像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，或Co和Cg）。样本矩阵元素的值为整数。亮色度之间的关系，包括原始信号的色度和转移特性等可在码流中定义，这些信息不影响解码过程。图像的解码处理包括解析过程和解码过程。

6.3.2 图像格式

6.3.2.1 YUV422 格式

对于YUV422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。

6.3.2.2 YUV444 格式

对于YUV444格式，Cb和Cr矩阵(Co和Cg矩阵)在水平和垂直方向的尺寸都与Y矩阵相同。

6.4 子图

子图是图像中的矩形区域，子图之间不应重叠。每个子图数据可独立解码，码流中子图按光栅扫描顺序排序。一种可能的子图结构见图2，解码顺序为子图0，子图1，子图2，子图3，子图4，子图5。

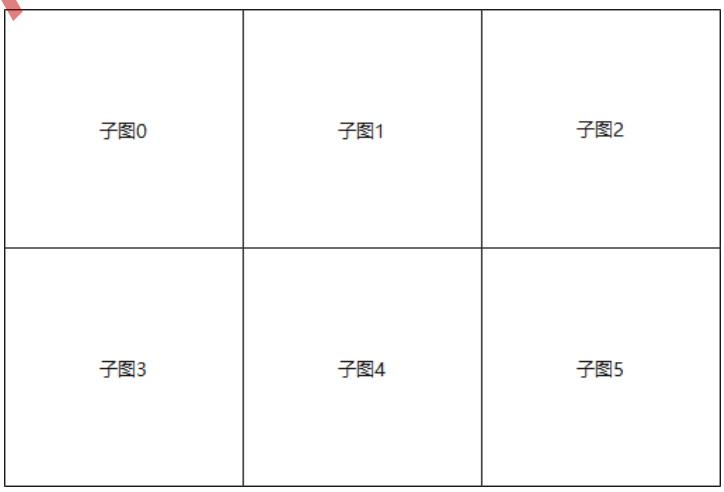

图2 子图结构

6.5 子带

子带包括低频子带（LL子带），三个高频子带（HL子带、LH子带、HH子带）。低频子带数据与高频子带数据可独立解码。HL子带表示水平变换的高频分量，垂直变换的低频分量的子带。LH子带表示水平变换的低频分量，垂直变换的高频分量的子带。HH子带表示水平变换的高频分量，垂直变换的高频分量的子带。

小波分解结构图见图3。一个子图生成4个子带。

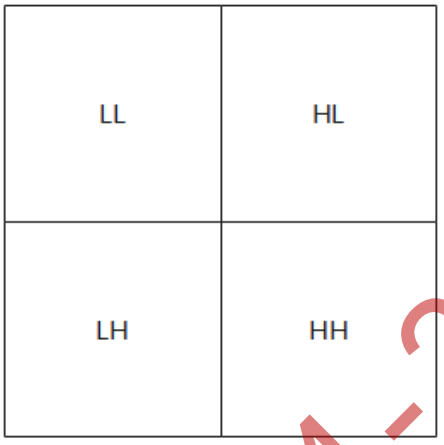

图3 小波分解结构

6.6 低频子带编码单元

低频子带编码单元由LL子带中宽为8高为8的亮度宏块及对应的色度宏块组成。

6.7 高频子带编码单元

高频子带编码单元由HL子带或LH子带或HH子带中宽为8高为8的亮度宏块及对应的色度宏块组成。

7 码流的语法和语义

7.1 语法描述

7.1.1 序列语法

7.1.1.1 序列定义

序列定义见表9。

表9 序列定义

序列定义	码流指针 ID	描述符
sequence () {		
sequence_header( )		
while(NumOfFrame--) {		
picture( )		
}		
}		

7.1.1.2 序列头定义

序列头定义见表10。

表10 序列头定义

序列头定义	码流指针 ID	描述符
sequence_header () {		
profile_idc	1	u(8)
level_idc	1	u(8)
num_of_frames_minus1	1	u(8)
NumOfFrame = num_of_frames_minus1 + 1		
frame_rate	1	u(8)
input_picture_width	1	u(16)
input_picture_height	1	u(16)
sub_pic_width_in_128_minus2	1	u(8)
sub_pic_height_in_128_minus1	1	u(8)
bit_depth_minus8	1	u(4)
chroma_format	1	u(4)
interlace_mode	1	u(2)
yuv444_packed_by_yuv422_flag	1	u(1)
for(i = 0; i < 69; i ++) {		
seq_header_reserved_flag[i]	1	u(1)
}		
rendering_information()		
}		

rendering_information()的定义见表 11

表11 渲染信息定义

渲染信息定义	码流指针 ID	描述符
rendering_information () {		
cicp_info_present_flag	1	u(1)
mdcv_info_present_flag	1	u(1)
dm_present_flag	1	u(1)

表 11 渲染信息定义（续）

渲染信息定义	码流指针 ID	描述符
render_reserved	1	u(5)
if(cicp_info_present_flag)		
cicp_rendering_info()		
if(mdcv_info_present_flag)		
hdr_static_metadata()		
if(dm_present_flag)		
dynamic_metadata()		
}		

cicp_rendering_info ()的定义见表 12

表12 色彩信息定义

色彩信息定义	码流指针 ID	描述符
cicp_rendering_info () {		
colour_primaries	1	u(8)
transfer_characteristics	1	u(8)
matrix_coefficients	1	u(8)
video_full_range_flag	1	u(1)
cicp_reserved	1	u(7)
}		

dynamic_metadata()的定义见表 13

表13 HDR 动态元数据信息定义

HDR 动态元数据信息定义	码流指针 ID	描述符
dynamic_metadata () {		
dm_type	1	u(8)
dm_size	1	u(16)
for(i = 0; i < dm_size; i++) {		
dm_data_byte[i]	1	b(8)

表13 HDR动态元数据信息定义（续）

HDR 动态元数据信息定义	码流指针 ID	描述符
}		
}		

7.1.2 图像语法

7.1.2.1 图像定义

图像定义见表14。

表14 图像定义

图像定义	码流指针 ID	描述符
picture () {		
picture_header(frame_idx)		
frameDataSize = 8		
curSubPicHeight = SubPictureHeight		
for (i = 0; i < NumSubPictureVer; ++i) {		
curSubPicWidth = SubPictureWidth		
for (j = 0; j < NumSubPictureHor; ++j) {		
if (j == NumSubPictureHor – 1) {		
curSubPicWidth = CodedPictureWidth – SubPictureWidth * j		
}		
if (i == NumSubPictureVer – 1) {		
curSubPicHeight = CodedPictureHeight – SubPictureHeight * i		
}		
subPicIdx = i * NumSubPictureHor + j		
BandWidth = curSubPicWidth >> 1		
BandHeight = curSubPicHeight>> 1		
sub_pic_info (subPicIdx)		
sub_pic_data(subPicIdx, curSubPicWidth >> 1, curSubPicHeight>> 1)		
frameDataSize += subpic_len		
}		
}		

表14 图像定义（续）

图像定义	码流指针 ID	描述符
while (frameDataSize < picture_len) {		
frame_zero_byte	1	u(8)
}		
}		

7.1.2.2 图像头及子图头定义

图像头定义见表15。

表15 图像头定义

图像头定义	码流指针 ID	描述符
picture_header () {		
picture_len	1	u(32)
frame_type	1	u(1)
alpha_map_flag	1	u(1)
alpha_map_16bit_flag	1	u(1)
alpha_map_code_mode	1	u(4)
mb_qp_delta_enabled_flag	1	u(1)
hf_transform_skip_enable_flag	1	u(1)
cclm_enable_flag	1	u(1)
pic_output_flag	1	u(1)
pic_header_reserved_bits	1	u(21)
}		

子图信息定义见表16。

表16 子图信息定义

子图信息定义	码流指针 ID	描述符
sub_pic_info (idx) {		
subpic_ll_qp_index[idx]	1	u(6)
subpic_hl_qp_index_offset_plus12[idx]	1	u(5)

表 16 子图信息定义（续）

子图信息定义	码流指针 ID	描述符
subpic_lh_qp_index_offset_plus12[idx]	1	u(5)
subpic_hh_qp_index_offset_plus12[idx]	1	u(5)
subpic_cb_qp_index_offset_plus12[idx]	1	u(5)
subpic_cr_qp_index_offset_plus12[idx]	1	u(5)
subpic_reserved_bits[idx]	1	u(9)
subpic_len[idx]	1	u(32)
ll_band_lbac_len[idx]	1	u(32)
ll_band_vlc_len[idx]	1	u(32)
hf_band_lbac_len[idx]	1	u(32)
if (alpha_map_flag) {		
hf_band_vlc_len[idx]	1	u(32)
}		
}		

7.1.3 子图语法

7.1.3.1 子图数据定义

子图数据定义见表17。

表17 子图数据定义

子图数据定义	码流指针 ID	描述符
sub_pic_data (subPicIdx, BandWidth, BandHeight) {		
offset1 = calcOffset(subPicIdx, LL_BAND, 0)		
offset2 = calcOffset(subPicIdx, LL_BAND, 1)		
offset3 = calcOffset(subPicIdx, HF_BAND, 0)		
offset4 = calcOffset(subPicIdx, HF_BAND, 1)		
if (alpha_map_flag) {		
offset5 = calcOffset(subPicIdx, ALPHA_MAP, 0)		
}		
ll_band_data(offset1, offset2, BandWidth, BandHeight)		
hf_band_data(offset3, offset4, BandWidth, BandHeight)		
if (alpha_map_flag) {		
alpha_map_data (offset5, BandWidth << 1, BandHeight << 1)		

表 17 子图数据定义（续）

子图数据定义	码流指针 ID	描述符
}		
}		

7.1.3.2 码流指针偏移量定义

码流指针偏移量定义见表18。

表18 码流指针偏移量定义

码流指针偏移量定义	码流指针 ID	描述符
calcOffset (subPicIdx, band, is_vlc) {		
for (i = 0; i < subPicIdx; ++i) {		
offset += subpic_len[i]		
llBandLbacLen = ll_band_lbac_len[subPicIdx]		
llBandVlcLen = ll_band_vlc_len[subPicIdx]		
hfBandLbacLen = hf_band_lbac_len[subPicIdx]		
if (alpha_map_flag) {		
hfBandVlcLen = hf_band_vlc_len[subPicIdx]		
}		
if (band == HF_BAND)		
offset += llBandLbacLen + llBandVlcLen		
else if (band == ALPHA_MAP)		
offset+= llBandLbacLen + llBandVlcLen + hfBandLbacLen + hfBandVlcLen		
if (is_vlc) {		
lbacLength = (band == HF_BAND) ? hfBandLbacLen : llBandLbacLen		
offset += lbacLength		
}		
}		

7.1.4 低频子带数据定义

低频子带数据定义见表19。

表19 低频子带数据定义

低频子带数据定义	码流指针 ID	描述符
ll_band_data (offset1, offset2, BandWidth, BandHeight) {		

表 19 低频子带数据定义（续）

低频子带数据定义	码流指针 ID	描述符
set_bitstream_pointer1(offset1)		
set_bitstream_pointer2(offset2)		
for (MbY=0; MbY < ((BandHeight + 7) >> 3); MbY++) {		
for (MbX=0; MbX < ((BandWidth + 7) >> 3); MbX ++) {		
ll_band_mb_data(MbX, MbY)		
}		
}		
band_stuffing_bit	1	ae(v)
band_stop_one_bit /* equal to 1 */	1	f(1)
while (!byte_aligned (1)) {		
zero_bit	1	r(1)
}		
while (!byte_aligned (2)) {		
zero_bit	2	r(1)
}		

7.1.5 高频子带数据定义

高频子带数据定义见表20。

表20 高频子带数据定义

高频子带数据定义	码流指针 ID	描述符
hf_band_data (offset3, offset4, BandWidth, BandHeight) {		
set_bitstream_pointer3(offset3)		
set_bitstream_pointer4(offset4)		
for (MbY=0; MbY < ((BandHeight + 7) >> 3); MbY++) {		
for (MbX=0; MbX < ((BandWidth + 7) >> 3); MbX ++) {		
if (mb_qp_delta_enabled_flag)		
hf_mb_qp_delta	4	se(v)
for (BandIdx = 0; BandIdx <3; BandIdx++) {		
hf_band_mb_data(BandIdx, MbX, MbY)		
}		
}		
}		
band_stuffing_bit	3	ae(v)

表 20 高频子带数据定义（续）

高频子带数据定义	码流指针 ID	描述符
band_stop_one_bit /* equal to 1 */	3	f(1)
while (!byte_aligned (3)) {		
zero_bit	3	r(1)
}		
while (!byte_aligned (4)) {		
zero_bit	4	r(1)
}		
}		

7.1.6 低频子带编码单元数据定义

低频子带编码单元数据定义见表21。

表21 低频子带编码单元数据定义

低频子带编码单元数据定义	码流指针 ID	描述符
ll_band_mb_data (x, y) {		
if (mb_qp_delta_enabled_flag) {		
ll_mb_qp_delta	2	se(v)
}		
if (frame_type == 1) {		
mb_mode	1	ae(v)
}		
if (mb_mode == MODE_INTER) {		
inter_no_residual_flag	1	ae(v)
mvd_flag	1	ae(v)
if (mvd_flag) {		
mvd_coding ( )		
}		
luma_tb_size = TB_SIZE8x8		
}		
else {		
luma_tb_size	1	ae(v)
intra_pred_mode_luma_first_flag	1	ae(v)

表 21 低频子带编码单元数据定义（续）

低频子带编码单元数据定义	码流指针ID	描述符
if (intra_pred_mode_luma_first_flag == 0)		
intra_pred_mode_luma_second_flag	1	ae(v)
else		
intra_pred_mode_luma_second_flag = 0		
intra_pred_mode_chroma_first_flag	1	ae(v)
if (intra_pred_mode_chroma_first_flag == 0 || cclm_enable_flag == 1)		
intra_pred_mode_chroma_second_flag	1	ae(v)
else		
intra_pred_mode_chroma_second_flag = 0		
}		
if (mb_mode == MODE_INTRA || ! inter_no_residual_flag) {		
for (comp = 0; comp < 3; comp++) {		
TbSize = (comp == 0) ? luma_tb_size : TbSizeChroma		
tbNum = 1		
if (comp == 0 && luma_tb_size == TB_SIZE4x4) {		
tbNum = 4		
}		
for (k = 0; k < tbNum; k++) {		
decode_coefficients(TbSize)		
}		
}		
}		
}		

低频子带宏块运动矢量差定义见表22。

表22 低频子带宏块运动矢量差定义

低频子带宏块运动矢量差定义	码流指针ID	描述符
mvd_coding ( ) {		
abs_mvd_grt0_flag[0]	1	ae(v)
if (abs_mvd_grt0_flag[0]) {		
abs_mvd_grt1_flag[0]	1	ae(v)
}		

表 22 低频子带宏块运动矢量差定义（续）

低频子带宏块运动矢量差定义	码流指针ID	描述符
if (abs_mvd_grt1_flag[0]) {		
abs_mvd_minus2[0]	2	se(v)
}		
MvDiffX = abs_mvd_grt0_flag[0] + abs_mvd_grt1_flag[0] + abs_mvd_minus2[0]		
if (abs_mvd_grt0_flag[0]) {		
mvd_sign_flag[0]	2	u(1)
MvDiffX = MvDiffX * (1 – 2 * mvd_sign_flag[0])		
}		
if (abs_mvd_grt0_flag[0]) {		
abs_mvd_grt0_flag[1]	1	ae(v)
}		
else {		
abs_mvd_grt0_flag[1] = 1		
}		
if (abs_mvd_grt0_flag[1]) {		
abs_mvd_grt1_flag[1]	1	ae(v)
}		
if (abs_mvd_grt1_flag[1]) {		
abs_mvd_minus2[1]	2	se(v)
}		
MvDiffY = abs_mvd_grt0_flag[1] + abs_mvd_grt1_flag[1] + abs_mvd_minus2[1]		
if (abs_mvd_grt0_flag[1]) {		
mvd_sign_flag[1]	2	u(1)
MvDiffY = MvDiffY * (1 – 2 * mvd_sign_flag[1])		
}		
}		

低频子带宏块量化系数定义见表23。

表23 低频子带宏块量化系数定义

低频子带宏块量化系数定义	码流指针ID	描述符
decode_coefficients (TbSize) {		
MaxNumCoeff = CoefNumInSize[TbSize]		

表 23 低频子带宏块量化系数定义（续）

低频子带宏块量化系数定义	码流指针 ID	描述符
coded_block_flag	1	ae(v)
for (i = 0; i < MaxNumCoeff; i++)		
CoeffLevel [i] = 0		
if (coded_block_flag) {		
last_coeff_nz_flag	1	ae(v)
if (last_coeff_nz_flag == 0) {		
last_nz_pos = MaxNumCoeff – 1		
}		
else {		
last_nz_pos	1	ae(v)
}		
RegularStopPosTable [6] = RegularStopPosTableDefault[TbSize]		
if (last_nz_pos >= MaxNumCoeff – 6) {		
RegularStopPos = RegularStopPosTable[MaxNumCoeff – 1 – last_nz_pos]		
}		
else {		
RegularStopPos = 1		
}		
if (last_nz_pos != 0) {		
coeff_abs_level_greater1_flag[last_nz_pos **]**	1	ae(v)
coeffAbsLevel[last_nz_pos] = 1 + coeff_abs_level_greater1_flag[last_nz_pos]		
}		
for (i = last_nz_pos – 1; i >= RegularStopPos; i– –) {		
PosCur = i		
significant_coeff_flag[i]	1	ae(v)
coeffAbsLevel[i] = significant_coeff_flag[i]		
if (significant_coeff_flag[i]) {		
coeff_abs_level_greater1_flag[i]	1	ae(v)
coeffAbsLevel[i] += coeff_abs_level_greater1_flag[i]		
}		
}		

