ICS 13.310
CCS L 64

团　　体　　标　　准

T/AI 109.8—2025
T/CSPIA 018—2026

信息技术 智能媒体编码 第8部分：公共安全音视频

Information technology – Intelligent media coding –

Part 8: Public security audio and video

T/AI 109.8-2025

2025－12 －26 发布　　2026－03－01 实施

中关村视听产业技术创新联盟
中国安全防范产品行业协会　　发　布

T/AI 109.8-2025

T/AI 109.8-2025



版权保护文件

版权所有归属于该标准的发布机构，除非有其他规定，否则未经许可，此发行物及其章节不得以其他形式或任何手段进行复制、再版或使用，包括电子版，影印件，或发布在互联网及内部网络等。使用许可可于发布机构获取。

T/AI 109.8-2025

目　　次

前言 IV
引言 V
1 范围 1
2 规范性引用文件 1
3 术语、定义和缩略语 1
3.1 术语和定义 1
3.2 缩略语 18
4 约定 20
4.1 概述 20
4.2 算术运算符 20
4.3 逻辑运算符 21
4.4 关系运算符 21
4.5 位运算符 21
4.6 赋值运算符 22
4.7 数学函数 22
4.8 结构关系符 24
4.9 位流语法、解析过程和解码过程的描述方法 24
5 视频编码位流的结构 28
5.1 位流格式 28
5.2 编码视频序列 28
5.3 图像 29
5.4 片 34
5.5 最大编码单元、编码树和编码单元 34
6 视频编码位流的语法和语义 34
6.1 语法描述 34
6.2 语义描述 104
7 视频解析过程 170
7.1 k 阶指数哥伦布码 170
7.2 ue(v)和 se(v)的解析过程 171
7.3 ae(v)的解析过程 172
8 视频解码过程 211
8.1 NAL 位流解码 211

T/AI 109.8-2025

8.2 序列解码 212
8.3 图像解码 213
8.4 片解码 219
8.5 最大编码单元解码 220
8.6 编码单元解码 220
8.7 变换块解码 295
8.8 帧内预测 303
8.9 帧间预测 344
8.10 预测补偿 393
8.11 去块效应滤波 395
8.12 样值偏移补偿 401
8.13 自适应修正滤波 411
附录 A （规范性） 扫描表 418
附录 B （规范性） 智能分割信息语法及语义补充 421
B.1 语法描述 421
B.2 语义描述 422
附录 C （资料性） 子位流抽取方法 424
C.1 概述 424
C.2 构建目标空域层列表 424
C.3 子位流抽取方法 424
附录 D （资料性） 视频数据加解密方法示例 426
D.1 位流加密 426
D.2 位流解密 427
附录 E （资料性） 视频数据认证方法示例 429
E.1 显示图像签名 429
E.2 显示图像认证 431
E.3 知识图像签名 432
E.4 知识图像认证 433
附录 F （规范性） 视频档次和级别 436
F.1 概述 436
F.2 档次 436
F.3 级别 440
附录 G （资料性） 隐私保护单层编码模式低权限解码填充方法 452
G.1 概述 452

T/AI 109.8-2025

G.2 方法 452
附录 H （规范性） 位流参考缓冲区管理 454
H.1 概述 454
H.2 约定 454
H.3 基本操作 454
H.4 缓冲区检测时间间隔 458
附录 I （规范性） 加权量化矩阵 459
附录 J （资料性） 解密正确性验证功能和解码正确性验证功能参考实现方法示例 460
J.1 概述 460
J.2 语法和语义 460
J.3 位流验证值计算过程 461
J.4 重建图验证值计算过程 461
附录 K （资料性） 特征张量图前处理、后处理和编码模式示例 463
K.1 概述 463
K.2 非对称特征编码技术 463
附录 L （规范性） 系数编码使用的数组 469
L.1 prefix_ctx 469
L.2 min_in_group 469
L.3 group_index 469
附录 M （资料性） 高级熵编码器解码器参考描述方法 470
附录 N （规范性） 反变换矩阵 472
N.1 DCT2 型反变换矩阵 472
N.2 DCT8 型反变换矩阵 481
N.3 DST7 型反变换矩阵 482
附录 O （资料性） 智能分割图表示编码模式示例 486
O.1 概述 486
O.2 方法 486

T/AI 109.8-2025

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 109《信息技术 智能媒体编码》的第8部分。T/AI 109已经发布了以下部分：

——第2部分：视频；

——第3部分：沉浸式音频；

——第4部分：符合性测试；

——第6部分：智能媒体格式；

——第7部分：音频封装与传输；

——第8部分：公共安全音视频；

——第10部分：实时通信语音编码。

本文件由数字音视频编解码技术标准工作组联合中国安全防范产品行业协会提出。

本文件由中关村视听产业技术创新联盟及中国安全防范产品行业协会归口。

本文件起草单位：北京大学、公安部第一研究所、鹏城实验室、海思技术有限公司、浙江大学、公安部安全与警用电子产品质量检测中心、杭州海康威视数字技术股份有限公司、北京博雅睿视科技有限公司、清华大学、浙江大华技术股份有限公司、天翼视联科技股份有限公司、北京大学深圳研究生院、华为技术有限公司、杭州恒生数字设备科技有限公司、深流微智能科技（深圳）有限公司、苏州科达科技股份有限公司、西安电子科技大学、中国电信股份有限公司、武汉中科通达高新技术股份有限公司、天地伟业技术有限公司、上海富瀚微电子股份有限公司、浙江宇视科技有限公司、北京联视神盾安防技术有限公司、三星电子株式会社、广东博华超高清创新中心有限公司、中山大学、北京字节跳动科技有限公司、上海大学、上海国茂数字技术有限公司、哈尔滨工业大学、中国科学院计算技术研究所、武汉大学、中国科学技术大学、电子科技大学、腾讯科技（深圳）有限公司、同济大学、绍兴文理学院、绍兴市北大信息技术科创中心、广东魅视科技股份有限公司、锚丁科技（北京）有限责任公司、富盛科技股份有限公司。

本文件主要起草人：高文、高磊、黄铁军、马思伟、王楠、杨海涛、虞露、王荣刚、王磊、孙煜程、王苫社、林聚财、赵寅、查敏中、张伟民、梁凡、陈焕浜、罗传飞、潘存斌、林冬、余子龙、王爱军、梁志坚、潘冬萍、江东、魏亮、曹小强、陈方栋、张坚、江靖华、邱天、杨冬梅、杨付正、范彬彬、沈佳峰、张园、李云龙、刘仕怡、袁文鸿、骆华敏、梁小明、刘志扬、韩巍、赵帅、夏宇、吴梦、陈晓春、胡开云、赵海英、崔晓冉、李晶、崔云红、董骞、郑建铧、何芸、赵海武、王稷、李忠良、赵日洋、范晓鹏、林和源、李革、张嘉琪、李俊儒、傅天亮、张玉槐、张莉、邓志玭、朴银姬、王振宇、范奎、马祥、郑建成、欧阳晓、王凡、王力强、牛犇犇、高小丁、谢熙、许桂森、徐巍炜、贾川民、吴文、吴参毅、雷萌、简云瑞、鲁晓牧、何鸣、马海川、李一鸣、王梦杨、刘杉、许晓中、鲁大军、王洵、王英彬、胡晔、王莉、修晓宇、陈伟、郭哲伟、方诚、张雪、方瑞东、林涛、周开伦、王淑慧、赵利平、于化龙、林翔宇、王迪、龙仕强、韦裕京、彭志丹、钟永强。

T/AI 109.8-2025

引 言

T/AI 109 《信息技术 智能媒体编码》旨在确立智能媒体压缩的方法，拟由10个部分构成。

——第 1 部分：系统。目的在于规定沉浸媒体系统中的虚拟现实视频和音频数据的封装格式、传输格式和信令格式。

——第 2 部分：视频。目的在于规定了适应多种位率、分辨率和质量要求的高效视频压缩方法的解码过程。

——第 3 部分：音频。目的在于描述面向全场景的下一代沉浸式音频高效压缩方法的解码过程。

——第 4 部分：视频符合性测试。目的在于定义测试验证编码器生成的位流和解码器是否符合《信息技术 智能媒体编码 第 2 部分：视频》和《信息技术 智能媒体编码 第 3 部分：音频》所规定要求的方法。

——第 5 部分：参考软件。目的在于定义满足《信息技术 智能媒体编码 第 2 部分：视频》和《信息技术 智能媒体编码 第 3 部分：音频》规定要求的参考软件。

——第 6 部分：智能媒体格式。目的在于规定智能媒体编码数据的 AVS3 视频文件格式、AVS3 视频 CMAF 轨道和媒体配置、DASH 传输信令、SMT 传输信令、传输流和节目流技术要求、RTP 传输技术要求。

——第 7 部分；音频封装与传输。目的在于确立异构网络中的沉浸式音频数据提供封装和传输。

——第 8 部分：公共安全音视频。目的在于规定安防音视频智能编解码的视频编码位流的结构、语法和语义、视频解析过程和视频解码过程的技术要求。

——第 10 部分；实时通信语音编码。目的在于确立面向实时通信的语音编码位流和解码过程。

本文件是针对安全防范视频图像智能化应用场景的音视频编解码标准。近年来，安防视频系统建设应用已经扩展到公共安全各行业、领域，成为新形势下维护社会安全稳定的重要手段，在打击犯罪、治安防范、社会管理、服务民生等方面发挥着积极作用，成为构建和谐社会的重要基础设施。

本文件主要技术特点包括：

——支持高精度视频数据编码，适应宽动态范围，保留更多的图像细节，满足忠实于场景的要求。视频支持 8 位～10 位数据；

——引入非显示知识图像切片模式，对背景帧做极高质量编码，将背景帧压缩位流切分得到多个位流切片，并将这些位流切片与指定时间窗内的多个显示图像压缩位流交织传输；

——支持显示知识图像模式，基于知识图像机制实现传统的长期参考帧功能，兼容部分现存的产品方案；

——支持可伸缩性视频编码（SVC），将时域可伸缩编码（TSVC）支持的层级扩展至 5 层，空域可伸缩编码（SSVC）支持的层级扩展至 4 层，支持各层独立编码模式及层间预测编码模式，满足不同传输网络带宽和数据存储环境的需求；

——引入四种隐私保护模式，满足用户对于隐私保护的需求。隐私保护模式 0 和模式 1 基于空域分层编码结构，通过对不同分层的不同权限等级设置来达到隐私保护的目的，兼容现有解决方案。隐私保护模式 2 和模式 3 将隐私区域的信息单独编入隐私位流，提升编解码器性能；

——支持语义分割图编码，在增加少量码率的代价下，将前端的语义分割信息随位流传输，支持后端在无额外 AI 算力消耗下的视频浓缩、视频标签等高级上层应用；

——支持数据安全保护，加强了对国密算法的支持，完善了安全参数集，增添了摘要、签名算法的标识等内容，并对密钥及数字证书相关信息的携带做了规范，支持视频数据加密、认证功能并

T/AI 109.8-2025

做多方面性能优化。

本文件的发布机构提请注意，声明符合本文件时，可能涉及到 5.1.3.4、5.2.2.3、5.2.2.9、5.3.5、5.4.2、5.4.3.2、5.4.3.6、5.5、6.1.2、6.1.3、6.1.6、6.1.10、6.1.11、6.1.12、6.1.13、6.1.14、6.1.15、6.1.16、6.1.17、6.1.18、6.1.20、6.1.7.4、6.1.7.5、6.1.7.7、6.1.7.10、6.1.7.11、6.1.8、6.1.9、6.2.2、6.2.3、6.2.6、6.2.7、6.2.8、6.2.9、6.2.11、6.2.12、6.2.13、6.2.14、6.2.17、6.2.20、7.3.1、7.3.3、7.3.4.1、7.3.4.2、7.3.4.20、8.2、8.3.3、8.3.4、8.3.5、8.3.6、8.3.7、8.5、8.6.1、8.6.2、8.6.3、8.6.4、8.6.5、8.6.6.3.3、8.6.6.4.2.1、8.6.6.4.3、8.6.6.5、8.6.6.7、8.6.7、8.7.2、8.7.3、8.8.1、8.8.3.2、8.8.3.3、8.9.2、8.9.3.1、8.9.3.2、8.9.3.4、8.9.4.3、8.10.3、8.11.2、8.11.4、8.11.5、8.11.8、8.11.9、8.11.11、8.12.1.4.2、8.12.2.3、8.12.3.3.2、8.12.3.3.3、8.13.1、8.13.2、8.13.5、附录 B、附录 F、附录 H 和附录 N 中如下 224 项与数字视频编解码技术相关的专利的使用。专利名称如下：

202311734090.2，视频编解码方法及其相关装置；202010446922.0，按图像内容特征进行帧级划分的快速编解码方法；202110628553.1，一种串长度参数编解码方法和装置；202011324094.X，一种串长度参数编码、解码方法和装置；202011490545.7，融合帧内块复制和帧内串复制的编码参数的编码、解码方法和装置；2020104939391，对分量下采样格式数据进行串预测的数据编码和解码方法；2021103764089，封存和激活越界等值串常现位置的数据编解码方法及装置；2022105216755，串矢量与像素值共用一个缓冲区的串预测解码方法及装置；ZL03157076.3，一种定位编码图像标识的方法；ZL201210546675.7，一种头信息编解码、解码方法及装置；ZL201110347691.9，一种视频图像滤波处理方法和装置；ZL202010509307.X，图像滤波方法、装置、设备及可读存储介质；ZL202010509322.4，图像滤波方法、装置、设备及存储介质；ZL201310008682.6，一种运动矢量预测的方法；ZL201910796327.7，参数获取方法、像素点对选择方法及相关设备；ZL202110855539.5，像素点对选择方法、设备及计算机可读存储介质；ZL201210546537.9，一种多方向的帧内预测编解码方法及装置；ZL201310128415.2，一种视频编解码环路滤波的实现方法；ZL202110220816.5，一种图像处理方法、装置、设备及可读存储介质；CN201910786742.4，一种编解码方法、装置及其设备；CN202111150945.8，一种编解码方法、装置及其设备；CN202111153196.4，一种编解码方法、装置及其设备；CN202111168217.X，一种编解码方法、装置及其设备；CN201910798693.6，解码方法、编码方法及装置；CN201911089646.0，解码方法、编码方法及装置；CN202211329855.X，解码方法、编码方法及装置；CN201910844633.3，一种编解码方法、装置及其设备；CN202111153141.3，一种编解码方法、装置及其设备；CN202111153142.8，一种编解码方法、装置及其设备；CN201910901820.0，一种编解码方法、装置及其设备；CN202111155057.5，一种编解码方法、装置及其设备；CN202111155083.8，一种编解码方法、装置及其设备；CN201910919775.1，一种解码、编码方法、装置及其设备；CN202111153097.6，一种解码、编码方法、装置及其设备；CN201911222274.4，解码方法、编码方法、装置及设备；CN202111143951.0，解码方法、编码方法、装置、设备及机器可读存储介质；CN201911223606.0，一种编码、解码方法、装置及其设备；CN202111150952.8，一种编码、解码方法、装置及其设备；CN201911343050.9，一种编解码方法、装置及其设备；CN202111152763.4，一种编解码方法、装置及其设备；CN202111155058.X，一种编解码方法、装置及其设备；CN202010220367.X，一种解码、编码、编解码方法、装置及其设备；CN202111150944.3，一种解码、编码、编解码方法、装置及其设备；CN202010220130.1，一种编解码方法、装置及其设备；CN202111152789.9，一种编解码方法、装置及其设备；CN202111143948.9，解码方法、编码方法、装置、设备及机器可读存储介质；CN202010486400.3，一种编解码方法、装置及其设备；CN202111155050.3，一种编解码方法、装置及其设备；CN202010508167.4，图像增强方法及装置；CN202111667155.7，图像增强方法、装置及机器可读存储介质；CN202010508168.9，编码方法、解码方法及装置；CN202111143932.8，编码方法、解码方法、装置及机器可读存储介质；CN202111146301.1，编码方法、解码方法、装置及机器可读存储介质；CN202010622752.7，一种编解码方法、装置及其设备；

T/AI 109.8-2025

CN202111155088.0，一种编解码方法、装置及其设备；CN202010635665.5，滤波处理方法及装置；CN202111673286.6，滤波处理方法、装置及机器可读存储介质；CN202010637298.2，一种运动信息候选列表的构建方法、装置及其设备；CN202111150918.0，一种运动信息候选列表的构建方法、装置及其设备；CN202111153145.1，一种运动信息候选列表的构建方法、装置及其设备；CN202010845243.0，一种编解码方法、装置及其设备；CN202111155082.3，一种编解码方法、装置及其设备；CN202010859387.1，滤波方法、装置及设备；CN202111146303.0，滤波方法、装置、设备及机器可读存储介质；CN202010970776.1，增强滤波方法及装置；CN202111673093.0，增强滤波方法、装置及机器可读存储介质；CN202111673156.2，预测模式解码方法、电子设备及机器可读存储介质；CN202011455990.X，一种编码、解码方法、装置及其设备；CN202111153132.4，一种编码、解码方法、装置及其设备；CN202111153185.6，一种编码、解码方法、装置及其设备；CN202011615201.4，一种编解码方法、装置及其设备；CN202111155067.9，一种编解码方法、装置及其设备；CN202110204154.2，编解码方法、装置及其设备；CN202111625855.X，解码方法、装置、设备及机器可读存储介质；CN202110204286.5，基于自适应帧内刷新机制的解码、编码方法及相关设备；CN202111123861.5，基于自适应帧内刷新机制的解码、编码方法及相关设备；CN202110247471.2，滤波方法、装置及设备；CN202111146284.1，滤波方法、装置、设备及机器可读存储介质；CN202110297136.3，解码方法、编码方法、装置及设备；CN202111143912.0，解码方法、编码方法、装置、设备及机器可读存储介质；CN202111143914.X，解码方法、编码方法、装置、设备及机器可读存储介质；CN202111146263.X，解码方法、编码方法、装置、设备及机器可读存储介质；CN202111146280.3，解码方法、编码方法、装置、设备及机器可读存储介质；CN202110298610.4，基于自适应帧内刷新机制的解码、编码方法及相关设备；CN202111123857.9，基于自适应帧内刷新机制的解码、编码方法及相关设备；CN202110397236.3，一种编解码方法、装置及其设备；CN202111152790.1，一种编解码方法、装置及其设备；CN202111155087.6，一种编解码方法、装置及其设备；CN202110739453.6，编解码方法、装置及其设备；CN202111628844.7，解码方法、装置、设备及机器可读存储介质；CN202110739455.5，解码方法、编码方法、装置及设备；CN202111146250.2，解码方法、编码方法、装置、设备及机器可读存储介质；CN202111146278.6，解码方法、装置、设备及机器可读存储介质；CN201910944559.2，一种编解码方法、装置及其设备；CN201910430956.8，进行解码、编码的方法、解码端和编码端；CN202010508179.7，一种解码、编码方法、装置及其设备；CN201810380105.2，一种图像解码、编码方法、装置及其设备；CN202210623408.9，一种图像解码、编码方法、装置及其设备；CN201811459061.9，一种运动信息确定方法及其设备；CN202210922647.4，一种运动信息确定方法、装置及其设备；CN202210923870.0，一种运动信息确定方法、装置及其设备；CN201910165085.1，一种编解码方法、装置及其设备；CN202211610564.8，一种编解码方法、装置及其设备；CN201910364638.6，一种矢量差解码方法、装置及电子设备；CN202011410129.1，一种编解码方法、装置及其设备；CN202211098101.8，一种编解码方法、装置及其设备；CN202210633928.8，一种编解码方法、装置及其设备；CN202310489496.2，一种编解码方法、装置及其设备；CN202210233926.X，编码方法、解码方法、装置及机器可读存储介质；CN202211146011.1，滤波方法、装置及设备；CN202211419754.1，一种图像解码、图像编码方法、装置及其设备；CN202211426315.3，滤波方法、设备及存储介质；CN202211421087.0，编解码方法、装置、设备及存储介质；CN202211422461.9，一种图像编解码方法及装置；CN202410089093.3，图像解码和编码方法、装置、设备及存储介质；CN202310872974.8，一种图像解码、图像编码方法、装置及其设备；CN202310561476.1，一种解码、编码方法、装置及其设备；CN202211423122.2，一种图像处理方法、装置、电子设备及存储介质；CN202510895602.6，滤波方法、设备及存储介质；CN202510887046.8，滤波方法、设备及存储介质；CN202510887381.8，滤波方法、设备及存储介质；CN202510887704.3，一种图像解码、图像编码方法、装置及其设备；CN202510897010.8，一种图像处理方法、装置、电子设备及存储介质；CN202510884205.9，

T/AI 109.8-2025

编解码方法、装置、设备及存储介质；CN202510886030.5，编解码方法、装置、设备及存储介质；CN202510887036.4，编解码方法、装置、设备及存储介质；CN202510884188.9，一种图像解码方法、装置、解码器和计算机可读存储介质；CN202510886943.7，一种图像解码方法、装置、解码器和计算机可读存储介质；CN202510887685.4，一种解码、编码方法、装置及其设备；CN202510896683.1，一种图像解码、图像编码方法、装置及其设备；CN202510890327.9，图像解码和编码方法、装置、设备及存储介质；CN202510895610.0，图像解码和编码方法、装置、设备及存储介质；200780000403.4，在编解码中的实现量化的方法和装置；201910679070.7，图像显示顺序的确定方法、装置和视频编解码设备；201910335981.8，视频编码器、视频解码器及相应方法；201910108004.4，一种帧间预测的方法和装置；201910888383.3，视频解码方法、视频编码方法、装置、设备及存储介质；201810581662.0，一种编解码方法及装置；201810278488.2，图像块编码中的变换方法、解码中的反变换方法及装置；202211333863.1，视频译码器及相应方法；201910246994.8，块划分方法、视频编解码方法、视频编解码器；201911286497.7，图像重建方法和装置；201910222973.2，视频解码方法、装置及解码设备；201910222962.4，预测模式确定方法、装置及编码设备和解码设备；201910457372.X，预测模式确定方法和装置；201580077673.X，图像预测方法和相关设备；201510391765.7，图像预测方法和相关设备；201510543542.8，图像预测的方法及设备；201510632589.1，预测图像编解码方法及装置；201711319298.2，视频数据的帧间预测方法和装置；201810990444.2，帧间预测方法、装置以及视频编码器和视频解码器；201880095581.8，视频编码器、视频解码器及相应方法；201811109475.9，划分标志位的上下文建模方法及装置；201910191131.5，图像划分方法及装置；201980062412.9，图像预测方法、装置以及相应的编码器和解码器；201911424851.8，视频编码器、视频解码器及相应方法；201611228200.8，图像编解码方法及装置；201510150090.7，图像编解码方法和相关装置；201010155175.1，频带加权量化编解码方法和装置；200810110177.1，视频数据的打包方法及装置；200810056977.X，一种码流标识方法、装置及编解码系统；201410638187.8，一种视频数据编码、解码的方法及装置；200680014013.8，一种实现算术编解码的方法及装置；202310491823.8，编解码方法及装置；202310731240.8，编解码方法及电子设备；202311138668.8，用于 TSVC 技术的认证方法、加密方法及相关装置；202311135747.3，分层编解码方法及装置；202311441353.0，视频数据编码方法、解码方法、可读介质和电子设备；202310703615.X，编解码方法及装置；202311138721.4，编解码方法及装置；202311133490.8，编解码方法及装置；202311131845.X，编解码方法及装置；202311468726.3，编解码方法和装置；202311692069.0，量化参数的编解码方法及电子设备；202311614706.2，码流的签名及认证方法；202311614838.5，码流的签名及认证方法；202311665406.7，码流的签名及认证方法；202311837246.X，一种编解码方法及装置；202410104708.5，加密方法、解密方法及装置；202410104632.6，一种位流、位流签名及检测方法；202410084689.4，一种解码方法及装置；202410178188.2，一种编解码方法及装置；202410178268.8，一种图像处理方法及装置；202410211671.6，一种位流处理方法及装置；202410205619.X，一种解码方法及装置；202410178335.6，一种编解码方法及装置；202410176180.2，一种签名方法、认证方法及装置；202410166342.4，加密方法、解密方法及装置；202410176090.3，一种位流、位流签名及认证方法；202410176997.X，一种位流、位流签名及认证方法；PCT/CN2024/096907，编解码方法及装置；202310841441.3，一种视频编码方法、解码方法和相关装置；ZL201680091245.7，用于对画面轮廓线的编码单元进行编码或解码的方法和装置；ZL201680091331.8，通过块映射来对图像进行编码或解码的方法和装置；ZL201580072015.1，使用高精度跳过编码的视频编码设备和视频解码设备及其方法；ZL201910011266.9，一种视频编解码中运动信息的编解码方法及装置；ZL201080036192.1，用于对视频进行编码的方法和设备以及用于对视频进行解码的方法和设备；ZL201080049482.X，对图像边界的编码单元进行编码和解码的方法和设备；ZL201180027840.1，用于基于变换和反变换执行内插的方法和设备；ZL201680059343.2，用于对图像进行编码或解码的方法和设备；ZL202010152701.2，模型更新方法、

T/AI 109.8-2025

装置、电子设备及计算机可读存储介质；ZL201880017180.0，视频解码方法和设备以及视频编码方法和设备；ZL201880089303.1，视频解码方法及其装置以及视频编码方法及其装置；ZL201880089530.4，视频解码方法及其装置和视频编码方法及其装置；ZL202010188807.8，一种帧内预测方法和装置；ZL201880084999.9，图像编码方法及其装置、图像解码方法及其装置；ZL201911078110.9，一种帧内预测的方法和装置；ZL201980074840.3，视频解码方法和设备以及视频编码方法和设备；ZL202010668733.8，用于对运动矢量进行编码/解码的方法和装置；ZL201880071146.1，用于对运动信息进行编码的设备和方法以及解码设备和方法；ZL201310746046.3，表示视频图像显示顺序的方法；ZL202411409642.7，一种视频传输方法和装置；ZL201811487546.9，指定参考图像的方法及装置及处理参考图像请求的方法及装置；ZL201811488779.0，处理和传输媒体数据的方法和装置；ZL201910002807.1，视频编解码方法与装置；ZL201910375478.5，位流检验方法、解码方法及其装置；ZL201911229769.X，一种检验被测解码器的方法和装置；ZL201910888383.3，视频解码方法、视频编码方法、装置、设备及存储介质；ZL201911424851.8，视频编码器、视频解码器及相应方法；ZL201910017708.0，视频图像解码、编码方法及装置；ZL201910503066.5，视频解码方法、视频编码方法、装置、设备及存储介质；ZL201510150090.7，图像编解码方法和相关装置；ZL201510271487.1，一种视频处理方法及装置；ZL201611228200.8，图像编解码方法及装置。

本文件的发布机构对于该专利的真实性、有效性和范围无任何立场。

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

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
联 系 人：李晶（中国安全防范产品行业协会）
通讯地址：北京市海淀区西三环北路 87 号 C 座 1401
邮政编码：100089
电子邮件：anfangxiehui@sina.com
电　　话：+8610-68730175
传　　真：+8610-68730788
网　　址：http://www.21csp.com.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。

T/AI 109.8-2025

T/AI 109.8-2025

信息技术 智能媒体编码 第8部分：公共安全音视频

1 范围

本文件规定了安防音视频智能编解码的视频编码位流的结构、语法和语义、视频解析过程和视频解码过程的技术要求。

本文件适用于安全防范视频图像智能化应用场景的音视频实时编解码、传输、播放和存储等业务，其他需要音视频编解码的领域可参考采用。

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 17975.3-2002 信息技术 运动图像及伴音信号的通用编码 第3部分：音频

GB/T 25724-2017 公共安全视频监控数字视音频编解码技术要求

T/BAX 0007-2025 SVAC扩展信息应用技术要求

T/AI 109.2-2021 信息技术 智能媒体编码 第2部分：视频

3 术语、定义和缩略语

3.1 术语和定义

下列术语和定义适用于本文件。

3.1.1

安全前端设备 secure front end device

具备安全密码部件（如安全电路和安全TF卡等）的安防视频系统前端设备。

注：该设备能够利用证书存储管理、密钥存储和管理、数据签名验签和数据加密等技术实现设备身份认证、视频签名和视频加密等功能。

3.1.2

B图像 B picture

帧间预测中最多使用两个运动矢量和参考索引来预测每个块的样本值的图像。

3.1.3

变换系数 transform coefficient

变换域上的一个标量。

[来源：T/AI 109.2-2021,3.2]

T/AI 109.8-2025

3.1.4

编码单元 coding unit

由最大编码单元划分得到，可包含一个亮度编码块和对应的色度编码块，也可只包含一个亮度编码块，或只包含色度编码块，由编码单元的分量模式决定。

[来源：T/AI 109.2-2021,3.3]

3.1.5

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。

[来源：T/AI 109.2-2021,3.4]

3.1.6

编码图像 coded picture

一幅图像的编码表示。

[来源：T/AI 109.2-2021,3.5]

3.1.7

编码视频序列 coded video sequence

编码位流的最高层次语法结构，包括一个或多个连续的访问单元。

3.1.8

编码视频序列起始图像 first picture of coded video sequence

编码视频序列的第一个图像。

注：该图像可以是IDR图像、RAPI图像、RL前置知识图像或显示知识图像。

3.1.9

编码片 coded patch

一个片的编码表示。

注：同一个编码片单元中的NAL单元应该相邻。

3.1.10

编码片 NAL 单元 NAL unit of coded patch

nal_unit_type 值等于 1，2，4，12，14，17，18 或 19 的 NAL 单元。

注：所有其他的NAL单元都称为非编码片NAL单元。

3.1.11

补偿后样本 compensated sample

经预测补偿得到的样本。

[来源：T/AI 109.2-2021,3.6]

3.1.12

T/AI 109.8-2025

不完全匹配串 incompletely matched sample string

至少包含一个未匹配样本的串。

[来源：T/AI 109.2-2021,3.7]

3.1.13

残差 residual

样本或数据元素的重建值与其预测值之差。

[来源：T/AI 109.2-2021,3.8]

3.1.14

参考索引 reference index

参考图像在参考图像队列中的编号。

注：在参考图像队列0中的编号称为L0参考索引，在参考图像队列1中的编号称为L1参考索引。

[来源：T/AI 109.2-2021,3.9]

3.1.15

参考图像 reference picture

解码过程中用于后续图像帧间预测的图像。

[来源：T/AI 109.2-2021,3.10]

3.1.16

参考图像队列 reference picture list

当前图像的参考图像所组成的图像队列。

注：可包括参考图像队列0和参考图像队列1。

[来源：T/AI 109.2-2021,3.11]

3.1.17

参考图像队列0 reference picture list 0

当前图像的参考图像所组成的图像队列，用于P图像的帧间预测或作为B图像帧间预测的第一个参考图像队列。

3.1.18

参考图像队列1 reference picture list 1

当前图像的参考图像所组成的图像队列，作为B图像帧间预测的第二个参考图像队列。

3.1.19

参考图像缓冲区 reference picture buffer

保存解码图像并用于预测的缓冲区。

[来源：T/AI 109.2-2021,3.14]

3.1.20

层单元 layer unit

T/AI 109.8-2025

根据指定规则相互关联的具有相同 layer_id 取值的 NAL 单元集合，并按照解码顺序连续。

3.1.21

常现位置 frequently occurring position

图像内的一个样本的位置。

注：这个位置的样本的值在同一图像内经常重复出现，可当作其他样本的预测样本。

[来源：T/AI 109.2-2021,3.17]

3.1.22

重建样本 reconstructed sample

由解码器根据位流解码得到并构成解码图像的样本。

[来源：T/AI 109.2-2021,3.18]

3.1.23

串复制帧内预测 intra string copy prediction

在相同解码图像中使用已解码的样本复制至当前样本区域，作为当前样本预测值的模式。

注：复制样本的区域的二维形状和样本数目与当前样本区域的二维形状和样本数量相同。

[来源：T/AI 109.2-2021,3.19]

3.1.24

串矢量 string vector

用于串复制帧内预测模式的二维矢量。

注：其值为当前串和参考串之间的坐标偏移量，其中，当前串与参考串均在当前图像中。

[来源：T/AI 109.2-2021,3.20]

3.1.25

单位基矢量串 unit basis vector string

串矢量等于(0, -1)的串。

[来源：T/AI 109.2-2021,3.21]

3.1.26

档次 profile

本文件规定的语法、语义及算法的子集。

[来源：T/AI 109.2-2021,3.49]

3.1.27

等值串 equal value string

所有样本的值都相等的串。

[来源：T/AI 109.2-2021,3.22]

3.1.28

点矢量 point vector

T/AI 109.8-2025

表示常现位置的二维坐标的矢量。

[来源：T/AI 109.2-2021,3.23]

3.1.29

点预测 point prediction

在串复制帧内预测中，一个串的所有参考样本是同一个常现位置上的样本，该串上所有像素的预测样本都是该常现位置的样本的值。

[来源：T/AI 109.2-2021,3.24]

3.1.30

点预测信息表 point prediction information list

用于存放一个使用非普通串子模式的编码单元中所使用的点矢量。

[来源：T/AI 109.2-2021,3.25]

3.1.31

二元符号 bin

组成二元符号串的符号，包括‘0’和‘1’。

[来源：T/AI 109.2-2021,3.26]

3.1.32

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。

[来源：T/AI 109.2-2021,3.27]

3.1.33

反变换 inverse transform

将变换系数矩阵转换成空域样值矩阵的过程。

[来源：T/AI 109.2-2021,3.28]

3.1.34

反量化 dequantization

对量化系数缩放后得到变换系数的过程。

[来源：T/AI 109.2-2021,3.29]

3.1.35

仿射控制点运动矢量组 affine control point motion vectors

用于计算仿射预测子块运动矢量的参考运动矢量，包含两个或三个运动矢量。

[来源：T/AI 109.2-2021,3.30]

3.1.36

仿射样本改善信息 affine sample refinement information

用于改善仿射预测单元预测样本的信息，包括仿射样本改善标志、仿射样本水平改善标志、仿射

T/AI 109.8-2025

样本垂直改善标志、第一偏差矩阵、第二偏差矩阵、第三偏差矩阵和第四偏差矩阵。

[来源：T/AI 109.2-2021,3.31]

3.1.37

仿射预测单元 affine prediction unit

由多个子块组成，每个子块由一个 4×4 或 8×8 的亮度预测块和对应的色度预测块组成。

注：各子块的运动矢量可不同。

[来源：T/AI 109.2-2021,3.32]

3.1.38

仿射运动信息 affine motion information

用于仿射预测单元的帧间预测的六元组，由仿射类型（四参数或六参数）、参考预测模式、L0 运动矢量集合、L1 运动矢量集合、L0 参考索引和 L1 参考索引构成。

[来源：T/AI 109.2-2021,3.33]

3.1.39

访问单元 access unit

根据指定规则相互关联的一组 NAL 单元，并按照解码顺序连续。

注1：IDR 图像、RL 图像、RAPI 图像、和显示知识图像的访问单元包括访问单元分界 NAL 单元（如果存在）、安全参数集 NAL 单元（如果存在）、序列参数集 NAL 单元、图像参数集 NAL 单元、图像头 NAL 单元、扩展数据 NAL 单元（如果存在）、补充增强信息 NAL 单元（如果存在）、认证数据 NAL 单元（如果存在）、编码片 NAL 单元、编码视频序列结尾 NAL 单元（如果存在）和流结尾 NAL 单元（如果存在）；NRAP 图像的访问单元包括访问单元分界 NAL 单元（如果存在）、图像参数集 NAL 单元（如果存在）、认证数据 NAL 单元（如果存在）、图像头 NAL 单元、扩展数据 NAL 单元（如果存在）、补充增强信息 NAL 单元（如果存在）、编码片 NAL 单元、编码视频序列结尾 NAL 单元（如果存在）和流结尾 NAL 单元（如果存在）。