表 23 低频子带宏块量化系数定义（续）

低频子带宏块量化系数定义	码流指针 ID	描述符
num4x4block = MaxNumCoeff >> 4		
for (subblockIdx = num4x4block; subblockIdx>0; subblockIdx– –) {		
for (i= min((subblockIdx<<4) – 1, last_nz_pos); i >= (subblockIdx-1) << 4; i– –){		
if (i < RegularStopPos){		
coeff_abs_level_remaining[i]	2	se(v)
coeffAbsLevel[i] = coeff_abs_level_remaining[i] + (i == last_nz_pos)		
}		
else if (coeffAbsLevel[i] == 2) {		
coeff_abs_level_remaining[i]	2	se(v)
coeffAbsLevel[i] += coeff_abs_level_remaining[i]		
}		
}		
for (i= min((subblockIdx<<4) – 1, last_nz_pos); i >= (subblockIdx-1)<<4; i– –){		
if (coeffAbsLevel[i]) {		
coeff_sign_flag[i]	2	u(1)
CoeffLevel[i] = coeffAbsLevel[i] * (1 - 2*coeff_sign_flag [i])		
}		
}		
}		
}		
}		

7.1.7 高频子带编码单元数据定义

高频子带编码单元数据定义见表24。

表24 高频子带编码单元数据定义

高频子带编码单元数据定义	码流指针 ID	描述符
hf_band_mb_data (BandIdx, x, y) {		
for (CompIdx = 0; CompIdx < 3; CompIdx ++) {		

表 24 高频子带编码单元数据定义（续）

高频子带编码单元数据定义	码流指针 ID	描述符
blockNum = (CompIdx == 0) ? 4 : HfBlkNumChroma		
for (i = 0; i < 16 * blockNum; i++)		
coeff_level[i] = 0		
mb_has_coef_flag	3	ae(v)
if (mb_has_coef_flag) {		
mb_all_one_flag	3	ae(v)
if (hf_transform_skip_enable_flag && CompIdx == Y_COMPONENT) {		
transform_skip_flag	3	ae(v)
}		
if (!mb_all_one_flag) {		
countzero = 0		
countone = 0		
for (i = 0; i < blockNum; i++) {		
if((countzero != (blockNum-1)) && (countone != (blockNum-1))) {		
significance_flag[i]	3	ae(v)
}		
else {		
significance_flag[i] = (countzero == blockNum -1)? 1 : 0		
}		
countzero += (significance_flag[i] == 0) ? 1: 0		
countone += (significance_flag[i] == 1) ? 1: 0		
}		
}		
else {		
for (i = 0; i < blockNum; i++)		
significance_flag[i] = 1		
}		
for (i = 0; i < blockNum; i++) {		
if (significance_flag[i]) {		
decode_hf_coef(BandIdx, i)		
}		
}		
}		
}		
}		

高频子带宏块量化系数定义见表25。

表25 高频子带宏块量化系数定义

高频子带宏块量化系数定义	码流指针ID	描述符
decode_hf_coef (BandIdx, block_idx) {		
block_mode_flag	3	ae(v)
if(block_mode_flag) {		
table_idx_flag	3	ae(v)
for (i = 0; i < 16; i++) {		
coeff_level[16 * block_idx + i]	4	ce(v)
}		
}		
else {		
for (SubBlkIdx = 0; SubBlkIdx < 4; SubBlkIdx ++) {		
sub_significance_flag[SubBlkIdx]	3	ae(v)
if (sub_significance_flag[SubBlkIdx]) {		
pattern_0001_flag	3	ae(v)
if (pattern_0001_flag) {		
pattern_0001_code	4	u(3)
}		
else {		
max_grt1_flag	3	ae(v)
for (k = 0; k < 4; k++) {		
coeff_level[16 * block_idx + 4 * SubBlkIdx + k]	4	ce(v)
}		
}		
}		
}		
}		
}		

7.2 语义描述

7.2.1 概述

本文件规定的码流支持可伸缩解码，水平和垂直方向上各进行一次小波分解得到4个小波子带，小波子带水平和垂直方向上各降采样到原尺寸的1/2。

7.2.2 序列头

档次标志　profile_idc
8位无符号整数。规定码流符合的档次，应符合附录A。

级别标志　level_idc
8位无符号整数。规定码流符合的级别，应符合附录A。

帧数　num_of_frames_minus1
8位无符号整数。序列帧数减1。

帧率　frame_rate
8位无符号整数。视频帧率，值为N，表示视频每一秒有N帧。

图像宽度　input_picture_width
16位无符号整数。输入图像宽度，最小图像宽度为256。当色度格式为YUV422时，图像宽度为偶数。
编码图像宽度CodedPictureWidth的计算如下：

CodedPictureWidth = (input_picture_width + 15)/ 16 * 16

当解码输出显示1/2下采样图像时，下采样图像宽度的计算如下：

if(chroma_format == YUV422)

DownsamplePictureWidth = (input_picture_width / 4) * 2

else

DownsamplePictureWidth = input_picture_width / 2

图像高度　input_picture_height
16位无符号整数。输入图像高度，最小图像高度为256。
编码图像高度CodedPictureHeight的计算如下：

CodedPictureHeight = (input_picture_height + 15) / 16 * 16

当解码输出显示1/2下采样图像时，下采样图像高度的计算如下：

DownsamplePictureHeight = input_picture_height / 2

子图宽度信息　sub_pic_width_in_128_minus2
8位无符号整数。用于确定子图宽度的值。
子图宽度SubPictureWidth和水平方向子图个数NumSubPictureHor的计算如下：

SubPictureWidth = (sub_pic_width_in_128_minus2 + 2) << 7

NumSubPictureHor = (CodedPictureWidth + SubPictureWidth - 1) / SubPictureWidth

子图高度信息　sub_pic_height_in_128_minus1
8位无符号整数。用于确定子图高度的值。
子图高度SubPictureHeight和垂直方向子图个数NumSubPictureVer的计算如下：

SubPictureHeight = (sub_pic_height_in_128_minus1+1) << 7

NumSubPictureVer = (CodedPictureHeight - SubPictureHeight / 4) / SubPictureHeight + 1

输入图像位宽信息　bit_depth_minus8
4位无符号整数。输入图像位宽减8的值，可选取值为0-8。输入图像位宽BitDepth等于bit_depth_minus8+8。

色度格式　chroma_format

4位无符号整数。值为‘0’表示YUV444，值为‘1’表示YUV422，值为‘2’表示RGB，值为3到15保留。
当chroma_format为YUV444时，变量FormatShiftX设置为0，变量FormatShiftY设置为0，变量TbSizeChroma设置为TB_SIZE8x8，变量HfBlkNumChroma设置为4。
当chroma_format为YUV422时，变量FormatShiftX设置为1，变量FormatShiftY设置为0，变量TbSizeChroma设置为TB_SIZE4x8，变量HfBlkNumChroma设置为2。
TB_SIZE4x4、TB_SIZE8x8和TB_SIZE4x8为助记符，TB_SIZE4x4=0，TB_SIZE8x8=1，TB_SIZE4x8=2。

YUV422封装YUV444标志位　yuv444_packed_by_yuv422_flag
1位无符号整数。将YUV422转化为YUV444的标志，值为‘0’表示无后处理转化，值为‘1’表示按照附录B方式转化YUV422为YUV444。当chroma_format不等于1时，yuv444_packed_by_yuv422_flag应为0。当yuv444_packed_by_yuv422_flag为1时，num_of_frames_minus1应等于1，并且序列中每一幅图像的帧类型均为I帧。

场编码模式　interlace_mode
2位无符号整数。值为‘0’表示逐行扫描模式，值为‘1’表示场编码模式且序列中的第一幅图像和第二幅图像分别为顶场和底场，‘2’表示场编码模式且序列中的第一幅图像和第二幅图像分别为底场和顶场，值为3保留。interlace_mode等于‘1’或‘2’时，num_of_frames_minus1应等于1，并且序列中的每一幅图像的帧类型均为I帧。

序列头预留标志位　seq_header_reserved_flag[i]
1位无符号整数。seq_header_reserved_flag[i]表示序列头的第i个预留位，预留值为‘0’，支持拓展。

CICP信息标志位　cicp_info_present_flag
1位无符号整数。值为‘1’表示包含cicp_rendering_info ()，值为‘0’表示不包含cicp_rendering_info ()。

显示与内容元数据信息标志位　mdcv_info_present_flag
1位无符号整数。值为‘1’表示包含hdr_static_metadata ()，值为‘0’表示不包含hdr_static_metadata ()。hdr_static_metadata()结构体内容应符合GB/T 46269.1—2025 高动态范围（HDR）视频技术 第1部分:元数据及适配标准定义.

动态元数据信息标志位　dm_present_flag
1位无符号整数。值为‘1’表示包含dynamic_metadata ()，值为‘0’表示不包含dynamic_metadata ()。

色彩信息　colour_primaries
8位无符号整数。表示Rec. ITU-T H.273 | ISO/IEC 23091-2定义的ColourPrimaries。

转换信息　transfer_characteristics
8位无符号整数。表示Rec. ITU-T H.273 | ISO/IEC 23091-2定义的TransferCharacteristics。

矩阵系数　matrix_coefficients
8位无符号整数。表示Rec. ITU-T H.273 | ISO/IEC 23091-2定义的MatrixCoefficients。

图像全范围标志　video_full_range_flag
1位无符号整数。表示Rec. ITU-T H.273 | ISO/IEC 23091-2定义的VideoFullRangeFlag。

动态元数据类型　dm_type
8位无符号整数。表示动态元数据类型。值为‘0’表示动态元数据dm_data_byte[i]应符GB/T 46269.1—2025 高动态范围（HDR）视频技术 第1部分:元数据及适配。其他取值为预留位。

动态元数据大小　dm_size
16 位无符号整数。表示动态元数据字节数。

动态元数据　dm_data_byte[i]
8 位无符号整数。表示动态元数据第 i 字节内容。

渲染信息预留位　render_reserved
5位无符号整数。渲染信息预留值为0，支持拓展。

CICP信息预留位　cicp_reserved
5位无符号整数。CICP信息预留值为0，支持拓展。

7.2.3　图像头

图像长度　picture_len
32 位无符号整数。规定当前图像的编码数据总字节数，包含 picture_header 字节数、所有子图长度、填充字节的总字节数。

帧类型　frame_type
1 位无符号整数。值为‘0’表示 I 帧，值为‘1’表示 P 帧。

Alpha 分量标志　alpha_map_flag
1 位无符号整数。值为‘1’表示当前图像存在 Alpha 分量数据，值为‘0’表示当前图像不存在 Alpha 分量数据。Alpha 分量的解码见附录 C。

Alpha 分量 16 比特数据标志　alpha_map_16bit_flag
1 位无符号整数。值为‘1’表示当前图像的 Alpha 分量为 16 比特数据，值为‘0’表示当前图像 Alpha 分量为 8 比特数据。

Alpha 分量编码方式　alpha_map_code_mode
4 位无符号整数。表示当前图像 Alpha 分量的编码方式对应的索引号。索引号为‘0’时的 Alpha 分量数据定义和解码方法，见 C.3　和 C.4　。其他索引值保留。

块级量化参数允许标志　mb_qp_delta_enabled_flag
1 位无符号整数。值为‘0’表示当前图像关闭块级量化参数，宏块码流中不包含块级量化参数，值为‘1’表示当前图像使用块级量化参数，宏块码流中包含 ll_mb_qp_delta 和 hf_mb_qp_delta 语法。

高频变换跳过允许标志　hf_transform_skip_enable_flag
1 位无符号整数。值为‘0’表示当前图像的高频子带所有宏块均进行变换，值为‘1’表示当前图像允许使用块级变换跳过，高频宏块码流中包含 transform_skip_flag 语法。

跨分量预测允许标志　cclm_enable_flag
1 位无符号整数。值为‘1’表示当前图像的 LL 子带的宏块允许使用跨分量预测模式，值为‘0’表示当前图像的 LL 子带的宏块不使用跨分量预测模式。

图像输出标志　pic_output_flag
1 位无符号整数。值为‘1’表示当前图像的解码显示，值为‘0’表示当前图像的不解码显示。

图像字节填充　frame_zero_byte

8 位无符号整数。值为‘0’，用于字节填充。

图像头预留位　pic_header_reserved_bits
21位无符号整数。预留值为‘0’，支持拓展。

7.2.4　子图

子图长度 subpic_len[idx]
32位无符号整数。第idx个子图的总字节数，即子图的sub_pic_info字节数、低频子带字节数、高频子带字节数和、填充字节的总字节数。

子图低频子带算术编码码流长度　ll_band_lbac_len[idx]
32 位无符号整数。第 idx 个子图的低频子带 BAC 码流的字节数（不包含 LL_band_lbac_len 语法元素）。

子图低频子带变长编码码流长度　ll_band_vlc_len[idx]
32 位无符号整数。第 idx 个子图的低频子带 VLC 码流的字节数（不包含 LL_band_vlc_len 语法元素）。

子图高频子带算术编码码流长度　hf_band_lbac_len[idx]
32 位无符号整数。第 idx 个子图的高频子带 BAC 码流的字节数（不包含 HF_band_lbac_len 语法元素）。

子图高频子带变长编码码流长度　hf_band_vlc_len[idx]
32 位无符号整数。第 idx 个子图的高频子带 VLC 码流的字节数（不包含 HF_band_vlc_len 语法元素）。

子图 LL 子带量化参数　subpic_ll_qp_index[idx]
6 位无符号整数。第 idx 个子图的 LL 子带亮度 QP 索引，范围为 0 到 39。

子图 HL 子带量化参数偏移量　subpic_hl_qp_index_offset_plus12[idx]
5 位无符号整数。第 idx 个子图的 HL 子带亮度 QP 相对 LL 子带亮度 QP 偏移量加 12，范围为 0 到 24。第 idx 个子图 HL 子带亮度分量 QP 索引 SubpicHFQPindex[0][0]的计算如下。

SubpicHFQPindex[0][0] = clip(0, 39, subpic_ll_qp_index + subpic_hl_qp_index_offset_plus12 – 12)

子图 LH 子带量化参数　subpic_lh_qp_index_offset_plus12[idx]
5 位无符号整数。第 idx 个子图的 LH 子带亮度 QP 相对 LL 子带亮度 QP 偏移量加 12，范围为 0 到 24。第 idx 个子图 LH 子带亮度分量 QP 索引 SubpicHFQPindex[1][0] 的计算如下。

SubpicHFQPindex[1][0] = clip(0, 39, subpic_ll_qp_index + subpic_lh_qp_index_offset_plus12 – 12)

子图 HH 子带量化参数　subpic_hh_qp_index_offset_plus12[idx]
5 位无符号整数。第 idx 个子图的 HH 子带亮度 QP 相对 LL 子带亮度 QP 偏移量加 12，范围为 0 到 24。第 idx 个子图 HH 子带亮度分量 QP 索引 SubpicHFQPindex[2][0] 的计算如下。

SubpicHFQPindex[2][0] = clip(0, 39, subpic_ll_qp_index + subpic_hh_qp_index_offset_plus12 – 12)

子图色度 cb 分量量化参数　subpic_cb_qp_index_offset_plus12[idx]
5 位无符号整数。第 idx 个子图的 cb 分量 QP 相对对应子带亮度 QP 偏移量加 12，范围为 0 到 24。第 idx 个子图 LL 子带色度 Cb 分量 QP 索引 SubpicLLcbQPindex、HL 子带色度 Cb 分量 QP 索引 SubpicHFQPindex[0][1]、LH 子带色度 Cb 分量 QP 索引 SubpicHFQPindex[1][1]、HH 子带色度 Cb 分量 QP 索引 SubpicHFQPindex[2][1]的计算如下。

SubpicLLcbQPindex = clip(0, 39, subpic_ll_qp_index + subpic_cb_qp_index_offset_plus12 – 12)

SubpicHFQPindex[0][1] = clip(0, 39, SubpicHFQPindex[0][0] + subpic_cb_qp_index_offset_plus12 – 12)

SubpicHFQPindex[1][1] = clip(0, 39, SubpicHFQPindex[1][0] + subpic_cb_qp_index_offset_plus12 – 12)

SubpicHFQPindex[2][1] = clip(0, 39, SubpicHFQPindex[2][0] + subpic_cb_qp_index_offset_plus12 – 12)

子图色度 cr 分量量化参数　subpic_cr_qp_index_offset_plus12[idx]

5 位无符号整数。第 idx 个子图的 cr 分量 QP 相对对应子带亮度 QP 偏移量加 12，范围为 0 到 24。

第 idx 个子图 LL 子带色度 Cr 分量 QP 索引 SubpicLLcrQPindex、HL 子带色度 Cr 分量 QP 索引 SubpicHFQPindex[0][2]、LH 子带色度 Cr 分量 QP 索引 SubpicHFQPindex[1][2]、HH 子带色度 Cr 分量 QP 索引 SubpicHFQPindex[2][2]的计算如下。

SubpicLLcrQPindex = clip(0, 39, subpic_ll_qp_index + subpic_cr_qp_index_offset_plus12 – 12)

SubpicHFQPindex[0][2] = clip(0, 39, SubpicHFQPindex[0][0] + subpic_cr_qp_index_offset_plus12 – 12)

SubpicHFQPindex[1][2] = clip(0, 39, SubpicHFQPindex[1][0] + subpic_cr_qp_index_offset_plus12 – 12)

SubpicHFQPindex[2][2] = clip(0, 39, SubpicHFQPindex[2][0] + subpic_cr_qp_index_offset_plus12 – 12)

子图头预留位　subpic_reserved_bits[idx]

9位无符号整数。第idx个子图sub_pic_info的预留位，预留值为0，支持拓展。

常值 0　zero_bit

1位无符号整数。值为‘0’，用于字节对齐。

7.2.5　子带

填充位　band_stuffing_bit

编码时插入码流的二元符号，解码时被丢弃，解析过程见8.1。

7.2.6　低频子带编码单元数据

宏块量化参数　ll_mb_qp_delta

当前宏块量化参数相对于预测量化参数的差值，解析过程见8.2。

预测模式　mb_mode

表示编码单元预测模式，解析过程见8.1。值为‘0’表示帧内预测，帧内预测记为MODE_INTRA，值为‘1’表示帧间预测，帧间预测记为MODE_INTER。如果码流不存在mb_mode，mb_mode的值设置为0。

亮度变换块大小　luma_tb_size

表示当前编码单元中亮度分量的变换块大小，解析过程见8.1。值为‘0’表示TB_SIZE4x4，值为‘1’表示TB_SIZE8x8。

帧间残差标志　inter_no_residual_flag

表示帧间编码模式的量化残差系数是否全零，解析过程见8.1。值为‘0’表示编码单元包含非零的量化残差，值为‘1’表示编码单元的量化残差全零。

运动矢量差标志　mvd_flag

表示运动矢量差是否为零，解析过程见8.1。值为‘0’表示运动矢量差为零，值为‘1’表示运动矢量差非零。

亮度帧内预测模式第一标志位　intra_pred_mode_luma_first_flag
与亮度帧内预测模式第二标志位共同表示亮度分量帧内预测模式，解析过程见8.1。

亮度帧内预测模式第二标志位　intra_pred_mode_luma_second_flag
与亮度帧内预测模式第一标志位共同表示亮度分量帧内预测模式，解析过程见8.1。intra_pred_mode_luma_first_flag为‘0’且intra_pred_mode_luma_second_flag为‘0’表示帧内垂直预测，intra_pred_mode_luma_first_flag为‘0’且intra_pred_mode_luma_second_flag为‘1’表示帧内直流预测。intra_pred_mode_luma_first_flag为‘1’表示帧内水平预测。如果码流不存在intra_pred_mode_luma_second_flag，宏块intra_pred_mode_luma_second_flag的值为‘0’。

色度帧内预测模式第一标志位　intra_pred_mode_chroma_first_flag
与色度帧内预测模式第二标志位共同表示色度分量帧内预测模式，解析过程见8.1。

色度帧内预测模式第二标志位　intra_pred_mode_chroma_second_flag
与色度帧内预测模式第一标志位共同表示色度分量帧内预测模式，解析过程见8.1。intra_pred_mode_chroma_first_flag为‘0’且intra_pred_mode_chroma_second_flag为‘0’表示帧内垂直预测，intra_pred_mode_chroma_first_flag为‘0’且intra_pred_mode_chroma_second_flag为‘1’表示帧内直流预测；intra_pred_mode_chroma_first_flag为‘1’且intra_pred_mode_chroma_second_flag为‘0’表示帧内水平预测;intra_pred_mode_chroma_first_flag为‘1’且intra_pred_mode_chroma_second_flag为‘1’表示帧内跨分量预测。如果码流不存在intra_pred_mode_chroma_second_flag，宏块intra_pred_mode_chroma_second_flag的值为‘0’。

运动矢量差大于0标志位　abs_mvd_grt0_flag
表示运动矢量差绝对值是否大于0，解析过程见8.1。abs_mvd_grt0_flag[0]为‘0’表示水平方向运动矢量差绝对值等于0，值为‘1’表示水平方向运动矢量差绝对值大于0。abs_mvd_grt0_flag[1]为‘0’表示垂直方向运动矢量差绝对值等于0，值为‘1’表示垂直方向运动矢量差绝对值大于0。

运动矢量差大于1标志位　abs_mvd_grt1_flag
表示运动矢量差绝对值是否大于1，解析过程见8.1。abs_mvd_grt1_flag[0]为‘0’表示水平方向运动矢量差绝对值等于1，值为‘1’表示水平方向运动矢量差绝对值大于1。abs_mvd_grt1_flag[1]为‘0’表示垂直方向运动矢量差绝对值等于1，值为‘1’表示垂直方向运动矢量差绝对值大于1。

运动矢量差幅值参数　abs_mvd_minus2
表示运动矢量差绝对值减2，解析过程见8.2。abs_mvd_minus2[0]表示水平方向运动矢量差绝对值减2，5位无符号整数。abs_mvd_minus2[1]表示垂直方向运动矢量差绝对值减2，4位无符号整数。

运动矢量差符号位　mvd_sign_flag
1位无符号整数，值为‘0’表示运动矢量差为正数，值为‘1’表示运动矢量差为负数。mvd_sign_flag[0]表示水平方向运动矢量差正负性，mvd_sign_flag[1]表示垂直方向运动矢量差正负性。

TB非零系数标志　coded_block_flag
表示变换块是否包含非零系数，解析过程见8.1。值为‘1’表示变换块含有非零系数，值为‘0’表示变换块所有系数均为0。

最后一个非零系数位置编码标志　last_coeff_nz_flag
表示是否编码扫描顺序最后一个位置，解析过程见8.1。值为‘0’表示编码扫描顺序最后一个位置为非零系数，值为‘1’表示需解码最后一个非零系数的位置。

最后一个非零系数位置　last_nz_pos

表示最后一个非零系数的位置，解析过程见8.1。最后一个非零系数的位置与变换块大小有关，语法表中CoefNumInSize和RegularStopPosTableDefault定义如下：

CoefNumInSize[3] = {16, 64, 32}

RegularStopPosTableDefault[3][6] = { {14, 9, 6, 4, 3, 2}
{45, 30, 20, 15, 10, 5}
{20, 14, 9, 6, 4, 2} }

系数绝对值大于0标志　significant_coeff_flag
表示当前量化系数的绝对值是否大于0，解析过程见8.1。值为‘0’表示绝对值等于0，值为‘1’表示绝对值大于0。

系数绝对值大于1标志　coeff_abs_level_greater1_flag
表示当前量化系数的绝对值是否大于1，解析过程见8.1。值为‘0’表示绝对值等于1，值为‘1’表示绝对值大于1。

系数剩余值　coeff_abs_level_remaining
表示当前量化系数的剩余值，解析过程见8.3.1。

系数符号位　coeff_sign_flag
1位无符号整数，表示当前量化系数的符号，值为‘0’表示正数，值为‘1’表示负数。

7.2.7　高频子带编码单元数据

宏块量化参数　hf_mb_qp_delta
高频子带中宏块量化参数相对于预测量化参数的差值，解析过程见8.2。

宏块全零标志　mb_has_coef_flag
二值变量，解析过程见8.1。值为‘0’表示宏块所有系数都为零。值为‘1’表示宏块至少有一个非零系数。

宏块全一标志　mb_all_one_flag
二值变量，解析过程见8.1。值为‘1’表示宏块中所有4x4块都包含非零系数。值为‘0’表示宏块中至少一个4x4块为全零块。

宏块变换跳过标志　transform_skip_flag
二值变量，解析过程见8.1。值为‘1’表示宏块跳过2x2的哈达玛变换，值为‘0’表示宏块执行2x2的哈达玛变换。如果码流不存在transform_skip_flag，宏块transform_skip_flag的值为0。色度宏块transform_skip_flag的值为0。

块重要性标志　significance_flag
二值变量，解析过程见8.1。值为‘0’表示4x4块内所有系数都为零。值为‘1’表示4x4块包含非零系数。

块稀疏性标志　block_mode_flag
二值变量，解析过程见8.1。值为‘0’表示4x4块系数稀疏，采用特殊模式解析系数值。值为‘1’表示4x4块系数密集，采用直接模式解析16个系数值。

块码表标志　table_idx_flag

二值变量，解析过程见8.1。值为'0'表示码表索引号的偏移量为0。值为'1'表示码表索引号的偏移量为1。

子块重要性标志　sub_significance_flag

二值变量，解析过程见8.1。值为‘0’表示一个系数组的4个系数都为零。值为‘1’表示一个系数组的4个系数包含非零系数。

子块系数绝对值标志　max_grt1_flag

二值变量，解析过程见8.1。值为‘0’表示一个系数组的4个系数的绝对值都小于等于1。值为‘1’表示一个系数组的4个系数至少一个系数绝对值大于1。

子块系数特殊模式标志　pattern_0001_flag

二值变量，解析过程见8.1。值为‘0’表示一个系数组的4个系数不是特殊模式。值为‘1’表示一个系数组的4个系数是0001模式。

子块系数特殊模式值　pattern_0001_code

3位无符号整数，表示当前系数组的四个系数值的模式索引值，基于模式索引值解析系数组的四个系数的过程见8.3.2.2.1　。

量化系数值　coeff_level

表示当前量化系数值，解析过程见8.3.2。

8 解析过程

8.1 ae(v) 的解析过程

8.1.1　概述

ae(v)描述的语法元素使用基于上下文的BAC编码。在解析每个子图的高频子带和低频子带前，需要初始化所有二元符号概率模型和熵编解码器，过程见8.1.2。解码ae(v)描述的语法元素时，首先从码流中依次解析得到二元符号串，过程见8.1.3。之后，对二元符号串进行反二值化操作，得到语法元素值，过程见8.1.4。

8.1.2　初始化

解码ae(v)描述的语法元素前，需要初始化数组biCtxLLArray和数组biCtxHFArray中的所有二元符号模型。其中，biCtxLLArray是保存低频子带二元符号概率模型的数组，biCtxHFArray是保存高频子带二元符号概率模型的数组。一个二元符号模型包括mps，lgPmps两个变量。其中，mps的位宽为1位，应初始化为0，lgPmps的位宽为9位，应初始化为255。

range、value、cFlag，tFlag是用于熵编码解码器的变量，其中，range位宽为10位,value位宽为9位，cFlag位宽为1位初始化过程用伪代码描述如下：

```
range = 0x1FF
value = read_bits(9)
```

8.1.3　二元符号串解析

8.1.3.1 解析步骤

解析二元符号串的步骤如下：

a)　设二元符号在串中的索引号binIndex为-1，二元符号串为空。

b)　binIndex的值加1，如果当前二元符号是band_stuffing_bit，则将StuffingBitFlag的值设为1，否则将StuffingBitFlag的值置为0。

c) 根据binIndex得到每个二元符号对应的唯一CtxIdxStart，并根据CtxIdxStart导出二元符号模型ctx（过程见8.1.3.2）。
d) 解析当前二元符号（过程见8.1.3.3）。
e) 将由步骤d)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。
f) 将由步骤e)得到的二元符号串与8.1.4中语法元素的反二值化结果进行比较，如果该二元符号串与语法元素的反二值化结果相匹配，则完成二元符号串的解析，否则返回步骤b)，继续解析下一个二元符号。

8.1.3.2 导出二元符号的概率模型

biCtxLLArray是保存低频子带二元符号概率模型的数组。低频子带语法元素的每个二元符号模型索引值CtxIdx等于CtxIdxDelta加CtxIdxStart，模型ctx为biCtxLLArray[CtxIdx]。

biCtxHFArray是保存高频子带二元符号概率模型的数组。高频子带语法元素的每个二元符号模型索引值CtxIdx等于CtxIdxDelta加CtxIdxStart，模型ctx为biCtxHFArray[CtxIdx]。

表26 LL 子带语法元素对应的概率模型索引号 CtxIdxStart 及模式数目 CtxNum

语法元素	CtxIdxDelta	CtxIdxStart	CtxNum
mb_mode	0	0	1
inter_no_residual_flag	0	1	1
mvd_flag	0	2	1
luma_tb_size	0	3	1
intra_pred_mode_luma_first_flag	0	4	1
intra_pred_mode_luma_second_flag	0	5	1
intra_pred_mode_chroma_first_flag	0	6	1
intra_pred_mode_chroma_second_flag	见 8.1.3.2.1	7	2
coded_block_flag	见 8.1.3.2.2	9	4
last_coeff_nz_flag	见 8.1.3.2.3	13	4
last_nz_pos	见 8.1.3.2.4	17	22
significant_coeff_flag	见 8.1.3.2.5	39	56
coeff_abs_level_greater1_flag	见 8.1.3.2.6	95	32
abs_mvd_grt0_flag	见 8.1.3.2.7	127	2
abs_mvd_grt1_flag	见 8.1.3.2.8	129	2

表27 HF 子带语法元素对应的概率模型索引号 CtxIdxStart 及模式数目 CtxNum

语法元素	CtxIdxDelta	CtxIdxStart	CtxNum
mb_has_coef_flag	3*BandIdx + CompIdx	0	9
mb_all_one_flag	3*BandIdx + CompIdx	9	9
significance_flag	BandIdx	18	3
block_mode_flag	BandIdx	21	3
transform_skip_flag	BandIdx	24	3
sub_significance_flag	见 8.1.3.2.9	27	39

表 27 HF 子带语法元素对应的概率模型索引号 CtxIdxStart 及模式数目 CtxNum（续）

语法元素	CtxIdxDelta	CtxIdxStart	CtxNum
pattern_0001_flag	BandIdx	66	3
table_idx_flag	BandIdx	69	3
max_grt1_flag	BandIdx	72	3

8.1.3.2.1 确定 intra_pred_mode_chroma_second_flag 的 CtxIdxDelta

根据以下方法确定intra_pred_mode_chroma_second_flag的CtxIdxDelta：

- CtxIdxDelta = 1 + intra_pred_mode_chroma_first_flag。

8.1.3.2.2 确定 coded_block_flag 的 CtxIdxDelta

根据以下步骤依次执行，确定coded_block_flag的CtxIdxDelta：

- 如果当前块是亮度宏块，CtxIdxDelta = (TbSize == TB_SIZE8x8)? 0 : 1
- 如果当前块是色度cb宏块，CtxIdxDelta = 2
- 如果当前块是色度cr宏块，CtxIdxDelta = 3

8.1.3.2.3 确定 last_coeff_nz_flag 的 CtxIdxDelta

根据以下步骤依次执行，确定last_coeff_nz_flag的CtxIdxDelta：

- 如果当前块是亮度宏块，CtxIdxDelta = (TbSize == TB_SIZE8x8)? 0 : 1
- 如果当前块是色度cb宏块，CtxIdxDelta = 2
- 如果当前块是色度cr宏块，CtxIdxDelta = 3

8.1.3.2.4 确定 last_nz_pos 的 CtxIdxDelta

根据以下步骤依次执行，令MaxBinNumLastPos表示最大bin数，确定last_nz_pos的CtxIdxDelta：

a) 确定MaxBinNumLastPos及模型偏移量ctxOffset，
   - 如果当前块是亮度宏块:
     MaxBinNumLastPos = (TbSize == TB_SIZE8x8)? 6 : 4
     ctxOffset = (TbSize == TB_SIZE8x8)? 0 : 6
   - 如果当前块是色度cb宏块:
     MaxBinNumLastPos = (TbSize == TB_SIZE8x8)? 6 : 5
     ctxOffset = 10
   - 如果当前块是色度cr宏块:
     MaxBinNumLastPos = (TbSize == TB_SIZE8x8)? 6 : 5
     ctxOffset = 16

b) For binIdx = 0,1,..., MaxBinNumLastPos-1:
   - CtxIdxDelta = ctxOffset + binIdx

注：若前MaxBinNumLastPos -1个二元符号串全为1时，最后一个bin数值设为0，不需要从码流中解析。

8.1.3.2.5 确定 significant_coeff_flag 的 CtxIdxDelta

根据以下步骤依次执行，确定当前变换块扫描顺序的第PosCur个系数的significant_coeff_flag的CtxIdxDelta：

a) 确定模型偏移量ctxBase:
- 如果当前块是亮度宏块:
  - ctxOffset = (TbSize == TB_SIZE8x8)? 0 : 14
  - 当TbSize等于TB_SIZE8x8，ctx_table[62] = [0, 1, 0, 1, 2, 3, 2, 3, 4, 5, 4, 5, 4, 5, 6, 7, 6, 7, 6, 7, 8, 9, 8, 9, 8, 9, 8, 9, 8, 9, 8, 9, 8, 9, 10, 11, 10, 11, 10, 11, 10, 11, 10, 11, 10, 11,10, 11, 12, 13, 12, 13, 12, 13, 12, 13, 12, 13, 12, 13, 12, 13]
  - 当TbSize等于TB_SIZE4x4，ctx_table[14] = [0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13]
- 如果当前块是色度cb宏块:
  - ctxOffset = 28
  - 当TbSize等于TB_SIZE4x8，ctx_table[30] = [0, 1, 2, 3, 4, 5, 6, 7, 6, 7, 6, 7, 8, 9, 8, 9, 8, 9, 10, 11, 10, 11, 10, 11, 12, 13, 12, 13, 12, 13]
  - 当TbSize等于TB_SIZE8x8，ctx_table[62] = [0, 1, 0, 1, 2, 3, 2, 3, 4, 5, 4, 5, 4, 5, 6, 7, 6, 7, 6, 7, 8, 9, 8, 9, 8, 9, 8, 9, 8, 9, 8, 9, 8, 9, 10, 11, 10, 11, 10, 11, 10, 11, 10, 11, 10, 11,10, 11, 12, 13, 12, 13, 12, 13, 12, 13, 12, 13, 12, 13, 12, 13]
- 如果当前块是色度cr宏块:
  - ctxOffset = 42
  - 当TbSize等于TB_SIZE4x8，ctx_table[30] = [0, 1, 2, 3, 4, 5, 6, 7, 6, 7, 6, 7, 8, 9, 8, 9, 8, 9, 10, 11, 10, 11, 10, 11, 12, 13, 12, 13, 12, 13]
  - 当TbSize等于TB_SIZE8x8，ctx_table[62] = [0, 1, 0, 1, 2, 3, 2, 3, 4, 5, 4, 5, 4, 5, 6, 7, 6, 7, 6, 7, 8, 9, 8, 9, 8, 9, 8, 9, 8, 9, 8, 9, 8, 9, 10, 11, 10, 11, 10, 11, 10, 11, 10, 11, 10, 11,10, 11, 12, 13, 12, 13, 12, 13, 12, 13, 12, 13, 12, 13, 12, 13]

b) 确定CtxIdxDelta:

CtxIdxDelta = ctxOffset + ctx_table[PosCur-1]

8.1.3.2.6 确定 coeff_abs_level_greater1_flag 的 CtxIdxDelta

a) 从last_nz_pos起始按逆扫描顺序扫描变换块，根据以下步骤确定变换块内第i个coeff_abs_level_greater1_flag[i]的CtxIdxDelta[i]，设his_val初始值为1，令N表示变换块内需要解码的coeff_abs_level_greater1_flag总个数： 确定模型偏移量ctxBase:
- 如果当前块是亮度宏块，ctxOffset = (TbSize == TB_SIZE8x8)? 0 : 8
- 如果当前块是色度cb宏块，ctxOffset = 16
- 如果当前块是色度cb宏块，ctxOffset = 24

b) 确定CtxIdxDelta:

```
for (int i = 0; i < N; i++) {
    CtxIdxDelta[i] = his_val + ctxOffset
    if (coeff_abs_level_greater1_flag[i] == 1)
            his_val = 0
    else if (his_val > 0)
            his_val = min(7, his_val+1)
  }
```

8.1.3.2.7 确定 abs_mvd_grt0_flag 的 CtxIdxDelta

根据以下方法确定abs_mvd_grt0_flag的CtxIdxDelta：

- abs_mvd_grt0_flag[0]的CtxIdxDelta等于0。
- abs_mvd_grt0_flag[1]的CtxIdxDelta等于1。

8.1.3.2.8 确定 abs_mvd_grt1_flag 的 CtxIdxDelta

根据以下方法确定abs_mvd_grt1_flag的CtxIdxDelta：

- abs_mvd_grt1_flag[0]的CtxIdxDelta等于0。
- abs_mvd_grt1_flag[1]的CtxIdxDelta等于1。

8.1.3.2.9 确定 sub_significance_flag 的 CtxIdxDelta

根据以下方式确定sub_significance_flag的CtxIdxDelta：

- 当transform_skip_flag等于1时，CtxIdxDelta = BandIdx + 36
- 当transform_skip_flag等于0时，CtxIdxDelta = 9 * SubBlkIdx + 3 * BandIdx + CompIdx

8.1.3.3 二元符号解析

8.1.3.3.1 解析过程

二元符号的解析过程如下：

a) 按照如下方式解析二元符号值binVal。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’。
   - 如果StuffingBitFlag的值为1，则执行decode_stuffing_bit过程（见8.1.3.3.3），得到binVal；
   - 否则，令cFlag等于1，tFlag等于0，执行decode_decision过程（见8.1.3.3.2），得到binVal。

b) 如果cFlag等于1，执行update_ctx过程对概率模型更新（见8.1.3.3.4）

8.1.3.3.2 decode_decision

decode_decision过程的输入是range、value、cFlag以及上下文模型ctx。decode_decision过程的输出是二元符号值binVal。decode_decision过程用伪代码描述如下：

```
decode_decision( ) {
    rLPS = ctx->lgPmps
    rMPS = range - rLPS
    s_flag = rMPS < 256 ? 1 : 0
    rMPS = rMPS | 0x100
    if (s_flag) {
        value = (value << 1) | read_bits(1)
    }
    if (value < rMPS) {
        binVal = ctx->mps
        range = rMPS
    }
    else {
```

```
            binVal = !ctx->mps
            range = (range << s_flag) - rMPS
            value = value - rMPS
            if (!tFlag) {
                while ( range < 256 ) {
                    range = range << 1
                    value = (value << 1) | read_bits(1)
                }
            }
        }
        if ( cFlag ) {
            ctx = update_ctx( binVal, ctx )
        }
        return (binVal)
    }
```

8.1.3.3.3 decode_stuffing_bit

decode_stuffing_bit 过程的输入是 range、value 和 cFlag。decode_aec_stuffing_bit 过程的输出是二元符号值 binVal。ctx0 是二元符号模型，令 ctx0->lgPmps 等于 1，ctx0->mps 等于 0。令 cFlag 等于 0，tFlag 等于 1，ctx 等于 ctx0，带入 decode_decision 过程实现 decode_stuffing_bit 过程。

8.1.3.3.4 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 ( binVal == ctx->mps ) {
            ctx->lgPmps = ctx->lgPmps – (ctx->lgPmps >> 4) - (ctx->lgPmps >> 6)
        }
        else {
            ctx->lgPmps = ctx->lgPmps + 23
            if ( ctx->lgPmps > 255 ) {
                ctx->lgPmps = 511 - ctx->lgPmps
                ctx->mps = 1 – ctx->mps
            }
        }
        return (ctx)
    }
```

8.1.4 反二值化

低频子带的MaxBinNumLastPos为6时，由二元符号串查表28得到last_nz_pos的值。
低频子带的MaxBinNumLastPos为5时，由二元符号串查表29得到last_nz_pos的值。

低频子带的MaxBinNumLastPos为4时，由二元符号串查表30得到last_nz_pos的值。

表28 MaxBinNumLastPos 为 6 时，last_nz_pos 的值与二元符号串的关系

last_nz_pos	binIdx0 的值	binIdx1 的值	binIdx2 的值	binIdx3 的值	binIdx4 的值	binIdx5 的值
0	0	0	0	0	0	0
1	0	0	0	0	0	1
2	0	0	0	0	1	0
3	0	0	0	0	1	1
4	0	0	0	1	0	0
…	…	…	…	…	…	…
60	1	1	1	1	0	0
61	1	1	1	1	0	1
62	1	1	1	1	1	-

表29 MaxBinNumLastPos 为 5 时，last_nz_pos 的值与二元符号串的关系

last_nz_pos	binIdx0 的值	binIdx1 的值	binIdx2 的值	binIdx3 的值	binIdx4 的值
0	0	0	0	0	0
1	0	0	0	0	1
2	0	0	0	1	0
3	0	0	0	1	1
4	0	0	1	0	0
…	…	…	…	…	…
28	1	1	1	0	0
29	1	1	1	0	1
30	1	1	1	1	-

表30 MaxBinNumLastPos 为 4 时，last_nz_pos 的值与二元符号串的关系

last_nz_pos	binIdx0 的值	binIdx1 的值	binIdx2 的值	binIdx3 的值
0	0	0	0	0
1	0	0	0	1
2	0	0	1	0
3	0	0	1	1
4	0	1	0	0
…	…	…	…	…
12	1	1	0	0

表 30 MaxBinNumLastPos 为 4 时，last_nz_pos 的值与二元符号串的关系（续）

last_nz_pos	binIdx0 的值	binIdx1 的值	binIdx2 的值	binIdx3 的值
13	1	1	0	1
14	1	1	1	-

8.2 se(v) k 阶指数哥伦布码

解析0阶指数哥伦布码时，首先从比特流的当前位置开始寻找第一个非零比特，并将找到的零比特个数记为leadingZeroBits，然后根据leadingZeroBits计算CodeNum。用伪代码描述如下：

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

表31给出了0阶指数哥伦布码的结构。指数哥伦布码的比特串分为“前缀”和“后缀”两部分。前缀由leadingZeroBits个连续的‘0’和一个‘1’构成。后缀由leadingZeroBits个比特构成，即表中的x_i串，x_i的值为‘0’或‘1’。

表31 0 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
	……	……

低频子带ll_mb_qp_delta、高频子带hf_mb_qp_delta依据0阶指数哥伦布码解析出数值codeNumLL和codeNumHF，ll_mb_qp_delta和hf_mb_qp_delta的取值范围为[-16, 15]，根据数值codeNumLL和codeNumHF计算的方式如下：

```
ll_mb_qp_delta = (codeNumLL&0x01) ? (codeNumLL + 1)>>1 : -(codeNumLL>>1)
hf_mb_qp_delta = (codeNumHF&0x01) ? (codeNumHF + 1)>>1 : -(codeNumHF>>1)
```

解析k阶（k大于0）指数哥伦布码时，首先从比特流的当前位置开始寻找第一个零比特，并将找到的一比特个数记为leadingOneBits，然后根据leadingOneBits计算CodeNum。用伪代码描述如下：

```
leadingOneBits = -1;
for ( b = 1; b; leadingOneBits++ )
    b = read_bits(1)
CodeNum = 2^(leadingOneBits + k) – 2^k + read_bits(leadingOneBits + k)
```

表32给出了1阶、2阶和3阶指数哥伦布码的结构。指数哥伦布码的比特串分为“前缀”和“后缀”两部分。前缀由leadingOneBits个连续的‘1’和一个‘0’构成。后缀由leadingOneBits + k个比特构成，即表中的x_i串，x_i的值为‘0’或‘1’。

表32 k 阶指数哥伦布码表

阶数	码字结构	CodeNum取值范围
k = 1	0 x_0	0～1

表32 k阶指数哥伦布码表（续）

阶数	码字结构	CodeNum取值范围
	1 0 x_1 x_0	2～5
	1 1 0 x_2 x_1 x_0	6～13
	1 1 1 0 x_3 x_2 x_1 x_0	14～29
	……	……
k = 2	0 x_1 x_0	0～3
	1 0 x_2 x_1 x_0	4～11
	1 1 0 x_3 x_2 x_1 x_0	12～27
	1 1 1 0 x_4 x_3 x_2 x_1 x_0	28～59
	……	……
k = 3	0 x_2 x_1 x_0	0～7
	1 0 x_3 x_2 x_1 x_0	8～23
	1 1 0 x_4 x_3 x_2 x_1 x_0	24～55
	1 1 1 0 x_5 x_4 x_3 x_2 x_1 x_0	56～119
	……	……

低频子带abs_mvd_minus2依据3阶指数哥伦布码解析。

8.3 ce(v)的解析过程

8.3.1 低频子带 coeff_abs_level_remaining 解析方法

当低频子带coeff_abs_level_remaining小于(3 << k)采用k阶截断莱斯码；否则，采用k阶截断莱斯码和k阶指数哥伦布码编码，k为莱斯参数。

8.3.1.1 K 阶截断莱斯码

令k阶截断莱斯码的门限参数值为cMax，cMax = 3 << k。解析k阶截断莱斯码时，首先从比特流的当前位置开始记录连续一比特个数记为leadingOneBits，当leadingOneBits为3时，停止记录。leadingOneBits小于3时，继续从码流中读取k比特后缀码。最后根据leadingOneBits计算CodeNum。用伪代码描述如下：

```
for ( leadingOneBits = 0; leadingOneBits < 3; leadingOneBits++ ){
    if（read_bits(1) == 0）
        break
}
if ( leadingOneBits < 3){
    CodeNum = (leadingOneBits << k) + read_bits(k)
}
else
{
    CodeNum = cMax
}
```

8.3.1.2 低频子带 coeff_abs_level_remaining 的解析过程

从last_nz_pos起始按逆扫描顺序扫描变换块，根据以下步骤解析变换块内第i个coeff_abs_level_remaining[i]的值，令N表示变换块内需要解码的coeff_abs_level_remaining总个数。

遍历 i 的取值从 0 到 N-1：

a) 首先，采用8.3.1.1所述k阶截断莱斯码解码方法从码流中解码得到CodeNumA，

b) 若CodeNumA等于(3 << k)，采用所述k阶指数哥伦布码从码流中解码得到CodeNumB，coeff_abs_level_remaining[i]等于CodeNumA与CodeNumB的和。

c) 若CodeNumA小于(3 << k)， coeff_abs_level_remaining[i]等于CodeNumA。

亮度或色度变换块的莱斯参数k初始值设置为0，若coeffAbsLevel[i]大于(3 << k),设置k等于k加一。对于非亮度DC系数，k的最大值为4。亮度DC系数解析时k值为当前值再加一，最大可能为5。

8.3.2 高频子带 coeff_level 解析方法

8.3.2.1 block_mode_flag 等于 1 的条件下 coeff_level 解析方法

令相同子带相同色彩分量上解码顺序前一个4x4块的系数绝对值的最大值为PrevCoeffMaxAbs。根据以下步骤依次执行，确定coeff_level的值：

- 当MbX等于0且当前系数为宏块内解码顺序的第一个系数时，初始化PrevCoeffMaxAbs为0。
- 当PrevCoeffMaxAbs小于等于5时，tableIdx = table_idx_flag
- 当PrevCoeffMaxAbs大于5且PrevCoeffMaxAbs小于等于15时，tableIdx = 1 + table_idx_flag
- 当PrevCoeffMaxAbs大于15时，tableIdx = 2 + table_idx_flag
- 由二元符号串查tableIdx所对应的码表得到coeff_level的值。

tableIdx等于0的码表，按以下方式解析coeff_level值，首先从比特流的当前位置开始寻找第一个非零比特，并将找到的零比特个数记为leadingZeroBits：

a) leadingZeroBits小于等于4时，比特串为leadingZeroBits个连续的‘0’和一个‘1’，由码字查表33得到coeff_level。

b) leadingZeroBits等于5时，继续从码流中读取符号位s，按如下过程得到coeff_level：

```
abs_level = (1 << (leadingZeroBits - 5)) + 2
coeff_level = abs_level * (1 - 2 * s)
```

c) leadingZeroBits大于5时，比特串分为“前缀码”、“后缀码”和“符号位”三部分。前缀由leadingZeroBits个连续的‘0’和一个‘1’构成。然后从比特流读取leadingZeroBits - 5个比特得到后缀码，即表中的x_i串，x_i的值为‘0’或‘1’，令后缀码二元串对应的十进制数记为SuffixCode。然后从比特流读取符号位，即表33中的s，数值为‘0’表示正数，‘1’表示负数，coeff_level按如下步骤得到：

```
abs_level = (1 << (leadingZeroBits - 5)) + 2 + SuffixCode;
coeff_level = abs_level * (1 - 2 * s)
```

表33 tableIdx 等于 0 的码表

coeff_level 取值范围	前缀码	后缀码	符号位
0	1	-	-
-1	01	-	-
1	001	-	-
-2	0001	-	-
2	00001	-	-
-3	000001	-	1
3	000001	-	0

表 33 tableIdx 等于 0 的码表（续）

coeff_level 取值范围	前缀码	后缀码	符号位
-4，4，-5，5	0000001	x_0	s
-9～-6，6～9	00000001	x_1 x_0	s
-17～-10，10～17	000000001	x_2 x_1 x_0	s
……	……	……	……

tableIdx等于1的码表，按以下方式解析coeff_level值，首先从比特流的当前位置开始读取2比特记为pre_value：

a) pre_value等于0时，coeff_level = 0。

b) pre_value等于1或2时，继续从码流中读取符号位s：coeff_level = pre_value*(1 – 2 *s)。

c) pre_value等于3时，

1) 从比特流的当前位置开始寻找第一个零比特，并将找到的非零比特个数记为leadingOneBits，用伪代码描述如下：

```
leadingOneBits = -1
for (b = 1; b; leadingOneBits++ )
    b = read_bits(1)
```

2) leadingOneBits小于等于2时，继续从码流中读取符号位s：coeff_level = （leadingOneBits + 3）*(1 – 2 * s)

3) leadingOneBits大于2时，比特串分为“前缀码”、“后缀码”和“符号位”三部分。前缀由leadingOneBits个连续的‘1’和一个‘0’构成。然后从比特流读取leadingOneBits – 2个比特得到后缀码，即表中的x_i串，x_i的值为‘0’或‘1’，令后缀码二元串对应的十进制数记为SuffixCode。最后从比特流读取符号位，即表34中的s，数值为‘0’表示正数，‘1’表示负数。coeff_level按如下步骤得到：

```
abs_level = (1 << (leadingOneBits – 2)) + 4 + SuffixCode;
coeff_level = abs_level * (1 – 2 * s)
```

表34 tableIdx 等于 1，coeff_level 的值与二元符号串的关系

coeff_level 取值范围	前缀码	后缀码	符号位
0	00	–	–
-1，1	01	–	s
-2，2	10	–	s
-3，3	110	–	s
-4，4	1110	–	s
-5，5	11110	–	s
-6，6，-7，7	111110	x_0	s
-11～-8，8～11	1111110	x_1 x_0	s
-19～-12，12～19	11111110	x_2 x_1 x_0	s
……	……	……	……

tableIdx等于2的码表，按以下方式解析coeff_level值，首先从比特流的当前位置开始读取2比特记为pre_value：

a) pre_value等于0时，coeff_level = 0。

b) pre_value等于1或2时，继续从码流中读取2个比特得到后缀码SuffixCode和符号位s：coeff_level = (2 * pre_value – 1 + SuffixCode) * (1 – 2 * s)。

c) pre_value等于3时，

1) 然后从比特流的当前位置开始寻找第一个零比特，并将找到的非零比特个数记为leadingOneBits，比特串分为“前缀码”、“后缀码”和“符号位”三部分。前缀由leadingOneBits个连续的‘1’和一个‘0’构成。leadingOneBits的获取用伪代码描述如下：