注2：非显示知识图像可包含在一至多个访问单元中，每个访问单元包括一个至多个连续的知识图像编码片 NAL 单元。如果一个非显示知识图像的编码片 NAL 包含在多个访问单元，第一个访问单元应包括知识图像的安全参数集 NAL 单元（如果存在）、序列参数集 NAL 单元、图像参数集 NAL 单元、图像头 NAL 单元和片索引为 0 的知识图像编码片 NAL 单元，其余访问单元应包括至少一个片索引不为零的知识图像编码片 NAL 单元。包含非显示知识图像的编码片 NAL 单元的访问单元应包含访问单元分界 NAL 单元。

注3：如果一个访问单元内包含一个随机访问点图像编码片 NAL 单元，该访问单元称为随机访问点访问单元。

注4：一个随机访问点访问单元内的非随机访问点图像仅允许参考该访问单元内的随机访问点图像。

注5：一个层单元内的所有编码片 NAL 单元，除隐私图像编码片 NAL 单元外，应具有相同的 NAL 单元类型。一个层单元中允许同时包含非显示知识图像编码片和非显示知识图像分界编码片。

3.1.40

非 RL 前置知识图像 non leading library picture of an RL picture

位流顺序在一个关联 RL 图像之前的非显示知识图像，该非显示知识图像的第一个知识图像编码片 NAL 单元所在的访问单元与关联 RL 图像的访问单元之间存在至少一个其它图像的访问单元。

注：非RL前置知识图像不是随机访问点图像。

3.1.41

T/AI 109.8-2025

非普通串子模式 non-ordinary string submode

使用等值串、单位基矢量串和未匹配样本串进行预测的模式。

[来源：T/AI 109.2-2021,3.34]

3.1.42

非显示知识图像 non output library picture

每帧图像对应一个序列参数集，且对应的序列集参数中知识位流标志为1、知识图像模式索引为0或2的编码图像。

注：非显示知识图像的编码片NAL单元类型为12或18。

3.1.43

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。

[来源：T/AI 109.2-2021,3.35]

3.1.44

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。

注：光栅中的行从左到右扫描。

[来源：T/AI 109.2-2021,3.36]

3.1.45

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。

[来源：T/AI 109.2-2021,3.37]

3.1.46

划分方式 partition type

划分获得的子集的组织方式。

[来源：T/AI 109.2-2021,3.38]

3.1.47

IDR图像 instantaneous decoding refresh picture

NAL单元类型为4的I图像。IDR图像是随机访问点图像。

注：如果序列参数集后的第一幅编码图像是IDR图像，位流中解码顺序在该IDR图像之后的任一图像的参考图像都为该IDR图像或者在该IDR图像解码顺序后的图像。

3.1.48

I图像 I picture

只使用帧内预测解码的编码图像。

[来源：T/AI 109.2-2021,3.39]

T/AI 109.8-2025

3.1.49

级别 level

在某一档次下对语法元素和语法元素参数值的限定集合。

[来源：T/AI 109.2-2021,3.40]

3.1.50

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。

[来源：T/AI 109.2-2021,3.41]

3.1.51

解码图像 decoded picture

解码器根据位流重建的图像。

[来源：T/AI 109.2-2021,3.42]

3.1.52

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。

注：包括参考图像缓冲区。

[来源：T/AI 109.2-2021,3.43]

3.1.53

解析过程 parse

由位流获得语法元素的过程。

[来源：T/AI 109.2-2021,3.44]

3.1.54

块 block

一个 M×N（M 列 N 行）的样值矩阵或者变换系数矩阵。

[来源：T/AI 109.2-2021,3.45]

3.1.55

块复制帧内预测 intra block copy prediction

在相同解码图像中使用已解码的样值复制至当前样本区域，作为当前样本预测值的模式。

注：复制样值的区域与当前样本区域大小相同。

[来源：T/AI 109.2-2021,3.46]

3.1.56

块矢量 block vector

用于块复制帧内预测模式的二维矢量，其值为当前块和参考块之间的坐标偏移量。

注：当前块与参考块均在当前图像中。

T/AI 109.8-2025

[来源：T/AI 109.2-2021,3.47]

3.1.57

块扫描 block scan
量化系数的特定串行排序方式。
[来源：T/AI 109.2-2021,3.48]

3.1.58

历史点预测信息表 historical point prediction information list
用于非普通串子模式，由预测单元的点预测信息表构成。
[来源：T/AI 109.2-2021,3.50]

3.1.59

历史运动信息表 historical motion information list
用于帧间预测，由预测单元的运动信息构成。
[来源：T/AI 109.2-2021,3.51]

3.1.60

历史帧内复制信息表 historical intra copy information list
用于块复制帧内预测或串复制帧内预测，由帧内复制信息构成。
[来源：T/AI 109.2-2021,3.52]

3.1.61

亮度 luma
表示图像明暗程度的信号。
注：符号为Y，表示亮度信号的样值矩阵或单个样值。
[来源：T/AI 109.2-2021,3.53]

3.1.62

量化参数 quantization parameter
在解码过程对量化系数进行反量化的参数。
[来源：T/AI 109.2-2021,3.54]

3.1.63

量化系数 quantization coefficient
反量化前变换系数的值。
[来源：T/AI 109.2-2021,3.55]

3.1.64

六参数仿射预测单元 affine prediction unit with six parameters
仿射控制点运动矢量组中的运动矢量个数为三个的仿射预测单元。
[来源：T/AI 109.2-2021,3.56]

T/AI 109.8-2025

3.1.65

滤波后样本 filtered sample

经去块效应滤波得到的样本。

[来源：T/AI 109.2-2021,3.57]

3.1.66

NAL 单元 NAL unit

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

3.1.67

NAL 单元流 NAL unit stream

由 NAL 单元组成的序列。

3.1.68

NRAP 图像 non random access point picture

P 图像或者 B 图像。

注：RL图像不是NRAP图像。

3.1.69

P 图像 P picture

帧间预测中最多使用一个运动矢量和参考索引来预测每个块的样本值的图像。

注：P图像不能使用显示顺序上位于当前图像之后的图像作为参考图像进行解码。

3.1.70

匹配串 ordinary string

使用任意合法串矢量进行预测的串。

[来源：T/AI 109.2-2021,3.59]

3.1.71

偏移后样本 offseted sample

经样值偏移补偿得到的样本。

[来源：T/AI 109.2-2021,3.60]

3.1.72

片 patch

图像中可被独立解码的矩形区域。

3.1.73

频率索引 frequency index

与解码过程中反变换之前的变换系数相关的一维或二维索引。

T/AI 109.8-2025

3.1.74

普通串子模式 ordinary string submode

使用匹配串和不完全匹配串进行预测的模式。

[来源：T/AI 109.2-2021,3.62]

3.1.75

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。

[来源：T/AI 109.2-2021,3.63]

3.1.76

RAPI图像 random access point I picture

NAL单元类型为2的I图像。

注1：RAPI图像是随机访问点图像。

注2：如果序列参数集后的第一幅编码图像是RAPI图像，在解码顺序在该RAPI图像之后、显示顺序在该RAPI图像之前的图像可能参考解码顺序在该RAPI图像之前的图像。

注3：解码顺序在RAPI之后、显示顺序在RAPI之后的图像，只能参考RAPI图像或者该RAPI图像解码顺序后的图像。

3.1.77

RL前置知识图像 leading library picture of an RL picture

位流顺序在一个关联RL图像之前的非显示知识图像或经过位流编辑新出现在一个RL图像之前的显示知识图像，该知识图像的第一个知识图像编码片NAL单元所在的访问单元与关联RL图像的访问单元之间不存在其它图像的访问单元。

注1：RL前置知识图像不是随机访问点图像。

注2：若RL前置知识图像为显示知识图像，此时显示知识图像无需输出，可通过显示知识图像显示信息SEI负载进行标识（library_display_flag应为‘0’）。

3.1.78

RL图像 refrence library picture

只使用知识图像作为参考图像进行帧间预测解码的P图像或B图像。

注1：RL图像是随机访问点图像。

注2：如果当前显示图像对应的序列参数集NAL单元后的第一幅编码图像为RL图像，并且当前显示图像的显示顺序在该RL图像之后，则当前显示图像的参考图像应在以下范围内：该RL图像、显示顺序在该RL图像之后的显示图像、该RL图像所参考的知识图像。

3.1.79

色度 chroma

用于表示亮度和红色、蓝色的差值的信号。

注：符号为Cr和Cb，表示两种色差信号的样值矩阵或单个样值。

T/AI 109.8-2025

[来源：T/AI 109.2-2021,3.65]

3.1.80

视频加密密钥 video encryption key

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

3.1.81

视频密钥加密密钥 video key encryption key

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

3.1.82

输出顺序 output order

输出解码图像的顺序，与显示顺序相同。

[来源：T/AI 109.2-2021,3.67]

3.1.83

四参数仿射预测单元 affine prediction unit with four parameters

仿射控制点运动矢量组中的运动矢量个数为两个的仿射预测单元。

[来源：T/AI 109.2-2021,3.68]

3.1.84

随机访问 random access

从某一点而非位流起始点开始对位流解码并恢复出解码图像的能力。

[来源：T/AI 109.2-2021,3.69]

3.1.85

填充位 stuffing bits

编码时插入位流中的位串，在解码时被丢弃。

[来源：T/AI 109.2-2021,3.70]

3.1.86

图像 picture

一幅图像指输入视频的一帧。

3.1.87

图像参数集 picture parameter set

一种语法结构，包含作用于一至多幅图像的语法元素。

注：每个编码视频序列至少包括一个图像参数集NAL单元。

3.1.88

T/AI 109.8-2025

图像头 picture header

一种语法结构，包含作用于一幅图像的语法元素。

注：每个编码图像包含且只包含一个图像头NAL单元。

3.1.89

往返扫描 traverse scan

将二维矩形映射到一维串，一维串的入口从二维矩形的起始行开始，逐行扫描整个二维矩形。

注：起始行的行扫描方向是从左到右，起始行的相邻行的行扫描方向是从右到左，任意两个相邻行具有相反的行扫描方向。

[来源：T/AI 109.2-2021,3.73]

3.1.90

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）。

[来源：T/AI 109.2-2021,3.74]

3.1.91

位流 bitstream

编码图像所形成的二进制数据流。

[来源：T/AI 109.2-2021,3.75]

3.1.92

位流缓冲区 bitstream buffer

存储位流的缓冲区。

[来源：T/AI 109.2-2021,3.76]

3.1.93

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。

[来源：T/AI 109.2-2021,3.77]

3.1.94

未匹配样本 unmatched sample

在串复制帧内预测中，没有参考样本的样本。

[来源：T/AI 109.2-2021,3.78]

3.1.95

未匹配样本串 unmatched sample string

只包含未匹配样本的串。

[来源：T/AI 109.2-2021,3.79]

3.1.96

T/AI 109.8-2025

显示顺序 display order

显示解码图像的顺序。

注：不应被输出的图像是不具有显示顺序的图像。

[来源：T/AI 109.2-2021,3.80]

3.1.97

显示图像 output picture

解码器解码后将重建图像输出的 RL 图像、IDR 图像、P 图像、B 图像或者 RAPI 图像。

注：显示知识图像和非显示知识图像都不属于本文所定义的显示图像。

3.1.98

显示图像对应的图像参数集 picture parameter set for output picture

解码顺序在当前显示图像之前的距离当前显示图像的编码图像最近的 libraryStreamFlag 为 0 的序列参数集 NAL 单元后最近的图像参数集 NAL 单元表示的图像参数集。

3.1.99

显示图像对应的图像头 picture header for output picture

解码顺序在当前显示图像之前的最近的 library_stream_flag 为 0 的序列参数集后最近的图像头。

3.1.100

显示图像对应的序列参数集 sequence parameter set for output picture

解码顺序在当前显示图像之前的最近的 library_stream_flag 为 0 的序列参数集。

3.1.101

显示图像集 output picture set

共享同一个序列参数集的一个或多个显示图像。

3.1.102

显示知识图像 output library picture

每帧图像对应一个序列参数集，且对应的序列集参数中知识位流标志为 1、知识图像模式索引为 1 的编码图像。

注1：显示知识图像的编码片 NAL 单元类型为 17。

注2：显示知识图像是随机访问点图像，作为 RL 前置知识图像的显示知识图像不是随机访问点图像。

3.1.103

序列参数集 sequence parameter set

一种语法结构，包含作用于一个编码视频序列的语法元素。

3.1.104

样本 sample

构成图像的基本元素。

T/AI 109.8-2025

[来源：T/AI 109.2-2021,3.81]

3.1.105

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÷v，其中 h 为水平方向样本个数；v 为垂直方向样本个数。

[来源：T/AI 109.2-2021,3.82]

3.1.106

样值 sample value

样本的幅值。

[来源：T/AI 109.2-2021,3.83]

3.1.107

游程 run

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

[来源：T/AI 109.2-2021,3.84]

3.1.108

预测 prediction

使用预测器对当前解码样值或者数据元素进行估计。

3.1.109

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。

[来源：T/AI 109.2-2021,3.86]

3.1.110

预测单元 prediction unit

可包含一个亮度预测块和对应的色度预测块，也可只包含一个亮度预测块，或只包括色度预测块，由编码单元划分得到。

[来源：T/AI 109.2-2021,3.87]

3.1.111

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。

[来源：T/AI 109.2-2021,3.89]

3.1.112

预测块 prediction block

一个使用相同预测过程的 M×N 的样值块，由编码单元划分得到。

T/AI 109.8-2025

[来源：T/AI 109.2-2021,3.90]

3.1.113

预测值 prediction value

在样值或数据元素的解码过程中，用到的先前已解码的样值或数据元素的组合。

[来源：T/AI 109.2-2021,3.91]

3.1.114

语法元素 syntax element

位流中的数据单元解析后的结果。

[来源：T/AI 109.2-2021,3.92]

3.1.115

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。

注：指向参考图像队列0和参考图像队列1中的参考图像的运动矢量分别称为L0运动矢量和L1运动矢量。

[来源：T/AI 109.2-2021,3.94]

3.1.116

运动信息 motion information

用于帧间预测的五元组，由预测参考模式、L0 运动矢量、L1 运动矢量、L0 参考索引和 L1 参考索引构成。

[来源：T/AI 109.2-2021,3.95]

3.1.117

帧 frame

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

[来源：T/AI 109.2-2021,3.96]

3.1.118

帧间编码 inter coding

使用帧间预测对编码单元或图像进行编码。

[来源：T/AI 109.2-2021,3.97]

3.1.119

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。

[来源：T/AI 109.2-2021,3.98]

3.1.120

帧内编码 intra coding

T/AI 109.8-2025

使用帧内预测对编码单元或图像进行编码。

注：使用普通帧内预测的编码单元称为普通帧内编码单元；使用块复制帧内预测的编码单元称为块复制帧内编码单元；使用串复制帧内预测的编码单元称为串复制帧内编码单元。

[来源：T/AI 109.2-2021,3.99]

3.1.121

帧内复制信息 intra copy information

用于块复制帧内预测或串复制帧内预测，包括位移矢量、位置、尺寸信息和重复次数。

[来源：T/AI 109.2-2021,3.100]

3.1.122

帧内模式频数表 intra mode frequency list

用于帧内预测，由预测单元的帧内预测模式的频数构成。

[来源：T/AI 109.2-2021,3.101]

3.1.123

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。

注：帧内预测分为普通帧内预测、块复制帧内预测和串复制帧内预测。

[来源：T/AI 109.2-2021,3.102]

3.1.124

知识图像 library picture

每帧图像对应一个序列参数集，且对应的序列集参数中知识位流标志为 1 的编码图像。

注：知识图像的编码片NAL单元类型为12、17或18，知识图像关联隐私编码片。

3.1.125

知识图像片 patch of library picture

知识图像的一个片。

3.1.126

知识图像编码片 coded patch of library picture

知识图像的编码片。

3.1.127

知识图像对应的图像参数集 picture parameter set for library picture

解码顺序在当前知识图像之前的距离当前知识图像的编码图像最近的 libraryStreamFlag 为 1 的序列参数集 NAL 单元后最近的图像参数集 NAL 单元表示的图像参数集。

3.1.128

知识图像对应的图像头 picture header for library picture

解码顺序在当前知识图像之前的最近的 library_stream_flag 为 1 的序列参数集后最近的图像

T/AI 109.8-2025

头。

3.1.129

知识图像对应的序列参数集 sequence parameter set for library picture

解码顺序在当前知识图像之前的最近的 library_stream_flag 为 1 的序列参数集。

3.1.130

知识位流 library stream

编码知识图像生成的位流。

3.1.131

知识位流片 library patch stream

编码知识图像片生成的位流。

注：知识图像的第一个编码片位流包括知识图像的序列集参数、图像集参数、补充增强信息、图像头的位流和片索引为0的编码片位流，其余编码片位流包括一个片索引不为零的编码片位流。

3.1.132

字节 byte

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

3.1.133

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。

[来源：T/AI 109.2-2021,3.109]

3.1.134

最大编码单元 largest coding unit

包括一个 LxL 的亮度样值块和对应的色度样值块，由图像划分得到。

[来源：T/AI 109.2-2021,3.110]

3.1.135

最大编码块 largest coding block

一个 KxK 的样值块，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。

[来源：T/AI 109.2-2021,3.111]

3.2 缩略语

下列缩略语适用于本文件。

ABVR：自适应块矢量精度(Adaptive Block Vector Resolution)

AEC：高级熵编码（Advanced Entropy Code）

AIR：自适应帧内刷新（Adaptive Intra Refresh）

ALF：自适应修正滤波（Adaptive Leveling Filter）

T/AI 109.8-2025

AMVR：自适应运动矢量精度（Adaptive Motion Vector Resolution）
ASR：仿射预测样本改善（Affine Sample Refinement）
AU：访问单元（Access Unit）
AWP：角度加权预测模式（Angular Weighted Prediction）
BBV：位流参考缓冲区管理（Bitstream Buffer Verifier）
BGC：双向梯度修正（Bi-directional Gradient Correction）
BIO：双向光流（BI-directional Optical flow）
CB：编码块（Coding Block）
CBR：恒定位率（Constant Bit Rate）
CBVP：基于类别的块矢量预测（Class based Block Vector Prediction）
CCSAO：跨分量样值偏移补偿（Cross-Component Sample Adaptive Offset）
CPLF：跨片环路滤波（Cross Patch Loop Filter）
CRC：循环冗余校验码(Cyclic Redundancy Code)
CU：编码单元（Coding Unit）
CUT：编码树（Coding Unit Tree）
CVS：编码视频序列（Coded Video Sequence）
DMVR：解码端运动矢量改良（Decoder-side Motion Vector Refinement）
DOI：解码顺序索引（Decode Order Index）
DPB：解码图像缓冲区（Decoded Picture Buffer）
DT：衍生模式树（Derived Tree）
EALF：增强自适应修正滤波（Enhanced Adaptive Leveling Filter）
EBSP：扩展字节序列载荷（Encapsulated Byte Sequence Payload）
EIPM：帧内预测扩展模式（Extended Intra Prediction Mode）
ESAO：增强样值偏移补偿（Enhanced Sample Adaptive Offset）
EVEK：加密后视频加密密钥（Encoded Video Encryption Key）
EVS：等值串（Equal Value String）
FFT：快速傅立叶变换(Fast Fourier Transform)
FIMC：基于频数信息的帧内编码（Frequency-based Intra Mode Coding）
FTM：特征张量图(Feature Tensor Map)
FT：特征张量(Feature Tensor)
HMVP：基于历史信息的运动矢量预测（History-based Motion Vector Prediction）
IBC：帧内块复制（Intra Block Copy）
IDR：即时解码刷新(Instantaneous Decoding Refresh)
IIP：改进型帧内预测（Improved Intra Prediction）
ISC：帧内串复制（Intra String Copy）
IST：隐择变换（Implicit Selected Transform）
IV：初始向量（Initial Vector）
LCB：最大编码块（Largest Coding Block）
LCU：最大编码单元（Largest Coding Unit）
MIPF：多组合帧内预测滤波器（Multiple Intra Prediction Filter）
MVAP：运动矢量角度预测（Motion Vector Angle Prediction）
NAL：网络抽象层(Network Abstraction Layer)

T/AI 109.8-2025

NRAP：非随机访问点（Non Random Access Point）
OBMC：重叠块运动补偿（Overlapped Block Motion Compensation）
OS：普通串（Ordinary String）
PB：预测块（Prediction Block）
PBT：基于位置的变换（Position Based Transform）
PCM：脉冲编码调制(Pulse Code Modulation)
PMC：跨多分量预测（Prediction from Multiple Cross-component）
POI：显示顺序索引（Present Order Index）
PU：预测单元（Prediction Unit）
RAP：随机访问点（Random Access Point）
RAS：随机访问片段（Random Access Segment）
RAPI：随机访问点 I 图像（Random Access Point I Picture）
RBSP：原始字节序列负载(Raw Byte Sequence Payload)
ROI：感兴趣区域（Region Of Interesting）
SAO：样值偏移补偿（Sample Adaptive Offset）
SAWP：空域角度加权预测（Spatial Angular Weighted Prediction）
SBT：子块变换（Sub-Block Transform）
SEI：补充增强信息(Supplement Enhancement Information)
SODB：数据位串(String Of Data Bits)
SRCC：基于扫描区域的系数编码（Scan Region based Coefficient Coding）
ST：二次变换（Secondary Transform）
TB：变换块（Transform Block）
TSCPM：跨分量两步预测模式（Two-Step Cross-component Prediction Mode）
TVC：变换域矢量编码(Transform Vector Coding)
UBVS：单位基矢量串（Unit Basis Vector String）
UMVE：高级运动矢量表达（Ultimate Motion Vector Expression）
VCL：视频编码层(Video Coding Layer)
VEK：视频加密密钥（Video Encryption Key）
VKEK：视频密钥加密密钥（Video Key Encryption Key）

4 约定

4.1 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。

4.2 算术运算符

算术运算符定义见表 1。

T/AI 109.8-2025

表1 算术运算符定义

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
$\lceil . \rceil$	上取整

4.3 逻辑运算符

逻辑运算符定义见表 2。

表2 逻辑运算符定义

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

4.4 关系运算符

关系运算符定义见表 3。

表3 关系运算符定义

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

4.5 位运算符

位运算符定义见表 4。

T/AI 109.8-2025

表4 位运算符定义

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

4.6 赋值运算符

赋值运算符定义见表 5。

表5 赋值运算符定义

运算符	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)

4.7 数学函数

数学函数定义见式（1）～式（13）。

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

式中：

x——自变量 x。

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

式中：

x——自变量 x。

$$\mathrm{Floor}(x)=\lfloor x \rfloor \qquad (3)$$

式中：

x——自变量 x。

$$\mathrm{Clip1}(x)=\mathrm{Clip3}(0,2^{B}-1,x) \qquad (4)$$

式中：

x——自变量 x；

B——编码样本精度，对应比特位数。

T/AI 109.8-2025

$$\mathrm{Clip3}(i,j,x)=\begin{cases} i & ; \quad x<i \\ j & ; \quad x>j \\ x & ; \quad \text{其他} \end{cases} \cdots\cdots (5)$$

式中：

x——自变量 x；

i——下界；

j——上界。

$$\mathrm{Median}(x,y,z)=x+y+z-\mathrm{Min}(x,\mathrm{Min}(y,z))-\mathrm{Max}(x,\mathrm{Max}(y,z)) \cdots\cdots (6)$$

式中：

x——自变量 x；

y——自变量 y；

z——自变量 z。

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

式中：

x——自变量 x；

y——自变量 y。

$$\mathrm{Max}(x,y)=\begin{cases} x & ; \quad x>=y \\ y & ; \quad x<y \end{cases} \cdots\cdots (8)$$

式中：

x——自变量 x；

y——自变量 y。

$$\mathrm{Sign}(x)=\begin{cases} 1 & ; \quad x>=0 \\ -1 & ; \quad x<0 \end{cases} \cdots\cdots (9)$$

式中：

x——自变量 x。

$$\mathrm{Log}(x)=\log_2 x \cdots\cdots (10)$$

式中：

x——自变量 x。

$$\mathrm{Ln}(x)=\log_e x \cdots\cdots (11)$$

式中：

x——自变量 x；

e——自然对数的底，其值为 2.718281828。

$$\mathrm{Rounding}(x,s)=\begin{cases} \mathrm{Sign}(x)\times((\mathrm{Abs}(x)+(1<<(s-1)))>>s) & s\geq 1 \\ x & s=0 \end{cases} \cdots\cdots (12)$$

式中：

x——自变量 x；

T/AI 109.8-2025

s ——自变量 s 。

$$\text{Exchange}(x, y) = (\text{交换}x\text{和}y\text{的值}) \quad (13)$$

式中：

x ——自变量 x ；

y ——自变量 y 。

4.8 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

4.9 位流语法、解析过程和解码过程的描述方法

4.9.1 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（用下划线分隔的英文字母组，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位流语法的超集，附加的语法限制在相关章条中说明。

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	
…	

T/AI 109.8-2025

表 7 语法描述的伪代码（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	
/*“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	

T/AI 109.8-2025

解析过程和解码过程用文字和类似 C 语言的伪代码描述。

4.9.2 函数

4.9.2.1 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。

4.9.2.2 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。

4.9.2.3 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。

4.9.2.4 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。

4.9.2.5 is_end_of_patch( )

在位流中检测是否已达到片的结尾，如果已到达片的结尾，返回 TRUE，否则返回 FALSE。此函数不修改位流指针。函数定义见表 8。

表8 is_end_of_patch 函数的定义

函数定义	描述符
is_end_of_patch( ) {	
if (byte_aligned( )) {	
if (next_bits(40) == 0x8000000001)	
return TRUE /* 片结束 */	
}	
else {	
if ((byte_aligned_next_bits(32) == 0x00000001) && is_stuffing_pattern( ))	
return TRUE /* 片结束 */	
}	
return FALSE;	
}	

4.9.2.6 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片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 9。

表9 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;	
}	

T/AI 109.8-2025

4.9.2.7 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。

4.9.2.8 more_rbsp_data( )

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

判断 RBSP 中是否有更多数据的方法不在本文件中规定。

4.9.2.9 more_data_in_byte_stream( )

如果位流中后续还有更多数据，more_data_in_byte_stream( )的返回值为 TRUE；否则，more_data_in_byte_stream( )的返回值为 FALSE。

4.9.3 描述符

描述符表示不同语法元素的解析过程，见表 10。

表10 描述符

描述符	说明
ae(v)	高级熵编码的语法元素。解析过程见7.3
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。解析过程在见7.2中
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。解析过程在见7.2中

4.9.4 保留、禁止和标记位

本文件规定的位流语法中，某些语法元素的值被标注为“保留”（reserved）或“禁止”（forbidden）。

“保留”规定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位流中。

“禁止”规定了一些特定语法元素值，这些值不应出现在符合本文件的位流中。

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。

“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文件的扩展，解码处理应忽略这些位。

T/AI 109.8-2025

5 视频编码位流的结构

5.1 位流格式

本条规定 NAL 单元流和字节流之间的关系，二者均可称为位流。

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

字节流可以用 NAL 单元流构造，通过将 NAL 单元按照解码顺序排列，并且为每个 NAL 单元添加一个起始码前缀和若干零值字节形成一个位流。NAL 单元流格式可以通过在位流中搜索唯一的起始码前缀，从位流格式中提取出来。除位流格式以外，NAL 单元的其他组织方法，本文件不做规定。位流格式在 6.1.2、6.2.1 和 6.2.2 中规定。

5.2 编码视频序列

5.2.1 概述

编码视频序列是位流的最高层次语法结构，包含一个或多个连续的访问单元。编码视频序列由 IDR 图像的访问单元、RAPI 图像的访问单元、RL 前置知识图像的访问单元或显示知识图像的访问单元开始，流结尾 NAL 单元或编码视频序列结尾 NAL 单元表明了一个编码视频序列的结束。每个编码视频序列最多包含一个 IDR 图像的访问单元、RAPI 图像的访问单元、RL 前置知识图像的访问单元或显示知识图像的访问单元。访问单元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。访问单元的构成在 6.2.2.2.2.3 中描述。

5.2.2 逐行编码视频序列

本文件支持逐行序列。

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描述，这些信息不影响解码过程。

解码器的输出是一系列解码图像。

5.2.3 编码视频序列与随机访问

编码视频序列的起始图像应为 IDR 图像、RAPI 图像、RL 前置知识图像或显示知识图像。

一个编码视频序列包含一个或多个随机访问片段，每个随机访问片段由一个随机访问点访问单元起始，包含至下一个随机访问点访问单元前的一个或多个连续的访问单元。

随机访问点图像应为 IDR 图像、RAPI 图像、RL 图像或显示知识图像。对于除 RL 图像外的其它随机访问点图像，显示顺序位于随机访问点图像之后的图像不允许参考显示顺序位于随机访问点之前的图像。显示顺序位于一幅 RL 图像之后的显示图像允许参考该 RL 图像、该 RL 图像所参考的知识图像、以及显示顺序位于该 RL 图像之后的图像。

可以参考一幅知识图像的图像包括按解码顺序从该知识图像之后的第一个 RL 图像开始，到下一个非 RL 图像的随机访问点图像或下一个知识图像后第一个 RL 图像之前的所有显示图像。

随机访问点图像和非显示知识图像的访问单元中应包含序列参数集 NAL 单元,其它访问单元不应包含序列参数集 NAL 单元。如果一幅知识图像的编码数据包含在一个访问单元中，该访问单元应包含该知识图像对应的序列参数集 NAL 单元。如果一幅知识图像的编码数据包含在多于一个访问单元中，这些访问单元中的第一个应包含该知识图像对应的序列参数集 NAL 单元。一幅显示图像对应的序列参数

T/AI 109.8-2025

集 NAL 单元，或者在该显示图像的访问单元中，或者为解码顺序位于该显示图像所在访问单元之前的最近的 LibraryStreamFlag 为 0 的序列参数集的序列参数集 NAL 单元。

在对位流进行随机访问时，通过以下操作得到的一个新的编码视频序列的位流仍应符合本文件：

a) 丢弃一个 IDR 图像访问单元前的所有访问单元；
b) 丢弃一个 RAPI 图像访问单元前的全部访问单元以及该 RAPI 图像访问单元之后的 RandomAccessDecodableFlag 为 0 的编码图像访问单元；
c) 丢弃一个 RL 前置知识图像访问单元前的全部访问单元；
d) 丢弃一个显示知识图像访问单元前的全部访问单元；
e) 丢弃一个 RL 图像访问单元前除该 RL 图像的参考的知识图像的访问单元外的其余访问单元以及该 RL 图像访问单元之后的 RandomAccessDecodableFlag 为 0 的编码图像访问单元。

5.3 图像

5.3.1 概述

一幅图像是编码视频序列的一帧，其编码数据包含在一个或多个访问单元中。其编码图像由图像头 NAL 单元、补充增强信息（如果存在）NAL 单元、扩展数据 NAL（如果存在）单元和该图像的所有编码片 NAL 单元构成。具体地，IDR 图像的编码图像包括图像头 NAL 单元、零个或多个该 IDR 图像的补充增强信息 NAL 单元和该 IDR 图像的所有 IDR 图像编码片 NAL 单元。RAPI 图像的编码图像包括图像头 NAL 单元、零个或多个该 RAPI 图像的补充增强信息 NAL 单元和该 RAPI 图像的所有 RAPI 图像编码片 NAL 单元。P 图像和 B 图像的编码图像包括图像头 NAL 单元、零个或多个该 P 图像或 B 图像的补充增强信息 NAL 单元和该 P 图像或 B 图像的所有 NRAP 图像编码片 NAL 单元。RL 图像的编码图像包括图像头 NAL 单元、零个或多个该 RL 图像的补充增强信息 NAL 单元和该 RL 图像的所有 RL 图像编码片 NAL 单元。知识图像的编码图像由图像头 NAL 单元和所有知识图像编码片 NAL 单元构成。RL 前置知识图像以及显示知识图像的编码图像中所有编码片 NAL 单元是连续的，其访问单元包含编码图像的所有 NAL 单元，非 RL 前置知识图像的编码片可以作为访问单元与显示图像的访问单元交织。

特别地，非 RL 前置知识图像的多个知识图像编码片 NAL 单元之间可交织一至多幅显示图像的访问单元，但交织的显示图像的访问单元不应为 RL 图像、IDR 图像或者 RAPI 图像的访问单元。一幅 RL 前置知识图像或显示知识图像的所有知识图像编码片 NAL 单元应连续，每个知识图像编码片 NAL 单元后可与隐私图像编码片 NAL 单元（若存在）交织，但不与显示图像的访问单元交织。

一幅知识图像的所有编码片 NAL 单元应位于参考该知识图像的第一个 RL 图像的访问单元前。不同知识图像的知识图像编码片 NAL 单元不可交织。一幅 RL 图像参考的知识图像是从位流中 RL 访问单元起依解码次序中逆序查找到的与该 RL 图像的 LayerId 相同的第一个知识图像的访问单元所表示的知识图像。

图像的解码处理包括解析过程和解码过程。

5.3.2 图像格式

5.3.2.1 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。

T/AI 109.8-2025

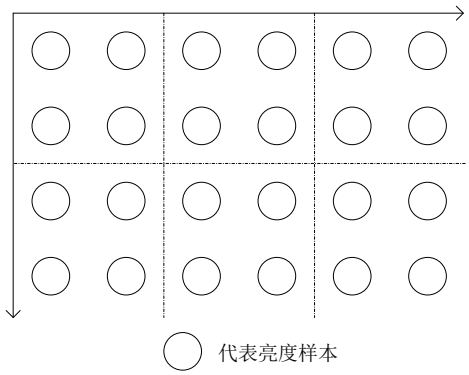


图1 4:0:0 格式下亮度样本位置

5.3.2.2 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。亮度和色度样本位置见图 2。

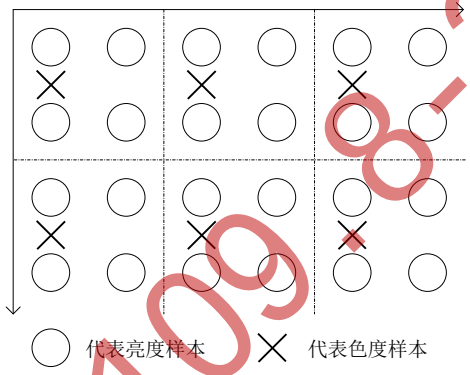


图2 4:2:0 格式下亮度和色度样本位置

5.3.2.3 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。亮度和色度样本位置见图 3。

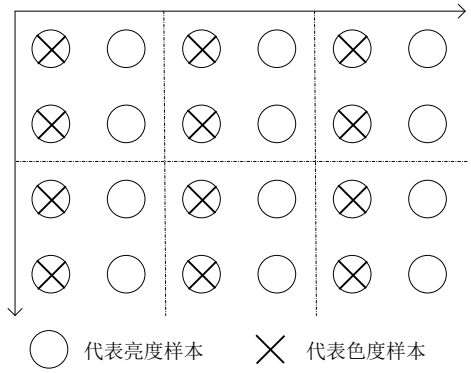


图3 4:2:2 格式下亮度和色度样本位置

5.3.2.4 4:4:4 格式

T/AI 109.8-2025

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。亮度和色度样本位置见图 4。

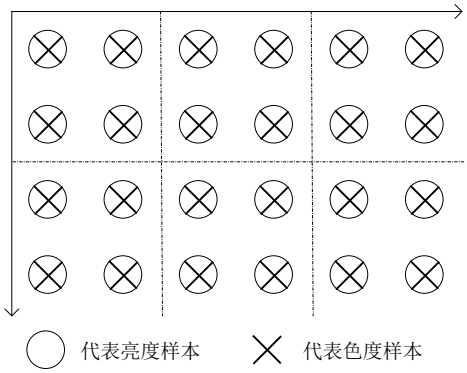


图4　4:4:4 格式下亮度和色度样本位置

5.3.3　图像类型

本文件定义了三种图像：

——I 图像；

——P 图像；

——B 图像。

注：IDR 图像、RAPI 图像和知识图像都是 I 图像。

5.3.4　图像间的顺序

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

位流中显示顺序在一个随机访问点图像之后的图像的解码顺序应在该随机访问点图像之后。

下面举例说明图像重排序。示例文字说明中图像编号中的数字为显示顺序号。

示例 1：IDR/RAPI/RL 图像的访问单元和 P 图像的访问单元之间有 3 幅 B 图像，两幅连续的 P 图像的访问单元之间也有 3 幅 B 图像的访问单元。按照解码顺序用图像 OI/RL 预测图像 4P，用图像 4P 和 OI/RL 预测图像 2B,用图像 2B 和 OI/RL 预测图像 1B，用图像 4P 和 2B 预测图像 3B。解码顺序是 OI/RL，4P，2B，1B,3B；显示顺序是 OI/RL，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	IDR/RAPI/RL	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	IDR/RAPI/RL	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：IDR/RAPI/RL 图像的访问单元和 P 图像的访问单元之间有 7 幅 B 图像，两幅连续的 P 图像的访问单元之间

T/AI 109.8-2025

也有 7 幅 B 图像。按照解码顺序用图像 OI/RL 预测图像 8P，用图像 8P 和 OI/RL 预测图像 4B,用图像 4B 和 OI/RL 预测图像 2B，用图像 2B 和 OI/RL 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 OI/RL，8P，4B，2B,1B，3B，6B,5B,7B；显示顺序是 OI/RL，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	IDR/RAPI/RL	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	IDR/RAPI/RL	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：IDR/RAPI/RL 图像的访问单元和 P 图像的访问单元之间有 2 幅 B 图像，两幅连续的 P 图像的访问单元之间也有 2 幅 B 图像的访问单元。按照解码顺序用图像 OI/RL 预测图像 3P，用图像 3P 和 OI/RL 预测图像 1B 和 2B。解码顺序是 OI/RL，3P，1B，2B；显示顺序是 OI/RL，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	IDR/RAPI/RL	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	IDR/RAPI/RL	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

示例 4：IDR/RAPI/RL 图像的访问单元后有 16 幅 B 图像的访问单元。

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	IDR/RAPI/RL	B	B	B	B	B	B	B	B	B	B	B	B	B	B	B	B
显示顺序	0	16	8	4	2	1	3	6	5	7	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	IDR/RAPI/RL	B	B	B	B	B	B	B	B	B	B	B	B	B	B	B	B
解码顺序	0	5	4	6	3	8	7	9	2	12	11	13	10	15	14	16	1

如果编码视频序列中包含非 RL 前置知识图像，在随机访问参考该知识图像的第一个 RL 图像时，

T/AI 109.8-2025

可应进行位流重排序，抽取该知识图像对应的序列参数集 NAL 单元、图像参数集 NAL 单元、图像头和所有知识图像位流片，使所有知识图像位流片按照片索引递增的顺序连续，形成 RL 前置知识图像的访问单元，并使参考该知识图像的第一个 RL 图像的访问单元紧跟该 RL 前置知识图像的访问单元。

如果编码视频序列中存在知识图像，位流中的知识图像位流片的次序应与知识图像编码片索引的递增次序一致，所有知识图像编码片应在以其为参考的第一个 RL 图像的访问单元之前解码。两个 RL 图像之间最多有一幅知识图像的位流，且该知识图像的所有位流片都应在两个 RL 图像的访问单元之间，并在满足以上约束的前提下，可以在两个 RL 图像的访问单元之间任意变动。

下面举例说明不同的知识位流在位流中的位置和知识位流的重排序（知识图像用“L”后跟知识图像索引表示）。

示例 5：L0 为非 RL 前置知识图像，其编码图像包含 3 个编码片 NAL 单元，每个编码片 NAL 单元紧跟一个显示图像的访问单元，最后一个编码片 NAL 单元后紧跟 RL 图像的访问单元。以参考 L0 的 RL 图像为随机访问点图像时，位流经过重组，将 L0 变为 RL 前置知识图像，其访问单元之前的所有位流可以丢弃。

重组前：

位流顺序	0	1	2	3	4	5	6	7	8	9	10	11	12
类型	IDR/RAPI/RL	P	P	P	L0(patch0)	P	L0(patch1)	P	L0(patch2)	RL	P	P	P
显示顺序	0	1	2	3		4		5		6	7	8	9

重组后：

位流顺序	0	1	2	3	4	5	6	7	8	9	10	11
类型	IDR/RAPI/RL	RL	P	P	P	P	P	L0(patch0、patch1、patch2)	RL	P	P	P
显示顺序		0	1	2	3	4	5		6	7	8	9

示例 6：L0 为非 RL 前置知识图像，其编码图像包含 3 个编码片 NAL 单元，每个编码片 NAL 单元紧跟一个显示图像的访问单元，该知识图像的最后一个编码片与参考该知识图像的第一个 RL 图像的访问单元之间还存在其余显示图像的访问单元。以参考 L0 的 RL 图像为随机访问点图像时，位流经过重组，将 L0 变为 RL 前置知识图像，其访问单元之前的所有位流可以丢弃。

重组前：

位流顺序	0	1	2	3	4	5	6	7	8	9	10	11	12	13
类型	IDR/RAPI/RL	P	P	P	P	L0(patch0)	P	L0(patch1)	P	L0(patch2)	P	P	RL	P
显示顺序	0	1	2	3	4		5		6		7	8	9	10

重组后：

位流顺序	0	1	2	3	4	5	6	7	8	9	10	11
类型	IDR/RAPI/RL	P	P	P	P	P	P	P	P	L0(patch0、patch1、patch2)	RL	P
显示顺序	0	1	2	3	4	5	6	7	8		9	10

示例 7：L0 为非 RL 前置知识图像，其编码图像包含 3 个编码片 NAL 单元，且三个编码片 NAL 单元连续，该知识图像的最后一个编码片与参考该知识图像的第一个 RL 图像的访问单元之间还存在其余显示图像的访问单元。以参考 L0 的 RL 图像为随机访问点图像时，位流经过重组，将 L0 变为 RL 前置知识图像，其访问单元之前的所有位流可以丢弃。

重组前：

T/AI 109.8-2025

位流顺序	0	1	2	3	4	5	6	7	8	9	10	11	12
类型	IDR/RAPI/RL	P	P	P	P	P	L0(patch0)	L0(patch1)	L0(patch2)	P	P	RL	P
显示顺序	0	1	2	3	4	5				6	7	8	9

重组后：

位流顺序	0	1	2	3	4	5	6	7	8	9	10
类型	IDR/RAPI/RL	P	P	P	P	P	P	P	L0(patch0、patch1、patch2)	RL	P
显示顺序	0	1	2	3	4	5	6	7		8	9

5.3.5 参考图像和参考图像队列

P 图像可有知识图像和显示顺序上位于当前图像之前的多幅参考图像，这些参考图像构成 P 图像的参考图像队列 0。

B 图像可有知识图像、多幅显示顺序位于当前图像之前的（过去的）参考图像和多幅显示顺序位于当前图像之后的参考图像，这些参考图像构成 B 图像的参考图像队列 0 和参考图像队列 1。

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。

5.4 片

片是图像中的矩形区域，包含若干最大编码单元在图像内的部分，片之间不应重叠。片结构见图5。

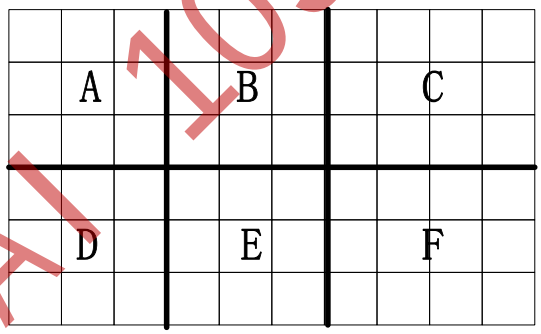


图5 片结构

5.5 最大编码单元、编码树和编码单元

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。

6 视频编码位流的语法和语义

6.1 语法描述

6.1.1 位流 NAL 单元语法

T/AI 109.8-2025

位流 NAL 单元语法见表 11。

表11 位流 NAL 单元语法表

位流 NAL 单元语法定义	描述符
byte_stream_nal_unit(NumBytesInNALunit) {	
while(next_bits(32) != 0x00000001)	
leading_zero_8bits /* 应等于 0x00 */	f(8)
zero_byte /* 应等于 0x00 */	f(8)
start_code_prefix_one_3bytes /* 应等于 0x000001 */	f(24)
nal_unit(NumBytesInNALunit)	
while(more_data_in_byte_stream( ) && next_bits(32) != 0x00000001)	
trailing_zero_8bits /* 应等于 0x00 */	f(8)
}	

6.1.2 NAL 单元语法

NAL 单元语法见表 12。

表12 NAL 单元语法表

NAL 单元语法定义	描述符
nal_unit(NumBytesInNALunit) {	
forbidden_zero_bit	f(1)
encryption_idc	u(2)
authentication_idc	u(2)
authentication_data_id	u(1)
layer_id	u(2)
temporal_id	u(3)
nal_unit_type	u(5)
NumBytesInPayload = 0	
for (i=2; i<NumBytesInNALunit; i++) {	
if (i+2 < NumBytesInNALunit && next_bits(24) == 0x000003) {	
payload_byte[NumBytesInPayload++]	b(8)
payload_byte[NumBytesInPayload++]	b(8)
i += 2	
emulation_prevention_three_byte /* 应等于 0x03 */	f(8)
}	
else	
payload_byte[NumBytesInPayload++]	b(8)
}	
}	

T/AI 109.8-2025

6.1.3 序列参数集 RBSP 定义

序列参数集 RBSP 定义见表 13。

表13 序列参数集 RBSP 定义

序列参数集RBSP定义	描述符
seq_parameter_set_rbsp( ) {	
profile_id	u(8)
level_id	u(8)
temporal_nesting_flag	u(1)
library_picture_enable_flag	u(1)
library_stream_flag	u(1)
if (LibraryPictureEnableFlag) {	
duplicate_sequence_header_flag	u(1)
library_picture_mode_index	u(2)
}	
rapi_nalu_type_enable_flag	u(1)
if (profile_id == 0x40 || profile_id == 0x42) {	
num_of_layers_minus1	u(2)
ssvc_enable_flag	u(1)
privacy_enable_flag	u(1)
if (PrivacyEnableFlag)	
privacy_mode_index	u(2)
if (!SsvcEnableFlag)	
reference_picture_resampling_enable_flag	u(1)
reserved_zero_bit /* 应等于0 */	f(1)
}	
if (NumOfLayers > 1) {	
sps_all_independent_layers_flag	u(1)
if (!SpsAllIndependentLayersFlag) {	
if (SsvcEnableFlag && PrivacyModeIndex != 1)	
reserved_zero_bit /* 应等于0 */	f(1)
for (i = 1; i < NumOfLayers; i++) {	
sps_independent_layer_flag[i]	u(1)
if (!SpsIndependentLayerFlag[i])	
ref_layer_id[i]	u(2)

T/AI 109.8-2025

表 13 序列参数集 RBSP 定义（续）

序列参数集RBSP定义	描述符
}	
}	
}	
for (i = 0; i < NumOfLayers; i++) {	
sps_horizontal_size[i]	u(14)
sps_vertical_size[i]	u(14)
sps_pic_width_in_luma[i]	u(14)
sps_pic_height_in_luma[i]	u(14)
if (ReferencePictureResamplingEnableFlag) {	
sps_rpr_pic_width_in_luma[i]	u(14)
sps_rpr_pic_height_in_luma[i]	u(14)
}	
}	
chroma_format	u(2)
sample_precision	u(3)
if (profile_id == 0x42)	
encoding_precision	u(3)
aspect_ratio	u(4)
frame_rate_code	u(4)
bit_rate_lower	u(18)
bit_rate_upper	u(12)
if (profile_id == 0x40 || profile_id == 0x42)	
low_delay	u(1)
temporal_id_enable_flag	u(1)
bbv_buffer_size	u(18)
max_dpb_size_minus1	u(4)
for (i = 0; i < NumOfLayers; i++) {	
rpl1_index_exist_flag[i]	u(1)
rpl1_same_as_rpl0_flag[i]	u(1)
num_ref_pic_list_set[i][0]	ue(v)
for (j = 0; j < NumRefPicListSet[i][0]; j++)	
reference_picture_list_set(i, 0, j)	
if (!Rpl1SameAsRpl0Flag[i]) {	
num_ref_pic_list_set[i][1]	ue(v)

T/AI 109.8-2025

表 13 序列参数集 RBSP 定义（续）

序列参数集RBSP定义	描述符
for (j = 0; j < NumRefPicListSet[i][1]; j++)	
reference_picture_list_set(i, 1, j)	
}	
num_ref_default_active_minus1[i][0]	ue(v)
num_ref_default_active_minus1[i][1]	ue(v)
}	
log2_lcu_size_minus2	u(3)
log2_min_cu_size_minus2	u(2)
log2_max_part_ratio_minus2	u(2)
max_split_times_minus6	u(3)
log2_min_qt_size_minus2	u(3)
log2_max_bt_size_minus2	u(3)
log2_max_eqt_size_minus3	u(2)
weight_quant_enable_flag	u(1)
if (WeightQuantEnableFlag) {	
load_seq_weight_quant_data_flag	u(1)
if (load_seq_weight_quant_data_flag == '1')	
weight_quant_matrix( )	
}	
st_enable_flag	u(1)
sao_enable_flag	u(1)
alf_enable_flag	u(1)
affine_enable_flag	u(1)
smvd_enable_flag	u(1)
ipcm_enable_flag	u(1)
amvr_enable_flag	u(1)
num_of_hmvp_cand	u(4)
umve_enable_flag	u(1)
if ((NumOfHmvpCand != 0) && AmvrEnableFlag)	
emvr_enable_flag	u(1)
intra_pf_enable_flag	u(1)
tscpm_enable_flag	u(1)
dt_enable_flag	u(1)
if (DtEnableFlag)	

T/AI 109.8-2025

表 13 序列参数集 RBSP 定义（续）

序列参数集RBSP定义	描述符
log2_max_dt_size_minus4	u(2)
pbt_enable_flag	u(1)
EipmEnableFlag = 1	
pmc_enable_flag	u(1)
if (TscpmEnableFlag)	
EnhancedTscpmEnableFlag = 1	
iip_enable_flag	u(1)
sawp_enable_flag	u(1)
mipf_enable_flag	u(1)
if (IntraPfEnableFlag)	
IntraPfChromaEnableFlag = 1	
if (UmveEnableFlag)	
UmveEnhancementEnableFlag = 1	
if (AffineEnableFlag)	
AffineUmveEnableFlag = 1	
if (AffineEnableFlag)	
asr_enable_flag	u(1)
awp_enable_flag	u(1)
tmvp_enable_flag	u(1)
if (TmvpEnableFlag) {	
sbtmvp_enable_flag	u(1)
etmvp_mvap_enable_flag	u(1)
}	
dmvr_enable_flag	u(1)
bio_enable_flag	u(1)
bgc_enable_flag	u(1)
inter_pf_enable_flag	u(1)
inter_pc_enable_flag	u(1)
obmc_enable_flag	u(1)
if (StEnableFlag)	
EnhancedStEnableFlag = 1	
sbt_enable_flag	u(1)
ist_enable_flag	u(1)
if (IstEnableFlag)	

T/AI 109.8-2025

表 13 序列参数集 RBSP 定义（续）

序列参数集RBSP定义	描述符
ts_enable_flag	u(1)
cu_dqp_enable_flag	u(1)
srcc_enable_flag	u(1)
maec_enable_flag	u(1)
dbr_enable_flag	u(1)
esao_enable_flag	u(1)
ccsao_enable_flag	u(1)
if (EsaoEnableFlag)	
SaoEnableFlag = 0	
if (AlfEnableFlag)	
ealf_enable_flag	u(1)
if (profile_id == 0x40 || profile_id == 0x42) {	
ibc_enable_flag	u(1)
isc_enable_flag	u(1)
if (IbcEnableFlag || IscEnableFlag)	
num_of_intra_hmvp_cand	u(4)
fimc_enable_flag	u(1)
nn_tools_set_hook	u(8)
if (NnFilterEnableFlag)	
num_of_nn_filter_minus1	ue(v)
}	
if (low_delay == '0')	
output_reorder_delay	u(5)
cross_patch_loop_filter_enable_flag	u(1)
ref_colocated_patch_flag	u(1)
for (i = 0; i < NumOfLayers; i++) {	
stable_patch_flag[i]	u(1)
if (stable_patch_flag[i] == '1') {	
uniform_patch_flag[i]	u(1)
if (uniform_patch_flag [i]== '1') {	
patch_width_minus1[i]	ue(v)
patch_height_minus1[i]	ue(v)
}	
}	

T/AI 109.8-2025

表 13 序列参数集 RBSP 定义（续）

序列参数集RBSP定义	描述符
}	
reserved_bits	u(2)
rbsp_trailing_bits( )	
}	

6.1.4 参考图像队列配置集定义

参考图像队列配置集定义见表 14。

表14 参考图像队列配置集定义

参考图像队列配置集定义	描述符
reference_picture_list_set(layerId, list, rpls) {	
if (LibraryPictureEnableFlag && !LibraryStreamFlag)	
reference_to_library_enable_flag[layerId]	u(1)
num_of_ref_pic[layerId][list][rpls]	ue(v)
for (i = 0; i < NumOfRefPic[layerId][list][rpls]; i++) {	
if (ReferenceToLibraryEnableFlag[layerId])	
library_index_flag[layerId][list][rpls][i]	u(1)
if (LibraryIndexFlag[layerId][list][rpls][i])	
referenced_library_picture_index[layerId][list][rpls][i]	ue(v)
else {	
abs_delta_doi[layerId][list][rpls][i]	ue(v)
if (abs_delta_doi[layerId][list][rpls][i] > 0)	
sign_delta_doi[layerId][list][rpls][i]	u(1)
}	
}	
}	

6.1.5 加权量化数据定义

自定义加权量化矩阵定义见表 15。

表15 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)	

T/AI 109.8-2025

表 15 自定义加权量化矩阵定义（续）

自定义加权量化矩阵定义	描述符
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	
}	
}	
}	
}	

6.1.6 安全参数集 RBSP 定义

安全参数集 RBSP 定义见表 16。

表16 安全参数集 RBSP 定义

安全参数集RBSP定义	描述符
sec_parameter_set_rbsp( ) {	
sec_is_library_flag	u(1)
reserved_para_set_id	u(2)
encryption_enable_flag	u(1)
authentication_enable_flag	u(1)
if (encryption_enable_flag) {	
encryption_unit_mode	u(2)
encryption_level_mode	u(2)
if (encryption_level_mode) {	
encryption_num_minus1	u(8)
encryption_base_byte	u(2)
}	
encryption_type	u(4)
vek_flag	u(1)
iv_flag	u(1)
if (vek_flag) {	
vek_encryption_type	u(4)
evek_length_minus1	u(8)
evek	f(n)
vkek_version_length_minus1	u(8)

T/AI 109.8-2025

表 16 安全参数集 RBSP 定义（续）

安全参数集RBSP定义	描述符
vkek_version	f(n)
}	
if (iv_flag) {	
iv_length_minus1	u(8)
iv	f(n)
}	
}	
if (authentication_enable_flag) {	
authentication_hash_mode	u(1)
hash_type	u(2)
if (!sec_is_library_flag)	
hash_discard_nrap_pictures_flag	u(1)
signature_type	u(2)
signature_fmt	u(2)
hash_period_in_doi_minus1	u(8)
camera_idc	f(152)
}	
if (vek_flag || authentication_enable_flag)	
camera_id	f(160)
rbsp_trailing_bits( )	
}	

T/AI 109.8-2025

6.1.7 补充增强信息 RBSP 定义

6.1.7.1 补充增强信息 RBSP 定义通则

补充增强信息 RBSP 定义、补充增强信息消息定义和补充增强负载定义。分别见表 17、表 18 和表 19。

表17 补充增强信息 RBSP 定义

补充增强信息 RBSP 定义	描述符
sei_rbsp( ) {	
do	
sei_message( )	
while(more_rbsp_data( ))	
rbsp_trailing_bits( )	
}	

表18 补充增强信息消息定义

补充增强信息消息语法定义	描述符
sei_message( ) {	
PayloadType = 0	
while(next_bits(8) == 0xFF) {	
ff_byte /* 应等于 0xFF */	f(8)
PayloadType += 255	
}	
last_payload_type_byte	u(8)
PayloadType += last_payload_type_byte	
PayloadSize = 0	
while(next_bits(8) == 0xFF) {	
ff_byte /* 应等于 0xFF */	f(8)
PayloadSize += 255	
}	
last_payload_size_byte	u(8)
PayloadSize += last_payload_size_byte	
sei_payload(PayloadType, PayloadSize)	
}	

表19 补充增强负载语法定义

补充增强负载语法定义	描述符
sei_payload(PayloadType, PayloadSize) {	
if (PayloadType == 0)	
buffering_period (PayloadSize)	
else if (PayloadType == 1)	
pic_timing (PayloadSize)	
else if (PayloadType == 2)	
spherical_pano_video_parameter_set (PayloadSize)	
else if (PayloadType == 3)	
gmv_parameters (PayloadSize)	
else if (PayloadType == 8)	
roi_parameters (PayloadSize)	
else if (PayloadType == 16)	
user_data (PayloadSize)	
else if (PayloadType == 17)	
temporal_scalability_info (PayloadSize)	
else if (PayloadType == 18)	
air_parameters (PayloadSize)	
else if (PayloadType == 19)	

T/AI 109.8-2025

表 19 补充增强负载语法定义（续）

补充增强负载语法定义	描述符
seq_privacy_parameters (PayloadSize)	
else if (PayloadType == 20)	
pic_privacy_parameters (PayloadSize)	
else if (PayloadType == 25)	
layer_info (PayloadSize)	
else if (PayloadType == 26)	
library_display_info (PayloadSize)	
else if (PayloadType == 27)	
sequence_display_extension (PayloadSize)	
else if (PayloadType == 28)	
mastering_display_and_content_metadata_extension (PayloadSize)	
else if (PayloadType == 29)	
hdr_dynamic_metadata_extension (PayloadSize)	
else if (PayloadType == 30)	
picture_display_extension (PayloadSize)	
else if (PayloadType == 127)	
reserved_version_info (PayloadSize)	
else	
reserved_sei_message(PayloadSize)	
if (!byte_aligned( )) {	
bit_equal_to_one /* 应等于1 */	f(1)
while(!byte_aligned( ))	
bit_equal_to_zero /* 应等于0 */	f(1)
}	
}	

6.1.7.2 感兴趣区域参数负载定义

感兴趣区域参数负载定义见表 20。

表20 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 (PayloadSize) {	
current_picture_roi_num	u(8)
roiIndex = 0	
if (PictureCodingType != 1) {	
prev_picture_roi_num	u(8)
for (i=0; i<PrevPictureROINum; i++) {	

T/AI 109.8-2025

表 20 感兴趣区域参数扩展定义（续）

感兴趣区域参数扩展定义	描述符
roi_skip_run	ue(v)
if (roi_skip_run != '0') {	
for (j=0; j<roi_skip_run; j++) {	
skip_roi_mode[i+j]	u(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	
}	
else {	
roi_axisx_delta	se(v)
roi_axisy_delta	se(v)
roi_width_delta	se(v)
roi_height_delta	se(v)
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)
roi_axisy	u(6)
roi_width	u(6)
roi_height	u(6)
ROIInfo[roiIndex]->asisx = roi_axisx	
ROIInfo[roiIndex]->asisy = roi_axisy	
ROIInfo[roiIndex]->width = roi_width	
ROIInfo[roiIndex]->height = roi_height	
roiIndex++	
}	

T/AI 109.8-2025

表 20 感兴趣区域参数扩展定义（续）

感兴趣区域参数扩展定义	描述符
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	
}	
}	

6.1.7.3 用户数据负载定义

用户数据负载定义见表21。

表21 用户数据负载定义

用户数据负载定义	描述符
user_data (PayloadSize) {	
for (i=0; i<PayloadSize; i++) {	
user_data	b(8)
}	
}	

6.1.7.4 高层序列隐私保护区域负载定义

序列级隐私保护区域负载定义见表 22。seq_privacy_parameters( )只应出现在随机访问点图像和非显示知识图像的访问单元。同一访问单元内 seq_privacy_parameters( )的位流顺序应在 pic_privacy_parameters( )之前。

知识图像与其作用域的 seq_privacy_parameters( )应相同。若解码过程跳过了 seq_privacy_parameters( )解析，则也应跳过 pic_privacy_parameters( )的解析。

表22 序列级隐私保护区域负载定义

序列级隐私保护区域负载定义	描述符
seq_privacy_parameters (PayloadSize) {	
privacy_level_num_minus2	u(3)
privacy_category_num_minus1	u(3)
privacy_coding_mode_index	u(2)
if (PrivacyCodingModeIndex== 'RECT_INFO_MODE')	
log2_min_rect_info_mode_size	u(3)
for (i = 0; i < PrivacyLevelNum; i++) {	
for (j = 0; j < PrivacyCategoryNum; j++) {	

T/AI 109.8-2025

表 22 序列级隐私保护区域负载定义（续）

序列级隐私保护区域负载定义	描述符
if (i==0)	
PrivacyLevelCategoryEnable[i][j] = (j==0) ? 1 : 0	
else if (i == PrivacyLevelNum-1)	
PrivacyLevelCategoryEnable[i][j] = 1	
else	
privacy_level_category_enable[i][j]	u(1)
}	
}	
}	

6.1.7.5 高层图像隐私保护区域负载定义

图像级隐私保护区域负载定义见表 23。

表23 图像级隐私保护区域负载定义

图像级隐私保护区域负载定义	描述符
pic_privacy_parameters(PayloadSize){	
if (PrivacyCodingModeIndex == 0) {	
for (i=1; i<PrivacyCategoryNum; i++) {	
privacy_region_num[i]	u(8)
for (j=0; j<PrivacyRegionNum[i]; j++){	
privacy_left	u(16)
privacy_top	u(16)
privacy_width	u(16)
privacy_height	u(16)
}	
}	
}	
}	

6.1.7.6 时域分层信息负载定义

时域分层信息负载定义见表 24。

T/AI 109.8-2025

表24 时域分层信息负载定义

时域分层负载定义	描述符
temporal_scalability_info (payloadSize) {	
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)
temporal_bit_rate_upper[i]	u(12)
}	
}	

6.1.7.7 自适应帧内刷新参数负载定义

自适应帧内刷新参数负载定义见表 25。

表25 自适应帧内刷新参数负载定义

自适应帧内刷新参数负载定义	描述符
air_parameters (payloadSize) {	
air_bound_x	u(16)
air_bound_y	u(16)
}	

6.1.7.8 编码防抖参数负载定义

编码防抖参数负载定义见表 26。

表26 编码防抖参数负载定义

编码防抖参数负载定义	描述符
gmv_parameters (payloadSize) {	
gmv_translational_x	i(8)
gmv_translational_y	i(8)
gmv_translational_z	i(8)
gmv_rotational_x	i(8)
gmv_rotational_y	i(8)
gmv_rotational_z	i(8)
}	

6.1.7.9 分层信息负载定义

分层信息参数负载定义见表 27。

T/AI 109.8-2025

表27 分层信息负载定义

分层信息负载定义	描述符
layer_info (payloadSize) {	
sei_num_of_layers_minus1	u(2)
for (i=0; i<=sei_num_of_layers_minus1; i++)	
sei_layer_id[i]	u(2)
}	

6.1.7.10 显示知识图像显示信息负载定义

显示知识图像显示信息负载定义见表 28。

表28 显示知识图像显示信息负载定义

显示知识图像显示信息负载定义	描述符
library_display_info (payloadSize) {	
library_display_flag	u(1)
reserved_bits	r(7)
}	

6.1.7.11 缓存周期负载定义

缓存周期负载 buffering_period(PayloadSize)定义应符合 GB/T 25724-2017 中 E.1.2 的规定。

6.1.7.12 图像定时负载定义

图像定时负载 pic_timing(PayloadSize)定义应符合 GB/T 25724-2017 中 E.1.3 的规定。

6.1.7.13 全景视频参数负载定义

全景视频参数负载 spherical_pano_video_parameter_set(PayloadSize)定义应符合 GB/T 25724-2017 中 E.1.4 的规定。

6.1.7.14 序列显示扩展负载定义

序列显示扩展负载定义见表 29。

表29 序列显示扩展负载定义

序列显示扩展负载定义	描述符
sequence_display_extension (payloadSize) {	
video_format	u(3)
sample_range	u(1)
colour_description	u(1)
if (colour_description) {	

T/AI 109.8-2025

表 29 序列显示扩展负载定义（续）

序列显示扩展负载定义	描述符
colour_primaries	u(8)
transfer_characteristics	u(8)
matrix_coefficients	u(8)
}	
display_horizontal_size	u(14)
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)
}	
}	

6.1.7.15 目标显示设备和内容元数据扩展负载定义

目标显示设备和内容元数据扩展负载定义见表 30。

表30 目标显示设备和内容元数据扩展负载定义

目标显示设备和内容元数据扩展负载定义	描述符
mastering_display_and_content_metadata_extension(payloadSize) {	
for (c=0; c<3; c++) {	
display_primaries_x[c]	u(16)
display_primaries_y[c]	u(16)
}	
white_point_x	u(16)
white_point_y	u(16)
max_display_mastering_luminance	u(16)
min_display_mastering_luminance	u(16)
max_content_light_level	u(16)
max_picture_average_light_level	u(16)
reserved_bits	r(16)
}	

6.1.7.16 高动态范围图像扩展负载定义

高动态范围图像扩展负载定义见表 31。

T/AI 109.8-2025

表31 高动态范围图像扩展负载定义

高动态范围图像扩展负载定义	描述符
hdr_dynamic_metadata_extension(payloadSize) {	
hdr_dynamic_metadata_type	u(4)
for (i=1; i< payloadSize; i++) {	
extension_data_byte	u(8)
}	
reserved_bits	r(4)
}	

6.1.7.17 图像显示扩展负载定义

图像显示扩展负载定义见表 32。

表32 图像显示扩展负载定义

图像显示扩展负载定义	描述符
picture_display_extension(payloadSize) {	
picture_centre_horizontal_offset	i(16)
picture_centre_vertical_offset	i(16)
}	

6.1.8 图像参数集 RBSP 定义

图像参数集 RBSP 语法定义见表 33。

表33 图像参数集 RBSP 定义

图像参数集RBSP定义	描述符
pic_parameter_set_rbsp ( ) {	
fixed_picture_qp_flag	u(1)
if (!FixedPictureQpFlag && CuDqpEnableFlag) {	
cu_delta_qp_flag	u(1)
if (CuDeltaQpFlag)	
cu_qp_group_size_log2_minus3	u(2)
}	
deblocking_filter_disable_flag	u(1)
if (!DeblockingFilterDisableFlag) {	
deblocking_filter_type	u(1)
deblocking_filter_parameter_flag	u(1)
}	
if (WeightQuantEnableFlag) {	

T/AI 109.8-2025

表 33 图像参数集 RBSP 定义（续）

图像参数集RBSP定义	描述符
picture_weight_quant_enable_flag	u(1)
if (PictureWeightQuantEnableFlag) {	
picture_weight_quant_data_index	u(2)
if (picture_weight_quant_data_index == '01') {	
reserved_bits	u(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 (picture_weight_quant_data_index == '10')	
weight_quant_matrix( )	
}	
}	
if (IbcEnableFlag)	
picture_ibc_enable_flag	u(1)
if (IscEnableFlag) {	
picture_isc_os_enable_flag	u(1)
picture_isc_evs_ubvs_enable_flag	u(1)
}	
if (FimcEnableFlag)	
picture_fimc_enable_flag	u(1)
if (AffineEnableFlag)	
affine_subblock_size_flag	u(1)
if (AwpEnableFlag || SawpEnableFlag)	
picture_awp_refine_index	u(1)
rbsp_trailing_bits( )	
}	

6.1.9 图像头 RBSP 定义

图像头 RBSP 语法定义见表 34。

T/AI 109.8-2025

表34 图像头 RBSP 定义

图像头RBSP定义	描述符
picture_header_rbsp( ) {	
decode_order_index	u(8)
if (LibraryStreamFlag) {	
library_picture_index	ue(v)
library_patch_num_minus1	u(8)
}	
picture_coding_type	u(2)
if (ReferencePictureResamplingEnableFlag)	
picture_rpr_flag	u(1)
if (profile_id != 0x30 && TmvpEnableFlag)	
picture_tmvp_enable_flag	u(1)
if (PictureCodingType != 1)	
random_access_decodable_flag	u(1)
bbv_delay	u(32)
if (PictureCodingType == 1) {	
time_code_flag	u(1)
if (time_code_flag == '1')	
time_code	u(24)
}	
if (low_delay == '0')	
picture_output_delay	ue(v)
if (low_delay == '1')	
bbv_check_times	ue(v)
ref_pic_list_set_flag[0]	u(1)
if (RefPicListSetFlag[0]) {	
if (NumRefPicListSet[LayerId][0] > 1)	
ref_pic_list_set_index[0]	ue(v)
}	
else	
reference_picture_list_set(LayerId, 0, NumRefPicListSet[LayerId][0])	
if (Rpl1IndexExistFlag)	
ref_pic_list_set_flag[1]	u(1)
if (RefPicListSetFlag[1]) {	
if (Rpl1IndexExistFlag && NumRefPicListSet[LayerId][1] > 1)	
ref_pic_list_set_index[1]	ue(v)
}	
else	
reference_picture_list_set(LayerId, 1, NumRefPicListSet[LayerId][1])	

T/AI 109.8-2025

表 34 图像头 RBSP 定义（续）

图像头RBSP定义	描述符
if (PictureCodingType != 1) {	
num_ref_active_override_flag	u(1)
if (num_ref_active_override_flag == '1') {	
num_ref_active_minus1[0]	ue(v)
if (PictureCodingType == 3)	
num_ref_active_minus1[1]	ue(v)
}	
}	
picture_qp	u(7)
if (PrivacyModeIndex == 2 || PrivacyModeIndex == 3)	
picture_privacy_enable_flag	
if (PictureCodingType == 2)	
reserved_bits	u(1)
if (!DeblckingFilterDisableFlag) {	
if (deblocking_filter_parameter_flag == '1') {	
alpha_c_offset	se(v)
beta_offset	se(v)
}	
if (DbrEnableFlag) {	
picture_dbr_v_enable_flag	u(1)
if (PictureDbrVEnableFlag) {	
dbr_v_offset0_minus1	u(2)
dbr_v_offset1_minus1	u(2)
}	
picture_alt_dbr_v_enable_flag	u(1)
if (PictureAltDbrVEnableFlag) {	
dbr_v_alt_offset0_minus1	u(2)
dbr_v_alt_offset1_minus1	u(2)
}	
if (PictureDbrVEnableFlag || PictureAltDbrVEnableFlag)	
dbr_v_threshold_minus1	u(1)
picture_dbr_h_enable_flag	u(1)
if (PictureDbrHEnableFlag) {	
dbr_h_offset0_minus1	u(2)
dbr_h_offset1_minus1	u(2)
}	
picture_alt_dbr_h_enable_flag	u(1)
if (PictureAltDbrHEnableFlag) {	