```
leadingOneBits = -1
for (b = 1; b; leadingOneBits++ )
    b = read_bits(1)
```

2) leadingOneBits等于0时，继续从码流中读取2个比特得到后缀码SuffixCode和符号位s：coeff_level = （SuffixCode + 5）*(1 – 2 * s)

3) leadingOneBits大于0时，继续从比特流读取leadingOneBits个比特得到后缀码，即表中的x_i串，x_i的值为‘0’或‘1’，令后缀码二元串对应的十进制数记为SuffixCode。最后从比特流读取符号位，即表35中的s，数值为‘0’表示正数，‘1’表示负数。coeff_level按如下步骤得到：

```
abs_level = (1 << leadingOneBits) + 5 + SuffixCode;
coeff_level = abs_level * (1 – 2 * s)
```

表35 tableIdx 等于 2，coeff_level 的值与二元符号串的关系

coeff_level 取值范围	前缀码	后缀码	符号位
0	00	–	–
-2，-1，1，2	01	x_0	s
-4，-3，3，4	10	x_0	s
-6，-5，5，6	110	x_0	s
-8，-7，7，8	1110	x_0	s
-12～-9，9～12	11110	x_1 x_0	s
-20～-13，13～20	111110	x_2 x_1 x_0	s
……	……	……	……

tableIdx等于3的码表，首先从比特流的当前位置开始读取2比特记为pre_value。如果pre_value等于0时，coeff_level = 0。如果pre_value大于0时，继续从码流中读取1个比特得到数值k，计算pre_value_sec = 2 * pre_value + k，然后按以下方式解析coeff_level值：

a) pre_value_sec等于7时，

1) 然后从比特流的当前位置开始寻找第一个零比特，并将找到的非零比特个数记为leadingOneBits，比特串分为“前缀码”、“后缀码”和“符号位”三部分。前缀由leadingOneBits个连续的‘1’和一个‘0’构成。

2) 然后从比特流读取leadingOneBits + 5个比特得到后缀码，即表中的x_i串，x_i的值为‘0’或‘1’，令后缀码二元串对应的十进制数记为SuffixCode。最后从比特流读取符号位，即表中的s，数值为‘0’表示正数，‘1’表示负数。

3) abs_level = (1 << (leadingOneBits + 5)) + SuffixCode;

4) coeff_level = abs_level * (1 – 2 * s)

b) pre_value_sec小于7时，

1) 然后从比特流读取pre_value_sec – 2个比特得到后缀码，即表中的x_i串，x_i的值为‘0’或‘1’,令后缀码二元串对应的十进制数记为SuffixCode。若pre_value_sec等于2，则SuffixCode为0。最后从比特流读取符号位，即表36中的s，数值为‘0’表示正数，‘1’表示负数。

2) abs_level = (1 << (pre_value_sec – 2)) + SuffixCode;

3) coeff_level = abs_level * (1 – 2 * s)

表36 tableIdx 等于 3，coeff_level 的值与二元符号串的关系

coeff_level 取值范围	前缀码	后缀码	符号位
0	00	–	–
-1，1	010	–	s
-3，-2，2，3	011	x_0	s
-7～-4，4～7	100	x_1 x_0	s
-15～-8，8～15	101	x_2 x_1 x_0	s
-31～-16，16～31	110	x_3 x_2 x_1 x_0	s
-63～-32，32～63	1110	x_4 x_3 x_2 x_1 x_0	s
-127～-64，64～127	11110	x_5 x_4 x_3 x_2 x_1 x_0	s
……	……	……	……

8.3.2.2 block_mode_flag 等于 0 的条件下 coeff_level 解析方法

8.3.2.2.1 0001 模式解析方法

由pattern_0001_code查表37确定系数组的四个系数的coeff_level的值：

表37 系数组的内四个系数的 coeff_level 的值与 pattern_0001_code 的关系

pattern_0001_code	coeff_level[0]	coeff_level[1]	coeff_level[2]	coeff_level[3]
0 0 0	1	0	0	0
0 0 1	-1	0	0	0
0 1 0	0	1	0	0
0 1 1	0	-1	0	0
1 0 0	0	0	1	0
1 0 1	0	0	-1	0
1 1 0	0	0	0	1
1 1 1	0	0	0	-1

8.3.2.2.2 max_grt1_flag 等于 0 的条件下 coeff_level 解析方法

根据以下步骤依次执行,确定coeff_level的值:首先从比特流的当前位置读取1个比特得到数值k。k等于0时，coeff_level等于0。k等于1时，继续从比特流中读取1个比特得到s，按如下方式计算coeff_level：

coeff_level = 1 - 2 * s。

8.3.2.2.3 max_grt1_flag 等于 1 的条件下 coeff_level 解析方法

由二元符号串查tableIdx等于1所对应的码表（表34）得到coeff_level的值。

9 解码过程

9.1 序列解码

序列的解码过程如下：

a) 解码序列头；
b) 计算当前序列的图像数量；
c) 解码当前序列中的图像（见9.2），得到各个图像的重建样本；
d) 重建样本构成解码序列并输出全分辨率解码序列或1/2下采样解码序列；

9.2 图像解码

图像的解码过程如下：

a) 解码图像头；
b) 计算当前图像的子图数量；
c) 解码当前图像的各个子图（见9.3），得到各个子图的重建样本；
d) 各个子图的重建样本构成解码图像；

9.3 子图解码

子图的解码过程如下：

a) 解码子图头；
b) 计算当前子图的低频子带BAC数据、低频子带VLC数据、高频子带BAC数据和高频频子带VLC数据的各自指针偏移量；
c) 依次解码当前图像的低频子带的编码单元（见9.4），得到低频子带重建图RecLL；
d) 若解码1/2下采样视频，对低频子带重建图RecLL进行数值运算，构成子图的低分辨率图（见9.7）；
e) 若解码全分辨率视频，执行如下步骤：
   1) 依次解码当前图像的高频子带的编码单元（见9.5），得到三个高频子带重建图HLSubPic、LHSubPic和HHSubPic；
   2) 进行小波反变换，构成重建子图（见9.6）；
f) 当存在alpha通道时，解码子图的alpha通道数据（见附录C）；

9.4 低频子带编码单元解码

9.4.1 解码步骤

低频子带编码单元中亮度分量宏块及色度分量宏块解码步骤如下：

a) 确定宏块的预测模式信息和量化参数相关信息（见9.4.2）
b) 变换块解码获得残差矩阵（见9.4.3）
c) 对宏块进行帧内预测（见9.4.4）或帧间预测（见9.4.5）获得预测矩阵
d) 进行预测补偿得到低频子带宏块的重建小波系数矩阵（见9.4.6）

9.4.2 确定编码单元类型和相关信息

9.4.2.1 宏块量化参数

宏块量化参数按如下步骤确定：

a) 若mb_qp_delta_enabled_flag为0，
   - 亮度宏块量化参数$MbQP_y$ = subpic_ll_qp_index
   - 色度Cb分量宏块量化参数$MbQP_{cb}$ = SubpicLLcbQPindex
   - 色度Cr分量宏块量化参数$MbQP_{cr}$ = SubpicLLcrQPindex

b) 若mb_qp_delta_enabled_flag为1，
   - 若当前宏块MbX为0：

     $MbQP_y$ = clip(0, 39, subpic_ll_qp_index + ll_mb_qp_delta)

     $MbQP_{cb}$ = clip(0, 39, SubpicLLcbQPindex + ll_mb_qp_delta)

     $MbQP_{cr}$ = clip(0, 39, SubpicLLcrQPindex + ll_mb_qp_delta)
   - 若当前宏块MbX不为0，令相同色彩分量上左侧宏块量化参数记为$LeftQP_x$，x代表亮度y或色度cb或色度cr：

     $MbQP_x$ = clip(0, 39, $LeftQP_x$ + ll_mb_qp_delta)

9.4.2.2 宏块的变换块大小及变换类型

变换块大小TbSize按如下方法确定：

- 对于亮度宏块，若luma_tb_size为0，变换块大小为4x4，TbSize记为TB_SIZE4x4；若luma_tb_size为1，变换块大小为8x8，TbSize记为TB_SIZE8x8。
- 对于色度宏块，若chroma_format为YUV444，变换块大小为8x8，TbSize记为TB_SIZE8x8；若chroma_format为YUV422，变换块大小为4x8，TbSize记为TB_SIZE4x8。

变换类型按如下步骤确定：

- 对于亮度宏块，不同预测模式及不同TbSize下水平变换和垂直变换类型如表38
- 对于色度宏块，不同预测模式及不同TbSize下水平变换和垂直变换类型如表39

表38 亮度宏块垂直方向和水平方向变换类型

TbSize	mb_mode	intra_pred_mode_luma_first_flag/ intra_pred_mode_luma_second_flag	垂直方向变换类型	水平方向变换类型
TB_SIZE4x4	0	0/0	$DST7_4$	$DCT2_4$
TB_SIZE4x4	0	0/1	$DCT2_4$	$DCT2_4$
TB_SIZE4x4	0	1/-	$DCT2_4$	$DST7_4$
TB_SIZE8x8	-	-	$DCT2_8$	$DCT2_8$

表39 色度宏块垂直方向和水平方向变换类型

TbSize	mb_mode	intra_pred_mode_chroma_first_flag	垂直方向变换类型	水平方向变换类型
TB_SIZE4x8	0	0	$DCT2_8$	$DCT2_4$
TB_SIZE4x8	0	1	$DCT2_8$	$DST7_4$
TB_SIZE4x8	1	-	$DCT2_8$	$DCT2_4$
TB_SIZE8x8	-	-	$DCT2_8$	$DCT2_8$

9.4.2.3 宏块的运动矢量信息

本条定义宏块运动矢量的确定方法。

若mb_mode等于1，MvDiffX表示水平方向半像素运动矢量差，MvDiffY表示垂直方向半像素运动矢量差，按如下步骤导出宏块半像素运动矢量（MbMvX，MbMvY），其中水平方向半像素运动矢量MbMvX取值范围为[-14, 14]，垂直方向半像素运动矢量MbMvY取值范围为[-6,6]：

- 若当前宏块MbX不为0且左侧宏块mb_mode等于1，令左侧宏块半像素运动矢量MV记为（leftMvX，leftMvY），则MVP_x = leftMvX，MVP_y = leftMvY。
- 若当前宏块MbX等于0或左侧宏块mb_mode等于0，则MVP_x = 0，MVP_y = 0。
- MbMvX = clip(-14, 14, MVP_x + MvDiffX)，MbMvY = clip(-6, 6, MVP_y + MvDiffY)

9.4.3 变换块解码

9.4.3.1 解码步骤

首先，对量化系数值数组进行扫描重排序，获得量化系数矩阵（见9.4.3.2）；然后，对量化系数矩阵进行反量化，获得变换系数矩阵（见9.4.3.3）；最后，对变换系数矩阵进行反变换，获得残差矩阵（见9.4.3.4）：

9.4.3.2 扫描重排序

本条定义码流中解码得到的LL子带宏块的量化系数值重排序转化为二维量化系数矩阵QuantCoeffMatrix的过程。

令宏块中4x4系数块数目记为coeffGroupNum。亮度宏块coeffGroupNum为4。当chroma_format为YUV444时，色度宏块coeffGroupNum为4。当chroma_format为YUV422时，色度宏块coeffGroupNum为2。

令4x4系数块的扫描矩阵记为ScanOrderLL,三种候选扫描矩阵ScanOrderCanditate定义如下：

$$\text{ScanOrderCanditate}[0] = \begin{bmatrix} 0 & 1 & 2 & 3 \\ 4 & 5 & 6 & 8 \\ 7 & 9 & 10 & 12 \\ 11 & 13 & 14 & 15 \end{bmatrix}$$

$$\text{ScanOrderCanditate}[1] = \begin{bmatrix} 0 & 4 & 7 & 11 \\ 1 & 5 & 9 & 13 \\ 2 & 6 & 10 & 14 \\ 3 & 8 & 12 & 15 \end{bmatrix}$$

$$\text{ScanOrderCanditate}[2] = \begin{bmatrix} 0 & 1 & 5 & 6 \\ 2 & 3 & 7 & 12 \\ 4 & 8 & 11 & 13 \\ 9 & 10 & 14 & 15 \end{bmatrix}$$

二维量化系数矩阵QuantCoeffMatrix的导出过程如下：

```
if (TbSize == TB_SIZE4x4) {
    xArray = [0, 4, 0, 4]
    yArray = [0, 0, 4, 4]
}
else {
    xArray = [0, 0, 4, 4]
    yArray = [0, 4, 0, 4]
}
for (n = 0; n < coeffGroupNum; n++) {
```

```
intraPredModeLuma = (intra_pred_mode_luma_first_flag << 1) + intra_pred_mode_luma_second_flag
intraPredModeChroma = (intra_pred_mode_chroma_first_flag<<1) + intra_pred_mode_chroma_second_flag
if (mb_mode == 0 && (intraPredModeLuma == 00 || intraPredModeChroma == 00)){
    ScanOrderLL = ScanOrderCanditate[0]
}
else if (mb_mode == 0 && (intraPredModeLuma ==10 || intraPredModeChroma == 10)){
    ScanOrderLL = ScanOrderCanditate[1]
}
else {
    if (TbSize == TB_SIZE8x8) {
        orderIdxList = [2, 0, 1, 2]
        ScanOrderLL = ScanOrderCanditate[orderIdxList[n]]
    }
    else {
        ScanOrderLL = ScanOrderCanditate[2]
    }
}
for (i = 0; i < 4; i++) {
    for (j = 0; j < 4; j++) {
        pos = ScanOrderLL[i][j]
        x_pos = xArray[n]
        y_pos = yArray[n]
        QuantCoeffMatrix[y_pos + i][x_pos + j] = CoeffLevel[16*n + pos]
    }
}
}
```

9.4.3.3 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。本条输入为宏块的量化参数QP和量化系数QuantCoeffMatrix。本条反量化输出为CoeffMatrix。QuantCoeffMatrix的元素的数据位宽记为InputBitDepth。本条输出的CoeffMatrix的元素的数据位宽记为OutputBitDepth。

低频子带宏块量化参数QP为$MbQP_x$，低频子带QuantCoeffMatrix的数据位宽InputBitDepth为输入图像位宽BitDepth加3，量化系数取值范围应为$-2^{BitDepth+2}$～$2^{BitDepth+2}-1$。低频子带CoeffMatrix的数据位宽OutputBitDepth为输入图像位宽BitDepth加6，变换系数取值范围为$-2^{BitDepth+5}$～$2^{BitDepth+5}-1$。

二维变换系数矩阵CoeffMatrix的计算如下：

```
shift = 4 – ((QP + 12) >> 3)
```

```
offset = (shift <= 0) ? 0 : (1 << (shift -1))
scale = ScaleTable[(QP + 12) & 7]
for (i=0; i<M2; i++) {
    for (j=0; j<M1; j++) {
        if (shift > 0) {
            CoeffMatrix[i][j] = clip(–2^OutputBitDepth - 1, 2^OutputBitDepth - 1 -1, (QuantCoeffMatrix[i][j] * scale + offset) >> shift))
        }
        else {
            CoeffMatrix[i][j] = clip(–2^OutputBitDepth - 1, 2^OutputBitDepth - 1 -1, (QuantCoeffMatrix[i][j] * scale) << (-shift))
        }
    }
}
```

若$M_1 \times M_2$宏块的变换块大小TbSize为TB_SIZE4x4时，ScaleTable={64, 70, 76, 83, 91, 99, 108, 117}，
若$M_1 \times M_2$宏块的变换块大小TbSize为TB_SIZE4x8时，ScaleTable={45, 49, 54, 58, 64, 69, 76, 83}，
若$M_1 \times M_2$宏块的变换块大小TbSize为TB_SIZE8x8时，ScaleTable={32, 35, 38, 41, 45, 49, 54, 59 }；

9.4.3.4 反变换

本条定义由$M_1 \times M_2$宏块的变换系数矩阵CoeffMatrix转化为残差矩阵ResidueMatrix的获取方法。令当前宏块的变换块块大小记为$N_1 \times N_2$。$M_1 \times M_2$变换系数矩阵CoeffMatrix水平方向变换块个数为M_1/N_1，垂直方向变换块个数为M_2/N_2。对每一个变换块按如下步骤进行反变换：

a) 由变换块变换系数矩阵TuMatrix和变换矩阵T进行垂直方向变换，得到矩阵V。
- 如果N_2的值等于4，且变换类型为DCT2，变换矩阵T是4×4反变换矩阵$DCT2_4$的转置。
- 如果N_2的值等于4，且变换类型为DST7，变换矩阵T是4×4反变换矩阵$DST7_4$的转置。
- 如果N_2的值等于8，变换矩阵T是8×8反变换矩阵$DCT2_8$的转置。
- $V = clip(-2^{BitDepth+5}, 2^{BitDepth+5}-1, (T \times CoeffMatrix + 16) >> 5)$

b) 由矩阵V和变换矩阵T进行水平方向变换，得到残差矩阵。
- 如果N_1的值等于4，且变换类型为DCT2，变换矩阵T是4×4反变换矩阵$DCT2_4$。
- 如果N_1的值等于4，且变换类型为DST7，变换矩阵T是4×4反变换矩阵$DST7_4$。
- 如果N_1的值等于8，变换矩阵T是8×8反变换矩阵$DCT2_8$。
- $ResidueMatrix = clip(-2^{BitDepth+3}, 2^{BitDepth+3}-1, (V \times T + 64) >> 7)$

$$DST7_4 = \begin{bmatrix} 15 & 27 & 37 & 42 \\ 37 & 37 & 0 & -37 \\ 42 & -15 & -37 & 27 \\ 27 & -42 & 37 & -15 \end{bmatrix}$$

$$DCT2_4 = \begin{bmatrix} 32 & 32 & 32 & 32 \\ 42 & 17 & -17 & -42 \\ 32 & -32 & -32 & 32 \\ 17 & -42 & 42 & -17 \end{bmatrix}$$

$$DCT2_8 = \begin{bmatrix} 32 & 32 & 32 & 32 & 32 & 32 & 32 & 32 \\ 44 & 38 & 25 & 9 & -9 & -25 & -38 & -44 \\ 42 & 17 & -17 & -42 & -42 & -17 & 17 & 42 \\ 38 & -9 & -44 & -25 & 25 & 44 & 9 & -38 \\ 32 & -32 & -32 & 32 & 32 & -32 & -32 & 32 \\ 25 & -44 & 9 & 38 & -38 & -9 & 44 & -25 \\ 17 & -42 & 42 & -17 & -17 & 42 & -42 & 17 \\ 9 & -25 & 38 & -44 & 44 & -38 & 25 & -9 \end{bmatrix}$$

9.4.4 帧内预测

9.4.4.1 概述

本条定义$M_1 \times M_2$宏块的帧内预测过程。宏块的帧内预测块大小等于宏块的变换块大小。本条输出$M_1 \times M_2$宏块的预测像素矩阵PredMatrix。

令当前宏块的预测块PredBlock大小记为$N_1 \times N_2$。$M_1 \times M_2$预测像素矩阵PredMatrix水平方向预测块个数为M_1/N_1，垂直方向预测块个数为M_2/N_2。按如下方法进行获得预测矩阵PredMatrix：

- for v = 0,..., M_2/N_2-1:
  - for h = 0,..., M_1/N_1-1:
    - 获取参考样本（见9.4.4.2）
    - 进行帧内预测获得预测块PredBlock（见9.4.4.3）
    - PredMatrix[y + v * N_2][x + h * N_1] = PredBlock[y][x],其中，y = 0,1,2..., N_2-1；x =0,1,2..., N_1-1。

9.4.4.2 参考样本的获得

令当前子图的低频子带重建图记为RecLL。令当前预测块在当前子图的低频子带中的坐标为（xPu，yPu）。令预测块上侧参考样本数组记为upRefPixel，令上侧参考样本是否可得标志位记为upAvailable。令预测块左侧参考样本数组记为leftRefPixel，令左侧参考样本是否可得标志位记为leftAvailable。

- 若yPu == 0，上侧参考样本不可得，设置upAvailable等于0。否则，设置upAvailable等于1，并按如下步骤获取上侧参考样本数组upRefPixel：