T/AI 109.8-2025

表 34 图像头 RBSP 定义（续）

图像头RBSP定义	描述符
dbr_h_alt_offset0_minus1	u(2)
dbr_h_alt_offset1_minus1	u(2)
}	
if (PictureDbrHEnableFlag || PictureAltDbrHEnableFlag)	
dbr_h_threshold_minus1	u(1)
}	
}	
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 (EsaoEnableFlag)	
esao_parameter_picture_header_set( )	
if (CcsaoEnableFlag)	
ccsao_parameter_picture_header_set( )	
if (AlfEnableFlag) {	
for (compIndex=0; compIndex<3; compIndex++)	
picture_alf_enable_flag[compIndex]	u(1)
if (PictureAlfEnableFlag[0] == 1 || PictureAlfEnableFlag[1] == 1 || PictureAlfEnableFlag[2] == 1)	
alf_parameter_set( )	
}	
if (PmcEnableFlag)	
picture_pmc_param_index	u(1)
if (UmveEnhancementEnableFlag && (PictureCodingType != 1))	
picture_umve_step_set_index	u(1)
if (ObmcEnableFlag && (PictureCodingType == 3))	
picture_obmc_blending_set_index	u(1)
if ((PictureCodingType != 1) && InterPcEnableFlag)	
picture_inter_pc_skip_flag	u(1)
if (TsEnableFlag)	
picture_ts_enable_flag	u(1)
if (NnFilterEnableFlag) {	
for (compIdx=0; compIdx<3; compIdx++) {	
picture_nn_filter_enable_flag[comIdx]	u(1)
if (PictureNnFilterEnableFlag[comIdx]) {	
picture_nn_filter_adaptive_flag[comIdx]	u(1)
if (PictureNnFilterAdaptiveFlag[comIdx] == 0) {	

T/AI 109.8-2025

表 34 图像头 RBSP 定义（续）

图像头RBSP定义	描述符
if (NumOfNnFilter > 1)	
picture_nn_filter_set_index[comIdx]	ue(v)
}	
}	
}	
}	
rbsp_trailing_bits( )	
}	

6.1.10 片 RBSP 定义

若 UserPermission 等于 1，且 PicturePrivacyEnableFlag 的值为 1，完整片定义由 patch_data_rbsp( )和 privacy_patch_rbsp( )组成，否则完整片定义内容等于片 RBSP 定义见表 35。

其中 privacy_patch_rbsp( )定义由连续的 privacy_unit_data( )定义组成。

注：当 PrivacyModeIndex 的值等于 2 且 UserPermission 的值为 1 且 PrivacyLevel 的值等于 1 且 IsPrivacyFlag 的值等于 0 时，coding_unit 解析得到的语法元素的值仅参与确定量化参数(见 8.6.2)。

表35 片 RBSP 定义

片RBSP定义	描述符
patch_data_rbsp( ) {	
patch_index	u(8)
if (!FixedPictureQpFlag) {	
fixed_patch_qp_flag	u(1)
patch_qp	u(7)
}	
if (SaoEnableFlag) {	
for (compIndex=0; compIndex<3; compIndex++)	
patch_sao_enable_flag[compIndex]	u(1)
}	
while(!byte_aligned( ))	
aec_byte_alignment_bit	f(1)
do {	
if (!FixedQP) {	
if (!CuDeltaQpFlag)	
lcu_qp_delta	ae(v)
PreviousDeltaQP = lcu_qp_delta	
}	
if (SaoEnableFlag) {	

T/AI 109.8-2025

表 35 片 RBSP 定义（续）

片RBSP定义	描述符
if (PatchSaoEnableFlag[0] || PatchSaoEnableFlag[1] || PatchSaoEnableFlag[2]) {	
if (MergeFlagExist)	
sao_merge_type_index	ae(v)
if (SaoMergeMode == 'SAO_NON_MERGE') {	
for (compIndex=0; compIndex<3; compIndex++) {	
if (PatchSaoEnableFlag[compIndex]) {	
sao_mode[compIndex]	ae(v)
if (SaoMode[compIndex] == 'SAO_Interval') {	
for (j=0; j<MaxOffsetNumber; j++) {	
sao_interval_offset_abs[compIndex][j]	ae(v)
if (SaoIntervalOffsetAbs[compIndex][j])	
sao_interval_offset_sign[compIndex][j]	ae(v)
}	
sao_interval_start_pos[compIndex]	ae(v)
sao_interval_delta_pos_minus2[compIndex]	ae(v)
}	
if (SaoMode[compIndex] == 'SAO_Edge') {	
for (j=0; j<MaxOffsetNumber; j++)	
sao_edge_offset[compIndex][j]	ae(v)
sao_edge_type[compIndex]	ae(v)
}	
}	
}	
}	
}	
}	
if (EsaoEnableFlag) {	
for(compIndex=0; compIndex<3; compIndex++) {	
if (PictureEsaoEnableFlag[compIndex] && PictureEsaoLcuControlFlag[compIndex]) {	
esao_lcu_enable_flag[compIndex][LcuIndex]	ae(v)
if (EsaoLcuEnableFlag[compIndex][LcuIndex] && PictureEsaoSetNum[compIndex]>1) {	
esao_lcu_set_index[compIndex][LcuIndex]	ae(v)
}	
}	
}	
}	
if (CcsaoEnableFlag) {	

T/AI 109.8-2025

表 35 片 RBSP 定义（续）

片RBSP定义	描述符
for(compIndex=1; compIndex<3; compIndex++) {	
if (PictureCcsaoEnableFlag[compIndex] && PictureCcsaoLcuControlFlag[compIndex]) {	
ccsao_lcu_enable_flag[compIndex][LcuIndex]	ae(v)
if (CcsaoLcuEnableFlag[compIndex][LcuIndex] && PictureCcsaoSetNum[compIndex]>1){	
ccsao_lcu_set_index[compIndex][LcuIndex]	ae(v)
}	
}	
}	
}	
for (compIndex=0; compIndex<3; compIndex++) {	
if (PictureAlfEnableFlag[compIndex] == 1)	
alf_lcu_enable_flag[compIndex][LcuIndex]	ae(v)
}	
for (compIdx=0; compIdx<3; compIdx++) {	
if (pictureNnFilterAdaptiveFlag[compIdx]) {	
nn_filter_lcu_enable_flag[compIdx][LcuIdx]	ae(v)
if (NnFilterLcuEnableFlag[compIdx][LcuIdx]) {	
if (NumOfNnFilter > 1)	
nn_filter_lcu_set_index[compIdx][LcuIdx]	ae(v)
}	
}	
}	
x0 = (LcuIndex % PictureWidthInLcu) * LcuSize	
y0 = (LcuIndex / PictureWidthInLcu) * LcuSize	
Component = 0	
coding_unit_tree(x0, y0, 0, 1<<LcuSizeInBit, 1<<LcuSizeInBit, 1, 'PRED_No_Constraint')	
if (PicturePrivacyEnableFlag && UserPermission == 1)	
privacy_unit_data(x0, y0, 1<<LcuSizeInBit, 1<<LcuSizeInBit, 'PRED_No_Constraint', 1, 1)	
aec_lcu_stuffing_bit	ae(v)
} while (! is_end_of_patch( ))	
rbsp_trailing_bits( )	
if (PicturePrivacyEnableFlag && UserPermission == 1)	
privacy_unit_data(x0, y0, 1<<LcuSizeInBit, 1<<LcuSizeInBit, 'PRED_No_Constraint', 1, 2)	
}	

T/AI 109.8-2025

6.1.11 编码树定义

编码树定义见表 36。

表36 编码树定义

编码树定义	描述符
coding_unit_tree(x0, y0, split, width, height, qt, mode) {	
isBoundary = ((x0+width) > PicWidthInLuma) || ((y0+height) > PicHeightInLuma)	
rightBoundary = ((x0+width) > PicWidthInLuma) && ((y0+height) <= PicHeightInLuma)	
bottomBoundary = ((x0 + width) <= PicWidthInLuma) && ((y0 + height) > PicHeightInLuma)	
allowNoSplit = 0	
allowSplitQt = 0	
allowSplitBtVer = 0	
allowSplitBtHor = 0	
allowSplitEqtVer = 0	
allowSplitEqtHor = 0	
if (isBoundary) {	
allowNoSplit = 0	
if ((PictureCodingType == 1 || PictureRprFlag == 1) && (width > 64) && (height > 64)) {	
allowSplitQt = 1	
allowNoSplit = 1	
}	
else if (((width == 64 && height > 64) || (height == 64 && width > 64)) && ((width <= MaxBtSize) && (height <= MaxBtSize))) {	
allowSplitBtHor = 1	
allowSplitBtVer = 1	
}	
else if ((!rightBoundary && ! bottomBoundary) || ((width > MaxBtSize) || (height > MaxBtSize)))	
allowSplitQt = 1	
else if (rightBoundary)	
allowSplitBtVer = 1	
else if (bottomBoundary)	
allowSplitBtHor = 1	
}	
else {	
if (((width == 64) && (height > 64)) || ((height == 64) && (width > 64))) {	
if((width <= MaxBtSize) && (height <= MaxBtSize)){	
allowSplitBtHor = 1	
allowSplitBtVer = 1	
}	
allowNoSplit = 1	
}	
else if (split >= MaxSplitTimes)	

T/AI 109.8-2025

表 36 编码树定义（续）

编码树定义	描述符
allowNoSplit = 1	
else if ((PictureCodingType == 1 || PictureRprFlag == 1) && (width == 128) && (height == 128)) {	
allowSplitQt = 1	
allowNoSplit = 1	
}	
else {	
if ((width <= height * MaxPartRatio) && (height <= width * MaxPartRatio))	
allowNoSplit = 1	
if ((width > MinQtSize) && qt)	
allowSplitQt = 1	
if ((width <= MaxBtSize) && (height <= MaxBtSize) && (width > MinBtSize) && (height < MaxPartRatio*width))	
allowSplitBtVer = 1	
if ((width <= MaxBtSize) && (height <= MaxBtSize) && (height > MinBtSize) && (width < MaxPartRatio*height))	
allowSplitBtHor = 1	
if ((width <= MaxEqtSize) && (height <= MaxEqtSize) && (height >= MinEqtSize*2) && (width >= MinEqtSize*4) && (height*4 <= MaxPartRatio*width))	
allowSplitEqtVer = 1	
if ((width <= MaxEqtSize) && (height <= MaxEqtSize) && (width >= MinEqtSize*2) && (height >= MinEqtSize*4) && (width*4 <= MaxPartRatio*height))	
allowSplitEqtHor = 1	
}	
}	
allowSplitBt = allowSplitBtVer || allowSplitBtHor	
allowSplitEqt = allowSplitEqtVer || allowSplitEqtHor	
if (allowSplitQt && (allowNoSplit || allowSplitBt || allowSplitEqt))	
qt_split_flag	ae(v)
if (!QtSplitFlag) {	
if (allowNoSplit && (allowSplitBt || allowSplitEqt))	
bet_split_flag	ae(v)
if (BetSplitFlag) {	
if (allowSplitBt && allowSplitEqt)	
bet_split_type_flag	ae(v)
if ((!BetSplitTypeFlag && allowSplitBtHor && allowSplitBtVer) || (BetSplitTypeFlag && allowSplitEqtHor && allowSplitEqtVer))	
bet_split_dir_flag	ae(v)
}	

T/AI 109.8-2025

表 36 编码树定义（续）

编码树定义	描述符
}	
if ((mode == 'PRED_No_Constraint') && (PictureCodingType != 1) && ((((BetSplitFlag && ! BetSplitTypeFlag) || QtSplitFlag) && (width * height == 64)) || (BetSplitTypeFlag && (width * height == 128)))) {	
root_cu_constraint	ae(v)
modeChild = root_cu_constraint ? 'PRED_Intra_Only' : 'PRED_Inter_Only'	
}	
else	
modeChild = mode	
if (ChildSizeOccur4) {	
if (Component == 0) {	
LumaWidth = width	
LumaHeight = height	
Component = 1	
}	
}	
if ((!FixedQP) && CuDeltaQpFlag && (width * height >= CuQpGroupAreaSize)) {	
NumDeltaQp = 0	
NumDeltaQpPrivacy = 0	
CuQpGroupX = x0	
CuQpGroupY = y0	
}	
if (BlockSplitMode == 'SPLIT_QT') {	
QtWidth = width / 2	
QtHeight = height / 2	
x1 = x0 + QtWidth	
y1 = y0 + QtHeight	
coding_unit_tree(x0, y0, split+1, QtWidth, QtHeight, 1, modeChild)	
if (x1 < PicWidthInLuma)	
coding_unit_tree(x1, y0, split+1, QtWidth, QtHeight, 1, modeChild)	
if (y1 < PicHeightInLuma)	
coding_unit_tree(x0, y1, split+1, QtWidth, QtHeight, 1, modeChild)	
if ((x1 < PicWidthInLuma) && (y1 < PicHeightInLuma))	
coding_unit_tree(x1, y1, split+1, QtWidth, QtHeight, 1, modeChild)	
}	
else if (BlockSplitMode == 'SPLIT_BT_VER') {	
x1 = x0 + width / 2	
coding_unit_tree(x0, y0, split+1, width/2, height, 0, modeChild)	

T/AI 109.8-2025

表 36 编码树定义（续）

编码树定义	描述符
if (x1 < PicWidthInLuma)	
coding_unit_tree(x1, y0, split+1, width/2, height, 0, modeChild)	
}	
else if (BlockSplitMode == 'SPLIT_BT_HOR') {	
y1 = y0 + height / 2	
coding_unit_tree(x0, y0, split+1, width, height/2, 0, modeChild)	
if (y1 < PicHeightInLuma)	
coding_unit_tree(x0, y1, split+1, width, height/2, 0, modeChild)	
}	
else if (BlockSplitMode == 'SPLIT_EQT_VER') {	
x1 = x0 + width / 4	
x2 = x0 + (3 * width / 4)	
y1 = y0 + height / 2	
coding_unit_tree(x0, y0, split+1, width/4, height, 0, modeChild)	
coding_unit_tree(x1, y0, split+1, width/2, height/2, 0, modeChild)	
coding_unit_tree(x1, y1, split+1, width/2, height/2, 0, modeChild)	
coding_unit_tree(x2, y0, split+ 1, width/4, height, 0, modeChild)	
}	
else if (BlockSplitMode == 'SPLIT_EQT_HOR') {	
x1 = x0 + width / 2	
y1 = y0 + height / 4	
y2 = y0 + (3 * height / 4)	
coding_unit_tree(x0, y0, split+1, width, height/4, 0, modeChild)	
coding_unit_tree(x0, y1, split+1, width/2, height/2, 0, modeChild)	
coding_unit_tree(x1, y1, split+1, width/2, height/2, 0, modeChild)	
coding_unit_tree(x0, y2, split+1, width, height/4, 0, modeChild)	
}	
else {	
if (PicturePrivacyEnableFlag)	
privacy_level	ae(v)
if (Component == 0)	
componentType = 'COMPONENT_LUMACHROMA'	
else	
componentType = 'COMPONENT_LUMA'	
IsPrivacyFlag = 0	
if (PrivacyLevel == 0)	
coding_unit(x0, y0, width, height, mode, componentType)	
else {	

T/AI 109.8-2025

表 36 编码树定义（续）

编码树定义	描述符
if (PrivacyModeIndex == 2) {	
coding_unit(x0, y0, width, height, mode, componentType)	
PriorCuModeLuma = IntraCuFlag ? 1 : 0	
}	
if (UserPermission == 1)	
privacy_unit_data(x0, y0, width, height, mode, componentType, 0)	
}	
}	
if (BlockSplitMode != 'NO_SPLIT' && ((LumaWidth == width) && (LumaHeight == height) && ChildSizeOccur4) {	
IsPrivacyFlag = 0	
if (PrivacyLevel == 0)	
coding_unit(x0, y0, width, height, 'PRED_No_Constraint', 'COMPONENT_Chroma')	
else {	
if (PrivacyModeIndex == 2) {	
coding_unit(x0, y0, width, height, 'PRED_No_Constraint', 'COMPONENT_Chroma')	
}	
if (UserPermission == 1)	
privacy_unit_data(x0, y0, width, height, 'PRED_No_Constraint', 'COMPONENT_Chroma', 0)	
}	
Component = 0	
}	
}	

T/AI 109.8-2025

6.1.12 隐私编码单元定义

隐私编码单元定义见表 37。

表37 隐私编码单元定义

隐私编码单元定义	描述符
privacy_unit_data(x0, y0, width, height, mode, Componentype, endType) {	
IsPrivacyFlag = 1	
if (endType == 1) /* LCU end */	
aec_lcu_stuffing_bit	ae(v)
else if (endType == 2) /* RBSP end */	
rbsp_trailing_bits( )	

表 37 隐私编码单元定义（续）

隐私编码单元定义	描述符
else	
coding_unit(x0, y0, width, height, mode, Componentype)	
IsPrivacyFlag = 0	
}	

6.1.13 编码单元定义

编码单元定义见表 38。

表38 编码单元定义

编码单元定义	描述符
coding_unit(x0, y0, width, height, mode, component) {	
CodingTreeComponent = component	
if (component == 'COMPONENT_Chroma') {	
if ((PriorCuMode != 0) && (chroma_format != '00'))	
intra_chroma_pred_mode_index	ae(v)
if ((TscpmEnableFlag || PmcEnableFlag) && (IntraChromaPredModeIndex == 1)) {	
if (EnhancedTscpmEnableFlag || PmcEnableFlag)	
chroma_eipm_index	ae(v)
if (PmcEnableFlag && TscpmEnableFlag && (EnhancedTscpmEnableFlag || (ChromaEipmIndex == 0)))	
chroma_pmc_index	ae(v)
if (ChromaPmcIndex)	
chroma_pmc_param_index	ae(v)
}	
NumOfTransBlocks = 3	
ctp_y[0] = 0	
CuCtp += ctp_y[0]	
if (IntraChromaPredMode != 'Intra_Chroma_PCM') {	
if (IntraChromaPredMode == 'Intra_Chroma_PMC' || IntraChromaPredMode == 'Intra_Chroma_PMC_LT' || IntraChromaPredMode == 'Intra_Chroma_PMC_T' || IntraChromaPredMode == 'Intra_Chroma_PMC_L' || IntraChromaPredMode == 'Intra_Chroma_EPMC' || IntraChromaPredMode == 'Intra_Chroma_EPMC_LT' || IntraChromaPredMode == 'Intra_Chroma_EPMC_L' || IntraChromaPredMode == 'Intra_Chroma_EPMC_T' || IntraChromaPredMode == 'Intra_Chroma_EPMC2' || IntraChromaPredMode == 'Intra_Chroma_EPMC2_LT' || IntraChromaPredMode == 'Intra_Chroma_EPMC2_L' || IntraChromaPredMode == 'Intra_Chroma_EPMC2_T')	
CuCtp += (1 << 1)	
else {	

T/AI 109.8-2025

表 38 编码单元定义（续）

编码单元定义	描述符
ctp_u	ae(v)
CuCtp += (ctp_u << 1)	
}	
ctp_v	ae(v)
CuCtp += (ctp_v << 2)	
}	
if (EnhancedStEnableFlag && IntraCuFlag && (IntraChromaPredMode != 'Intra_Chroma_PCM') && (! IntraPfFlag) && (! IipFlag)) {	
if (ctp_u || ctp_v)	
chroma_st_flag	ae(v)
}	
IstTuFlag = 0	
isIstApply = 0	
for (i=NumOfTransBlocks–2; i<NumOfTransBlocks; i++) {	
IsPcmMode[i] = (IntraChromaPredMode == 'Intra_Chroma_PCM')	
IsChroma = 0	
if (i == NumOfTransBlocks -1 || i == NumOfTransBlocks -2)	
IsChroma = 1	
block(i, width, height, CuCtp, IsChroma, IsPcmMode[i], isIstApply, IntraCuFlag, component)	
}	
}	
else {	
if (PictureCodingType != 1) {	
if (mode != 'PRED_Intra_Only')	
skip_flag	ae(v)
AllowAwp = AwpEnableFlag && (width >= 8) && (height >= 8) && (width <= 64) && (height <= 64)	
AllowEtmvp = EtmvpMvapEnableFlag && PictureTmvpEnableFlag && (width >= 8) && (height >= 8)	
if (SkipFlag) {	
if (UmveEnableFlag || AllowAwp || AllowEtmvp)	
advanced_pred_flag	ae(v)
if (AdvancedPredFlag && AllowEtmvp && (UmveEnableFlag || AllowAwp))	
etmvp_flag	ae(v)
if (AdvancedPredFlag && (! EtmvpFlag) && UmveEnableFlag && AllowAwp)	
awp_flag	ae(v)
if (AffineEnableFlag && (! AdvancedPredFlag) && (width >= 16) && (height >= 16))	
affine_flag	ae(v)

T/AI 109.8-2025

表 38 编码单元定义（续）

编码单元定义	描述符
if (AffineUmveEnableFlag && AffineFlag)	
affine_umve_flag	ae(v)
if (InterPcEnableFlag && PictureInterPcSkipFlag && width * height >= 64 && width <= 128 && height <= 128 && !AffineFlag && !AwpFlag &&!EtmvpFlag) {	
inter_pc_flag	ae(v)
if (InterPcFlag)	
inter_pc_index	ae(v)
}	
}	
if (!SkipFlag) {	
if (mode != 'PRED_Intra_Only')	
direct_flag	ae(v)
if (DirectFlag) {	
if (UmveEnableFlag || AllowAwp || AllowEtmvp)	
advanced_pred_flag	ae(v)
if (AdvancedPredFlag && AllowEtmvp && (UmveEnableFlag || AllowAwp))	
etmvp_flag	ae(v)
if (AdvancedPredFlag && (! EtmvpFlag) && UmveEnableFlag && AllowAwp)	
awp_flag	ae(v)
if (AffineEnableFlag && (! AdvancedPredFlag) && (width >= 16) && (height >= 16))	
affine_flag	ae(v)
if (AffineUmveEnableFlag && AffineFlag)	
affine_umve_flag	ae(v)
if (InterPfEnableFlag && (! AwpFlag) && (! AffineFlag) && (! EtmvpFlag) && (width * height >= 64) && (width <= 64) && (height <= 64)) {	
inter_pf_flag	ae(v)
if (InterPfFlag)	
inter_pf_index	ae(v)
}	
if (InterPcEnableFlag && width * height >= 64 && width <= 128 && height <= 128 && !AffineFlag && !AwpFlag &&!EtmvpFlag && !InterPfFlag) {	
inter_pc_flag	ae(v)
if (InterPcFlag)	
inter_pc_index	ae(v)
}	
}	
if ((!DirectFlag) && (mode == 'PRED_No_Constraint'))	

T/AI 109.8-2025

表 38 编码单元定义（续）

编码单元定义	描述符
intra_cu_flag	ae(v)
}	
}	
PartSize = 'SIZE_2Mx2N'	
if (!SkipFlag && ! DirectFlag) {	
ibcAllowedFlag = PictureIbcEnableFlag && (mode == 'PRED_No_Constraint' ‖ mode == 'PRED_Intra_Only') && IntraCuFlag && width <= 64 && height <= 64	
iscAllowedFlag = PictureIscEnableFlag && (mode == 'PRED_No_Constraint' ‖ mode == 'PRED_Intra_Only') && IntraCuFlag && width <=32 && height <= 32	
if (ibcAllowedFlag && iscAllowedFlag) {	
isc_ibc_cu_flag	ae(v)
if (IscIbcCuFlag)	
ibc_cu_flag	ae(v)
}	
else if ((!ibcAllowedFlag) && iscAllowedFlag)	
isc_ibc_cu_flag	ae(v)
else if (ibcAllowedFlag && (! iscAllowedFlag))	
ibc_cu_flag	ae(v)
if (IbcCuFlag) {	
if (HbvpEmptyFlag && NumOfIntraHmvpCand > 0)	
cbvp_index	ae(v)
if (PictureIbcEnableFlag)	
abvr_index	ae(v)
mv_diff_x_abs_bv	ae(v)
if (MvDiffXAbsBv)	
mv_diff_x_sign_bv	ae(v)
mv_diff_y_abs_bv	ae(v)
if (MvDiffYAbsBv)	
mv_diff_y_sign_bv	ae(v)
}	
if (IscCuFlag) {	
if (PictureIscEvsUbvsEnableFlag && PictureIscOsEnableFlag)	
isc_subtype_flag	ae(v)
if (!IscSubtypeFlag)	
isc_ordinary_sp_coding_unit(x0, y0, width, height, component)	
else	
isc_evs_ubvs_coding_unit(x0, y0, width, height, component)	
}	

T/AI 109.8-2025

表 38 编码单元定义（续）

编码单元定义	描述符
}	
if (SawpEnableFlag && IntraCuFlag && (width >= 8) && (height >= 8) && (width <= 32) && (height <= 32) && (component != 'COMPONENT_CHROMA') && ! IbcCuFlag && ! IscCuFlag) {	
sawp_flag	ae(v)
if (SawpFlag) {	
sawp_index	ae(v)
sawp_pred_mode0_index	ae(v)
sawp_pred_mode1_index	ae(v)
}	
}	
if (DtEnableFlag && IntraCuFlag && ! SawpFlag && ! IbcCuFlag && ! IscCuFlag) {	
allowDtHorSplit = (height >= DtMinSize) && (height <= DtMaxSize) && (width / height < 4) && (width <= DtMaxSize)	
allowDtVerSplit = (width >= DtMinSize) && (width <= DtMaxSize) && (height / width < 4) && (height <= DtMaxSize)	
if (allowDtHorSplit || allowDtVerSplit) {	
dt_split_flag	ae(v)
if (DtSplitFlag) {	
if (allowDtHorSplit && allowDtVerSplit)	
dt_split_dir	ae(v)
else if (allowDtHorSplit)	
DtSplitDir = 1	
else	
DtSplitDir = 0	
if (DtSplitDir) {	
dt_split_hqt_flag	ae(v)
if (!DtSplitHqtFlag)	
dt_split_hadt_flag	ae(v)
}	
else {	
dt_split_vqt_flag	ae(v)
if (!DtSplitVqtFlag)	
dt_split_vadt_flag	ae(v)
}	
}	
}	
}	
if (UmveFlag) {	

T/AI 109.8-2025

表 38 编码单元定义（续）

编码单元定义	描述符
umve_mv_index	ae(v)
umve_step_index	ae(v)
umve_dir_index	ae(v)
}	
else if ((SkipFlag || DirectFlag) && EtmvpFlag)	
cu_etmvp_cand_index	ae(v)
else if ((SkipFlag || DirectFlag) && AffineFlag) {	
cu_affine_cand_index	ae(v)
if (AffineUmveFlag){	
affine_umve_step_index0	ae(v)
affine_umve_dir_index0	ae(v)
affine_umve_step_index1	ae(v)
affine_umve_dir_index1	ae(v)
}	
}	
else if ((SkipFlag || DirectFlag) && AwpFlag) {	
if (PictureCodingType == 3) {	
awp_mvr_cand_flag0	ae(v)
if (AwpMvrCandFlag0) {	
awp_mvr_cand_step0	ae(v)
awp_mvr_cand_dir0	ae(v)
}	
awp_mvr_cand_flag1	ae(v)
if (AwpMvrCandFlag1) {	
awp_mvr_cand_step1	ae(v)
awp_mvr_cand_dir1	ae(v)
}	
}	
awp_index	ae(v)
awp_cand_index0	ae(v)
awp_cand_index1	ae(v)
}	
else if (SkipFlag || DirectFlag)	
cu_subtype_index	ae(v)
if (!SkipFlag && ! DirectFlag && ! IbcCuFlag && ! IscCuFlag) {	
if (!IntraCuFlag) {	
if (AffineEnableFlag && (width >= 16) && (height >= 16))	
affine_flag	ae(v)

T/AI 109.8-2025

表 38 编码单元定义（续）

编码单元定义	描述符
if (AmvrEnableFlag) {	
if (EmvrEnableFlag && ! AffineFlag)	
extend_mvr_flag	ae(v)
if (AffineFlag)	
affine_amvr_index	ae(v)
else	
amvr_index	ae(v)
}	
if (PictureCodingType == 3)	
inter_pred_ref_mode_index	ae(v)
if (SmvdEnableFlag && SmvdApplyFlag && ! AffineFlag && (InterPredRefMode == 2) && ! ExtendMvrFlag)	
smvd_flag	ae(v)
if (MvExistL0) {	
if ((!SmvdFlag) && NumRefActive[0] > 1)	
pu_reference_index_l0	ae(v)
mv_diff_x_abs_l0	ae(v)
if (MvDiffXAbsL0)	
mv_diff_x_sign_l0	ae(v)
mv_diff_y_abs_l0	ae(v)
if (MvDiffYAbsL0)	
mv_diff_y_sign_l0	ae(v)
if (AffineFlag) {	
mv_diff_x_abs_l0_affine	ae(v)
if (MvDiffXAbsL0Affine)	
mv_diff_x_sign_l0_affine	ae(v)
mv_diff_y_abs_l0_affine	ae(v)
if (MvDiffYAbsL0Affine)	
mv_diff_y_sign_l0_affine	ae(v)
}	
}	
if (MvExistL1 && (!SmvdFlag)) {	
if (NumRefActive[1] > 1)	
pu_reference_index_l1	ae(v)
mv_diff_x_abs_l1	ae(v)
if (MvDiffXAbsL1)	
mv_diff_x_sign_l1	ae(v)
mv_diff_y_abs_l1	ae(v)

T/AI 109.8-2025

表 38 编码单元定义（续）

编码单元定义	描述符
if (MvDiffYAbsL1)	
mv_diff_y_sign_l1	ae(v)
if (AffineFlag) {	
mv_diff_x_abs_l1_affine	ae(v)
if (MvDiffXAbsL1Affine)	
mv_diff_x_sign_l1_affine	ae(v)
mv_diff_y_abs_l1_affine	ae(v)
if (MvDiffYAbsL1Affine)	
mv_diff_y_sign_l1_affine	ae(v)
}	
}	
if (BgcEnableFlag && MvExistL0 && MvExistL1 && (width * height >= 256)) {	
bgc_flag	ae(v)
if (BgcFlag)	
bgc_index	ae(v)
}	
}	
else {	
TuOrder = 0	
if (!SawpFlag){	
for (i=0; i<NumOfIntraPredBlock; i++) {	
intra_luma_pred_mode_index	ae(v)
if (EipmEnableFlag && (IntraLumaPredModeIndex > 1))	
luma_eipm_index	ae(v)
}	
}	
if (PartSize == 'SIZE_2Mx2N') {	
IsPcmMode[TuOrder] = (IntraLumaPredMode == 'Intra_Luma_PCM')	
TuOrder++	
}	
else {	
IsPcmMode[0] = 0	
IsPcmMode[1] = 0	
IsPcmMode[2] = 0	
IsPcmMode[3] = 0	
TuOrder=3	
}	

T/AI 109.8-2025

表 38 编码单元定义（续）

编码单元定义	描述符
if (IntraCuFlag && (chroma_format != '00') && (component == 'COMPONENT_LUMACHROMA')) {	
intra_chroma_pred_mode_index	ae(v)
if ((TscpmEnableFlag || PmcEnableFlag) && (IntraChromaPredModeIndex == 1)) {	
if (EnhancedTscpmEnableFlag || PmcEnableFlag)	
chroma_eipm_index	ae(v)
if (PmcEnableFlag && TscpmEnagleFlag && (EnhancedTscpmEnableFlag || (ChromaEipmIndex == 0)))	
chroma_pmc_index	ae(v)
if (ChromaPmcIndex)	
chroma_pmc_param_index	ae(v)
}	
IsPcmMode[TuOrder+1] = (IntraChromaPredMode == 'Intra_Chroma_PCM')	
IsPcmMode[TuOrder+2] = (IntraChromaPredMode == 'Intra_Chroma_PCM')	
}	
if (IntraPfEnableFlag && (PartSize == 'SIZE_2Mx2N') && (! IsPcmMode[0]) && (! SawpFlag))	
intra_pf_flag	ae(v)
if (IipEnableFlag && (PartSize == 'SIZE_2Mx2N') && (!IsPcmMode[0]) && (! IntraPfFlag) && (width * height >= 64) && (width <=64) && (height <=64) && (! SawpFlag))	
iip_flag	ae(v)
}	
}	
CuCtp = 0	
if ((!IntraCuFlag && ! SkipFlag) || IbcCuFlag) {	
if (!DirectFlag && component == 'COMPONENT_LUMACHROMA')	
ctp_zero_flag	ae(v)
if (!CtpZeroFlag) {	
if (PbtEnableFlag && (width / height < 4) && (height / width < 4) && (width >= 8) && (width <= 32) && (height >= 8) && (height <= 32) && (! InterPfFlag) && (! InterPcFlag))	
pbt_cu_flag	ae(v)
if (!PbtCuFlag) {	
if (component == 'COMPONENT_LUMACHROMA') {	
ctp_u	ae(v)
ctp_v	ae(v)
}	
CuCtp = ctp_u << (NumOfTransBlocks–2)	
CuCtp += (ctp_v << (NumOfTransBlocks–1))	

T/AI 109.8-2025

表 38 编码单元定义（续）

编码单元定义	描述符
if (((ctp_u != 0) || (ctp_v != 0)) || (component != 'COMPONENT_LUMACHROMA')) {	
ctp_y[0]	ae(v)
CuCtp += ctp_y[0]	
}	
else	
CuCtp += ctp_y[0]	
if (SbtEnableFlag && (width <= 64) && (height <= 64) && (width > 4 || height > 4) && (! InterPfFlag) && (! InterPcFlag) && ctp_y[0] && (! IbcCuFlag)) {	
sbt_cu_flag	ae(v)
if (SbtCuFlag) {	
SbtVerHalf = width >= 8 && (height * 2 <= width * MaxPartRatio)	
SbtVerQuad = width >= 16 && (height * 4 <= width * MaxPartRatio)	
SbtHorHalf = height >= 8 && (width * 2 <= height * MaxPartRatio)	
SbtHorQuad = height >= 16 && (width * 4 <= height * MaxPartRatio)	
if ((SbtVerHalf || SbtHorHalf) && (SbtVerQuad || SbtHorQuad))	
sbt_quad_flag	ae(v)
if ((SbtQuadFlag && SbtVerQuad && SbtHorQuad) || (!SbtQuadFlag && SbtVerHalf && SbtHorHalf))	
sbt_dir_flag	ae(v)
}	
}	
}	
else {	
if (component == 'COMPONENT_LUMACHROMA') {	
ctp_u	ae(v)
ctp_v	ae(v)
}	
CuCtp = ctp_u << (NumOfTransBlocks–2)	
CuCtp += (ctp_v << (NumOfTransBlocks–1))	
for (i=0; i<NumOfTransBlocks–2; i++) {	
ctp_y[i]	ae(v)
CuCtp += (ctp_y[i] << i)	
}	
}	
}	
}	
else if (!SkipFlag && ! IscCuFlag) {	

T/AI 109.8-2025

表 38 编码单元定义（续）

编码单元定义	描述符
CuCtp = 0	
if (!IsPcmMode[0]) {	
for (i=0; i<NumOfTransBlocks–2; i++) {	
ctp_y[i]	ae(v)
CuCtp += (ctp_y[i] << i)	
}	
if (PartSize == 'SIZE_2Mx2N' && ctp_y[0] && (width < 64) && (height < 64) && IntraCuFlag && PictureTsEnableFlag)	
enhanced_ts_flag	ae(v)
}	
if ((component == 'COMPONENT_LUMACHROMA') && (IntraChromaPredMode != 'Intra_Chroma_PCM')) {	
if (IntraChromaPredMode == 'Intra_Chroma_PMC' || IntraChromaPredMode == 'Intra_Chroma_PMC_LT' || IntraChromaPredMode == 'Intra_Chroma_PMC_T' || IntraChromaPredMode == 'Intra_Chroma_PMC_L' || IntraChromaPredMode == 'Intra_Chroma_EPMC' || IntraChromaPredMode == 'Intra_Chroma_EPMC_LT' || IntraChromaPredMode == 'Intra_Chroma_EPMC_L' || IntraChromaPredMode == 'Intra_Chroma_EPMC_T' || IntraChromaPredMode == 'Intra_Chroma_EPMC2' || IntraChromaPredMode == 'Intra_Chroma_EPMC2_LT' || IntraChromaPredMode == 'Intra_Chroma_EPMC2_L' || IntraChromaPredMode == 'Intra_Chroma_EPMC2_T')	
ctp_u = 1	
else	
ctp_u	ae(v)
ctp_v	ae(v)
}	
CuCtp += (ctp_u << (NumOfTransBlocks–2))	
CuCtp += (ctp_v << (NumOfTransBlocks–1))	
}	
if ((!FixedQP) && CuDeltaQpFlag && CuCtp && (component != 'COMPONENT_CHROMA')) {	
cu_qp_delta_abs	ae(v)
if (cu_qp_delta_abs)	
cu_qp_delta_sign	ae(v)
if (IsPrivacyFlag)	
NumDeltaQpPrivacy += 1	
else	
NumDeltaQp += 1	
}	

T/AI 109.8-2025

表 38 编码单元定义（续）

编码单元定义	描述符
if (EnhancedStEnableFlag && (component == 'COMPONENT_LUMACHROMA) && IntraCuFlag && (!IbcCuFlag) && (IntraChromaPredMode != 'Intra_Chroma_PCM') && (! IntraPFlag) && (! IipFlag) && (! SawpFlag)) {	
if (ctp_u || ctp_v)	
chroma_st_flag	ae(v)
}	
IstTuFlag=0	
if (PartSize == 'SIZE_2Mx2N' && ctp_y[0] && (width < 64) && (height < 64) && IbcCuFlag && PictureTsEnableFlag &&IstEnableFlag) {	
ts_cu_flag	ae(v)
IstTuFlag = TsCuFlag	
}	
for (i=0; i<NumOfTransBlocks; i++) {	
if (i < NumOfTransBlocks–2) {	
blockWidth = ((TransformSplitDirection == 0) || (TransformSplitDirection == 2)) ? width : (TransformSplitDirection == 1 ? width >> 1 : width >> 2)	
blockHeight = ((TransformSplitDirection == 0) || (TransformSplitDirection == 3)) ? height : (TransformSplitDirection == 1 ? height >> 1 : height >> 2)	
blockX = x0 + (((TransformSplitDirection == 0) || (TransformSplitDirection == 2)) ? 0 : (TransformSplitDirection == 1 ? ((blockWidth >> 1) * (i % 2)) : ((blockWidth >> 2) * i)))	
blockY = y0 + (((TransformSplitDirection == 0) || (TransformSplitDirection == 3)) ? 0 : (TransformSplitDirection == 1 ? ((blockHeight >> 1) * (i / 2)) : ((blockHeight >> 2) * i)))	
if (SbtCuFlag) {	
blockWidth = SbtDirFlag ? blockWidth : (SbtQuadFlag ? blockWidth / 4 : blockWidth / 2)	
blockHeight = ! SbtDirFlag ? blockHeight : (SbtQuadFlag ? blockHeight / 4 : blockHeight / 2)	
}	
}else {	
blockWidth = width	
blockHeight = height	
}	
IsChroma = 0	
if (i == NumOfTransBlocks -1 || i == NumOfTransBlocks -2)	
IsChroma = 1	
isIstApply = ! DtSplitFlag && ! SbtCuFlag && ! IbcCuFlag && !PbtCuFlag && (blockWidth<64 && blockHeight<64) && IstEnableFlag	

T/AI 109.8-2025

表 38 编码单元定义（续）

编码单元定义	描述符
block(i, blockWidth, blockHeight, CuCtp, IsChroma, IsPcmMode[i], isIstApply, IntraCuFlag, component)	
if ((i < NumOfTransBlocks–2) && SbtCuFlag) {	
blockX = (SbtDirFlag ‖ (! SbtPosFlag)) ? 0 : (SbtQuadFlag ? blockWidth * 3 / 4: blockWidth / 2)	
blockY = (!SbtDirFlag ‖ (! SbtPosFlag)) ? 0 : (SbtQuadFlag ? blockHeight * 3 / 4: blockHeight / 2)	
}	
}	
}	
}	

6.1.14 变换块定义

变换块定义见表 39。

表39 变换块定义

变换块定义	描述符
block(i, blockWidth, blockHeight, CuCtp, isChroma, isPcm, isIstApply, isIntra, component) {	
M_1 = blockWidth	
M_2 = blockHeight	
for (x=0; x<M_1; x++) {	
for (y=0; y<M_2; y++)	
QuantCoeffMatrix[x][y] = 0	
}	
if (!isPcm) {	
if (CuCtp & (1 << i)) {	
if (SrccEnableFlag) {	
scan_region_x	ae(v)
scan_region_y	ae(v)
initScanOrder(ScanOrder, ScanRegionX + 1, ScanRegionY + 1，blockWidth)	
lastScanPos = ((ScanRegionX + 1) * (ScanRegionY + 1)) – 1	
lastSet = lastScanPos >> 4	
is_last_x = 0	
is_last_y = 0	
EscapeDataPresent = 0	
NumEvenCoeff = 0	
Pre5Gt0[5] = {0, 0, 0, 0, 0}	

T/AI 109.8-2025

表 39 变换块定义（续）

变换块定义	描述符
Pre5GtX[5] = {0, 0, 0, 0, 0}	
for(i = lastset; i >= 0; i—) {	
setPos = i << 4	
set_nz[i] = 0	
for(n = (i == lastSet ? lastScanPos – setPos : 15); n >= 0; n—) {	
blkpos = ScanOrder[setPos + n];	
PosX = blkpos & (blockWidth – 1)	
PosY = blkpos >> Log(blockWidth)	
if ((PosX == 0 && PosY == ScanRegionY && is_last_y == 0) || (PosY == 0 && PosX == ScanRegionX && is_last_x == 0))	
sig_flag[blkpos] = 1	
else {	
significant_flag	ae(v)
sig_flag[blkpos] = SignificantFlag	
}	
if (sig_flag[blkpos]) {	
if (PosX == ScanRegionX)	
is_last_x = 1	
if (PosY == ScanRegionY)	
is_last_y = 1	
set_nz[i]++	
}	
for (j=4; j > 0; j--)	
Pre5Gt0[j] = Pre5Gt0[j-1]	
Pre5Gt0[0] = sig_flag[blkpos]	
}	
if (set_nz[i]) {	
EscapeDataPresent = 0	
if (PictureTsEnableFlag) {	
for (n=(i == lastSet ? lastScanPos – setPos : 15); n>=0; n—) {	
blkpos = ScanOrder[setPos + n]	
if (sig_flag[blkpos]) {	
coeff_abs_level_greater1_flag	ae(v)
if (CoeffAbsLevelGt1Flag) {	
coeff_abs_level_greater2_flag	ae(v)
if (CoeffAbsLevelGt2Flag) {	
coeff_abs_level_greater4_flag	ae(v)
if (CoeffAbsLevelGt4Flag) {	

T/AI 109.8-2025

表 39 变换块定义（续）

变换块定义	描述符
coeff_abs_level_greater8_flag	ae(v)
if (CoeffAbsLevelGt8Flag) {	
EscapeDataPresent = 1	
abs_coef[blkpos] = 9	
}	
else {	
base_level = 5	
coeff_abs_level_remaining	ae(v)
abs_coef[blkpos] = base_level + CoeffAbsLevelRem	
}	
}	
else {	
base_level = 3;	
coeff_abs_level_remaining	ae(v)
abs_coef[blkpos] = base_level + CoeffAbsLevelRem	
}	
}	
else	
abs_coef[blkpos] = 2	
}	
else	
abs_coef[blkpos] = 1	
for (j=4; j > 0; j--)	
Pre5GtX[j] = Pre5GtX[j-1]	
Pre5GtX[0] = abs_coef [blkpos]	
}	
}	
}	
else {	
for (n=(i == lastSet ? lastScanPos – setPos : 15); n >= 0; n—) {	
blkpos = ScanOrder[setPos + n]	
if (sig_flag[blkpos]) {	
coeff_abs_level_greater1_flag	ae(v)
if (coeffAbsLevelGt1Flag) {	
coeff_abs_level_greater2_flag	ae(v)
if (CoeffAbsLevelGt2Flag) {	

T/AI 109.8-2025

表 39 变换块定义（续）

变换块定义	描述符
EscapeDataPresent = 1	
abs_coef[blkpos] = 3	
}	
else	
abs_coef[blkpos] = 2	
}	
else	
abs_coef[blkpos] = 1	
for (j=4; j > 0; j--)	
Pre5GtX[j] = Pre5GtX[j-1]	
Pre5GtX[0] = abs_coef [blkpos]	
}	
}	
}	
}	
if (EscapeDataPresent) {	
if (PictureTsEnableFlag)	
base_level = 9	
else	
base_level = 3	
for(n = (i == lastSet ? lastScanPos – setPos: 15); n >= 0; n—) {	
blkpos = ScanOrder[setPos + n]	
if (sig_flag[blkpos]) {	
if (abs_coef[blkpos] == base_level) {	
coeff_abs_level_remaining	ae(v)
abs_coef[blkpos] += CoeffAbsLevelRem	
}	
}	
}	
}	
for (n = (i == lastSet ? lastScanPos – setPos : 15); n >= 0; n—) {	
blkpos = ScanOrder[setPos + n]	
if (sig_flag[blkpos]) {	
PosX = blkpos & (blockwidth - 1)	
PosY = blkpos >> Log(blockWidth)	
coeff_sign	ae(v)
QuantCoeffMatrix[PosX][PosY] = CoeffSign ? –abs_coef[blkpos] : abs_coef[blkpos]	

T/AI 109.8-2025

表 39 变换块定义（续）

变换块定义	描述符
}	
}	
}	
for (x=0; x<M_1; x++) {	
for (y=0; y<M_2; y++){	
if (QuantCoeffMatrix[x][y] % 2 == 0)	
NumEvenCoeff++	
}	
}	
if (SbtCuFlag)	
SbtPosFlag = NumEvenCoeff % 2 ? 0 : 1	
if (isIstApply) {	
IstTuFlag = (!PictureTsEnableFlag && (ScanRegionX >=16 || ScanRegionY >=16)) ? 0 : NumEvenCoeff % 2 ? 0 : 1	
if (PictureTsEnableFlag && IntraCuFlag)	
IstTuFlag = EnhancedTsFlag ? (IstTuFlag ? 2 : 1) : 0	
}	
}	
else {	
blockWidth = isChroma ? blockWidth / 2 : blockWidth	
blockHeight = isChroma ? blockHeight / 2 : blockHeight	
indexW = Log(blockWidth) – 1	
indexH = Log(blockHeight) – 1	
NumOfCoeff = blockWidth * blockHeight	
NumEvenCoeff = 0	
ScanPosOffset = 0	
CoeffRestrict = 0	
do {	
coeff_run	ae(v)
coeff_level_minus1	ae(v)
coeff_sign	ae(v)
AbsLevel = coeff_level_minus1 + 1	
ScanPosOffset = ScanPosOffset + coeff_run	
PosxInBlk = InvScanCoeffInBlk[indexW][indexH][ScanPosOffset][0]	
PosyInBlk = InvScanCoeffInBlk[indexW][indexH][ScanPosOffset][1]	
if (PosxInBlk >= 16 || PosyInBlk >= 16)	
CoeffRestrict = 1	
QuantCoeffMatrix[PosxInBlk][PosyInBlk] = coeff_sign ? –AbsLevel : AbsLevel	

T/AI 109.8-2025

表 39 变换块定义（续）

变换块定义	描述符
if (ScanPosOffset >= NumOfCoeff – 1)	
break	
coeff_last	ae(v)
ScanPosOffset = ScanPosOffset + 1	
} while (! coeff_last)	
for(x=0; x<M_1; x++) {	
for(y=0; y<M_2; y++){	
if (QuantCoeffMatrix[x][y] % 2 == 0)	
NumEvenCoeff++	
}	
}	
if (SbtCuFlag)	
SbtPosFlag = NumEvenCoeff % 2 ? 0 : 1	
if (isIstApply)	
IstTuFlag = NumEvenCoeff % 2 ? 0 : 1	
if (CoeffRestrict && ! PictureTsEnableFlag)	
IstTuFlag = 0	
if (isIstApply && PictureTsEnableFlag && IntraCuFlag)	
IstTuFlag = EnhancedTsFlag ? (IstTuFlag ? 2 : 1) : 0	
}	
if (EnhancedStEnableFlag && (! isChroma)) {	
if (isIntra && (! DtSplitFlag) && (! IstTuFlag) && (! IbcCuFlag))	
est_tu_flag	ae(v)
}	
}	
}	
else {	
if ((component != 'COMPONENT_CHROMA' && i == 0) || (component == 'COMPONENT_CHROMA' && i == 1)) {	
aec_ipcm_stuffing_bit	ae(v)
while (! byte_aligned( ))	
aec_byte_alignment_bit0	f(1)
}	
M_1 = isChroma ? blockWidth / 2 : blockWidth	
M_2= isChroma ? blockHeight / 2 : blockHeight	
xMin = Min(32, M_1)	
yMin = Min(32, M_2)	
for (yStep=0; yStep<M_2/yMin; yStep++) {	

T/AI 109.8-2025

表 39 变换块定义（续）

变换块定义	描述符
for (xStep=0; xStep<M_1/xMin; xStep++) {	
for (y=0; y<yMin; y++) {	
for (x=0; x<xMin; x++) {	
pcm_coeff	f(n)
QuantCoeffMatrix[x+xStep*xMin][y + yStep*yMin] = pcm_coeff	
}	
}	
}	
}	
}	
}	

6.1.15 自适应修正滤波参数定义

自适应修正滤波参数定义见表 40。

表40 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 (EalfEnableFlag) {	
coeffNum = 15	
FilterNum = 64	
}	
else {	
coeffNum = 9	
FilterNum = 16	
}	
if (PictureAlfEnableFlag[0] == 1) {	
alf_filter_num_minus1	ue(v)
if (EalfEnableFlag)	
alf_region_order_index	u(2)
for (i=0; i<alf_filter_num_minus1+1; i++) {	
if ((i > 0) && (alf_filter_num_minus1 != (FilterNum-1)))	
alf_region_distance[i]	ue(v)
for (j=0; j<coeffNum; j++)	
alf_coeff_luma[i][j]	se(v)
if (EalfEnableFlag)	
alf_luma_shift_num_minus6[i]	se(v)

T/AI 109.8-2025

表 40 自适应修正滤波参数定义（续）

自适应修正滤波参数定义	描述符
}	
}	
if (PictureAlfEnableFlag[1] == 1) {	
for (j=0; j<coeffNum; j++)	
alf_coeff_chroma[0][j]	se(v)
if (EalfEnableFlag)	
alf_chroma_shift_num_minus6[0]	se(v)
}	
if (PictureAlfEnableFlag[2] == 1) {	
for (j=0; j<coeffNum; j++)	
alf_coeff_chroma[1][j]	se(v)
if (EalfEnableFlag)	
alf_chroma_shift_num_minus6[1]	se(v)
}	
}	

6.1.16 增强样值偏移补偿参数定义

增强样值偏移补偿图像头参数定义见表 41。

表41 增强样值偏移补偿图像头参数定义

增强样值偏移补偿图像头参数定义	描述符
esao_parameter_picture_header_set( ) {	
for (compIndex=0;compIndex<3;compIndex++){	
picture_esao_enable_flag[compIndex]	u(1)
if (PictureEsaoEnableFlag[compIndex]){	
picture_esao_lcu_control_flag[compIndex]	u(1)
if (PictureEsaoLcuControlFlag[compIndex])	
picture_esao_set_num_minus1[compIndex]	u(1)
for (setIndex=0; setIndex<PictureEsaoSetNum[compIndex]; setIndex++) {	
if (compIndex == 0) {	
picture_esao_adaptive_param_minus1[setIndex]	u(4)
esao_luma_param_type[setIndex]	u(1)
}	
else {	
picture_esao_adaptive_param_index[compIndex][setIndex]	u(7)
esao_chroma_band_mode_flag[compIndex][setIndex]	u(1)
if (EsaoChromaBandModeFlag[compIndex]**[setIndex]**) {	

T/AI 109.8-2025

表 41 增强样值偏移补偿图像头参数定义（续）

增强样值偏移补偿图像头参数定义	描述符
esao_chroma_start_band[compIndex][setIndex]	ue(v)
esao_chroma_band_num[compIndex][setIndex]	ue(v)
}	
}	
numCoeff = (compIndex == 0) ? ((EsaoLumaParamType[setIndex] ? 9 : 17) * PictureEsaoAdaptiveParam[setIndex]) : EsaoTableUV[PictureEsaoAdaptiveParamIndex[compIndex][setIndex]]	
startIndex = (compIndex == 0) ? 0 : (EsaoChromaBandModeFlag[compIndex][setIndex] ? EsaoChromaStartBand[compIndex][setIndex] : 0)	
numCoeff = (compIndex != 0 && EsaoChromaBandModeFlag[compIndex][setIndex]) ? EsaoChromaBandNum[compIndex][setIndex] : numCoeff	
for (i=startIndex; i<startIndex+numCoeff; i++)	
esao_filter_coeff[compIndex][setIndex][i]	se(v)
}	
}	
}	
}	

6.1.17 跨分量样值偏移补偿参数定义

跨分量样值偏移补偿图像头参数定义见表 42。

表42 跨分量样值偏移补偿图像头参数定义

跨分量样值偏移补偿参图像头参数定义	描述符
ccsao_parameter_picture_header_set( ) {	
for (compIndex=1; compIndex<3;compIndex++) {	
picture_ccsao_enable_flag[compIndex]	u(1)
if (PictureCcsaoEnableFlag[compIndex]) {	
picture_ccsao_lcu_control_flag[compIndex]	u(1)
if (PictureCcsaoLcuControlFlag[compIndex])	
picture_ccsao_set_num_minus1[compIndex]	u(2)
for (setIndex=0; setIndex<PictureCcsaoSetNum[compIndex]; setIndex++) {	
picture_ccsao_mode[compIndex][setIndex]	u(1)
if (PictureCcsaoMode[compIndex][setIndex]) {	
picture_ccsao_edge_type[compIndex][setIndex]	u(2)
picture_ccsao_edge_band_num_index[compIndex][setIndex]	u(2)
picture_ccsao_edge_thr_index[compIndex][setIndex]	u(3)
}	

T/AI 109.8-2025

表 42 跨分量样值偏移补偿图像头参数定义（续）

跨分量样值偏移补偿参图像头参数定义	描述符
else {	
picture_ccsao_cand_index[compIndex][setIndex]	u(4)
picture_ccsao_luma_band_num_minus1[compIndex][setIndex]	u(4)
picture_ccsao_chroma_band_num_minus1[compIndex][setIndex]	u(1)
}	
for (i=0; i<PictureCcsaoClassNum[compIndex][setIndex]; i++)	
ccsao_offset[compIndex][setIndex][i]	se(v)
}	
}	
}	
}	

6.1.18 串复制帧内预测定义

串复制帧内预测的普通串子模式编码单元定义见表 43。其中，扫描表 TravScan[4][4][1024][2]按附录 A 生成。

表43 串复制帧内预测的普通串子模式编码单元定义

串复制帧内预测的普通串子模式编码单元定义	描述符
isc_ordinary_sp_coding_unit(x0,y0,width,height,component) {	
NumCodedPixel = 0	
NumTotalPixel = width * height	
i = 0	
while (NumCodedPixel < NumTotalPixel) {	
StrXInCu = TravScan[Log(width) - 2][Log(height) - 2] [NumCodedPixel][0]	
StrYInCu = TravScan[Log(width) - 2][Log(height) - 2] [NumCodedPixel][1]	
NumMatchedPixel[i] = 0	
isc_match_type[i]	ae(v)
if (IscMatchType[i] == 1) {	
CurRemainingPixelQuarter = (NumTotalPixel – NumCodedPixel) >> 2	
if (CurRemainingPixelQuarter ==1)	
StrLen[i] = NumTotalPixel – NumCodedPixel	
else {	
isc_next_remaining_pixel_in_cu[i]	ae(v)
StrLen[i] = NumTotalPixel - NumCodedPixel – (IscNextRemainingPixelInCu[i] << 2)	
}	
}	
else {	
StrLen[i] = 4	

T/AI 109.8-2025

表 43 串复制帧内预测的普通串子模式编码单元定义（续）

串复制帧内预测的普通串子模式编码单元定义	描述符
for (j=0; j<StrLen[i]; j++) {	
isc_pixel_match_type[i][j]	ae(v)
if (IscPixelMatchType[i][j])	
NumMatchedPixel[i] += 1	
}	
}	
if (IscMatchType[i] || NumMatchedPixel) {	
isc_sv_above_flag[i]	ae(v)
if (!IscSvAboveFlag) {	
isc_sv_recent_flag[i]	ae(v)
if (IscSvRecentFlag)	
isc_sv_recent_index[i]	ae(v)
else {	
isc_sv_y_abs[i]	ae(v)
if (IscSvYAbs[i])	
isc_sv_y_sign[i]	ae(v)
if (IscSvYAbs[i] == 0) {	
isc_sv_x_sign[i]	ae(v)
isc_sv_x_abs_minus1[i]	ae(v)
}	
else {	
if (!IscSvYSign[i])	
isc_sv_x_abs_minus1[i]	ae(v)
else {	
isc_sv_x_non_zero_flag[i]	ae(v)
if (IscSvXNonZeroFlag[i]) {	
isc_sv_x_sign[i]	ae(v)
isc_sv_x_abs_minus1[i]	ae(v)
}	
}	
}	
}	
}	
}	
NumCodedPixel += StrLen[i]	
i++	
}	
IscPartNum = i	

T/AI 109.8-2025

表 43 串复制帧内预测的普通串子模式编码单元定义（续）

串复制帧内预测的普通串子模式编码单元定义	描述符
NumCodedPixel = 0	
i = 0	
while (NumCodedPixel < NumTotalPixel) {	
if (IscMatchType[i] == 0) {	
for (j=0; j<StrLen[i]; j++) {	
if (!IscPixelMatchType[i][j]) {	
StrXInCu = TravScan[Log(width) - 2][Log(height) - 2] [NumCodedPixel + j][0]	
StrYInCu = TravScan[Log(width) - 2][Log(height) - 2] [NumCodedPixel + j][1]	
isc_unmatched_pixel_y[i][j]	ae(v)
if (component == 'COMPONENT_LUMACHROMA' && ! (StrXInCu & 0x1 ‖ StrYInCu & 0x1)) {	
isc_unmatched_pixel_cb[i][j]	ae(v)
isc_unmatched_pixel_cr[i][j]	ae(v)
}	
}	
}	
}	
NumCodedPixel += StrLen[i]	
i++	
}	
}	

串复制帧内非普通串子模式编码单元定义见表 44。

表44 串复制帧内预测非普通串子模式编码单元定义

串复制帧内预测非普通串子模式编码单元定义	描述符
isc_evs_ubvs_coding_unit(x0,y0,width,height,component) {	
i = 0	
NumTotalPixel = width * height	
NumCodedPixel = 0	
isc_ups_present_flag	ae(v)
PvNum = 0	
isc_num_of_new_pv	ae(v)
if (PrevPvBufSize > 0 && IscNumOfNewPv < 15)	
isc_num_of_reused_pv	ae(v)
for (k = 0; PvNum < IscNumOfReusedPv; k++) {	
isc_prev_pv_not_reused_run	ae(v)
k += IscPrevPvNotReusedRun	

T/AI 109.8-2025

表 44 串复制帧内预测非普通串子模式编码单元定义（续）

串复制帧内预测非普通串子模式编码单元定义	描述符
PpInfoList[PvNum][0] = PrevPpInfoList[k][0]	
PpInfoList[PvNum][1] = PrevPpInfoList[k][1]	
FopYonly[PvNum] = PrevFopYonly[k]	
EvsDpbIndex[PvNum] = PrevEvsDpbIndex[k]	
EvsDpbReactivatedYonly[PvNum] = PrevEvsDpbReactivatedYonly[k]	
CompLumaFreqOccurPos[PvNum] = PrevCompLumaFreqOccurPos[k]	
PvNum ++	
PrevPpInfoList[k][0] = -1	
PrevPpInfoList[k][1] = -1	
}	
MaxValPvAddress = IscNumOfReusedPv – 1 + IscUpsPresentFlag	
MaxValPvAddressUpperBound = MaxValPvAddress + IscNumOfNewPv	
if (IscUpsPresentFlag)	
isc_ups_max_length_minus1	ae(v)
if (IscNumOfNewPv > 0)	
MaxValPvAddress++	
while (NumCodedPixel < width)	
isc_evs_ups_coding(x0,y0,width,height,component)	
FirstStringNumFlag = 1	
while (NumCodedPixel < NumTotalPixel) {	
isc_evs_ups_num_minus1	ae(v)
if (FirstStringNumFlag) {	
StringNum = IscEvsUPSNumMinus1	
FirstStringNumFlag = 0	
}	
else	
StringNum = IscEvsUPSNumMinus1 + 1	
for (k = 0; k < StringNum; k++)	
isc_evs_ups_coding(x0,y0,width,height,component)	
if (NumCodedPixel < NumTotalPixel) {	
maxStrLength = NumTotalPixel – NumCodedPixel	
maxValPrefix = Ceil(Log(maxStrLength)) – 1	
k = 0	
if (maxStrLength>2)	
str_length_minus1_prefix	ae(v)
if (StrLengthMinus1Prefix==0)	
maxValInfix = maxStrLength==1? 0:1	
else {	

T/AI 109.8-2025

表 44　串复制帧内预测非普通串子模式编码单元定义（续）

串复制帧内预测非普通串子模式编码单元定义	描述符
StrLengthMinus1Prefix += 1	
maxValInfix = min(maxStrLength - 1 - (1 << (StrLengthMinus1Prefix - 1)), (1 << (StrLengthMinus1Prefix - 1)) - 1)	
}	
n = Ceil(Log(maxValInfix+1))	
if (n >1)	
str_length_minus1_infix	ae(v)
if (!(StrLengthMinus1Infix < ((1<<n) – maxValInfix – 1) || maxValInfix == 0)) {	
k = 1	
str_length_minus1_suffix	ae(v)
}	
if (StrLengthMinus1Prefix ==0)	
StrLengthMinus1 = StrLengthMinus1Suffix	
else	
StrLengthMinus1 =(1 << (StrLengthMinus1Prefix -1)) + (StrLengthMinus1Infix << k) + StrLengthMinus1Suffix - ((1<<n) - maxValInfix-1) × k	
StrLen[i] = StrLengthMinus1 + 1	
StringType[i] = 0	
for (k=0; k<StrLen[i]; ++k) {	
sxInCu = TravScan[Log(width) - 2][Log(height) - 2][NumCodedPixel][0]	
syInCu = TravScan[Log(width) - 2][Log(height) - 2][NumCodedPixel][1]	
offsetAbove = (syInCu & 1) == 0 ? 2*sxInCu + 1 : 2*(width - sxInCu) - 1	
PvFlag[sxInCu][syInCu] = 0	
if (PixelType[NumCodedPixel - offsetAbove] == 2) {	
PixelType[NumCodedPixel] = 2	
UnmatchedPixelIndex[NumCodedPixel] = UnmatchedPixelIndex[NumCodedPixel - offsetAbove]	
}	
else {	
PixelType[NumCodedPixel] = 0	
PvAddr[NumCodedPixel] = PvAddr[NumCodedPixel - offsetAbove]	
if (component == 'COMPONENT_LUMACHROMA' && ((sxInCu | syInCu) & 0x1) == 0 && EvsDpbReactivatedYonly[PvAddr[NumCodedPixel]] == 1) {	
xInLcuR = x0 + sxInCu – LcuRx0	
yInLcuR = y0 + syInCu – LcuRy0	
LcuRowBufY[xInLcuR][yInLcuR] = LcuRowBufY[PPInfoList[PvAddr[NumCodedPixel]][0]][PPInfoList[PvAddr[NumCodedPixel]][1]]	
LcuRowBufU[xInLcuR>>1][yInLcuR>>1] = EvsDpb[1][EvsDpbIndex[PvAddr[NumCodedPixel]]]	

T/AI 109.8-2025

表 44 串复制帧内预测非普通串子模式编码单元定义（续）

串复制帧内预测非普通串子模式编码单元定义	描述符
LcuRowBufV[xInLcuR>>1][yInLcuR>>1] = EvsDpb[2][EvsDpbIndex[PvAddr[NumCodedPixel]]]	
PpInfoList[PvAddr[NumCodedPixel]][0] = xInLcuR	
PpInfoList[PvAddr[NumCodedPixel]][1] = yInLcuR	
EvsDpbIndex[PvAddr[NumCodedPixel]] = 28	
EvsDpbReactivatedYonly[PvAddr[NumCodedPixel]] = 0	
PvFlag[sxInCu][syInCu] = 1	
}	
}	
NumCodedPixel++	
}	
i ++	
}	
}	
IscPartNum = i	
i = 0	
NumCodedPixel = 0	
while (NumCodedPixel < NumTotalPixel) {	
if (StringType[i] == 2) {	
for (k=0; k<StrLen[i]; k++) {	
isc_nos_up_y[i+k]	ae(v)
if (!upYonly[i+k]) {	
isc_nos_up_cb[i+k]	ae(v)
isc_nos_up_cr[i+k]	ae(v)
}	
}	
NumCodedPixel += StrLen[i]	
i += k	
}	
else if (StringType[i] == 1) {	
if (PvType[i] == 1 || PvType[i] == 3) {	
isc_fopixel_y	ae(v)
LcuRowBufY[PpInfoList[PvAddr[NumCodedPixel]][0]][PpInfoList[PvAddr[NumCodedPixel]][1]] = IscFopixelY	
}	
if (PvType[i] == 2 || PvType[i] == 3) {	
isc_fopixel_cb	ae(v)
isc_fopixel_cr	ae(v)

T/AI 109.8-2025

表 44 串复制帧内预测非普通串子模式编码单元定义（续）

串复制帧内预测非普通串子模式编码单元定义	描述符
LcuRowBufU[PpInfoList[PvAddr[NumCodedPixel]][0]>>1][PpInfoList[PvAddr[NumCodedPixel]][1]>>1] = IscFopixelCb	
LcuRowBufV[PpInfoList[PvAddr[NumCodedPixel]][0]>>1][PpInfoList[PvAddr[NumCodedPixel]][1]>>1] = IscFopixelCr	
}	
NumCodedPixel += StrLen[i]	
i++	
}	
else {	
NumCodedPixel += StrLen[i]	
i++	
}	
}	
}	

串复制帧内预测等值串与未匹配样本串编码过程定义见表 45。

表45 串复制帧内预测等值串与未匹配样本串编码过程定义

串复制帧内预测等值串与未匹配样本串编码过程定义	描述符
isc_evs_ups_coding(x0,y0,width,height,component) {	
if (IscNumOfNewPv + IscNumOfReusedPv == 0) {	
maxStrLength = IscUpsMaxLengthMinus1 + 1	
maxValPrefix = Ceil(Log(maxStrLength))	
k = 0	
if (maxStrLength > 1)	
str_length_minus1_prefix	ae(v)
if (StrLengthMinus1Prefix == 0)	
maxValInfix = 0	
else	
maxValInfix = min(maxStrLength - 1 - (1 << (StrLengthMinus1Prefix - 1)), (1 << (StrLengthMinus1Prefix - 1)) - 1)	
n = Ceil(Log(maxValInfix +1))	
if (n >1)	
str_length_minus1_infix	ae(v)
if (!(StrLengthMinus1Infix < ((1<<n) – maxValInfix – 1) || maxValInfix == 0)) {	
k=1	
str_length_minus1_suffix	ae(v)
}	
if (StrLengthMinus1Prefix == 0)	

T/AI 109.8-2025

表 45 串复制帧内预测等值串与未匹配样本串编码过程定义（续）

串复制帧内预测等值串与未匹配样本串编码过程定义	描述符
StrLengthMinus1 = 0	
else	
StrLengthMinus1 =(1 << (StrLengthMinus1Prefix -1)) + (StrLengthMinus1Infix << k) + StrLengthMinus1Suffix - ((1<<n) - maxValInfix-1) × k	
StrLen[i] = StrLengthMinus1 + 1	
for (k=0; k< StrLengthMinus1+1; k++) {	
sxInCu = TravScan[Log(width) - 2][Log(height) - 2][NumCodedPixel][0]	
syInCu = TravScan[Log(width) - 2][Log(height) - 2][NumCodedPixel][1]	
StringType[i] = 2	
PixelType[NumCodedPixel] = 2	
UnmatchedPixelIndex[NumCodedPixel] = i	
upYonly[i] = 1	
if (component == 'COMPONENT_LUMACHROMA' && ((sxInCu | syInCu) & 0x1) == 0) {	
upYonly[i] = 0	
}	
PvAddr[NumCodedPixel] = 0	
PvFlag[sxInCu][syInCu] = 0	
PvType[i] = 0	
i++	
NumCodedPixel++	
}	
}	
else {	
MaxVal=MaxValPvAddress-1	
n = Ceil(Log(MaxValPvAddress + 1))	
k = 0	
if (n > 1)	
pv_address_prefix	ae(v)
if (!(PvAddressPrefix < ((1<<n) – MaxVal-1) || MaxVal == 0)) {	
k = 1	
pv_address_infix	ae(v)
}	
offsetAbove = (TravScan[Log(width) - 2][Log(height) - 2][NumCodedPixel][1] & 1) == 0 ? 2×TravScan[Log(width) - 2][Log(height) - 2][NumCodedPixel][0] + 1 : 2×(width - TravScan[Log(width) - 2][Log(height) - 2][NumCodedPixel][0]) - 1	
if ((NumCodedPixel == 0 || NumCodedPixel > 0 && StringType[i - 1] == 0 && PixelType[NumCodedPixel-offsetAbove]==2) && PvAddressPrefix==0 && PvAddressInfix==0 && MaxValPvAddress!=0)	