```
for( i = 0; i < N1; i++) {
    upRefPixel[i] = RecLL[yPu – 1][ xPu + i]
}
```

- 若xPu == 0，左侧参考样本不可得，设置leftAvailable等于0。否则，设置leftAvailable等于1，并按如下步骤获取上侧参考样本数组leftRefPixel：

```
for( i = 0; i < N2; i++) {
    leftRefPixel[i] = RecLL[yPu + i][xPu – 1]
}
```

9.4.4.3 预测块的预测方法

- $N_1 \times N_2$预测块PredBlock的元素[i][j]的默认值IntraDefault计算方法如下：

$$IntraDefault = (2^{BitDepth-1} << PixelPrecision) + LLbandOffset = 2^{BitDepth+2}$$

- 亮度帧内垂直预测或色度帧内垂直预测，$N_1 \times N_2$预测块PredBlock的元素[i][j]的样本值获取方法如下，其中i的取值范围为0~ N_2-1，j的取值范围为0~ N_1-1：

```
if ( upAvailable ) {
        PredBlock[i][j] = upRefPixel[j]
}
else {
        PredBlock[i][j] = IntraDefault
}
```

- 亮度帧内水平预测或色度帧内水平预测，$N_1 \times N_2$预测块PredBlock的元素[i][j]的样本值获取方法如下，其中i的取值范围为0~ N_2-1，j的取值范围为0~ N_1-1：

```
if ( leftAvailable ) {
    PredBlock[i][j] = leftRefPixel[i]
}
else {
    PredBlock[i][j] = IntraDefault
}
```

- 亮度帧内直流预测或色度帧内直流预测，$N_1 \times N_2$预测块PredBlock的元素[i][j]的样本值获取方法如下，其中i的取值范围为0~ N_2-1，j的取值范围为0~ N_1-1：

```
if ( upAvailable && leftAvailable) {
    if (N1 == N2) {
        sum = ∑_{j=1}^{N1/2} upRefPixel[2 * j − 1] + ∑_{i=1}^{N2/2} leftRefPixel[2 * i − 1]
        shift = (N1 == 8) ? 3 : 2
    }
    else {
        sum = ∑_{j=0}^{3} upRefPixel[j] + ∑_{i=1}^{4} leftRefPixel[2 * i − 1]
        shift = 3
    }
    PredBlock[i][j] = ( sum + (1 << (shift - 1))) >> shift
}
else if ( upAvailable ) {
    sum = ∑_{i=0}^{N1−1} upRefPixel[i]
    shift = (N1 == 8) ? 3 : 2
    PredBlock[i][j] = ( sum + (1 << (shift - 1))) >> shift
}
else if (leftAvailable ) {
    sum = ∑_{i=0}^{N2−1} leftRefPixel[i]
    shift = (N2 == 8) ? 3 : 2
    PredBlock[i][j] = ( sum + (1 << (shift - 1))) >> shift
}
else {
    PredBlock[i][j] = IntraDefault
}
```

- 色度 Cb 分量与色度 Cr 帧内跨分量预测方法相同，按 9.4.4.2 所述方法获取色度 Cb 分量（或色度 Cr 分量）的左侧参考样本数组 leftRefPixelChroma、上侧参考样本数组 upRefPixelChroma、标志符 leftAvailable 和 upAvailable。$N_1 \times N_2$预测块 PredBlock 等于色度宏块大小。leftAvailable 和 upAvailable 均为 0 时，$N_1 \times N_2$预测块 PredBlock 的元素[i][j]的样本值设置为 IntraDefault，否则按如下步骤获取 PredBlock[i][j]的样本值：
  - 按 9.4.4.2 所述方法获取亮度分量左侧参考样本数组 leftRefPixelLuma 和上侧参考样

本数组 upRefPixelLuma；

- 根据 leftRefPixelLuma、upRefPixelLuma、leftRefPixelChroma、upRefPixelChroma、leftAvailable 和 upAvailable，按照以下步骤获得色度 Cb 分量（或色度 Cr 分量）的 $N_1 \times N_2$ 预测块 PredBlock 的元素[i][j]的样本值，其中 i 的取值范围为 0~ N_2-1，j 的取值范围为 0~ N_1-1：
  - 计算亮度分量的三个代表样本值和色度分量的三个代表样本值：

if (upAvailable && leftAvailable) {

$R_Y^0 = (\text{upRefPixelLuma}[0] + \text{leftRefPixelLuma}[0]) \gg 1$

$R_C^0 = (\text{upRefPixelChroma}[0] + \text{leftRefPixelChroma}[0]) \gg 1$

$R_Y^1 = (N_1 == 8) ? \text{upRefPixelLuma}[7] : ((\text{upRefPixelLuma}[6] + \text{upRefPixelLuma}[7]) \gg 1)$

$R_C^1 = \text{upRefPixelChroma}[N_1 - 1]$

$R_Y^2 = \text{leftRefPixelLuma}[7]$

$R_C^2 = \text{leftRefPixelChroma}[N_2 - 1]$

}

else if (upAvailable) {

$R_Y^0 = (N_1 == 8) ? \text{upRefPixelLuma}[0] : (\text{upRefPixelLuma}[0] + \text{upRefPixelLuma}[1]) \gg 1$

$R_C^0 = \text{upRefPixelChroma}[0]$

$R_Y^1 = (N_1 == 8) ? \text{upRefPixelLuma}[4] : (\text{upRefPixelLuma}[4] + \text{upRefPixelLuma}[5]) \gg 1$

$R_C^1 = \text{upRefPixelChroma}[N_1/2]$

$R_Y^2 = (N_1 == 8) ? \text{upRefPixelLuma}[7] : ((\text{upRefPixelLuma}[6] + \text{upRefPixelLuma}[7]) \gg 1)$

$R_C^2 = \text{upRefPixelChroma}[N_1 - 1]$

}

else if (leftAvailable) {

$R_Y^0 = \text{leftRefPixelLuma}[0]$

$R_C^0 = \text{leftRefPixelChroma}[0]$

$R_Y^1 = \text{leftRefPixelLuma}[4]$

$R_C^1 = \text{leftRefPixelChroma}[N_2/2]$

$R_Y^2 = \text{leftpRefPixelLuma}[7]$

$R_C^2 = \text{leftRefPixelChroma}[N_2 - 1]$

}

  - 根据亮度分量的三个代表样本值大小关系得到样本索引号idx[3]:

if ($R_Y^0 \le R_Y^1 \le R_Y^2$) {

$idx[0] = 0$

$idx[1] = 1$

$idx[2] = 2$

}

else if ($R_Y^0 \le R_Y^2 \le R_Y^1$) {

$idx[0] = 0$

$idx[1] = 2$

$idx[2] = 1$

}

else if ($R_Y^1 \le R_Y^0 \le R_Y^2$) {

$idx[0] = 1$

$idx[1] = 0$

$idx[2] = 2$

}

else if ($R_Y^1 \le R_Y^2 \le R_Y^0$) {

$idx[0] = 1$

$idx[1] = 2$

$idx[2] = 0$

}

else if ($R_Y^2 \le R_Y^0 \le R_Y^1$) {

$idx[0] = 2$

$idx[1] = 0$

$idx[2] = 1$

}

else if ($R_Y^2 \le R_Y^1 \le R_Y^0$) {

$idx[0] = 2$

$idx[1] = 1$

$idx[2] = 0$

}

– 根据$R_Y^{idx[0]}$、$R_Y^{idx[1]}$、$R_Y^{idx[2]}$算关键点和三点距离：

if ($2*R_Y^{idx[1]} > R_Y^{idx[0]} + R_Y^{idx[2]}$) {

$R_C^{key} = (R_C^{idx[1]} + R_C^{idx[2]}) \gg 1$

$R_Y^{key} = (R_Y^{idx[1]} + R_Y^{idx[2]}) \gg 1$

$D_Y = -3 \times R_Y^{idx[0]} + R_Y^{idx[1]} + 2 \times R_Y^{idx[2]}$

$D_C = (-3 \times R_C^{idx[0]} + R_C^{idx[1]} + 2 \times R_C^{idx[2]}$

}

else {

$R_C^{key} = (R_C^{idx[0]} + R_C^{idx[1]}) \gg 1$

$R_Y^{key} = (R_Y^{idx[0]} + R_Y^{idx[1]}) \gg 1$

$D_Y = -2 \times R_Y^{idx[0]} - R_Y^{idx[1]} + 3 \times R_Y^{idx[2]}$

$$D_C = -2 \times R_C^{idx[0]} - R_C^{idx[1]} + 3 \times R_C^{idx[2]}$$

}

$$D_Y = Clip(0, (2^{10} - 16) - 1, D_Y >> (BitDepth - 7))$$

$$D_C = Clip(-2^{\mathrm{BitDepth}+2}, 2^{\mathrm{BitDepth}+2} - 1, D_C)$$

- 计算PredBlock的元素[i][j]的样本值：

Table = floor($\log_2$ (D_Y + 16) − 4)

index = ((D_Y + 16) >> Table)&15

scale = (ScaleListCCLM[index] * D_C) >> 10

if ((N_1 == 8) && (N_2 == 8)) {

recY[i][j] = RecLL[yPu + i][xPu + j]

}

if ((N_1 == 4) && (N_2 == 8)) {

recY[i][j] = (RecLL[yPu + i][xPu + 2 * j] + RecLL[yPu + i][xPu + 2 * j + 1]) >> 1

}

PredBlock[i][j] = (((recY[i][j] − R_Y^{key}) * scale >> (BitDepth − 7 + Table)) + R_c^{key}

PredBlock[i][j] = Clip(0, $2^{\mathrm{BitDepth}+3}$ − 1, PredBlock[i][j])

其中，ScaleListCCLM[16] = {63, 63, 61, 57, 54, 51, 49, 47, 45, 43, 41, 39, 38, 37, 35, 34}。

9.4.5 帧间预测

9.4.5.1 概述

本条定义M_1×M_2宏块的帧间预测过程。宏块的帧间预测块大小等于宏块大小。

9.4.5.2 预测样本的获得

本条输入M_1×M_2宏块半像素运动矢量(MbMvX, MbMvY)，宏块在子图中水平方向和垂直方向索引号(MbX, MbY)，子图索引号subPicIdx。

本条输出M_1×M_2宏块的预测像素矩阵PredMatrix。

预测像素矩阵PredMatrix的元素[i][j]是参考帧半像素精度样本矩阵中(xPos + 2 * j, yPos + 2 * i)位置的样本值，该样本值按9.4.5.3所述过程获得。

对于亮度块，按如下方法计算xPos及yPos ：

xPos = (MbX << 4) + MbMvX

yPos = (MbY << 4) + MbMvY

对于色度块，按如下步骤计算xPos及yPos

- 计算色度块水平方向和垂直方向的运动矢量整像素值：

xMVint = MbMvX >> (1+FormatShiftX)

yMVint = MbMvY >> (1+FormatShiftY)

- 计算色度块水平方向和垂直方向的运动矢量分像素值：

xMVfrac = (MbMvX % (1 <<(1+FormatShiftX)) > 0) ? 1 :0

yMVfrac = (MbMvY % (1 <<(1+FormatShiftY)) > 0) ? 1 :0

- 计算xPos及yPos：

  xPos = (MbX << (4 - FormatShiftX)) + (xMVint << 1) + xMVfrac

  yPos = (MbY << (4 - FormatShiftY)) + (yMVint << 1) + yMVfrac

9.4.5.3 预测样本的插值方法

本条定义亮度分量及色彩分量预测样本半像素精度插值方法。A,B,C,D是相邻整像素样本。dx与dy是整像素样本A周边分像素样本a(dx,dy)与A的水平和垂直距离，dx等于x & 1，dy等于y & 1，其中(x,y)表示该像素的半像素精度位置。

样本a(dx,dy)的样本值计算方法如下：

a(0,0) = (A << 2)

a(1,0) = (A + B) << 1

a(0,1) = (A + C) << 1

a(1,1) = A + B + C + D

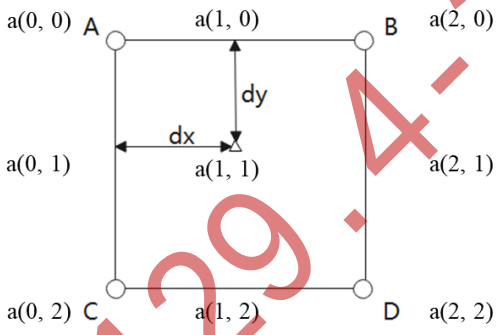

图4 整像素及分像素样本位置示意图

参考子图外的整数样本应使用该参考子图内距离该样本最近的整数样本（边缘或角样本）代替，如图5所示。像素填充数量应满足运动矢量范围约束，亮度分量水平方向单侧像素填充数量不小于7，垂直方向单侧像素填充数量不小于3；色度分量水平方向单侧像素填充数量不小于（（7 + FormatShiftX）>> FormatShiftX），垂直方向单侧像素填充数量不小于3。

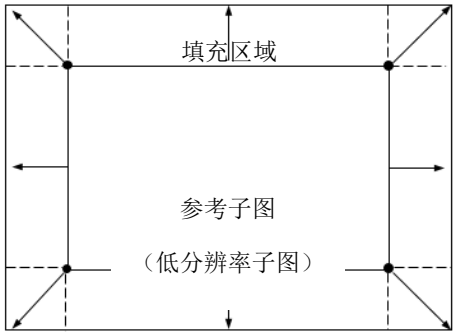

图5 参考子图上下左右填充示意图

9.4.6 预测补偿

本条输入宏块残差矩阵ResidueMatrix和预测矩阵PredMatirx。

本条输出宏块重建小波系数矩阵RecMatrix。M_1×M_2重建块RecMatrix的元素[i][j]的重建值按如下步骤导出，其中i= 0，...，M_2-1；j= 0，...，M_1-1：

对于帧内编码模式，按如下步骤计算重建小波系数矩阵RecMatrix的元素[i][j]的样本值：

$$RecMatrix[i][j] = clip(0, 2^{BitDepth+3}-1, ResidueMatrix[i][j] + PredMatirx [i][j])$$

对于帧间编码模式，按如下步骤计算重建矩阵RecMatrix的元素[i][j]的样本值：

$$RecMatrix[i][j] = clip(0, 2^{BitDepth+3}-1, ResidueMatrix[i][j] + PredMatirx [i][j] + LLbandOffset)$$

9.5 高频子带编码单元解码

9.5.1 解码步骤

高频子带HL子带、LH子带及HH子带编码单元解码中亮度分量宏块及色度分量宏块解码包括:

a) 确定宏块的变换模式信息和量化参数相关信息（见9.5.2）

b) 变换块解码获得高频子带宏块的重建小波系数矩阵（见9.5.3）

9.5.2 确定编码单元类型和相关信息

三个高频子带（HL，LH或HH）及三种色彩分量（X为Y、Cb或Cr）宏块量化参数获取步骤相同，令子带索引号记为hfband，hfband为0表示HL子带，hfband为1表示LH子带，hfband为2表示HH子带。色彩分量索引号记为comp，comp为0表示亮度分量，comp为1表示色彩Cb分量，comp为2表示色彩Cr分量。

宏块量化参数MbQP[hfband][comp],按如下步骤确定：

- 若mb_qp_delta_enabled_flag为0:
  - MbQP[hfband][comp] = SubpicHFQPindex[hfband][comp]
- 若mb_qp_delta_enabled_flag为1:
  - 若当前宏块MbX为0：
    - MbQP[hfband][comp] = clip(0, 39, SubpicHFQPindex[hfband][comp] + hf_mb_qp_delta)
  - 若当前宏块MbX不为0，令相同子带相同色彩分量的左侧宏块量化参数记为LeftQP[hfband][comp]:
    - MbQP[hfband][comp] = clip(0, 39, LeftQP[hfband][comp] + hf_mb_qp_delta)

9.5.3 变换块解码

9.5.3.1 解码步骤

首先，对量化系数值数组进行扫描重排序，获得量化系数矩阵（见9.5.3.2）；然后，对量化系数矩阵进行反量化，获得变换系数矩阵（见9.5.3.3）；最后，对变换系数矩阵进行反变换，获得重建矩阵（见9.5.3.4）：

9.5.3.2 扫描重排序

本条定义码流中解码得到的高频子带宏块的量化系数值重排序转化为二维量化系数矩阵QuantCoeffMatrix的过程。

令宏块中4x4系数块数目记为coeffGroupNum。亮度宏块coeffGroupNum为4。当chroma_format为YUV444时，色度宏块coeffGroupNum为4。当chroma_format为YUV422时，色度宏块coeffGroupNum为2。

令4x4系数块的扫描矩阵记为ScanOrderHF,不同子带及transform_skip_flag不同取值下ScanOrderHF数值定义如下：

若transform_skip_flag为0且当前宏块为HL子带的宏块时，

$$ScanOrderHF = \begin{bmatrix} 4 & 12 & 5 & 13 \\ 0 & 8 & 1 & 9 \\ 6 & 14 & 7 & 15 \\ 2 & 10 & 3 & 11 \end{bmatrix}。$$

若transform_skip_flag为0且当前宏块为LH子带的宏块时，

$$ScanOrderHF = \begin{bmatrix} 4 & 0 & 5 & 1 \\ 12 & 8 & 13 & 9 \\ 6 & 2 & 7 & 3 \\ 14 & 10 & 15 & 11 \end{bmatrix}。$$

若transform_skip_flag为0且当前宏块为HH子带的宏块时，

$$ScanOrderHF = \begin{bmatrix} 0 & 4 & 1 & 5 \\ 12 & 8 & 13 & 9 \\ 2 & 6 & 3 & 7 \\ 14 & 10 & 15 & 11 \end{bmatrix}。$$

若当前宏块为transform_skip_flag为1的亮度宏块时，

$$ScanOrderHF = \begin{bmatrix} 0 & 1 & 4 & 5 \\ 2 & 3 & 6 & 7 \\ 8 & 9 & 12 & 13 \\ 10 & 11 & 14 & 15 \end{bmatrix}。$$

二维量化系数矩阵QuantCoeffMatrix的导出过程如下：