T/AI 109.8-2025

表 45　串复制帧内预测等值串与未匹配样本串编码过程定义（续）

串复制帧内预测等值串与未匹配样本串编码过程定义	描述符
pv_address_suffix	ae(v)
PvAddress = (PvAddressPrefix << k) + PvAddressInfix – ((1<<n)-MaxVal-1) × k - PvAddressSuffix	
if (NumCodedPixel > 0) {	
if (StringType[i - 1] != 0) {	
if ((PvAddress += (PvAddr[NumCodedPixel-1]+1)) >= (MaxVal +2))	
PvAddress -= (MaxVal + 2)	
}	
else if (PixelType[NumCodedPixel - offsetAbove] != 2) {	
if ((PvAddress += (PvAddr[NumCodedPixel-offsetAbove]+1)) >= (MaxVal +2))	
PvAddress -= (MaxVal + 2)	
}	
else {	
if ((PvAddress += (MaxVal + 1 +1)) >= (MaxVal +2))	
PvAddress -= (MaxVal + 2)	
}	
}	
else {	
if ((PvAddress += (MaxVal + 1 + 1)) >= (MaxVal +2))	
PvAddress -= (MaxVal + 2)	
}	
PvType[i] = 0	
if ((PvNum < IscNumOfReusedPv + IscNumOfNewPv) && (PvAddress == (MaxValPvAddress-IscUpsPresentFlag))) {	
maxStrLength = NumTotalPixel – NumCodedPixel	
maxValPrefix = Ceil(Log(maxStrLength))	
k = 0	
if (maxStrLength > 1)	
str_length_minus1_prefix	ae(v)
if (StrLengthMinus1Prefix == 0)	
maxValInfix = 0	
else	
maxValInfix = min(maxStrLength - 1 - (1 << (StrLengthMinus1Prefix - 1)), (1 << (StrLengthMinus1Prefix - 1)) - 1)	
n = Ceil(Log(maxValInfix +1))	
if (n >1)	
str_length_minus1_infix	ae(v)
if (!(StrLengthMinus1Infix < ((1<<n) – maxValInfix – 1) || maxValInfix == 0)) {	
k = 1	

T/AI 109.8-2025

表 45 串复制帧内预测等值串与未匹配样本串编码过程定义（续）

串复制帧内预测等值串与未匹配样本串编码过程定义	描述符
str_length_minus1_suffix	ae(v)
}	
if (StrLengthMinus1Prefix == 0)	
StrLengthMinus1 = 0	
else	
StrLengthMinus1 = (1 << (StrLengthMinus1Prefix -1)) + (StrLengthMinus1Infix << k) + StrLengthMinus1Suffix - ((1<<n) - maxValInfix-1) × k	
StrLen[i] = StrLengthMinus1 + 1	
xInCu = TravScan[Log(width) - 2][Log(height) - 2][NumCodedPixel][0]	
yInCu = TravScan[Log(width) - 2][Log(height) - 2][NumCodedPixel][1]	
xInLcuR = x0 + xInCu – LcuRx0	
yInLcuR = y0 + yInCu – LcuRy0	
PpInfoList[PvNum][0] = xInLcuR	
PpInfoList[PvNum][1] = yInLcuR	
PvType[i] = 1	
FopYonly[PvNum] = component == 'COMPONENT_LUMA' ? 0 : 1	
EvsDpbIndex[PvNum] = 28	
EvsDpbReactivatedYonly[PvNum] = 0	
CompLumaFreqOccurPos[PvNum] = component == 'COMPONENT_LUMA' ? 1 : 0	
for (k=0; k<StrLen[i]; k++) {	
xInCu2 = TravScan[Log(width) – 2][Log(height) – 2][NumCodedPixel+k][0]	
yInCu2 = TravScan[Log(width) – 2][Log(height) – 2][NumCodedPixel+k][1]	
PvFlag[xInCu2][yInCu2] = 0	
}	
PvNum++	
NumOfNewPvReceived++	
if (MaxValPvAddress < MaxValPvAddressUpperBound)	
MaxValPvAddress++	
}	
else {	
if (PvAddress < PvNum) {	
maxStrLength = NumTotalPixel – NumCodedPixel	
maxValPrefix = Ceil(Log(maxStrLength))	
k = 0	
if (maxStrLength > 1)	
str_length_minus1_prefix	ae(v)
if (StrLengthMinus1Prefix = =0)	
maxValInfix = 0	

T/AI 109.8-2025

表 45　串复制帧内预测等值串与未匹配样本串编码过程定义（续）

串复制帧内预测等值串与未匹配样本串编码过程定义	描述符
else	
maxValInfix = min(maxStrLength - 1 - (1 << (StrLengthMinus1Prefix - 1)), (1 << (StrLengthMinus1Prefix - 1)) - 1)	
n = Ceil(Log(maxValInfix +1))	
if (n >1)	
str_length_minus1_infix	ae(v)
if (!(StrLengthMinus1Infix < ((1 << n) – maxValInfix – 1) || maxValInfix==0)) {	
k=1	
str_length_minus1_suffix	ae(v)
}	
if (StrLengthMinus1Prefix == 0)	
StrLengthMinus1 = 0	
else	
StrLengthMinus1 = (1 << (StrLengthMinus1Prefix -1)) + (StrLengthMinus1Infix << k) + StrLengthMinus1Suffix - ((1<<n) - maxValInfix-1) × k	
StrLen[i] = StrLengthMinus1 + 1	
}	
}	
if (PvAddress < PvNum) {	
if (component == 'COMPONENT_LUMACHROMA' && FopYonly[PvAddress]) {	
log2Size = log(min(LcuSize, 64))	
PvX = PpInfoList[PvAddress][0]	
PvY = PpInfoList[PvAddress][1]	
if (PvX >> log2Size != (x0-LcuRx0) >> log2Size || PvY >> log2Size != (y0-LcuRy0) >> log2Size) {	
xInCu = TravScan[Log(width) – 2][Log(height) – 2][NumCodedPixel][0]	
yInCu = TravScan[Log(width) – 2][Log(height) - 2][NumCodedPixel][1]	
xInLcuR = x0 + xInCu – LcuRx0	
yInLcuR = y0 + yInCu – LcuRy0	
if (EvsDpbIndex[PvAddress] < 28) {	
LcuRowBufY[xInLcuR][yInLcuR] = EvsDpb[0][EvsDpbIndex[PvAddress]]	
EvsDpbIndex[PvAddress] = 28	
}	
else	
LcuRowBufY[xInLcuR][yInLcuR] = LcuRowBufY[PpInfoList[PvAddress][0]][PpInfoList[PvAddress][1]]	
PpInfoList[PvAddress][0] = xInLcuR	
PpInfoList[PvAddress][1] = yInLcuR	

T/AI 109.8-2025

表 45 串复制帧内预测等值串与未匹配样本串编码过程定义（续）

串复制帧内预测等值串与未匹配样本串编码过程定义	描述符
}	
for (k=0; k<StrLen[i]; k++) {	
xInCu = TravScan[Log(width)-2][Log(height)-2][NumCodedPixel+k][0]	
yInCu = TravScan[Log(width)-2][Log(height)-2][NumCodedPixel+k][1]	
if (((xInCu | yInCu) & 0x1) == 0) {	
xInLcuR = x0 + xInCu – LcuRx0	
yInLcuR = y0 + yInCu – LcuRy0	
LcuRowBufY[xInLcuR][yInLcuR] = LcuRowBufY[PpInfoList[PvAddress]][0]][PpInfoList[PvAddress]][1]]	
PpInfoList[PvAddress][0] = xInLcuR	
PpInfoList[PvAddress][1] = yInLcuR	
PvFlag[xInCu][yInCu] = 1	
if (PvType[i] == 1)	
PvType[i] = 3	
else	
PvType[i] = 2	
FopYonly[PvAddress] = 0	
break	
}	
}	
}	
else if (component == 'COMPONENT_LUMACHROMA' && PvAddress < IscNumOfReusedPv) {	
if (EvsDpbIndex[PvAddress] < 28) {	
xInCu = TravScan[Log(width) – 2][Log(height) – 2][NumCodedPixel][0]	
yInCu = TravScan[Log(width) – 2][Log(height) - 2][NumCodedPixel][1]	
xInLcuR = x0 + xInCu – LcuRx0	
yInLcuR = y0 + yInCu – LcuRy0	
PpInfoList[PvAddress][0] = xInLcuR	
PpInfoList[PvAddress][1] = yInLcuR	
LcuRowBufY[xInLcuR][yInLcuR] = EvsDpb[0][EvsDpbIndex[PvAddress]]	
EvsDpbReactivatedYonly[PvAddress] = 1	
}	
for (k=0; k<StrLen[i]; k++) {	
xInCu = TravScan[Log(width)-2][Log(height)-2][NumCodedPixel+k][0]	
yInCu = TravScan[Log(width)-2][Log(height)-2][NumCodedPixel+k][1]	
log2Size = log(min(LcuSize,64))	
PvX = PpInfoList[PvAddress][0]	

T/AI 109.8-2025

表 45　串复制帧内预测等值串与未匹配样本串编码过程定义（续）

串复制帧内预测等值串与未匹配样本串编码过程定义	描述符
PvY = PpInfoList[PvAddress][1]	
if (((xInCu | yInCu) & 0x1) == 0 && ((PvX >> log2Size != (x0-LcuRx0) >> log2Size || PvY >> log2Size != (y0-LcuRy0) >> log2Size) || (EvsDpbIndex[PvAddress] < 28))) {	
xInLcuR = x0 + xInCu – LcuRx0	
yInLcuR = y0 + yInCu – LcuRy0	
PvFlag[xInCu][yInCu] = 1	
LcuRowBufY[xInLcuR][yInLcuR] = LcuRowBufY[PPInfoList[PvAddress][0]][PPInfoList[PvAddress][1]]	
if (EvsDpbIndex[PvAddress] < 28) {	
LcuRowBufU[xInLcuR >>1][yInLcuR >> 1] = EvsDpb[1][EvsDpbIndex[PvAddress]]	
LcuRowBufV[xInLcuR >> 1][yInLcuR >> 1] = EvsDpb[2][EvsDpbIndex[PvAddress]]	
EvsDpbIndex[PvAddress] = 28	
EvsDpbReactivatedYonly[PvAddress] = 0	
}	
else {	
LcuRowBufU[xInLcuR >> 1][yInLcuR >> 1] = LcuRowBufU[PpInfoList[PvAddress][0] >> 1][PpInfoList[PvAddress][1] >> 1]	
LcuRowBufV[xInLcuR >> 1][yInLcuR >> 1] = LcuRowBufV[PpInfoList[PvAddress][0] >> 1][PpInfoList[PvAddress][1] >> 1]	
}	
PpInfoList[PvAddress][0] = xInLcuR	
PpInfoList[PvAddress][1] = yInLcuR	
break	
}	
}	
}	
StringType[i] = 1	
for (k=0; k<StrLen[i]; k++) {	
PixelType[NumCodedPixel] = 1	
PvAddr[NumCodedPixel] = PvAddress	
NumCodedPixel++	
}	
i++	
}	
else {	
maxStrLength = IscUpsMaxLengthMinus1 + 1	

T/AI 109.8-2025

表 45　串复制帧内预测等值串与未匹配样本串编码过程定义（续）

串复制帧内预测等值串与未匹配样本串编码过程定义	描述符
maxValPrefix = Ceil(Log(maxStrLength))	
k=0	
if (maxStrLength>1)	
str_length_minus1_prefix	ae(v)
if (StrLengthMinus1Prefix==0)	
maxValInfix = 0	
else	
maxValInfix = min(maxStrLength - 1 - (1 << (StrLengthMinus1Prefix - 1)), (1 << (StrLengthMinus1Prefix - 1)) - 1)	
n = Ceil(Log(maxValInfix +1))	
if (n>1)	
str_length_minus1_infix	ae(v)
if (!(StrLengthMinus1Infix < ((1<<n) – maxValInfix – 1) || maxValInfix==0)) {	
k=1	
str_length_minus1_suffix	ae(v)
}	
if (StrLengthMinus1Prefix ==0)	
StrLengthMinus1 = 0	
else	
StrLengthMinus1 =(1 << (StrLengthMinus1Prefix -1)) + (StrLengthMinus1Infix << k) + StrLengthMinus1Suffix - ((1<<n) - maxValInfix-1) × k	
StrLen[i] = StrLengthMinus1 + 1	
for (k=0; k< StrLengthMinus1 + 1; k++) {	
sxInCu = TravScan[Log(width) - 2][Log(height) - 2][NumCodedPixel][0]	
syInCu = TravScan[Log(width) - 2][Log(height) - 2][NumCodedPixel][1]	
StringType[i] = 2	
PixelType[NumCodedPixel] = 2	
UnmatchedPixelIndex[NumCodedPixel] = i	
upYonly[i] = 1	
if (component == 'COMPONENT_LUMACHROMA' && ((sxInCu | syInCu) & 0x1) == 0) {	
upYonly[i] = 0	
}	
PvFlag[sxInCu][syInCu] = 0	
PvAddr[NumCodedPixel] = PvAddress	
i++	
NumCodedPixel++	
}	

T/AI 109.8-2025

表 45　串复制帧内预测等值串与未匹配样本串编码过程定义（续）

串复制帧内预测等值串与未匹配样本串编码过程定义	描述符
} }	
}	

6.1.19　扩展数据 RBSP 语法

6.1.19.1　扩展数据 BRSP 语法通则

扩展数据定义应符合如下语法结构，见表 46。

表46　TLV 语法结构表

surveillance_extension_TLV( ) {	描述符
extension_id	u(8)
extension_length	u(v)
extension_data	u(v)
}	

扩展数据RBSP语法见表47。

表47　扩展数据单元语法表

surveillance_extension_rbsp( ) {	描述符
while(next_bits(8) != 0x80) {	
if (next_bits(8) == 0x04)	
time_extension( )	
else if (next_bits(8) == 0x10)	
gis_extension( )	
else if (next_bits(8) == 0x12)	
osd_extension( )	
else if (next_bits(8) == 0xE1)	
analysis_extension( )	
else if (next_bits(8) == 0xE3)	
ftm_extension( )	
else	
reserved_extension( )	
}	
surveillance_extension_stop_byte	f(8)
}	

6.1.19.2　绝对时间信息扩展语法

T/AI 109.8-2025

绝对时间信息扩展语法见表48。

表48 绝对时间信息扩展语法表

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

6.1.19.3 地理信息扩展语法

地理信息扩展语法见表49。

表49 地理信息扩展语法表

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

6.1.19.4 OSD 信息扩展语法

T/AI 109.8-2025

OSD 信息扩展语法格式应符合 GB/T 25724-2017 中 5.2.3.5.4 的规定。

6.1.19.5 视频结构化信息扩展语法

视频结构化信息扩展语法见表 50。

其中，analysis_id 为 0x01 至 0x0C 对应的 analysis_data 语法应符合 T/BAX 0007-2025 中 5.3.5.2 至 5.2.5.13 描述。analysis_id 为 0x0D 对应的 analysis_data[i][j]语法见附录 B。

表50 视频结构化信息扩展语法表

analysis_extension( ) {	描述符
extension_id	u(8)
extension_length	u(32)
subtype	u(8)
analysis_num	u(8)
for (i=0; i<analysis_num; i++) {	
analysis_id[i]	u(8)
data_length[i]	u(32)
for(j=0; j<data_length; j++)	
analysis_data[i][j]	u(8)
}	
}	

6.1.19.6 特征张量图信息扩展语法

特征张量图信息扩展语法见表 51。

表51 特征张量图信息扩展语法表

ftm_extension( ) {	描述符
extension_id	u(8)
extension_length	u(32)
ftm_subtype_index	u(5)
ftm_ra_flag	u(1)
if (Ftm_ra_flag == 1) {	
ftm_type	u(8)
ftm_width	u(16)
ftm_height	u(16)
ftm_depth	u(4)
ftm_ipack_mode	u(4)
ftm_iquant_mode	u(4)
ft_num	u(4)
for (i=0; i<ft_num; i++) {	
ft_channel[i]	u(16)
ft_width[i]	u(16)

T/AI 109.8-2025

表 51 特征张量图信息扩展语法表（续）

ftm_extension( ) {	描述符
ft_height[i]	u(16)
}	
}	
reserved_bits	u(4)
ftm_lag_picture_order_index	u(6)
ftm_iquant_max	u(32)
ftm_iquant_min	u(32)
ftm_data_length	u(32)
ftm_data	u(v)
}	

6.1.20 认证数据 RBSP 定义

认证数据 RBSP 定义见表 52。

表52 认证数据 RBSP 定义

认证数据RBSP定义	描述符
authentication_data_rbsp( ) {	
for_current_ras_idc	u(1)
auth_is_library_flag	u(1)
if (AuthIsLibraryFlag)	
authenticaion_library_picture_index	ue(v)
authentication_data_id	u(1)
authentication_hash_list_flag	u(1)
if (authentication_hash_list_flag) {	
authentication_hash_number_minus1	u(8)
for (i=0; i<authentication_hash_number_minus1+1; i++)	
authentication_hash[i]	u(v)
}	
authentication_data_length_minus1	u(8)
for (i=0; i<authentication_data_length_minus1+1; i++)	
authentication_data[i]	u(8)
rbsp_trailing_bits( )	
}	

6.1.21 访问单元分界 RBSP 定义

访问单元分界 RBSP 定义见表 53。

T/AI 109.8-2025

表53 访问单元分界 RBSP 定义

访问单元分界RBSP定义	描述符
access_unit_delimiter_rbsp( ) {	
}	

6.1.22 编码视频序列结尾 RBSP 定义

编码视频序列结尾 RBSP 定义见表 54。

表54 编码视频序列结尾 RBSP 定义

编码视频序列结尾RBSP定义	描述符
end_of_cvs_rbsp( ) {	
}	

6.1.23 流结尾 RBSP 定义

流结尾 RBSP 定义见表 55。

表55 流结尾 RBSP 定义

流结尾RBSP定义	描述符
end_of_stream_rbsp( ) {	
}	

6.1.24 RBSP 尾位定义

RBSP 尾位定义见表 56。

表56 RBSP 尾位定义

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

6.2 语义描述

6.2.1 位流 NAL 单元语义描述

前导零码 leading_zero_8bits

位串‘0x00’，只可能在流的第一个位流 NAL 单元里出现。

T/AI 109.8-2025

零码 zero_byte

位串‘0x00’。

起始码前缀 start_code_prefix_one_3bytes

位串‘0x000001’，3 字节的固定值序列。

后缀零码 trailing_zero_8bits

位串‘0x00’。

子位流抽取方法见附录 C。

6.2.2 NAL 单元语义描述

6.2.2.1 NAL 单元语义

NumBytesInNALunit 指示 NAL 单元的长度，单位为字节。一个 NAL 单元由单元头部分和单元负载部分组成，其中单元负载部分包含一个 RBSP 语法结构及可能存在的认证数据负载，同时还可能包含一些 emulation_prevention_three_byte。为了能推导出 NumBytesInNALunit，需要对 NAL 单元的边界进行划分，8.1 条规定了一种划分方法，其他划分方法可能在本文件之外给出。

零禁止码 forbidden_zero_bit

二值变量。应等于‘0’。

NAL 单元类型标志 nal_unit_type

5 位无符号整数。表示 NAL 单元中的 RBSP 数据结构的类型，见表 57。在不影响 nal_unit_type 不等于 11 的 NAL 单元的解码过程和不影响本文件一致性的前提下，nal_unit_type 等于 11 的 NAL 单元可以被解码器丢弃。当一个编码片 NAL 单元的 nal_unit_type 值等于 1、2、4、12 或 17 时，编码同一图像的其它所有编码片 NAL 单元的 nal_unit_type 值都应相同。若 UserPermission 等于 0，nal_unit_type 值等于 19 的 NAL 单元可以被解码器丢弃。当一个编码片 NAL 单元的 nal_unit_type 值等于 19 时，RBSP 数据包含图像的编码片中若干编码单元数据。

一个知识图像的编码片的个数等于 1 时，该知识图像编码片 NAL 单元的 nal_unit_type 应为 12 或 17。一个非显示知识图像的编码片的个数大于 1 时，按解码顺序第一个与最后一个知识图像编码片 NAL 单元的 nal_unit_type 应为 18，其余知识图像编码片 NAL 单元的 nal_unit_type 应为 12。

表57 NAL 单元类型表

nal_unit_type	NAL单元中RBSP语法结构的内容
0	禁止
1	NRAP图像的编码片 patch_data_rbsp( )
2	RAPI图像的编码片 patch_data_rbsp( )
3	图像头 picture_header_rbsp( )
4	IDR图像的编码片 patch_data_rbsp( )
5	扩展数据单元 surveillance_extension_rbsp( )

T/AI 109.8-2025

表 57 NAL 单元类型表（续）

nal_unit_type	NAL单元中RBSP语法结构的内容
6	补充增强信息 sei_rbsp( )
7	序列参数集 seq_parameter_set_rbsp( )
8	图像参数集 pic_parameter_set_rbsp ( )
9	安全参数集 sec_parameter_set_rbsp( )
10	认证数据 authentication_data_rbsp( )
11	流结尾 end_of_stream_rbsp( )
12	非显示知识图像的编码片 patch_data_rbsp( )
13	符合GB/T 17975.3-2002标准技术规定
14	RL图像的编码片 patch_data_rbsp( )
15	访问单元分界 access_unit_delimiter_rbsp( )
16	编码视频序列结尾 end_of_cvs_rbsp( )
17	显示知识图像的编码片 patch_data_rbsp( )
18	非显示知识图像的分界编码片 patch_data_rbsp( )
19	隐私图像的编码片 privacy_patch_rbsp( )
20~31	保留

T/AI 109.8-2025

加密标志 encryption_idc

2 位无符号整数，取值范围为 0～1。表示 NAL 单元是否加密。值为‘0’表示该 NAL 单元中的 RBSP 没有加密，值不为‘0’则表示该 NAL 单元中的 RBSP 以安全参数集中指定的加密方法加密。RBSP 的最后一个字节不加密。数据加解密方法见附录 D。

当一个 NAL 单元的 nal_unit_type 为 3，7，8，9，10，11,15,16 时，encryption_idc 应为 0。

当一个 NAL 单元的 nal_unit_type 为 6 且包含 PayloadType 等于 25 或 26 或 127 的负载时，encryption_idc 应为 0。

认证标志 authentication_idc

2 位无符号整数，取值范围为 0～1。表示 NAL 单元是否认证，此时值为‘0’表示该 NAL 单元未经认证，值不为‘0’则表示该 NAL 单元以安全参数集指定的认证方法认证。数据认证方法示例见附录

E。

当一个 NAL 单元的 nal_unit_type 为 10 时，该 NAL 单元不参与签名认证。

当一个 NAL 单元的 nal_unit_type 为 11,15 或 16 时，authentication_idc 应为 0。

当一个 NAL 单元的 nal_unit_type 为 6 且包含 PayloadType 等于 25 或 26 的负载时，authentication_idc 应为 0。

认证数据标识 authentication_data_id

1 位无符号整数。取值范围 0～1，为该 NAL 单元参与的签名认证的认证数据的标识，应与其参与签名认证的认证数据 RBSP 中的 authentication_data_id 一致。同一个认证数据对应的参与认证的 NAL 单元的 authentication_data_id 应相同，且与认证数据中的 authentication_data_id 一致。一组参与共同签名认证的显示图像编码片 NAL 单元的 authentication_data_id 应与上一组参与共同签名认证的显示图像编码片 NAL 单元的 authentication_data_id 不同。

当一个 NAL 单元的 nal_unit_type 为 10 时，此 authentication_data_id 应与该 NAL 单元认证数据 RBSP 中的 authentication_data_id 保持一致。

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。时间层标识为 0 说明是最低层。当一个 NAL 单元的 nal_unit_type 为 7,8,9,10,11 或 16 时，temporal_id 应为 0。一个访问单元内的所有图像头 NAL 单元和所有编码片 NAL 单元的 temporal_id 应相同。一个访问单元的 temporal_id 为该访问单元内的编码片 NAL 单元的 temporal_id。如果一个访问单元包含 nal_unit_type 为 7,8 或 9 的 NAL 单元，该访问单元的 temporal_id 应为 0。

当一个 NAL 单元的 nal_unit_type 为 3,5,6 或 15 时，该 NAL 单元的 temporal_id 应等于其所在的访问单元的 temporal_id。

分层标识 layer_id

2 位无符号整数。说明当前图像的分层标识。分层标识的取值范围是 0～NumOfLayers-1。一个编码图像的图像头 NAL 单元和所有编码片 NAL 单元的 layer_id 应相同。LayerId 的值等于 layer_id 的值。一个编码图像或层单元的 LayerId 为该编码图像或层单元内的编码片 NAL 单元的 LayerId。

当一个 NAL 单元的 nal_unit_type 为 5,7,8,9,10 或 15 时，LayerId 应为 0。

当一个 NAL 单元的 nal_unit_type 为 6，且该 NAL 单元包括 PayloadType 为 19，25，26 或 127 的补充增强负载时，LayerId 应为 0。

负载的第 i 个字节 payload_byte[i]

8 位任意变量。表示一个 NAL 单元负载的第 i 个字节，等于 rbsp_byte[i]。一个 NAL 单元负载定义为一个字节的有序序列，包括一个 RBSP（如果 encryption_idc 等于 1，则为 RBSP 加密后生成的字节序列）。

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

一个 RBSP 定义为一个字节的有序序列，包含一个 SODB，如下所述：

a) 如果 SODB 为空（长度为 0 位），RBSP 也为空；

b) 否则 RBSP 包括如下 SODB：

RBSP 的第一个字节包括（最高位在前）8 位的 SODB；RBSP 的下一个字节应包括接下来的 8 位的 SODB，依此类推，直到剩下的 SODB 少于 8 位；

rbsp_trailing_bits( )用于 SODB 之后：最后 RBSP 字节中前面的(从最高位算起)位包括 SODB 的剩下的位（如果有的话）；下一位为单个 rbsp_stop_one_bit，其值为 1，并且当

T/AI 109.8-2025

rbsp_stop_one_bit 不是一个字节对齐的字节的最后一个位时，后面应存在一个或多个 rbsp_alignment_zero_bit 以形成一个字节对齐。

具有这些 RBSP 属性的语法结构在语法表中用“_rbsp”后缀表示。这些结构在 NAL 单元中作为 rbsp_byte[j]数据字节的内容携带。NAL 单元到 RBSP 语法结构的关联见表 57。

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

防伪码 emulation_prevention_three_byte

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

0x000000

0x000001

0x000002

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

0x00000300

0x00000301

0x00000302

0x00000303

6.2.2.2 NAL 单元的封装及约束

6.2.2.2.1 将 RBSP 封装为 NAL 单元

使用 emulation_prevention_three_byte 将一个 RBSP 封装到一个 NAL 单元中的目的：

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

b) 通过在 RBSP 的结尾查找其位 rbsp_stop_one_bit，以便能够识别 NAL 单元中 SODB 的结尾。

编码器通过下列步骤能够从一个 RBSP 中产生一个 NAL 单元：

从 RBSP 中查找字节对齐的下面二进制位串‘00000000 00000000 000000xx’，其中 xx 代表任意 2 位串‘00’、‘01’、‘10’或‘11’，并且在其中插入一个等于 0x03 的字节，形成‘00000000 00000000 00000011 000000xx’，得到的字节序列加上包含标识 RBSP 数据结构类型的 NAL 单元的单元头部分即形成了整个 NAL 单元。

该过程允许任何 SODB 在一个 NAL 单元中出现，同时可以确保：

a) 该 NAL 单元中没有字节对齐的伪起始码；

b) 无论是否字节对齐，在 NAL 单元中没有 8 个值为 0 的位后跟随起始码的序列。

6.2.2.2.2 NAL 单元的顺序及其与编码图像、访问单元和编码视频序列的关系

6.2.2.2.2.1 序列、图像参数及 RBSP 的顺序及生效

一个序列参数集 RBSP 包括的参数可以被一个或多个图像参数集 RBSP 和编码图像使用。LibraryStreamFlag 为 0 的序列参数集只对显示图像的编码图像生效，LibraryStreamFlag 为 1 的序列参数集只对知识图像的编码图像生效。

每个 LibraryStreamFlag 为 0 的序列参数集在其 RBSP 被解码器收到的同时生效，并且会导致先前的 LibraryStreamFlag 为 0 的有效序列参数集（如果有的话）失效。至多一个 LibraryStreamFlag 为

T/AI 109.8-2025

0 的序列参数集在解码过程中的指定时刻为有效的。每个 LibraryStreamFlag 为 1 的序列参数集在解码知识图像时生效，并在解码完一帧知识图像后立即失效。至多一个 LibraryStreamFlag 为 1 的序列参数集在解码过程中的指定时刻为有效的。

一个图像参数集 RBSP 包括的参数可以被一个或多个编码图像使用。紧跟在 LibraryStreamFlag 为 0 的序列参数集 NAL 单元后的图像参数集 NAL 单元表示的图像参数集只对显示图像的编码图像生效，紧跟在 LibraryStreamFlag 为 1 的序列参数集 NAL 单元后的图像参数集 NAL 单元表示的图像参数集只对知识图像的编码图像生效。

每个紧跟在 LibraryStreamFlag 为 0 的序列参数集 NAL 单元后的图像参数集 NAL 单元表示的图像参数集在其 RBSP 被解码器收到的同时生效，并且会导致先前的紧跟在 LibraryStreamFlag 为 0 的序列参数集 NAL 单元后的图像参数集 NAL 单元表示的有效图像参数集（如果有的话）失效。至多一个紧跟在 LibraryStreamFlag 为 0 的序列参数集 NAL 单元后的图像参数集 NAL 单元表示的图像参数集在解码过程中的指定时刻为有效的。每个紧跟在 LibraryStreamFlag 为 1 的序列参数集 NAL 单元后的图像参数集 NAL 单元表示的图像参数集在解码知识图像时生效，并在解码完一帧知识图像后立即失效。至多一个紧跟在 LibraryStreamFlag 为 1 的序列参数集 NAL 单元后的图像参数集 NAL 单元表示的图像参数集在解码过程中的指定时刻为有效的。

对序列参数集、图像参数集中的语法元素值与其他语法元素之间的关系所作出的规定，仅针对有效序列参数集和有效图像参数集。

在解码过程中，有效图像头和有效图像参数集的参数值应保持有效。

6.2.2.2.2.2 安全参数集 RBSP 的顺序及生效

安全参数集 RBSP 包括一些参数，这些参数可以被一个或多个其他类型的 NAL 单元使用。知识图像独立于显示图像单独加密或认证，以安全参数据 RBSP 中的 sec_is_library_flag 来区分是否为知识图像的安全参数集。RAS 的随机访问点访问单元或首个知识图像访问单元内前应存在安全参数集 NAL 单元，作用于当前的 RAS 或知识图像，安全参数集为当前 RAS 或知识图像访问单元的加密和认证提供参数；如果 RAS 的随机访问点访问单元内不存在安全参数集 NAL 单元，则认为当前的 RAS（不包括非显示知识图像访问单元）不加密和不参与认证；如果首个知识图像访问单元内不存在安全参数集 NAL 单元，则认为当前的知识图像不加密和不参与认证。安全参数集 NAL 单元应与序列参数集 NAL 单元位于同一访问单元内，且安全参数集 NAL 单元应位于序列参数集 NAL 单元前。如该访问单元内包括访问单元分界 NAL 单元，安全参数集 NAL 单元应位于访问单元分界 NAL 单元后。

注：当编码视频序列中存在非显示知识图像时，对于非 RL 前置知识图像，应有安全参数集 NAL 单元位于其包含 patch_index 的值为 0 的访问单元之内，对于 RL 前置知识图像，应有安全参数集 NAL 单元位于包含 patch_index 的值为 0 的访问单元之内，位于序列参数集 NAL 单元之前。

6.2.2.2.2.3 NAL 单元的顺序及其与访问单元和层单元的关系

一个访问单元包括连续的一个或多个 NAL 单元，且应包含至少一个编码片 NAL 单元。一个访问单元包括一个或多个层单元。一个层单元所包含的所有编码片 NAL 单元的 layer_id 取值应相同，一个访问单元内的不同层单元所包含的编码片 NAL 单元的 layer_id 取值应不同。一个层单元最多包含一个空域层编码图像。一个访问单元内的所有编码片 NAL 单元应按 layer_id 非降序排列。在一个随机访问片段内，每个访问单元中包含编码片 NAL 单元的层单元的数量应相同。

对于一个显示图像和显示知识图像，其所在访问单元应包含一个或多个空域层编码图像的全部编码片 NAL 单元。对于知识图像，其访问单元应包含一个或多个空域层编码图像的全部或部分编码片 NAL 单元。若一个知识图像的访问单元包含一个或多个空域层编码图像的 patch_index 的值为 0 的编

T/AI 109.8-2025

码片 NAL 单元，该访问单元应包含一个序列参数集 NAL 单元、一个图像参数集 NAL 单元和一个或多个空域层编码图像的图像头 NAL 单元。

一个层单元中的 NAL 单元应按照图像头 NAL 单元和编码片 NAL 单元的顺序排列。一个 IDR 图像、RAPI 图像、RL 图像和知识图像的访问单元中的安全参数集 NAL 单元（如存在）、序列参数集 NAL 单元、图像参数集 NAL 单元应依次排列并包含在访问单元的第一个层单元。一个 NRAP 图像的图像参数集 NAL 单元（如存在）应包含在访问单元的第一个层单元。

如果一个层单元中存在安全参数集 NAL 单元、序列参数集 NAL 单元或图像参数集 NAL 单元，这些 NAL 单元应位于该层单元的图像头 NAL 单元之前。

若一个访问单元中包含具有不同 layer_id 取值的编码片 NAL 单元，具有较小 layer_id 取值的编码片 NAL 单元应位于具有较大 layer_id 取值的编码片 NAL 单元前。

如果一个编码片 NAL 单元与前一个图像头 NAL 单元之间不存在其他编码片 NAL 单元，该编码片 NAL 单元为其所在的层单元的第一个编码片 NAL 单元。

如果一个层单元的第一个编码片 NAL 单元满足以下条件之一，那该编码片 NAL 单元是该层单元所在的访问单元的第一个编码片 NAL 单元：

——该编码片 NAL 单元的 layer_id 小于或等于按照解码顺序的前一个编码片 NAL 单元的 layer_id；

——该编码片 NAL 单元所在的编码图像的 DOI 不等于按照解码顺序的前一个编码片 NAL 单元所在的编码图像的 DOI。

对于一个访问单元的第一个编码片 NAL 单元 FirstCpNalUnit，如果按照顺序，存在以下 NAL 单元中第一个 NAL 单元，使其依解码次序位于 FirstCpNalUnit 之前并且位于 FirstCpNalUnit 前的逆序第一个编码片 NAL 单元之后，则指定该 NAL 单元为访问单元的开始：

a) 具有 nal_unit_type 等于 15 的 NAL 单元；
b) 具有 nal_unit_type 等于 9 的 NAL 单元；
c) 具有 nal_unit_type 等于 7 的 NAL 单元；
d) 具有 nal_unit_type 等于 8 的 NAL 单元；
e) 具有 nal_unit_type 等于 3 的 NAL 单元；
f) 具有 nal_unit_type 等于 6 的 NAL 单元。

对于 nal_unit_type 等于 12 或 18 的 NAL 单元的非显示知识图像所在的访问单元中的第一个编码片 NAL 单元 FirstCpNalUnit，应存在 nal_unit_type 等于 15 的 NAL 单元，使其依解码次序位于 FirstCpNalUnit 之前并且位于 FirstCpNalUnit 前的逆序一个编码片 NAL 单元之后，并指定该 NAL 单元为访问单元的开始。