```
xArray[4] = [0, 0, 4, 4]
yArray[4] = [0, 4, 0, 4]
for (n = 0; n < coeffGroupNum; n++) {
    for (i = 0; i < 4; i++) {
        for (j = 0; j < 4; j++) {
            pos = ScanOrderHF[i][j]
            xPos = xArray[n]
            yPos = yArray[n]
            QuantCoeffMatrix[yPos + i][ xPos + j] = coeff_level[16*n + pos]
        }
    }
}
```

9.5.3.3 反量化

本条定义高频子带$M_1 \times M_2$宏块的量化系数矩阵QuantCoeffMatrix转化为高频子带宏块的变换系数矩阵CoeffMatrix的方法。

与9.4.3.3描述的低频子带反量化过程相同，高频子带量化参数获取方式如9.5.2所述，高频子带宏块量化参数QP为MbQP[BandIdx][CompIdx]。高频子带QuantCoeffMatrix的数据位宽InputBitDepth为BitDepth减1，量化系数取值范围应为$-2^{BitDepth-2} \sim 2^{BitDepth-2}-1$。高频子带CoeffMatrix的数据位宽OutputBitDepth为BitDepth加4，变换系数取值范围为$-2^{BitDepth+3} \sim 2^{BitDepth+3}-1$。ScaleTable={64,70,76,83,91,99 ,108 ,117 }。

9.5.3.4 反变换

本条定义高频子带M_1×M_2宏块的变换系数矩阵CoeffMatrix转化为高频子带宏块的重建小波系数矩阵WaveMatrix的方法。

高频子带变换块大小为2x2，高频子带M_1×M_2宏块水平方向变换块个数nHor为$M_1/2$，垂直方向变换块个数nVer为$M_2/2$。

若高频子带宏块的transform_skip_flag为0，高频子带宏块M_1×M_2变换系数矩阵CoeffMatrix中每个2x2块按如下步骤执行变换：

$$-\begin{bmatrix} y[0][0] \\ y[0][1] \\ y[1][0] \\ y[1][1] \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & -1 & 1 & -1 \\ 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 \end{bmatrix} \times \begin{bmatrix} x[0][0] \\ x[0][1] \\ x[1][0] \\ x[1][1] \end{bmatrix}$$

$$-y[i][j] = (y[i][j] > 0)\ ?\ (y[i][j] + 1) \gg 1) : -((-y[i][j] + 1) \gg 1))$$

$$-y[i][j] = clip(-2^{BitDepth+2}, 2^{BitDepth+2} - 1, y[i][j])$$

若高频子带宏块的transform_skip_flag为1，高频子带宏块M_1×M_2变换系数矩阵CoeffMatrix中每个数据执行clip操作：

$$-y[i][j] = clip(-2^{BitDepth+2}, 2^{BitDepth+2} - 1, y[i][j])$$

9.6 小波反变换

9.6.1 概述

本条输入低频子带重建图RecLL，HL子带重建图HLSubPic，LH子带重建图LHSubPic，HH子带重建图HHSubPic。RecLL、HLSubPic、LHSubPic和HHSubPic的宽均为BandWidth，RecLL、HLSubPic、LHSubPic和HHSubPic的高为BandHeight。对应的小波正变换过程实现参考附录D。

本条输出重建子图，具体步骤如下：

a) 低频子带偏移量记为LLbandOffset，LLbandOffset为$2^{BitDepth+1}$。

b) 图像精度记为PixelPrecision，PixelPrecision为2。

c) 亮度分量与色度分量均按如下步骤执行垂直方向5/3小波反变换，得到矩阵L和矩阵H：

1) LL子带与LH子带逐列按如下步骤执行一维小波反变换，得到矩阵L：

– RecLL的第i列数据减去低频子带偏移量LLbandOffset得到的数组作为低频数组S，LHSubPic第i列数据作为高频数组，经如9.6.2所述的一维小波反变换得到矩阵L的第i列数据。

S[n] = RecLL[n][i] − LLbandOffset，其中n = 0,…, BandHeight-1

2) HL子带与HH子带逐列按如下步骤执行一维小波反变换，得到矩阵H：

– HLSubPic第i列数据作为低频数组，HHSubPic第i列数据作为高频数组，经如9.6.2所述的一维小波反变换得到矩阵H的第i列数据。

d) 水平方向执行小波反变换得到重建子图：

1) 矩阵L和矩阵H逐行按如下步骤执行一维小波反变换，得到矩阵R。

– 矩阵L的第i行数据作为低频数组，矩阵H的第i行数据作为高频数组。若当前分量为亮度分量，则经如9.6.3所述的一维9/7小波反变换得到矩阵R的第i行数据。若当前分量为色度分量，则经如9.6.2所述的一维5/3小波反变换得到矩阵R的第i行数据。

e) 矩阵R经如下处理得到重建子图RecImg：

RecImg[i][j] = clip(0, $2^{BitDepth}$-1, (R[i][j] + 2) >> PixelPrecision)，其中i = 0,…, BandHeight-1; j = 0,…, BandWidth-1

9.6.2 5/3 小波反变换

本条定义5/3小波反变换的具体方法，输入低频数组和高频数组，令低频数组记为S，令高频数组记为D，令输入数组长度为L；

本条输出为长度为2L的数组X：

for (i = 0 ; i < L; i++) {
    D[i] = D[i] << 1
}
X[0] = S[0] – ((2 * D[0] + 2) >> 2)
X[0] = clip($-2^{BitDepth+4}, 2^{BitDepth+4}-1, X[0]$)
for (i = 2 ; i <= 2L – 2; i = i + 2) {
    X[i] = S[i/2] – ((D[(i – 1) / 2] + D[(i + 1) / 2] + 2) >> 2)
    X[i] = clip($-2^{BitDepth+4}, 2^{BitDepth+4}-1, X[i]$)
}
for (i = 1 ; i < 2L – 1; i = i + 2) {
    X[i] = D[i / 2] + ((X[i – 1] + X[i + 1] + 1) >> 1)
    X[i] = clip($-2^{BitDepth+4}, 2^{BitDepth+4}-1, X[i]$)
}
X[2L – 1] = D[(2L – 1) / 2] + ((2 * X[2L – 2] + 1) >> 1)
X[2L – 1] = clip($-2^{BitDepth+4}, 2^{BitDepth+4}-1, X[2L-1]$)

9.6.3 9/7 小波反变换

本条定义9/7小波反变换的具体方法，输入低频数组和高频数组，令低频数组记为S，令高频数组记为D，令输入数组长度为L；

本条输出为长度为2L的数组X：

for (i = 0 ; i <= 2L – 2; i = i + 2){
    a = (i/2 < 1) ? D[0] : D[i/2 – 1]
    X[i] = S[i/2] – ((D[i/2] + a + 1) >> 1))
    X[i] = clip($-2^{BitDepth+4}, 2^{BitDepth+4}-1$,X[i])
}
for (i = 1 ; i <= 2L – 1; i = i + 2){
    $s_{i/2-1}$ = (i/2 < 1) ? S[1] : S[i/2 – 1]
    if (i/2 – 1 <0) {
        $d_{i/2-2}$ = D[1]
        $d_{i/2-1}$ = D[0]
    }
    else if (i/2 – 2 < 0) {
        $d_{i/2-2}$ = D[0]
        $d_{i/2-1}$ = D[i/2 – 1]
    }
    else {
        $d_{i/2-2}$ = D[i/2 – 2]
        $d_{i/2-1}$ = D[i/2 – 1]
    }
    if (i/2+1 > L – 1) {
        $s_{i/2+1}$ = S[L – 1]
        $s_{i/2+2}$ = S[L – 2]

$d_{i/2+1} = D[L-2]$

$d_{i/2+2} = D[L-3]$

}

else if (i/2+2 > L – 1) {

$s_{i/2+1} = S[i/2+1]$

$s_{i/2+2} = S[L-1]$

$d_{i/2+1} = D[i/2+1]$

$d_{i/2+2} = D[L-2]$

}

else {

$s_{i/2+1} = S[i/2+1]$

$s_{i/2+2} = S[i/2+2]$

$d_{i/2+1} = D[i/2+1]$

$d_{i/2+2} = D[i/2+2]$

}

$X[i] = (((9*(S[i/2] + s_{i/2+1}) - s_{i/2-1} - s_{i/2+2} + 8) >> 4) + ((d_{i/2-2} + d_{i/2+2} - ((d_{i/2-1} + d_{i/2+1}) << 3) + 16) >> 5) + ((23 * D[i/2] + 8) >> 4)$

$X[i] = \mathrm{clip}(-2^{BitDepth+4}, 2^{BitDepth+4} - 1, X[i])$

}

9.7 子图的低分辨率图解码

本条输入子图的低频子带重建图 RecLL，RecLL 的宽为 BandWidth，RecLL 的高为 BandHeight。

本条输出子图的 1/2 下采样图 RecDownPic，RecDownPic 的宽为 BandWidth，RecDownPic 的高为 BandHeight。

RecDownPic 的元素[i][j]的样本值确定方法如下：

RecDownPic[i][j] = clip(0, $2^{BitDepth}$-1, (RecLL[i][j] – LLbandOffset + 2) >> PixelPrecision)，其中，i = 0，…，BandHeight-1；j = 0，…，BandWidth-1。

若当前图像帧类型为 I 帧，RecDownPic 是序列中后续 P 帧对应子图的 LL 子带的参考子图。

附 录 A
（规范性）
档次和级别

A.1 概述

档次和级别提供了一种定义本文件的语法和语义的子集的手段。档次和级别对码流进行了各种限制，同时也就规定了对某一特定码流解码所需要的解码器能力。档次是本文件规定的语法、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。级别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。在给定档次的情况下，不同级别往往意味着对解码器能力和存储器容量的不同要求。

本附录描述了不同档次和级别所对应的各种限制。所有未被限定的语法元素和参数可以取任何本文件所允许的值。如果一个解码器能对某个档次和级别所规定的语法元素的所有允许值正确解码，则称此解码器在这个档次和级别上符合本文件。如果一个码流中不存在某个档次和级别所不允许的语法元素，并且其所含有的语法元素的值不超过此档次和级别所允许的范围，则认为此码流在这个档次和级别上符合本文件。

profile_idc和level_idc定义了码流的档次和级别。

A.2 档次

本文件定义的档次见表A.1.

表 A.1 档次

profile_idc的值	档次
0x00	基础帧内档次（Main Intra profile）
0x01	扩展帧内档次（Extended Intra profile）
0x02	基础档次（Main profile）
0x03	扩展档次（Extended profile）
0x10	基础帧内含透明度档次（Main Intra with Alpha profile）
0x11	扩展帧内含透明度档次（Extended Intra with Alpha profile）
0x12	基础含透明度档次（Main with Alpha profile）
0x13	扩展含透明度档次（Extended with Alpha profile）
其他	保留

对于一个给定的档次，不同的级别支持相同的语法子集。

基础帧内档次的码流应满足以下条件：
——profile_idc的值为0x00。
——num_of_frames_minus1的值为0或1。
——bit_depth_minus8为2。
——chroma_format为1。
——序列内所有图像的frame_type为0。
——alpha_map_flag为0。

扩展帧内档次的码流应满足以下条件：
——profile_idc的值应为0x01。
——num_of_frames_minus1的值应为0或1。
——bit_depth_minus8应为2或4。
——chroma_format应为0～2。
——序列内所有图像的frame_type应为0。
——alpha_map_flag应为0。

基础档次的码流应满足以下条件：
——profile_idc的值应为0x02。
——num_of_frames_minus1的值应为0或1。
——bit_depth_minus8应为2。
——chroma_format应为1。
——序列内第一幅图像的frame_type应为0。
——alpha_map_flag应为0。

扩展档次的码流应满足以下条件：
——profile_idc的值应为0x03。
——num_of_frames_minus1的值应为0或1。
——bit_depth_minus8应为2或4。
——chroma_format应为0～2。
——序列内第一幅图像的frame_type应为0。
——alpha_map_flag应为0。

基础帧内含透明度档次的码流应满足以下条件：
——profile_idc的值应为0x10。
——num_of_frames_minus1的值应为0或1。
——bit_depth_minus8应为2。
——chroma_format应为1。
——序列内所有图像的frame_type应为0。

扩展帧内含透明度档次档次的码流应满足以下条件：
——profile_idc的值应为0x11。
——num_of_frames_minus1的值应为0或1。
——bit_depth_minus8应为2或4。
——chroma_format应为0～2。
——序列内所有图像的frame_type应为0。

基础含透明度档次的码流应满足以下条件：
——profile_idc的值应为0x12。
——num_of_frames_minus1的值应为0或1。
——bit_depth_minus8应为2。
——chroma_format应为1。
——序列内第一幅图像的frame_type应为0。

扩展含透明度档次的码流应满足以下条件：
——profile_idc的值应为0x13。
——num_of_frames_minus1的值应为0或1。
——bit_depth_minus8应为2或4。
——chroma_format应为0～2。
——序列内第一幅图像的frame_type应为0。

A.3 级别

本文件定义的级别见表A.2。

表 A.2 级别

level_idc的值	级别
10	1
11	1.1
12	1.2
20	2
21	2.1
22	2.2
30	3
31	3.1
32	3.2
40	4
41	4.1
42	4.2
50	5
51	5.1
52	5.2
60	6
61	6.1
62	6.2
70	7
71	7.1
72	7.2
250	25
251	25.1
252	25.2
255	25.5

各级别的参数限制见表A.3。

表 A.3 级别的参数限制

级别	每秒最大编码单元处理数量（注：为各个子带编码单元的总数）	帧级最低压缩倍率（注：YUV/RGB数据压缩率，不含alpha分量）	最大子图宽度	单帧最大子图个数（注：W、H为图像宽和高）	注：级别对应的最大分辨率、帧率
1	1044480	12	1024	ceil(sqrt(W*H)/180)	2048x1088@30
1.1		8	1024	ceil(sqrt(W*H)/180)	
1.2		6	2048	无限制	
2	2088960	12	1024	ceil(sqrt(W*H)/180)	2048x1088@60
2.1		8	1024	ceil(sqrt(W*H)/180)	
2.2		6	2048	无限制	
3	4177920	12	1024	ceil(sqrt(W*H)/180)	4096x2160@30，2048x1088@120
3.1		8	1024	ceil(sqrt(W*H)/180)	
3.2		6	4096	无限制	
4	8355840	12	1024	ceil(sqrt(W*H)/180)	4096x2160@60
4.1		8	1024	ceil(sqrt(W*H)/180)	
4.2		6	4096	无限制	
5	16711680	12	1024	ceil(sqrt(W*H)/180)	8192x4320@30，4096x2160@120

表 A.3 级别的参数限制（续）

级别	每秒最大编码单元处理数量（注：为各个子带编码单元的总数）	帧级最低压缩倍率（注：YUV/RGB数据压缩率，不含alpha 分量）	最大子图宽度	单帧最大子图个数（注：W、H为图像宽和高）	注：级别对应的最大分辨率、帧率
5.1		8	1024	ceil(sqrt(W*H)/180)	
5.2		6	8192	无限制	
6	33423360	12	1024	ceil(sqrt(W*H)/180)	8192x4320@60
6.1		8	1024	ceil(sqrt(W*H)/180)	
6.2		6	8192	无限制	
7	66846720	12	1024	ceil(sqrt(W*H)/180)	8192x4320@120
7.1		8	1024	ceil(sqrt(W*H)/180)	
7.2		6	8192	无限制	
25	无限制	12	1024	无限制	16K或更高
25.1	无限制	8	1024	无限制	
25.2	无限制	6	无限制	无限制	
25.5	无限制	无限制	无限制	无限制	

注：各级别下每帧YUV或RGB数据的最大压缩比特数MaxBits等于W*H*SampleNumber*BitDepth/CR，其中W为图像宽，H为图像高，SampeleNumber为2（当chroma_format为1时）或3（当chroma_format为0或2时），BitDepth等于图像的位宽，CR为表A.3中规定的各级别对应的帧级最低压缩倍率。例如，对于级别4.1，当视频分辨率为宽W等于3840，高H等于2160，chroma_format等于1时，MaxBits=2073600。对序列中的第一帧，它的YUV或RGB数据的最大压缩比特数MaxBits为序列sequence_header( )语法结构的比特数加上该帧picture( )语法结构的比特数减去该帧alpha_map_data( )语法结构的比特数；对序列中的其余帧，它的YUV或RGB数据的最大压缩比特数MaxBits为该帧picture( )语法结构的比特数减去该帧alpha_map_data( )语法结构的比特数。

附 录 B
（资料性）
由序列中两幅 YUV422 图像拼装 YUV444 图像

B.1 概述

标准的扩展帧内档次、扩展档次、扩展帧内含透明度档次、扩展含透明度档次支持YUV444格式的编码。除此之外，考虑到支持基础帧内档次、基础档次、基础帧内含透明度档次、基础含透明度档次的解码器数量可能多于支持更高能力的扩展帧内档次、扩展档次、扩展帧内含透明度档次、扩展含透明度档次的解码器，本文件还推荐另一种支持YUV444视频编码的方式：通过序列中包含两幅YUV422图像并经过解码后处理拼装成一幅YUV444图像的方式。当序列头中yuv444_packed_by_yuv422_flag为1时，表示当前序列的码流中包含2幅YUV422格式的图像，可将它们拼装成为一幅YUV444格式的图像来显示。可使用本附录规定的方式进行解码后处理，将两幅YUV422格式的图像拼装成一幅YUV444格式的图像，再显示YUV444图像，而这两幅解码的YUV422格式图像宜不显示。需要注意的是，当yuv444_packed_by_yuv422_flag为1时，如果序列中第二幅图像的pic_output_flag为0，按照本附录的处理方法，仍需要解码第二幅图像，并与第一幅图像一起按照B.2 规定的方法拼装成一幅YUV444图像。

当序列头中yuv444_packed_by_yuv422_flag为1时，如果序列同时含有Alpha分量数据，则第一幅图像的alpha_map_flag宜为1，第二幅图像的alpha_map_flag宜为0。

B.2 解码后处理拼装 YUV444 图像的处理

对于解码后的每两幅YUV422图像，将第一幅YUV422图像的Y、U、V分量分别记为Y0、U0、V0，第二幅YUV422图像的Y、U、V分量分别记为Y1、U1、V1，则YUV444图像的Y、U、V分量的导出方式如下（如图C.1所示）：

a) 将Y0作为YUV444图像的Y分量；
b) 将U0的各列复制到YUV444图像的U分量的奇数列（即第1列、第3列、第5列等），将U1的各列复制到YUV444图像的U分量的偶数列（即第2列、第4列、第6列等）；
c) 将V0的各列复制到V分量的奇数列（即第1列、第3列、第5列等），将V1的各列复制到V分量的偶数列（即第2列、第4列、第6列等）。

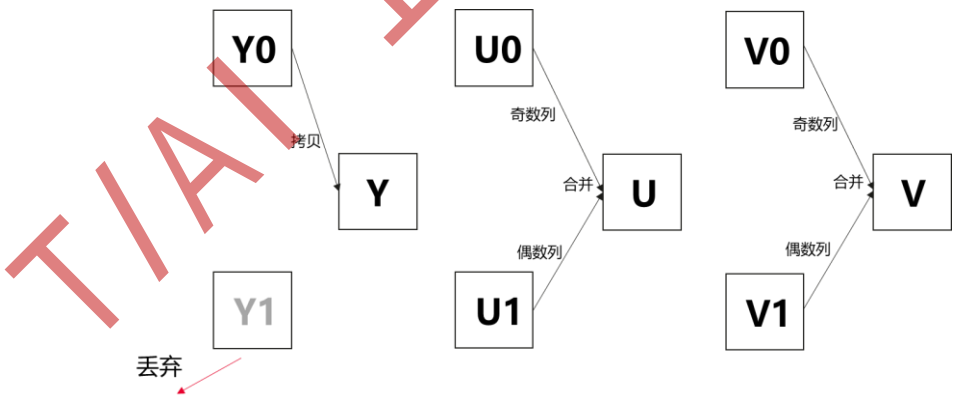

图 C.1 两幅 YUV422 图像拼装一幅 YUV444 图像的处理示意图

B.3 编码器拆分 YUV422 图像及编码处理

编码器将一幅YUV444图像拆分成两幅YUV422图像，再将这两幅YUV422图像依次做为序列的第一幅图像和第二幅图像，配置chroma_format为1、yuv444_packed_by_yuv422_flag为1，进行编码，产生序列的码流。其中，第一幅YUV422的Y分量复制为YUV444图像的Y分量，第二幅YUV422的Y分量可复制为YUV444图像的Y分量（此方式序列的压缩码流较大）或者将其所有数值设置为同一个数值，例如为512（此方式序列的压缩码流较小）。第一幅YUV422图像的U分量复制为YUV444图像的U分量的奇数列，第二幅YUV422

图像的U分量复制为YUV444图像的U分量的偶数列。第一幅YUV422图像的V分量复制为YUV444图像的V分量的奇数列，第二幅YUV422图像的V分量复制为YUV444图像的V分量的偶数列。

编码两幅YUV422图像时，宜将第一幅图像各个宏块的量化参数或与其相近的量化参数应用于第二幅图像对应位置宏块的量化参数，从而使U0和U1对应位置的解码重建像素质量接近，拼装成的YUV444图像具有较好的主观效果。

附 录 C
（规范性）
Alpha 分量的解码方法

C.1 概述

当图像头中的语法元素alpha_map_flag为1时，表示当前图像存在Alpha分量数据。可使用本附录中相应的方法对Alpha分量进行解码。

C.2 Alpha 分量的语法表和语义

Alpha分量数据定义见表B.1。

表 B.1 Alpha 分量数据定义

Alpha 分量数据定义	码流指针 ID	描述符
alpha_map_data (offset5, AlphaWidth, AlphaHeight) {		
if (alpha_map_code_mode == 0)		
alpha_map_data_mode0 (offset5, AlphaWidth, AlphaHeight)		
}		

C.3 Alpha 分量在编码方式 0 时的语法表和语义

当alpha_map_code_mode等于0时，Alpha分量数据定义见表B.2。

表 B.2 alpha_map_code_mode 等于 0 时的 Alpha 分量数据定义

Alpha 分量数据定义	码流指针 ID	描述符
alpha_map_data_mode0 (offset5, curSubpicWidth, curSubpicHeight) {		
set_bitstream_pointer5(offset5)		
elementNum = min(curSubpicWidth * 16, 2^{14})		
groupNum = (curSubpicHeight * curSubpicWidth + elementNum – 1) / elementNum		
iValNum = 0		
for(i = 0; i < groupNum; i++) {		
if (i == groupNum-1)		
curElementNum = curSubpicWidth * curSubpicHeight – i * elementNum		
else		
curElementNum = elementNum		
if (alpha_map_16bit_flag)		
alpha_val[iValNum]	5	u(16)
else		
alpha_val[iValNum]	5	u(8)
alpha_group_flag	5	u(1)
if (!alpha_group_flag) {		
curNumbers = 0		
while(curNumbers < curElementNum){		

表 B.2 alpha_map_code_mode 等于 0 时的 Alpha 分量数据定义（续）

Alpha 分量数据定义	码流指针 ID	描述符
preVal = alpha_val		
repeat_flag[iValNum]	5	u(1)
if (curNumbers != 0) {		
alpha_diff_sign	5	u(1)
abs_alpha_diff_minus1	5	se(v)
diff = (abs_alpha_diff_minus1 + 1) * (alpha_diff_sign ? 1 : -1)		
alpha_val[iValNum] = (alpha_map_16bit_flag) ? (preVal + diff + 2^{16}) % 2^{16} : (preVal + diff + 256) % 256		
}		
iValNum++		
curNumbers += 1		
if (repeat_flag){		
runlength_minus2	5	se(v)
curNumbers += runlength_minus2+1		
}		
}		
}		
else {		
iValNum++		
}		
}		
while (!byte_aligned (5)) {		
zero_bit	5	r(1)
}		
}		

Alpha分量组同一数据标志 alpha_group_flag

1位无符号整数，表示Alpha分量当前块组中的所有数值是否相同，值为'1'表示当前块组中所有数值都相同，值为'0'表示当前块组中存在不同数值。

Alpha分量数据值 alpha_val

当alpha_map_16bit_flag为1时为16位无符号整数，否则为8位无符号整数，表示Alpha分量当前位置的数据值。

Alpha分量数据重复标志 repeat_flag

1位无符号整数，表示Alpha分量当前位置的数据值是否与后续位置数据值相同，值为'1'表示当前位置与之后存在连续相同的数据值，值为'0'表示当前位置的数据值与下一位置的不同。

Alpha分量数据差值符号位 alpha_diff_sign

1位无符号整数，表示Alpha分量当前位置的数据值与前一位置数值的循环差值的符号，值为’1’表示循环差值为正，值为’0’表示循环差值为负。

Alpha分量数据差值绝对值 abs_alpha_diff_minus1

当alpha_map_16bit_flag为1时为15位无符号整数，否则为7位无符号整数，表示Alpha分量当前位置的数据值与前一位置数值的循环差值的绝对值减1。alpha_val依据0阶指数哥伦布码解析。

Alpha分量数据游程长度 runlength_minus2

14bit无符号数，表示当前位置的数据值连续出现的次数长度减2。runlength_minus2依据3阶指数哥伦布码解析。

C.4 Alpha 分量在编码方式 0 时的解码方法

当alpha_map_code_mode=0时，代表当前图像的Alpha分量使用了索引为0的编码方法，应当采用本章节的解码方法解码Alpha分量的码流， 子图的Alpha分量数据按光栅扫描顺序得到一维向量AlphaMap。

Alpha分量的语法表见C.3 。

Alpha分量按照以下步骤，依次解码各个块组得到AlphaMap：

```
iNum = 0
for( j = 0 ; j < groupNum – 1; j++){
    if(alpha_group_flag == 1){
        AlphaMap[k+ j*elementNum] = alpha_val[iNum],其中k = 0,1,2,…, curElementNum -1
        iNum++
    }
    else{
        for (i = 0 , i < curElementNum – 1; i++){
            if(repeat_flag[iNum] == 1){
                AlphaMap[i + j * elementNum + k] = alpha_val[iNum], 其中k = 0, 1, …,
                runlength_minus2+1
                i = i + runlength_minus2 + 2
                iNum++
            }
            else{
                AlphaMap[i + j * elementNum] = alpha_val[iNum]
                i = i + 1
                iNum++
            }
        }
    }
}
```

附 录 D
（资料性）
编码器实现和优化

D.1 概述

本附录提供若干对编码器的实现和优化的建议，供实施符合标准的编码器时参考。

D.2 小波正变换

本条输入待编码的子图P。

本条输出LL子带、LH子带、HL子带、HH子带，具体步骤如下。

a) 子图P的像素值左移两位：

P[i][j] = P[i][j] << PixelPrecision

b) 亮度分量与色度分量按如下步骤执行水平方向小波变换，得到矩阵L和矩阵H

1) 亮度分量逐行按如下步骤执行一维9/7小波正变换，得到矩阵L和矩阵H：

– 子图第i行像素经如D.4 所述的一维9/7小波正变换得到数组S及数组D，数组S为矩阵L的第i行数据，数组D为矩阵H的第i行数据。

2) 色度分量逐行按如下步骤执行一维5/3小波正变换，得到矩阵L和矩阵H：

– 子图第i行像素经如D.3 所述的一维5/3小波正变换得到数组S及数组D，数组S为矩阵L的第i行数据，数组D为矩阵H的第i行数据。

c) 垂直方向执行5/3小波正变换得到LL子带、LH子带、HL子带和HH子带。

1) 矩阵L逐列按如下步骤执行一维5/3小波正变换，得到LL子带和LH子带。

– 矩阵L的第i列数据经如D.3 所述的一维5/3小波正变换得到数组S及数组D，数组S加上LLbandOffset为LL子带的第i列数据，数组D为LH子带的第i列数据：

LL[n][i] = S[n] + LLbandOffset，其中n = 0,…, BandHeight-1

$$LL[n][i] = clip(0, 2^{BitDepth+3} - 1, LL[n][i])$$

2) 矩阵H逐列按如下步骤执行一维5/3小波正变换，得到HL子带和HH子带。

– 矩阵L的第i列数据经如D.3 所述的一维5/3小波正变换得到数组S及数组D，数组S为HL子带的第i列数据，数组D为HH子带的第i列数据。

D.3 5/3 小波正变换

本节介绍9.6.2规定的5/3小波反变换对应的5/3小波正变换处理。采用本条建议的小波正变换，包括边界像素的中心对称padding方法，使得小波正反变换在子图边界处可逆。

本条输入数组X，记输入数组长度为2N；

本条输出为长度为N的数组S和数组D：

for (i = 0 ; i < N – 1; i++) {

$$D[i] = X[2i + 1] - ((X[2i] + X[2i + 2] + 1) \gg 1)$$

}

$$D[N - 1] = X[2N - 1] - ((2 \times X[2N - 2] + 1) \gg 1)$$

$$S[0] = X[0] + ((2D[0] + 2) \gg 2)$$

for (i = 1 ; i < N; i ++) {

$$S[i] = X[2i] + ((D[i] + D[i - 1] + 2) \gg 2)$$

}

for (i = 0 ; i < N; i++) {

$$D[i] = D[i] \gg 1$$

}

D.4 9/7 小波正变换

本节介绍9.6.3规定的9/7小波反变换对应的9/7小波正变换处理。采用本条建议的小波正变换，包括边界像素的中心对称padding方法，使得小波正反变换在子图边界处可逆。

本条输入数组X，记输入数组长度为2N；

本条输出为长度为N的数组S和数组D：

$$S[0] = (23X[0] \gg 5) + (X[1] \gg 1) - (X[2] \gg 2) + (X[4] \gg 5)$$

$$S[1] = (X[2] \gg 6) - (X[0] \gg 3) + (X[1] \gg 2) + (23X[2] \gg 5) + (X[3] \gg 2) - (X[4] \gg 3) + (X[6] \gg 6)$$

for (i = 2 ; i < N – 2; i++) {

$$S[i] = (X[2i-4] \gg 6) - (X[2i-2] \gg 3) + (X[2i-1] \gg 2) + (23X[2i] \gg 5) + (X[2i+1] \gg 2) - (X[2i+2] \gg 3) + (X[2i+4] \gg 6)$$

}

$$S[N-2] = (X[2N-8] \gg 6) - (X[2N-6] \gg 3) + (X[2N-5] \gg 2) + (23X[2N-4] \gg 5) + (X[2N-3] \gg 2) - (X[2N-2] \gg 3) + (X[2N-2] \gg 6)$$

$$S[N-1] = (X[2N-6] \gg 6) - (X[2N-4] \gg 3) + (X[2N-3] \gg 2) + (23X[2N-2] \gg 5) + (X[2N-1] \gg 2) - (X[2N-4] \gg 3) + (X[2N-4] \gg 6)$$

$$D[0] = (X[2] \gg 5) - (9X[0] \gg 5) + (X[1] \gg 1) - (9X[2] \gg 5) + (X[4] \gg 5)$$

for (i = 1 ; i < N – 1; i ++) {

$$D[i] = (X[2i-2] \gg 5) - (9X[2i] \gg 5) + (X[2i+1] \gg 1) - (9X[2i+2] \gg 5) + (X[2i+4] \gg 5)$$

}

$$D[N-1] = (X[2N-4] \gg 5) - (9X[2N-2] \gg 5) + (X[2N-1] \gg 1) - (9X[2N-2] \gg 5) + (X[2N-4] \gg 5)$$

D.5 子图交界区域的量化参数配置

当一幅图像由多个子图构成时，由于子图独立编码，为了避免产生子图边界处重建像素的视觉不连续效应，特别是子图边界附近平坦区域的视觉不连续效应，宜采用以下量化参数配置方法：

——对于两个相邻子图交界处的低频子带宏块，基于宏块方差判定平坦区域，例如当宏块方差小于一定阈值时判定其为平坦区域；

——对判定为平坦区域的低频子带宏块进行更高的编码质量保护，例如可以调整这些平坦区域宏块的宏块级量化参数，使得宏块的量化参数为一个较小的数值（例如相对帧级QP的偏移量为-4或-6），从而控制这些宏块的重建失真在一个较小的范围。