访问单元与 NAL 单元的约束如下：

a) 如果存在具有 nal_unit_type 等于 15 的 NAL 单元，则其必须为访问单元的起始 NAL 单元，且一个访问单元中最多有一个 nal_unit_type 等于 15 的 NAL 单元；
b) 如果存在具有 nal_unit_type 等于 16 的 NAL 单元且该 NAL 单元之后不是具有 nal_unit_type 等于 11 的 NAL 单元，则该 NAL 单元应为访问单元的最后一个 NAL 单元；
c) 如果存在具有 nal_unit_type 等于 11 的 NAL 单元,则该 NAL 单元应为访问单元的最后一个 NAL 单元；
d) 如果存在具有 nal_unit_type 等于 19 的 NAL 单元，则该 NAL 单元应与所在访问单元中 nal_unit_type 等于 1，2，4，12，14，17 和 18 的 NAL 单元相邻；
e) 应在每个随机访问点图像所在的访问单元中传输包含 extension_id 为 4 的 nal_unit_type 为 5 的 NAL 单元；

T/AI 109.8-2025

f) 如果存在包含 PayloadType 为 25 的 nal_unit_type 为 6 的 NAL 单元，则该 NAL 单元应在每个序列起始图像和随机访问点图像所在的第一个访问单元中传输，且位于该访问单元第一个编码片 NAL 单元 FirstCpNalUnit 之前；

g) 如果存在包含 PayloadType 为 26 的 nal_unit_type 为 6 的 NAL 单元，则该 NAL 单元只应出现在显示知识图像的访问单元中；

h) 如果存在 nal_unit_type 等于 12、17、18 的 NAL 单元，访问单元内不应有除隐私图像编码片 NAL 单元外的其他类型的编码片 NAL 单元。

6.2.2.2.2.4 编码片 NAL 单元的顺序及其与编码图像的关系

每个编码片 NAL 单元都是一个编码图像的一部分。一个 IDR 图像的编码片 NAL 单元的 nal_unit_type 应等于 4，一个 RAPI 图像的编码片 NAL 单元的 nal_unit_type 应等于 2，一个 NRAP 图像的编码片 NAL 单元的 nal_unit_type 应等于 1，一个 RL 图像的编码片 NAL 单元的 nal_unit_type 应等于 14，一个非显示知识图像的编码片 NAL 单元的 nal_unit_type 应等于 12 或 18，一个显示知识图像的编码片 NAL 单元的 nal_unit_type 应等于 17。

一个编码图像中的编码片 NAL 单元的顺序规定如下：

a) 一个编码图像的编码片 NAL 单元顺序应按该图像的片索引递增的顺序；

b) 一个编码图像的图像头 NAL 单元应位于一个编码图像的第一个编码片 NAL 单元之前；

c) 具有 nal_unit_type 等于 6 的 NAL 单元应位于编码图像的第一个编码片 NAL 单元之前；

d) 具有 nal_unit_type 等于 12，18 的 NAL 单元可能与其余编码图像位流交织；

e) 具有 nal_unit_type 等于 19 的 NAL 单元与编码图像的同一编码片中 nal_unit_type 等于 1，2，4，12，14，17 和 18 的编码片 NAL 单元相邻。

6.2.3 序列参数集 RBSP 语义

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 F。

时域切换允许标志 temporal_nesting_flag

二值变量。值为‘1’指示对当前序列参数集作用范围内的所有 temporal_id > 0 的图像，图像 A 的时域 id 为 tIdA，图像 B 的时域 id 为 tIdB，图像 C 的时域 id 为 tIdC，其中 tIdB 小于等于 tIdA，tIdC 小于 tIdB，解码顺序为图像 B、图像 C、图像 A，则图像 A 不能参考图像 B，此时当前序列参数集作用范围内的所有 temporal_id > 0 的图像都可以允许作为时域向上切换点，即允许从图像 C 切换至图像 A；值为‘0’指示当前序列参数集作用范围内的所有 temporal_id > 0 的图像不添加额外的帧间预测限制，此时不允许当前序列参数集作用范围内的所有 temporal_id > 0 的图像作为时域向上切换点。

知识位流标志 library_stream_flag

二值变量。值为‘1’表示当前序列参数集是知识图像对应的序列参数集；值为‘0’表示当前序列参数集是显示图像对应的序列参数集。LibraryStreamFlag 的值等于 library_stream_flag 的值。

知识图像允许标志 library_picture_enable_flag

二值变量。值为‘1’表示编码视频序列中可存在使用知识图像作为参考图像的帧间预测图像；值为‘0’表示编码视频序列中不应存在使用知识图像作为参考图像的帧间预测图像。

T/AI 109.8-2025

LibraryPictureEnableFlag 的值等于 library_picture_enable_flag 的值。

知识位流重复序列参数集标志 duplicate_sequence_header_flag

二值变量。值为‘1’表示当前显示图像对应的序列参数集中除 library_stream_flag、bit_rate_lower、bit_rate_upper、cross_patch_loop_filter_enable_flag、ref_colocated_patch_flag、stable_patch_flag、uniform_patch_flag、patch_width_minus1、patch_height_minus1、duplicate_sequence_header_flag 和参考图像队列配置集中的参数外的所有语法元素的值均应与当前显示图像集所参考的知识图像对应的序列参数集中对应语法元素的值相同；值为‘0’表示当前知识图像对应的序列参数集中除 library_stream_flag、bit_rate_lower、bit_rate_upper cross_patch_loop_filter_enable_flag、ref_colocated_patch_flag、stable_patch_flag、uniform_patch_flag、patch_width_minus1、patch_height_minus1、duplicate_sequence_header_flag 和参考图像队列配置集中的参数外的其它语法元素的值可与当前显示图像集所参考的知识图像的序列参数集中对应语法元素的值不同。

知识图像模式索引 library_picture_mode_index

2 位无符号整数，取值范围 0～3。指示知识图像模式，见表 58。值为‘0’表示知识图像单流模式，值为‘1’表示长期参考图像模式，值为‘2’表示知识图像双流模式，值‘3’为预留。LibraryPictureModeIndex 的值等于 library_picture_mode_index 的值，如果位流中不存在 library_picture_mode_index，LibraryPictureModeIndex 的值等于 0。

注：知识图像单流模式下，编码视频序列中仅允许存在非显示知识图像，不允许存在显示知识图像，非显示知识图像编码片 NAL 单元可与显示图像编码片 NAL 单元交织构成一个编码视频序列。长期参考模式下，编码视频序列中仅允许存在显示知识图像，不允许存在非显示知识图像，显示知识图像的所有编码片 NAL 单元按照编码片索引递增的顺序连续，不应与显示图像的访问单元交织。知识图像双流模式下，主位流编码视频序列中仅允许存在显示图像编码 NAL 单元，知识位流编码图像序列中仅允许存在知识图像编码片 NAL 单元，根据 LibraryStreamFlag 判断当前编码视频序列位于主位流或是知识位流中。

表58 知识图像模式索引

library_picture_mode_index 值	含义
0	知识图像单流模式
1	长期参考帧模式
2	知识图像双流模式
3	保留

RAPI NAL 单元类型使能标志 rapi_nalu_type_enable_flag

二值变量。值为‘1’表示编码视频序列中允许存在值为 2 的 NAL 单元类型。值为‘0’表示编码视频序列中不允许存在 nal_unit_type 的值为 2 的 NAL 单元类型。

空域分层层数减1 num_of_layers_minus1

2 位无符号整数。加 1 标识空域分层结构的层数。NumOfLayers 的值等于 num_of_layers_minus1+1，如果位流中不存在 num_of_layers_minus1，NumOfLayers 的值等于 1。NumOfLayers 取值范围应为 1～MAX_LAYER。

符合本文件的位流应满足以下约束：

——如果 SsvcEnableFlag 为 1，NumOfLayers 的值应大于 1；

——如果 SsvcEnableFlag 为 1，且 PrivacyModeIndex 的值为 0 或 1，NumOfLayers 的值应为 4；

——如果 SsvcEnableFlag 为 0，且 PrivacyModeIndex 的值为 0 或 1，NumOfLayers 的值应为 2；

T/AI 109.8-2025

——如果 SsvcEnableFlag 为 0，且 PrivateModeIndex 的值不为 0 或 1，NumOfLayers 应为 1。

空域分层编码允许标志 ssvc_enable_flag

二值变量。值为‘1’表示开启空域分层编码；值为‘0’表示不开启空域分层编码。SsvcEnableFlag 的值等于 ssvc_enable_flag 的值，如果位流中不存在 ssvc_enable_flag，SsvcEnableFlag 的值为 0。

权限保护允许标志 privacy_enable_flag

二值变量。值为‘1’表示可使用权限保护；值为‘0’表示不应使用权限保护。PrivacyEnableFlag 的值等于 privacy_enable_flag 的值，如果位流中不存在 privacy_enable_flag，PrivacyEnableFlag 的值为 0。

权限保护模式索引 privacy_mode_index

2 位无符号整数，取值范围 0～3。指示权限保护模式。值为‘0’表示隐私保护分层独立编码模式，值为‘1’表示隐私保护分层参考编码模式，值为‘3’表示隐私保护单层编码模式，值为‘2’表示支持遮蔽隐私的编码模式。PrivacyModeIndex 的值等于 privacy_mode_index 的值，如果位流中不存在 privacy_mode_index，PrivacyModeIndex 的值等于-1。

如果 PrivacyModeIndex 的值等于 0 或 1，LayerId 等于 0 和 2（如存在）的空域层为低权限编码层，LayerId 等于 1 和 3（如存在）的空域层为高权限编码层。隐私保护单层编码模式低权限解码填充方法见附录 G。

变分辨率编码开启标志 reference_picture_resampling_enable_flag

二值变量，值为‘1’表示当前序列允许使用变分辨率编码；值为‘0’表示当前序列不允许使用变分辨率编码。ReferencePictureResamplingEnableFlag 的值等于 reference_picture_resampling_enable_flag。如果位流中不存在 reference_picture_resampling_enable_flag，ReferencePictureResamplingEnableFlag 的值等于 0。

零保留码 reserved_zero_bit

二值变量。应等于‘0’。

空域层独立编码总标志 sps_all_independent_layers_flag

二值变量。值为‘1’表示所有的空域层独立编码，不采用层间依赖编码。值为‘0’表示存在层间依赖编码。SpsAllIndependentLayersFlag 的值等于 sps_all_independent_layers_flag 的值，如果位流中不存在 sps_all_independent_layers_flag，SpsAllIndependentLayersFlag 的值等于 1。

如果满足以下任意条件之一，SpsAllIndependentLayersFlag 的值应为 1：

--PrivacyModeIndex 的值等于 0；

--LibraryPictureEnableFlag 的值等于 1；

--ReferencePictureResamplingEnableFlag 的值等于 1。

空域第i层独立编码标志 sps_independent_layer_flag[i]

二值变量。值为‘1’表示 LayerId 等于 i 的空域层独立编码，不采用层间依赖编码。值为‘0’表示 LayerId 等于 i 的空域层可以采用层间依赖编码。SpsIndependentLayerFlag[i]的值等于 sps_independent_layer_flag[i]的值，如果位流中不存在 sps_independent_layer_flag[i]，SpsIndependentLayerFlag[i]的值等于 1。如果 PrivacyModeIndex 的值等于 1，且 i 等于 1 或 3，SpsIndependentLayerFlag[i]的值应为 0。

如果 PrivacyModeIndex 的值等于 1，且 i 等于 0 或 2，SpsIndependentLayerFlag[i]的值应为 1。

空域第i层依赖层号 ref_layer_id[i]

T/AI 109.8-2025

2 位无符号整数。规定 LayerId 等于 i 的空域层的层间依赖编码的参考层的 LayerId 的值。ref_layer_id[i]的取值范围为 0～i-1。RefLayerId[i]的值等于 ref_layer_id[i]的值，如果位流中不存在 ref_layer_id[i]，RefLayerId[i]的值等于 i。如果 PrivacyModeIndex 的值等于 1，且 i 等于 1 或 3，RefLayerId[i]的值应为 i-1。

序列参数集水平尺寸 sps_horizontal_size[i]

14 位无符号整数。规定 LayerId 等于 i 的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。SpsHorizontalSize[i]的值等于 sps_horizontal_size。

SpsHorizontalSize[LayerId]不应为‘0’，且必须是 4 的整数倍。SpsHorizontalSize[LayerId]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。

序列参数集垂直尺寸 sps_vertical_size[i]

14 位无符号整数。规定 LayerId 等于 i 的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。SpsVerticalSize[i]的值等于 sps_vertical_size。

SpsVerticalSize[LayerId]不应为 0，且必须是 4 的整数倍。SpsVerticalSize[LayerId]的单位应是图像样本的行数。

序列参数集水平编码尺寸 sps_pic_width_in_luma[i]

14 位无符号整数。规定 LayerId 等于 i 的编码图像宽度。SpsPicWidthInLuma[i]的值等于 sps_pic_width_in_luma[i]。SpsPicWidthInLuma[i]的值不能为 0，且必须为 MiniSize 的整数倍。

SpsHorizontalSize[i]的值必须小于或等于 SpsPicWidthInLuma[i]的值。

序列参数集垂直编码尺寸 sps_pic_height_in_luma[i]

14 位无符号整数。规定 LayerId 等于 i 的编码图像高度。SpsPicHeightInLuma[i]的值等于 sps_pic_height_in_luma[i]。SpsPicHeightInLuma[i]的值不能为 0，且必须为 MiniSize 的整数倍。

SpsVerticalSize[i]的值必须小于或等于 SpsPicHeightInLuma[i]的值。

序列参数集变分辨率水平编码尺寸 sps_rpr_pic_width_in_luma[i]

14 位无符号整数，规定 LayerId 等于 i 的图像的变分辨率后的编码图像宽度。SpsRprPicWidthInLuma[i]的值等于 sps_rpr_pic_width_in_luma[i]。SpsRprPicWidthInLuma[i]的值不能为 0，且必须为 MiniSize 的整数倍。SpsRprPicWidthInLuma[i]的值应小于或等于(SpsPicWidthInLuma[i] >> 1) + (1 << LcuSizeInBit)。SpsRprPicWidthInLuma[i]的值应大于或等于(SpsPicWidthInLuma[i] >> 1)。

序列参数集变分辨率垂直编码尺寸 sps_rpr_pic_height_in_luma[i]

14 位无符号整数，规定 LayerId 等于 i 的图像的变分辨率后的编码图像高度。SpsRprPicHeightInLuma[i]的值等于 sps_rpr_pic_height_in_luma[i]。SpsRprPicHeightInLuma[i]的值不能为 0，且必须为 MiniSize 的整数倍。SpsRprPicHeightInLuma[i]的值应小于或等于(SpsPicHeightInLuma[i] >> 1) + (1 << LcuSizeInBit)。SpsRprPicHeightInLuma[i]的值应大于或等于(SpsPicHeightInLuma[i] >> 1)。

符合本文件的位流需要满足以下约束：

a) 如果 NumOfLayers 大于 1，对于任意小于 NumOfLayers 的无符号整数 k 和 m（k 小于 m），应满足：

SpsHorizontalSize[k] <= SpsHorizontalSize[m]

SpsVerticalSize[k] <= SpsVerticalSize[m]

SpsPicWidthInLuma[k] <= SpsPicWidthInLuma[m]

SpsPicHeightInLuma[k] <= SpsPicHeightInLuma[m]

SpsRprPicWidthInLuma[k] <= SpsRprPicWidthInLuma[m]

T/AI 109.8-2025

SpsRprPicHeightInLuma[k] <= SpsRprPicHeightInLuma[m]

b) 如果 PrivacyModeIndex 的值为 0 或 1，对于 i 等于 1 或 3，应满足：

SpsHorizontalSize[i] = SpsHorizontalSize[i-1]

SpsVerticalSize[i] = SpsVerticalSize[i-1]

SpsPicWidthInLuma[i] = SpsPicWidthInLuma[i-1]

SpsPicHeightInLuma[i] = SpsPicHeightInLuma[i-1]

SpsRprPicWidthInLuma[i] = SpsRprPicWidthInLuma[i-1]

SpsRprPicHeightInLuma[i] = SpsRprPicHeightInLuma[i-1]

c) 如果 PrivacyModeIndex 的值不为 1，且 SpsIndependentLayerFlag[i]的值为 0（i=1～NumOfLayers-1），应满足：

SpsHorizontalSize[i] = SpsHorizontalSize[RefLayerId[i]] * 2

SpsVerticalSize[i] = SpsVerticalSize[RefLayerId[i]] * 2

色度格式 chroma_format

2 位无符号整数。规定色度分量的格式，见表 59。

表59 色度格式

chroma_format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 60。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60 样本精度

sample_precision值	含义
000	禁止
001	亮度和色度均为8 bit精度
010	亮度和色度均为10 bit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 61。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，BitDepth 的值为 8。BitDepth 的值不应小于 SamplePrecision 的值。

T/AI 109.8-2025

表61 编码样本精度

encoding_precision值	含义
000	禁止
001	亮度和色度均为8 bit精度
010	亮度和色度均为10 bit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 62。

表62 宽高比

aspect_ratio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。样本宽高比为：

SAR DAR *SpsVerticalSize[LayerId] ÷ SpsHorizontalSize[LayerId]

注：在这种情况下，SpsHorizontalSize[LayerId]和SpsVerticalSize[LayerId]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，样本宽高比为：

SAR DAR * display_vertical_size ÷ display_horizontal_size

帧率代码 frame_rate_code

4 位无符号整数。规定帧率，见表 63。

表63 帧率代码

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

T/AI 109.8-2025

表 63 帧率代码（续）

frame_rate_code值	帧率
1001	100
1010	120
1011	200
1100	240
1101	300
1110	120000÷1001 (119.88...)
1111	保留

连续两帧之间的时间间隔是帧率的倒数。

帧周期等于帧率的倒数。

位率低位 bit_rate_lower

BitRate 的低 18 位。

位率高位 bit_rate_upper

BitRate 的高 12 位。BitRate 为：

```
BitRate = (bit_rate_upper << 18) + bit_rate_lower
```

BitRate 以 400bit/s 为单位计算视频位流的位率，并向上取整。BitRate 不应为 0。若 LibraryPictureEnableFlag 的值为 1 且 LibraryPictureModeIndex 的值为 2，视频位流的位率包含了该主位流以及所参考的知识位流的总位率。

低延迟 low_delay

二值变量。值为‘1’表示参考图像队列 0 和参考图像队列 1 中均不包含显示顺序上位于当前图像之后的图像，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 H）；值为‘0’表示参考图像队列 0 或参考图像队列 1 中可包含显示顺序上将来的图像，存在图像重排序延时，位流中不包含所谓“大图像”（见附录 H）。如果位流中不存在 low_delay，low_delay 的值等于 1。

时间层标识允许标志 temporal_id_enable_flag

二值变量。值为‘1’表示编码视频序列允许使用时间层标识；值为‘0’表示编码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。规定了位流参考解码器对编码视频序列解码的位流缓冲区尺寸（见附录 H）。BBS 是位流参考解码器对编码视频序列解码所需的位流缓冲区最小尺寸（按位计算）：

```
BBS     16 * 1024 * bbv_buffer_size
```

最大解码图像缓冲区大小 max_dpb_size_minus1

4 位无符号整数。表示解码当前位流所需要的最大的解码图像缓冲区的大小（以单幅图像存储缓冲区大小为单位）。max_dpb_size_minus1 的值应大于 0，小于 16 且小于当前位流的 level_id 对应的最大 DPB 值（具体由级别规定，见附录 F）。MaxDpbSize 的值等于 max_dpb_size_minus1 加 1。

参考图像队列 1 索引存在标志 rpl1_index_exist_flag[i]

二值变量。值为‘0’表示 LayerId 等于 i 的图像位流中不应出现 ref_pic_list_set_flag[1]和 ref_pic_list_set_index[1]；值为‘1’表示可出现。Rpl1IndexExistFlag[i]的值等于 rpl1_index_exist_flag[i]的值。

参考图像队列相同标志 rpl1_same_as_rpl0_flag[i]

T/AI 109.8-2025

二值变量。值为‘0’表示 LayerId 等于 i 的图像位流中应出现 num_ref_pic_list_set[i][1]和 reference_picture_list_set(i, 1, rplsIndex)；值为‘1’表示不应出现。Rpl1SameAsRpl0Flag[i]值等于 rpl1_same_as_rpl0_flag[i]的值。

如果 Rpl1SameAsRpl0Flag[i]的值为‘1’，则 num_ref_pic_list_set[i][1]值等于 num_ref_pic_list_set[i][0]值，且参考图像队列配置集 reference_picture_list_set(i, 1,rplsIndex)中每个语法元素的值与参考图像队列配置集 reference_picture_list_set(i, 0,rplsIndex)中对应语法元素的值相同，其中 rplsIndex 的取值范围是 0～(num_ref_pic_list_set[i][0]－1)。

参考图像队列配置集数 num_ref_pic_list_set[i][0]、num_ref_pic_list_set[i][1]

表示 LayerId 等于 i 的图像的参考图像队列配置集的数量。解析过程见 7.2。取值范围是 0～64。NumRefPicListSet[i][0]的值等于 num_ref_pic_list_set[i][0]值。如果 Rpl1SameAsRpl0Flag[i]的值为 1，NumRefPicListSet[i][1]的值等于 num_ref_pic_list_set[i][0]的值；否则 NumRefPicListSet[i][1]的值等于 num_ref_pic_list_set[i][1]值。

默认活跃参考图像数 num_ref_default_active_minus1[i][0]、num_ref_default_active_minus1[i][1]

表示解码图像时，LayerId 等于 i 的图像的参考图像队列中参考索引（pu_reference_index_l0、pu_reference_index_l1）默认的最大值。解析过程见 7.2。取值范围是 0～14。

NumRefDefaultActive[i][0]的值等于 num_ref_default_active_minus1[i][0]，NumRefDefaultActive[i][1]的值等于 num_ref_default_active_minus1[i][1]。

最大编码单元尺寸 log2_lcu_size_minus2

3 位无符号整数。表示最大编码单元的大小，取值范围是 3～5。LcuSizeInBit 的值等于 log2_lcu_size_minus2 的值加 2。MaxQtSize 的值等于 $2^{\text{LcuSizeInBit}}$。

最小编码单元尺寸 log2_min_cu_size_minus2

2 位无符号整数。表示最小编码单元的大小，取值范围是 0～2。MinCuSize、MinBtSize 和 MinEqtSize 的值均等于 $2^{\text{log2_min_cu_size_minus2+2}}$。

划分单元最大比例 log2_max_part_ratio_minus2

2 位无符号整数。表示最大编码单元宽比高或者高比宽的比值，MaxPartRatio 的值等于 $2^{\text{log2_max_part_ratio_minus2+2}}$。

编码树最大划分次数 max_split_times_minus6

3 位无符号整数。表示所允许的编码单元最大划分次数。四叉树、二叉树、扩展四叉树每划分一次，编码单元划分深度加 1。MaxSplitTimes 的值等于 max_split_times 的值加 6。

最小四叉树尺寸 log2_min_qt_size_minus2

3 位无符号整数。表示所允许的四叉树划分最小编码单元尺寸，取值范围是 0～5。MinQtSize 的值等于 $2^{\text{log2_min_qt_size_minus2+2}}$。

最大二叉树尺寸 log2_max_bt_size_minus2

3 位无符号整数。表示所允许的二叉树划分最大编码单元尺寸，取值范围是 0～5。MaxBtSize 的值等于 $2^{\text{log2_max_bt_size_minus2+2}}$。

最大扩展四叉树尺寸 log2_max_eqt_size_minus3

2 位无符号整数。表示所允许的扩展四叉树最大编码单元尺寸，取值范围是 0～3。MaxEqtSize 的值等于 $2^{\text{log2_max_eqt_size_minus3+3}}$。

加权量化允许标志 weight_quant_enable_flag

二值变量。值为‘1’表示编码视频序列允许使用加权量化；值为‘0’表示编码视频序列不使用

T/AI 109.8-2025

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表 15（见 6.1.5）从序列参数集中加载；值为‘0’表示 4×4 和 8×8 变换块的加权量化矩阵根据附录 I 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。

二次变换允许标志 st_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。StEnableFlag 的值等于 st_enable_flag 的值。

样值偏移补偿允许标志 sao_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 sao_enable_flag 的值。

跨分量样值偏移补偿允许标志位 ccsao_enable_flag

二值变量。值为‘1’表示可使用跨分量样值偏移补偿；值为‘0’表示不应使用跨分量样值偏移补偿。CcsaoEnableFlag 的值等于 ccsao_enable_flag 的值。如果位流中不存在 ccsao_enable_flag，CcsaoEnableFlag 的值等于 0。

自适应修正滤波允许标志 alf_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 alf_enable_flag 的值。

仿射运动补偿允许标志 affine_enable_flag

二值变量。值为‘1’表示可使用仿射运动补偿；值为‘0’表示不应使用仿射运动补偿。AffineEnableFlag 的值等于 affine_enable_flag 的值。

仿射预测样本改善允许标志 asr_enable_flag

二值变量。值为‘1’表示可使用仿射预测样本改善；值为‘0’表示不应使用仿射预测样本改善。AsrEnableFlag 的值等于 asr_enable_flag 的值。如果位流中不存在 asr_enable_flag，AsrEnableFlag 的值为 0。

对称运动矢量差模式允许标志 smvd_enable_flag

二值变量。值为‘1’表示可使用对称运动矢量差模式；值为‘0’表示不应使用对称运动矢量差模式。SmvdEnableFlag 的值等于 smvd_enable_flag 的值。

脉冲编码调制模式允许标志 ipcm_enable_flag

二值变量。值为‘1’表示可使用脉冲编码调制模式；值为‘0’表示不应使用脉冲编码调制模式。IpcmEnableFlag 的值等于 ipcm_enable_flag 的值。

自适应运动矢量精度允许标志 amvr_enable_flag

二值变量。值为‘1’表示可使用自适应运动矢量精度；值为‘0’表示不应使用自适应运动矢量精度。AmvrEnableFlag 的值等于 amvr_enable_flag 的值。

帧间预测候选历史运动信息数 num_of_hmvp_cand

4 位无符号整数。NumOfHmvpCand 的值等于 num_of_hmvp_cand 的值，取值范围是 0～8。NumOfHmvpCand 的值为 0 表示进行帧间预测时不应使用历史运动信息。

帧内预测滤波允许标志 intra_pf_enable_flag

二值变量。值为‘1’表示可使用帧内预测滤波；值为‘0’表示不应使用帧内预测滤波。IntraPfEnableFlag 的值等于 intra_pf_enable_flag。

高级运动矢量表达模式允许标志 umve_enable_flag

T/AI 109.8-2025

二值变量。值为‘1’表示可使用高级运动矢量表达模式；值为‘0’表示不应使用高级运动矢量表达模式。UmveEnableFlag 的值等于 umve_enable_flag 的值。

运动矢量精度扩展模式允许标志 emvr_enable_flag

二值变量。值为‘1’表示可使用运动矢量精度扩展模式；值为‘0’表示不应使用运动矢量精度扩展模式。EmvrEnableFlag 的值等于 emvr_enable_flag 的值。

跨分量两步预测模式允许标志 tscpm_enable_flag

二值变量。值为‘1’表示可使用跨分量两步预测模式；值为‘0’表示不应使用跨分量两步预测模式。TscpmEnableFlag 的值等于 tscpm_enable_flag 的值。

帧内衍生模式允许标志 dt_enable_flag

二值变量。值为‘1’表示可使用帧内衍生模式；值为‘0’表示不应使用帧内衍生模式。DtEnableFlag 的值等于 dt_enable_flag 的值。

衍生模式待划分边长最大尺寸 log2_max_dt_size_minus4

2 位无符号整数。表示所允许的衍生模式待划分边长的最大值。取值范围是 0～2。DtMaxSize 的值等于 $2^{\text{log2_max_dt_size_minus4}+4}$。DtMinSize 的值等于 16。

基于位置的变换允许标志 pbt_enable_flag

二值变量。值为‘1’表示可使用基于位置的变换；值为‘0’表示不应使用基于位置的变换。PbtEnableFlag 的值等于 pbt_enable_flag 的值。

重叠块运动补偿允许标志 obmc_enable_flag

二值变量。值为‘1’表示可使用重叠块运动补偿模式；值为‘0’表示不应使用重叠块运动补偿模式。ObmcEnableFlag 的值等于 obmc_enable_flag 的值。如果位流中不存在 obmc_enable_flag，ObmcEnableFlag 的值为 0。

帧间预测滤波允许标志 inter_pf_enable_flag

二值变量。值为‘1’表示可使用帧间预测滤波；值为‘0’表示不应使用帧间预测滤波。InterPfEnableFlag 的值等于 inter_pf_enable_flag 的值。如果位流中不存在 inter_pf_enable_flag，InterPfEnableFlag 的值为 0。

帧间预测修正允许标志 inter_pc_enable_flag

二值变量。值为‘1’表示可使用帧间预测修正；值为‘0’表示不应使用帧间预测修正。InterPcEnableFlag 的值等于 inter_pc_enable_flag 的值。如果位流中不存在 inter_pc_enable_flag，InterPcEnableFlag 的值为 0。

隐择变换允许标志 ist_enable_flag

二值变量。值为‘1’表示可使用隐择变换；值为‘0’表示不应使用隐择变换。IstEnableFlag 的值等于 ist_enable_flag。如果位流中不存在 ist_enable_flag，IstEnableFlag 的值为 0。

块复制帧内预测允许标志 ibc_enable_flag

二值变量。值为‘1’表示可使用块复制帧内预测；值为‘0’表示不应使用块复制帧内预测。IbcEnableFlag 的值等于 ibc_enable_flag 的值。如果位流中不存在 ibc_enable_flag，IbcEnableFlag 的值为 0。

串复制帧内预测允许标志 isc_enable_flag

二值变量。值为‘1’表示可使用串复制帧内预测；值为‘0’表示不应使用串复制帧内预测。IscEnableFlag 的值等于 isc_enable_flag 的值。如果位流中不存在 isc_enable_flag，IscEnableFlag 的值为 0。

帧内预测候选历史运动信息数 num_of_intra_hmvp_cand

4 位无符号整数。NumOfIntraHmvpCand 的值等于 num_of_intra_hmvp_cand 的值，取值范围是 0～

T/AI 109.8-2025

12。NumOfIntraHmvpCand 的值为 0 表示进行帧内预测时不应使用历史运动信息。

基于频数信息的帧内编码允许标志 fimc_enable_flag

二值变量。值为‘1’表示可使用基于频数信息的帧内编码；值为‘0’表示不应使用基于频数信息的帧内编码。FimcEnableFlag 的值等于 fimc_enable_flag 的值。如果位流中不存在 fimc_enable_flag，FimcEnableFlag 的值等于 0。

子块变换允许标志 sbt_enable_flag

二值变量。值为‘1’表示可使用子块变换。值为‘0’表示不应使用子块变换。SbtEnableFlag 的值等于 sbt_enable_flag 的值。如果位流中不存在 sbt_enable_flag，SbtEnableFlag 的值为 0。

双向光流允许标志 bio_enable_flag

二值变量。值为‘1’表示可使用双向光流；值为‘0’表示不应使用双向光流。BioEnableFlag的值等于bio_enable_flag的值。如果位流中不存在bio_enable_flag，BioEnableFlag的值为0。

解码端运动矢量改良允许标志 dmvr_enable_flag

二值变量。值为‘1’表示可使用解码端运动矢量改良；值为‘0’表示不应使用解码端运动矢量改良。DmvrEnableFlag 的值等于 dmvr_enable_flag 的值。如果位流中不存在 dmvr_enable_flag，DmvrEnableFlag的值为0。

双向梯度修正允许标志 bgc_enable_flag

二值变量。值为‘1’表示可使用双向梯度修正；值为‘0’表示不应使用双向梯度修正。BgcEnableFlag 的值等于bgc_enable_flag的值。如果位流中不存在bgc_enable_flag，BgcEnableFlag的值为0。

跨多分量预测允许标志 pmc_enable_flag

二值变量。值为‘1’表示可使用跨多分量预测；值为‘0’表示不应使用跨分量预测。PmcEnableFlag 的值等于 pmc_enable_flag 的值。如果位流中不存在 pmc_enable_flag，PmcEnableFlag 的值为 0。

改进型帧内预测允许标志 iip_enable_flag

二值变量。值为‘1’表示可使用改进型帧内预测；值为‘0’表示不应使用改进型帧内预测。IipEnableFlag 的值等于 iip_enable_flag。如果位流中不存在 iip_enable_flag，IipEnableFlag 的值为 0。

空域角度加权预测模式允许标志 sawp_enable_flag

二值变量。值为‘1’表示可使用空域角度加权预测模式；值为‘0’表示不应使用空域角度加权预测模式。SawpEnableFlag 的值等于 sawp_enable_flag 的值。如果位流中不存在 sawp_enable_flag，SawpEnableFlag 的值为 0。

多组合帧内预测滤波允许标志 mipf_enable_flag

值为‘1’表示可使用多组合帧内预测滤波；值为‘0’表示不应使用多组合帧内预测滤波。MipfEnableFlag 的值等于 mipf_enable_flag 的值。如果位流中不存在 mipf_enable_flag，MipfEnableFlag 的值为 0。

角度加权预测模式允许标志 awp_enable_flag

二值变量。值为‘1’表示可使用角度加权预测模式；值为‘0’表示不应使用角度加权预测模式。AwpEnableFlag 的值等于 awp_enable_flag 的值。如果位流中不存在 awp_enable_flag，AwpEnableFlag 的值为 0。

时域运动信息预测允许标志 tmvp_enable_flag

值为‘1’表示可使用时域运动信息预测；值为‘0’表示不应使用时域运动信息预测。TmvpEnableFlag 的值等于 tmvp_enable_flag 的值。如果位流中不存在 tmvp_enable_flag，TmvpEnableFlag 的值为 0。

T/AI 109.8-2025

子块时域运动信息预测允许标志 sbtmvp_enable_flag

值为‘1’表示可使用子块时域运动信息预测；值为‘0’表示不应使用子块时域运动信息预测。SbTmvpEnableFlag 的值等于 sbtmvp_enable_flag 的值。如果位流中不存在 sbtmvp_enable_flag，SbTmvpEnableFlag 的值为 0。

变换跳过模式允许标志 ts_enable_flag

二值变量。值为‘1’表示可使用变换跳过模式；值为‘0’表示不应使用变换跳过模式。TsEnableFlag 的值等于 ts_enable_flag 的值。如果位流中不存在 ts_enable_flag，TsEnableFlag 的值为 0。

编码单元量化参数可变允许标志 cu_dqp_enable_flag

二值变量。值为‘1’表示可使用编码单元量化参数可变；值为‘0’表示不可使用编码单元量化参数可变。CuDqpEnableFlag 的值等于 cu_dqp_enable_flag 的值。如果位流中不存在 cu_dqp_enable_flag，CuDqpEnableFlag 的值为 0。

基于扫描区域的系数编码允许标志 srcc_enable_flag

二值变量。值为‘1’表示可使用基于扫描区域的系数编码；值为‘0’表示不应使用基于扫描区域的系数编码。SrccEnableFlag 的值等于 srcc_enable_flag 的值。如果位流中不存在 srcc_enable_flag，SrccEnableFlag 的值为 0。

多假设概率模型高级熵编码允许标志 maec_enable_flag

二值变量。值为‘1’表示可使用多假设概率模型高级熵编码；值为‘0’表示不应使用多假设概率模型高级熵编码。MaecEnableFlag 的值等于 maec_enable_flag 的值。如果位流中不存在 maec_enable_flag，MaecEnableFlag 的值为 0。

改进型去块滤波允许标志 dbr_enable_flag

二值变量。值为‘1’表示可使用改进型去块滤波；值为‘0’表示不应使用改进型去块滤波。DbrEnableFlag 的值等于 dbr_enable_flag 的值。如果位流中不存在 dbr_enable_flag，DbrEnableFlag 的值为 0。

增强样值偏移补偿允许标志 esao_enable_flag

二值变量。值为‘1’表示可使用增强样值偏移补偿；值为‘0’表示不应使用增强样值偏移补偿。EsaoEnableFlag 的值等于 esao_enable_flag 的值。如果位流中不存在 esao_enable_flag，EsaoEnableFlag 的值为 0。如果 EsaoEnableFlag 的值为 1，SaoEnableFlag 的值应为 0。

增强时域运动矢量预测和运动矢量角度预测允许标志 etmvp_mvap_enable_flag

二值变量。值为‘1’表示可使用增强时域运动矢量预测和运动矢量角度预测；值为‘0’表示不应使用增强时域运动矢量预测和运动矢量角度预测。EtmvpMvapEnableFlag 的值等于 etmvp_mvap_enable_flag 的值。如果位流中不存在 etmvp_mvap_enable_flag，EtmvpMvapEnableFlag 的值为 0。如果 EtmvpMvapEnableFlag 的值为 1，NumOfMvapCand 的值等于 5；否则 NumOfMvapCand 的值等于 0。

增强自适应修正滤波允许标志 ealf_enable_flag

二值变量。值为‘1’表示应使用增强自适应修正滤波；值为‘0’表示不应使用增强自适应修正滤波。EalfEnableFlag 的值等于 ealf_enable_flag 的值。如果位流中不存在 ealf_enable_flag，EalfEnableFlag 的值为 0。

神经网络工具集 nn_tools_set_hook

8 位无符号整数。标识是否使用神经网络的工具。如果位流中不存在 nn_tools_set_hook，nn_tools_set_hook 的值应为‘00000000’。解码处理应该忽略这些位。

nn_tools_set_hook 最低位的值为‘1’表示可使用神经网络滤波；nn_tools_set_hook 最低位的

T/AI 109.8-2025

值为‘0’表示不应使用神经网络滤波。NnFilterEnableFlag 的值等于 nn_tools_set_hook & 0x01 的值。如果位流中不存在 nn_tools_set_hook，NnFilterEnableFlag 的值等于 0。其他位保留。

神经网络滤波模型数 num_of_nn_filter_minus1

表示神经网络滤波可使用的滤波模型数量。解析过程见 7.2。NumOfNnFilter 的值等于 num_of_nn_filter_minus1 的值加 1。如果位流中不存在 num_of_nn_filter_minus1，NumOfNnFilter 的值等于 0。解码处理应该忽略这些位。

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。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。

跨片环路滤波允许标志 cross_patch_loop_filter_enable_flag

二值变量。值为‘1’时表示可跨越片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’表示不应跨越片边界进行去块效应滤波、样本偏移补偿及自适应修正滤波。CplfEnableFlag 的值等于 cross_patch_loop_filter_enable_flag 的值。

片划分一致性标志 stable_patch_flag[i]

二值变量。LibraryStreamFlag 等于 0 时，值为‘1’时表示 LayerId 等于 i 的当前显示图像集中所有图像划分为片的方式均应相同；值为‘0’表示当前显示图像集中图像划分为片的方式可不相同。LibraryStreamFlag 的值为 1 时，stable_patch_flag[i]的值应为‘1’。

参考同位置片标志 ref_colocated_patch_flag

二值变量。值为‘1’时表示进行帧间预测时只使用参考图像同位置片边界内的采样值进行参考；值为‘0’表示进行帧间预测时可使用参考图像同位置片边界外的采样值进行参考。

统一片大小标志 uniform_patch_flag[i]

二值变量。值为‘1’时表示 LayerId 等于 i 的图像中除最右边和最下边的片外，图像中其他片的大小应相同；值为‘0’表示片的大小可不同。

片宽度 patch_width_minus1[i]

片高度 patch_height_minus1[i]

LayerId 等于 i 的图像中片的宽度和高度，以 LCU 为单位。patch_width_minus1 的值应小于 256，patch_height_minus1 的值应小于 144。PatchWidth[i]的值等于 patch_width_minus1[i]加 1，PatchHeight[i]的值等于 patch_height_minus1[i]加 1。

图像中各个片以 LCU 为单位的宽度、高度和位置按以下方法获得：

EipmEnableFlag 的值为 0 表示不应使用亮度帧内预测扩展模式；值为 1 表示应使用亮度帧内预测扩展模式。

IntraPfChromaEnableFlag 的值为 0 表示不应使用帧内色度预测滤波；值为 1 表示应使用帧内色度预测滤波。

UmveEnhancementEnableFlag 的值为 0 表示不应使用高级运动矢量表达增强模式；值为 1 表示可使用高级运动矢量表达增强模式。

AffineUmveEnableFlag 的值为 0 表示不应使用仿射高级运动矢量表达模式；值为 1 表示可使用仿射高级运动矢量表达模式。

EnhancedStEnableFlag 的值为 0 表示不应使用增强二次变换；值为 1 表示应使用增强二次变换。

EnhancedTscpmEnableFlag 的值为 0 表示不应使用增强的跨分量两步预测模式；值为 1 表示可使用增强的跨分量两步预测模式。

T/AI 109.8-2025

6.2.4 参考图像队列配置集语义

参考知识图像标志 reference_to_library_enable_flag[layerId]

二值变量。值为‘1’表示第 layerId 层的当前参考图像队列配置集中可有参考图像是知识图像；值为‘0’表示当前参考图像队列配置集的所有参考图像均不应是知识图像。ReferenceToLibraryEnableFlag[layerId]的值等于 reference_to_library_enable_flag[layerId]的值。如果位流中不存在 reference_to_library_enable_flag[layerId]，则 ReferenceToLibraryEnableFlag[layerId]的值等于 0。

知识图像索引标志 library_index_flag[layerId][list][rpls][i]

二值变量。值为‘1’表示第 layerId 层的索引为 list 的参考图像队列配置集中的第 rpls 个参考图像队列中第 i 幅参考图像是知识图像；值为‘0’表示第 layerId 层的索引为 list 的参考图像队列配置集中的第 rpls 个参考图像队列中第 i 幅参考图像不是知识图像。LibraryIndexFlag[layerId][list][rpls][i] 的值等于 library_index_flag[layerId][list][rpls][i] 的值。如果位流中不存在 library_index_flag[layerId][list][rpls][i]，则 LibraryIndexFlag[layerId][list][rpls][i]的值为 0。

被参考的知识图像索引 referenced_library_picture_index[layerId][list][rpls][i]

表示参考第 layerId 层的图像队列 list 的第 rpls 个参考图像配置集中第 i 幅参考图像的知识图像索引。取值范围是 0～511。ReferencedLibraryPictureIndex[layerId][list][rpls][i]的值等于 referenced_library_picture_index[layerId][list][rpls][i]的值。

参考图像数 num_of_ref_pic[layerId][list][rpls]

表示第 layerId 层的参考图像配置集中参考图像的数量。解析过程见 7.2。NumOfRefPic[layerId][list][rpls] 的值等于 num_of_ref_pic[layerId][list][rpls] 的值。num_of_ref_pic[layerId][list][rpls]的值应小于MaxDpbSize的值。

参考图像DOI差值绝对值 abs_delta_doi[layerId][list][rpls][i]

如果第i幅参考图像是参考图像队列配置集中出现的第一幅显示图像的参考图像，表示当前图像的DOI与该参考图像的DOI的差值的绝对值,且如果layerId为0,abs_delta_doi[layerId][list][rpls][i]的值不应为0；否则，表示在参考图像队列list中，离该参考图像最近的前一幅显示图像的参考图像的DOI与该参考图像的DOI的差值的绝对值。解析过程见7.2。取值范围是0～2^8-1。

参考图像DOI差值符号 sign_delta_doi[layerId][list][rpls][i]

二值变量。值为‘0’表示 DeltaDoi[layerId][list][rpls][i] 的值等于 abs_delta_doi[layerId][list][rpls][i]；值为‘1’表示DeltaDoi[layerId][list][rpls][i]的值等于-abs_delta_doi[layerId][list][rpls][i]。

如果Rpl1SameAsRpl0Flag[layerId]的值为1，进行以下操作：

——将 NumOfRefPic[layerId][1][rpls]的值设置为 NumOfRefPic[layerId][0][rpls]的值，rpls 的取值范围是 0～(NumRefPicListSet[layerId][0]-1)；

——将 DeltaDoi[layerId][1][rpls][i]的值设置为 DeltaDoi[layerId][0][rpls][i]的值，rpls 的取值范围是 0～(NumRefPicListSet[layerId][0]-1)，i 的取值范围是 0～NumOfRefPic[layerId][1][rpls]。

6.2.5 自定义加权量化矩阵语义

加权量化矩阵系数 weight_quant_coeff

T/AI 109.8-2025

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。

6.2.6 安全参数集 RBSP 语义

知识图像标识 sec_is_library_flag

二值变量。值为‘1’表示此安全参数集作用于知识图像，值为‘0’表示此安全参数集作用于显示图像。

保留参数集ID reserved_para_set_id

2 位无符号整数。规定取值为 1。

加密使能标志 encryption_enable_flag

二值变量。值为‘1’表示支持对显示图像编码片，或显示图像序列参数集，或显示图像图像参数集，或非显示知识图像编码片，或显示知识图像编码片，或知识图像序列参数集，或知识图像图像参数集，或扩展数据单元进行加密，即 NAL 单元中的 RBSP 可能经过加密。值为‘0’表示不支持对 NAL 单元中的 RBSP 进行加密。

认证使能标志 authentication_enable_flag

二值变量。值为‘1’表示支持对当前 RAS 或知识图像进行认证，可参与认证的 NAL 单元包括当前 RAS 中的显示图像或知识图像的编码片，以及在该访问单元传输的序列参数集、图像参数集、安全参数集、扩展数据单元以及补充增强信息等。认证数据通过 nal_unit_type 等于 10 的 NAL 单元传输。值为‘0’表示当前安全参数集不支持对 RAS 或知识图像进行认证，不应存在使用该安全参数集生成的该 RAS 或知识图像的 nal_unit_type 等于 10 的 NAL 单元。

知识图像仅支持独立签名认证，显示图像支持共同签名认证。参与共同签名认证的多个显示图像访问单元应位于同一个 RAS。参与共同签名的多个访问单元中的图像类型可以是显示图像。知识图像独立签名认证，采用独立于显示图像的安全参数集做独立签名认证，在安全参数集中使用 sec_is_library_flag 区分。

如果一个访问单元中存在 authentication_idc 等于 1 且 nal_unit_type 等于 1～9，12,14,17,18 和 19 的 NAL 单元，对该访问单元中每个层单元的 authentication_idc 等于 1 且具有相同 layer_id 取值的 NAL 单元按位流顺序排列后做摘要计算，产生该访问单元的 NumOfLayers 个层单元的摘要数据。摘要计算方法由 hash_type 规定。

按照哈希周期hash_period_in_doi_minus1以及随机访问点图像的DOI确定一起共同签名认证的一组访问单元（见哈希周期hash_period_in_doi_minus1定义）。hash_period_in_doi_minus1等于0时，依位流次序计算每个访问单元中每个层单元的摘要排列为初次摘要列表形成该访问单元的初次摘要，对初次摘要进行数字签名；hash_period_in_doi_minus1大于0时，针对hash_period_in_doi_minus1+1个访问单元，依位流次序计算每个访问单元的初次摘要;然后依次按authentication_hash_mode所示的方法计算二次摘要。

对二次摘要值进行数字签名，生成认证数据RBSP，认证数据作用范围不应跨RAS。

注：如果位流中安全参数集的 authentication_enable_flag 和 encryption_enable_flag 取值等于 1 时，即当前 RAS 或知识图像既支持加密又支持认证，则应先对其加密再对其认证，即用于做认证的数据应为加密后的 NAL 单元。

加密基本单元 encryption_unit_mode

2 位无符号整数。指示加密基本单元，值为‘0’表示以 NAL 为单位进行加密；值为‘1’表示以访问单元为单位加密，将访问单元中所有 NAL 单元 RBSP 需加密的部分按位流顺序拼接在一起加密，加密后恢复到加密的 NAL 单元中；值为‘2’表示以层单元为单位加密，将层单元中所有 NAL 单元 RBSP 需加密的部分按位流顺序拼接在一起加密，加密后恢复到加密的 NAL 单元中；值为‘3’保留。每次加

T/AI 109.8-2025

密需要重新初始化 IV。

加密级别模式 encryption_level_mode

2位无符号整数。指示加密级别模式，值为‘0’表示对所有类型NAL单元加密时，加密对象为全部RBSP数据（除了RBSP最后一个字节）；值为‘1’表示对nal_unit_type等于1,2,4,12,14,17,18和19的NAL单元加密时，加密对象为RBSP前encryptionByte个字节，对其他类型NAL单元加密时，加密对象为全部RBSP数据（除了RBSP最后一个字节）；值为‘2’表示对nal_unit_type等于1,2,4,5,12,14,17,18和19的NAL单元加密时，加密对象为RBSP前encryptionByte个字节，对其他类型NAL单元加密时，加密对象为全部RBSP数据（除了RBSP最后一个字节）；值为‘3’保留。其中，encryptionByte = Min(EncryptionNum * EncryptionBaseByte, NumBytesInPayload-1)。

加密基本字节长度数量 encryption_num_minus1

8位无符号整数。指示加密基本字节长度的数量，加密基本字节长度的数量EncryptionNum的值等于encryption_num_minus1的值加1。

加密基本字节长度 encryption_base_byte

2 位无符号整数。指示加密基本字节长度，值为‘0’表示加密基本字节长度 EncryptionBaseByte 的取值为 16，值为‘1’表示加密基本字节长度 EncryptionBaseByte 的取值为 64，值为‘2’表示加密基本字节长度 EncryptionBaseByte 的取值为 256，值为‘3’表示加密基本字节长度 EncryptionBaseByte 的取值为 1024。

加密类型 encryption_type

4 位无符号整数。指示加密所采用的算法，具体对应关系见表 64。

表64 加密类型与具体加密算法的对应关系

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

视频加密密钥标志 vek_flag

二值变量。值为‘1’表示携带 vek，值为‘0’表示不携带 vek。本规范中取值应为 1。

初始向量标志 iv_flag

二值变量。值为‘1’表示携带 iv，值为‘0’表示不携带 iv。本规范中取值应为 1。

视频加密密钥加密类型 vek_encryption_type

4 位无符号整数。指示视频加密密钥的加密类型，具体对应关系见表 64。

加密后视频加密密钥长度 evek_length_minus1

8 位无符号整数。指示加密后的视频加密密钥长度，以字节为单位。

加密后视频加密密钥 evek

n 位无符号整数。表示加密后的视频加密密钥，用于加密计算，长度为 evek_length_minus1 加 1 字节。

视频加密密钥版本号长度 vkek_version length_minus1

8 位无符号整数。指示视频加密密钥版本号长度，以字节为单位。

视频加密密钥版本号 vkek_version

n 位无符号整数。指示视频加密密钥版本号，长度为 vkek_version_length_minus1 加 1 字节。

初始向量长度 iv_length_minus1

8 位无符号整数。指示初始向量长度，以字节为单位。

T/AI 109.8-2025

初始向量 iv

n 位无符号整数。指示初始向量，用于分组加密，长度为 iv_length_minus1 加 1 字节。

哈希类型 hash_type

2 位无符号整数。指示认证所采用的算法，具体对应关系如表 65 所示。

表65 哈希类型与具体算法的对应关系

hash_type值	认证算法	摘要数据长度(字节)
0	SM3	32
1～3	保留	保留

认证摘要计算模式 authentication_hash_mode

二值变量。标识二次摘要计算的方式，值为‘0’表示使用连接方式计算二次摘要，即对各访问单元初次摘要值 Hpic1，Hpic2，…，Hpicn 按照位流顺序参考图 6 使用连接方式做二次摘要；值为‘1’表示使用树顶方式计算二次摘要，即对访问单元初次摘要值 Hpic1，Hpic2，…，Hpicn 按照位流顺序参考图 7 使用树顶方式做二次摘要。

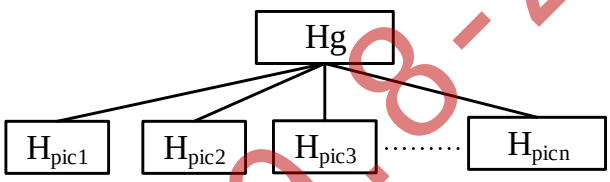


图6 连接摘要示意图

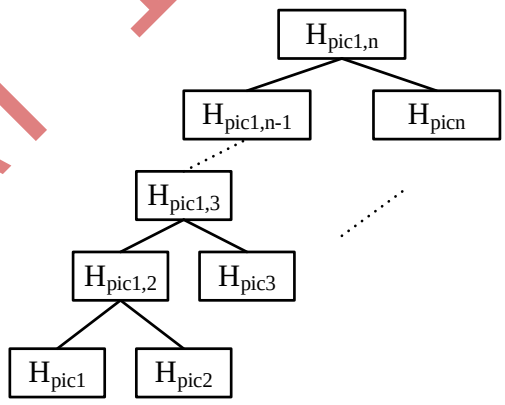


图7 树顶摘要示意图

非随机访问点图像哈希认证标志 hash_discard_nrap_pictures_flag

二值变量。值为‘1’表示对非随机访问点图像不进行认证；等于 0 表示可对非随机访问点图像进行认证。如果 hash_discard_nrap_pictures 不在位流中，默认其值等于 1。

哈希周期 hash_period_in_doi_minus1

8 位无符号整数，取值范围 0～(MAX_HASH_PERIOD/NumOfLayers-1)，MAX_HASH_PERIOD 设置为 256。hash_period_in_doi_minus1 指示与一个认证数据相关的一个位流片段中访问单元的 DOI 区间；

T/AI 109.8-2025

该位流片段所包含的共同参与签名认证的访问单元的个数最多为 hash_period_in_doi_minus1+1，也就是一个哈希周期；该位流片段中的任意两个共同参与签名认证的访问单元以 DOI 指示的时域距离应小于等于 hash_period_in_doi_minus1。一个随机访问片段内的随机访问点图像是第一个参与签名认证的图像，可根据随机访问点图像的 DOI 与 hash_period_in_doi_minus1 确定一组访问单元，这一组访问单元共同签名认证。一个哈希周期内共同参与一次签名认证的任意访问单元的 DOI 的范围为[((hash_period_in_doi_minus1+1)*n+RAP_DOI)%256, ((hash_period_in_doi_minus1+1)*(n+1)-1+RAP_DOI)%256]，其中 RAP_DOI 为随机访问点图像的 DOI，n 为第几个一起签名认证的位流片段，从 0 开始取值。

HashPeriodInDoi 等于 hash_period_in_doi_minus1+1。

HashPeriodInDoi 为 1 表示对单个访问单元独立签名，携带该签名的认证数据 NAL 单元应位于该签名所关联的访问单元或其后第一个访问单元。HashPeriodInDoi 大于 1 表示对多个访问单元共同签名。

数字签名类型 signature_type

2 位无符号整数。指示对图像的摘要数据进行数字签名的算法，如表 66 所示。

表66 数字签名类型与具体加密算法的对应关系

signature_type值	签名算法
0	SM2
1～3	保留

签名数据格式 signature_fmt

2 位无符号整数。指示签名数据格式，signature_fmt 取值含义与 signature_type 句法对应关系的具体规定如表 67 所示。

表67 签名数据格式

signature_type值	signature_fmt值	签名数据格式
0	0	R|S
	1	ASN.1 DER
	2～3	保留
1～3	0～3	保留

注： R|S 表示由 SM2 加密算法输出的秘文 R 和 S 直接拼接的结果，具体参考标准《GM/T 0003.1-2012 SM2 椭圆曲线公钥密码算法 第 1 部分：总则》4.2.2 整数到字节串的转换，和《GM/T 0003.2-2012 SM2 椭圆曲线公钥密码算法 第 2 部分：数字签名算法》6.1 数字签名的生成算法。

ASN.1 DER 编码是关于每个元素的标记、长度和值的编码系统，具体参考标准《GM/T 0009-2012 SM2 密码算法使用规范》7.3 签名数据格式。

摄像机证书标识 camera_idc

152 位字符串。指示图像来源的摄像机的证书标识。

摄像机标识 camera_id

160 位字符串。指示图像来源的摄像机 ID。

6.2.7 补充增强信息 RBSP 语义

T/AI 109.8-2025

FF 码 ff_byte
位串‘0xFF’。
尾负载类型码 last_payload_type_byte
8 位整数。表示负载类型的最后一个字节。
尾负载大小码 last_payload_size_byte
8 位整数。表示负载大小的最后一个字节。
一位 bit_equal_to_one
二值变量。应等于‘1’。
零位 bit_equal_to_zero
二值变量。应等于‘0’。

6.2.7.1 用户数据负载语义

用户数据字节 user_data
8位整数。用户数据的含义由用户根据需要自行定义。
注：用户数据的一种应用方式示例“利用用户数据负载验证解密解码正确性”，参见附录 J。

6.2.7.2 感兴趣区域参数负载语义

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。roi_skip_run 的值应小于 256。
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

T/AI 109.8-2025

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 PicHorizontalSize，ROIInfo[roiIndex]->height 的值不应大于 PicVerticalSize，ROIInfo[roiIndex]->asisx+ ROIInfo[roiIndex]->width 的值不应大于 PicHorizontalSize，ROIInfo[roiIndex]->asisy+ ROIInfo[roiIndex]->height 的值不应大于 PicVerticalSize。

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 PicHorizontalSize，ROIInfo[roiIndex]->height 的值不应大于 PicVerticalSize，ROIInfo[roiIndex]->asisx+ ROIInfo[roiIndex]->width 的值不应大于 PicHorizontalSize，ROIInfo[roiIndex]->asisy+ ROIInfo[roiIndex]->height 的值不应大于 PicVerticalSize。

6.2.7.3 高层序列隐私保护区域负载语义

隐私等级 privacy_level_num_minus2

3 位无符号整数。表示序列隐私保护等级个数。PrivacyLevelNum 的值等于 privacy_level_num_minus2 加 2。若位流中不存在 privacy_level_num_minus2，PrivacyLevelNum 的值等于 2。

隐私类别 privacy_category_num_minus1

3 位无符号整数。表示序列隐私保护类别个数，指明将当前序列隐私保护类别数量。PrivacyCategoryNum 的值等于 privacy_category_num_minus1 的值加 1。PrivacyCategoryNum 的值不应等于 1。若位流中不存在 privacy_category_num_minus1，PrivacyCategoryNum 的值等于 2。

隐私编码模式索引 privacy_coding_mode_index

2 位无符号整数。表示隐私保护编码模式，见表 68。PrivacyCodingModeIndex 的值等于 privacy_coding_mode_index 的值。若位流中不存在 privacy_coding_mode_index，

T/AI 109.8-2025

PrivacyCodingModeIndex 的值等于 0。

表68 隐私编码模式索引

privacy_coding_mode_index 值	编码模式
0	‘RECT_INFO_MODE’
其他	保留

矩形表示最小单元尺寸 log2_min_rect_info_mode_size

3 位无符号整数。表示使用矩形表示模式下最小单元尺寸，取值范围是 0～7。Log2MinRectInfoModeSize 的值等于 log2_min_rect_info_mode_size 的值。若位流中不存在 log2_min_rect_info_mode_size，Log2MinRectInfoModeSize 的值等于 0。

隐私保护权限类别使能标志 privacy_level_category_enable[i][j]

二值变量。privacy_level_category_enable[i][j]表示当前序列下等级权限 i 是否拥有类别 j 的访问权限，值为‘1’表示等级权限 i 拥有类别 j 的访问权限，值为‘0’表示等级权限 i 没有类别 j 的访问权限。PrivacyLevelCategoryEnable[i][j]的值等于 privacy_level_category_enable[i][j]的值。若位流中不存在 privacy_level_category_enable[i][j]，PrivacyLevelCategoryEnable[0][0]等于 1，PrivacyLevelCategoryEnable[0][1]等于 0，PrivacyLevelCategoryEnable[0][1]等于 1。PrivacyLevelCategoryEnable[1][1]等于 1。

6.2.7.4 高层图像隐私保护区域负载语义

隐私保护类别区域数量 privacy_region_num[i]

8 位无符号整数。表示类别 i 下隐私区域个数。

隐私保护矩形的左上角横坐标 privacy_left

16 位无符号整数。表示一个隐私保护矩形左上角在图像中的横坐标。PrivacyLeft 的值等于 privacy_left << Log2MinRectInfoModeSize 的值。

隐私保护矩形的左上角纵坐标 privacy_top

16 位无符号整数。表示一个隐私保护矩形左上角在图像中的纵坐标。PrivacyTop 的值等于 privacy_top << Log2MinRectInfoModeSize 的值。

隐私保护矩形的宽度 privacy_width

16 位无符号整数。表示一个隐私保护矩形在图像中的宽度。PrivacyWidth 的值等于 privacy_width << Log2MinRectInfoModeSize 的值。

隐私保护矩形的长度 privacy_height

16 位无符号整数。表示一个隐私保护矩形在图像中的高度。PrivacyHeight 的值等于 privacy_height << Log2MinRectInfoModeSize 的值。

根据隐私保护矩形区域可以确定其覆盖像素的类别，如果同一个像素具有多个类别，则以类别索引更大的为准。

6.2.7.5 时域分层信息负载语义

时间层数量 num_of_temporal_level_minus1

3 位无符号整数。表示编码视频序列包含的时间层数量。时间层数量等于编码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

T/AI 109.8-2025

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，见表 63。i 表示编码视频序列中所有图像的最高时间层标识。

时间层位率低位 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 表示编码视频序列中所有图像的最高时间层标识。

时间层位率高位 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 表示编码视频序列中所有图像的最高时间层标识。

6.2.7.6 自适应帧内刷新参数负载语义

自适应帧内刷新边界横坐标 air_bound_x

16 位无符号整数。表示已刷新区域和未刷新区域的边界在图像中的横坐标，以 LCU 的宽度为单位。AirBoundX 的值等于 air_bound_x 的值。如果位流中不存在 air_bound_x，AirBoundX 的值为 0。AirBoundX 的值应小于或等于 PictureWidthInLcu。

自适应帧内刷新边界纵坐标 air_bound_y

16 位无符号整数。表示已刷新区域和未刷新区域的边界在图像中的纵坐标，以 LCU 的高度为单位。AirBoundY 的值等于 air_bound_y 的值。如果位流中不存在 air_bound_y，AirBoundY 的值为 0。AirBoundY 的值应小于或等于 PictureHeightInLcu。

在自适应帧内刷新过程中，当前图像可划分为已刷新区域和未刷新区域，见图 8，已刷新区域仅利用当前图像或其他图像已刷新区域进行解码，未刷新区域可利用当前图像或其他图像进行解码。

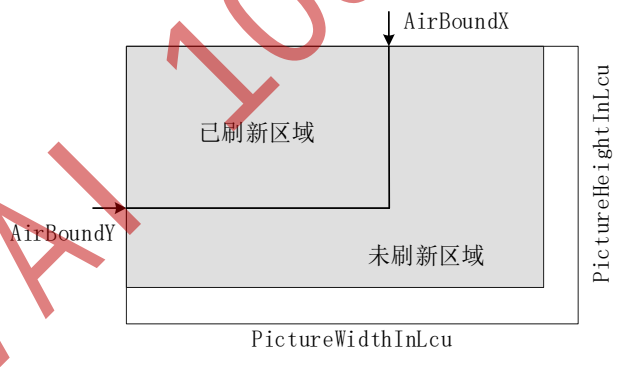


图8 AIR 技术刷新区域划分

6.2.7.7 编码防抖参数负载语义

全局运动矢量信息 x 方向平移分量 gmv_translational_x

8 位有符号整数。表示图像的全局运动向量在 x 方向上的平移分量。

全局运动矢量信息 y 方向平移分量 gmv_translational_y

8 位有符号整数。表示图像的全局运动向量在 y 方向上的平移分量。

全局运动矢量信息 z 方向平移分量 gmv_translational_z

8 位有符号整数。表示图像的全局运动向量在 z 方向上的平移分量。

全局运动矢量信息 x 轴旋转分量 gmv_rotational_x

T/AI 109.8-2025

8 位有符号整数。表示图像的全局运动向量绕 x 轴的旋转分量。

全局运动矢量信息 y 轴旋转分量 gmv_rotational_y

8 位有符号整数。表示图像的全局运动向量绕 y 轴的旋转分量。

全局运动矢量信息 z 轴旋转分量 gmv_rotational_z

8 位有符号整数。表示图像的全局运动向量绕 z 轴的旋转分量。

6.2.7.8 分层信息负载语义

分层层数 sei_num_of_layers_minus1

2 位无符号整数。加 1 标识分层结构的层数。sei_num_of_layers_minus1 的值应小于 NumOfLayers。

分层标识 sei_layer_id[i]

2 位无符号整数。表示第 i 层的 LayerId。sei_layer_id[i]应小于 NumOfLayers。对于任意无符号整数 k 和 m，当 k 小于 m，sei_layer_id[k]应小于 sei_layer_id[m]。

6.2.7.9 显示知识图像显示信息负载语义

显示知识图像显示标志 library_display_flag

二值变量。值为‘1’表示当前访问单元的显示知识图像需要显示。值为‘0’表示当前访问单元的显示知识图像不需要显示。LibraryDisplayFlag 的值等于 library_display_flag 的值。如果位流中不存在 library_display_flag，LibraryDisplayFlag 的值等于 1。

6.2.7.10 缓存周期负载语义

缓存周期负载语义应符合 GB/T 25724-2017 中 E.2.1 的规定。

6.2.7.11 图像定时负载语义

图像定时负载语义应符合 GB/T 25724-2017 中 E.2.2 的规定。

6.2.7.12 全景视频参数负载语义

全景视频参数负载语义应符合 GB/T 25724-2017 中 E.2.3 的规定。

6.2.7.13 序列显示扩展负载语义

序列显示扩展负载语义应符合 T/AI 109.2-2021 中 7.2.2.6 的规定。

6.2.7.14 目标显示设备和内容元数据扩展负载语义

目标显示设备和内容元数据扩展负载语义应符合 T/AI 109.2-2021 中 7.2.2.10 的规定。

6.2.7.15 高动态范围图像扩展负载语义

高动态范围图像扩展负载语义应符合 T/AI 109.2-2021 中 7.2.2.9 的规定。

6.2.7.16 图像显示扩展负载语义

图像显示扩展负载语义应符合 T/AI 109.2-2021 中 7.2.3.3 的规定。

6.2.8 图像参数集 RBSP 语义

T/AI 109.8-2025

固定图像量化因子标志 fixed_picture_qp_flag

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。FixedPictureQpFlag 的值等于 fixed_picture_qp_flag 的值。

编码单元量化参数可变标志 cu_delta_qp_flag

二值变量。值为‘1’表示图像中一棵编码树内的编码单元的量化参数可能不同；值为‘0’表示图像中任意一棵编码树内的所有编码单元的量化参数相同。CuDeltaQpFlag 的值等于 cu_delta_qp_flag。如果位流中不存在 cu_delta_qp_flag，CuDeltaQpFlag 的值为 0。

图像级编码单元量化组尺寸 cu_qp_group_size_log2_minus3

2 位无符号整数。给出编码单元量化组的尺寸，取值范围是 0～3。

cu_qp_group_size_log2_minus3 的值应小于或等于(log2_lcu_size_minus2 - 1)。

CuQpGroupAreaSize 的值等于 1 << ((cu_qp_group_size_log2_minus3 + 3) × 2)。

去块滤波禁用标志 deblocking_filter_disable_flag

二值变量。值为‘1’表示不应使用去块效应滤波；值为‘0’表示应使用去块效应滤波。DeblockingFilterDisableFlag 的值等于 deblocking_filter_disable_flag 的值。

去块滤波类型 deblocking_filter_type

1 位无符号整数。值为‘1’表示当前图像应使用改进型去块滤波；值为‘0’表示当前图像不应使用改进型去块滤波。DeblockingFilterType 的值等于 deblocking_filter_type。如果位流中不存在 deblocking_filter_type，DeblockingFilterType 的值为 0。

去块滤波参数标志 deblocking_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’表示位流中不存在 alpha_c_offset 和 beta_offset。

图像加权量化允许标志 picture_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。PictureWeightQuantEnableFlag 的值等于 picture_weight_quant_enable_flag。如果当前图像的图像头位流中不存在 picture_weight_quant_enable_flag，则 PictureWeightQuantEnableFlag 的值应为 0。

图像加权量化数据加载索引 picture_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 7.2。解码过程见 8.3.7。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。

图像级块复制帧内预测允许标志 picture_ibc_enable_flag

T/AI 109.8-2025

二值变量。值为‘1’表示当前图像可使用块复制帧内预测；值为‘0’表示当前图像不应使用块复制帧内预测。PictureIbcEnableFlag 的值等于 picture_ibc_enable_flag 的值。如果位流中不存在 picture_ibc_enable_flag，PictureIbcEnableFlag 的值为 0。

图像级串复制帧内预测普通串子模式允许标志 picture_isc_os_enable_flag

二值变量。值为‘1’表示当前图像可使用串复制帧内预测中的普通串子模式；值为‘0’表示当前图像不应使用串复制帧内预测中的普通串子模式。PictureIscOsEnableFlag 的值等于 picture_isc_os_enable_flag 的值。如果位流中不存在 picture_isc_os_enable_flag，PictureIscOsEnableFlag 的值为 0。

图像级串复制帧内预测非普通串子模式允许标志 picture_isc_evs_ubvs_enable_flag

二值变量。值为‘1’表示当前图像可使用串复制帧内预测中的非普通串子模式；值为‘0’表示当前图像不应使用串复制帧内预测中的非普通串子模式。PictureIscEvsUbvsEnableFlag 的值等于 picture_isc_evs_ubvs_enable_flag 的值。如果位流中不存在 picture_isc_evs_ubvs_enable_flag，PictureIscEvsUbvsEnableFlag 的值为 0。PictureIscEnableFlag 的值等于(PictureIscOsEnableFlag || PictureIscEvsUbvsEnableFlag)。

图像级基于频数信息的帧内编码允许标志 picture_fimc_enable_flag

二值变量。值为‘1’表示当前图像可使用基于频数信息的帧内编码；值为‘0’表示当前图像不应使用基于频数信息的帧内编码。PictureFimcEnableFlag 的值等于 picture_fimc_enable_flag 的值。如果位流中不存在 picture_fimc_enable_flag，PictureFimcEnableFlag 的值为 0。

仿射预测子块尺寸标志 affine_subblock_size_flag

二值变量。值为‘0’表示当前图像仿射预测子块最小尺寸为 4×4；值为‘1’表示最小尺寸为 8×8。AffineSubblockSizeFlag 的值等于 affine_subblock_size_flag 的值。如果位流中不存在 affine_subblock_size_flag，AffineSubblockSizeFlag 的值为 0。

图像级角度加权预测权重索引 picture_awp_refine_index

1 位无符号整数。用于确定角度加权预测模式的参考权重。PictureAwpRefineIndex 的值等于 picture_awp_refine_index 的值。如果位流中不存在 picture_awp_refine_index，PictureAwpRefineIndex 的值为 0。

6.2.9 图像头 RBSP 语义

本条中 compIndex 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。

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。在一个访问单元中，所有图像的 DOI 应相同。

图像编码方式 picture_coding_type

2 位无符号整数。规定图像的类型及 PictureCodingType 的值，见表 69。其中 NAL 单元类型为 0、2 和 12 对应的图像头的 picture_coding_type 值应为‘01’，NAL 单元类型为 1 和 14 对应的图像头的 picture_coding_type 值应为‘10’或者‘11’。

图像变分辨率编码标志 picture_rpr_flag

二值变量。

值为‘1’表示当前图像的水平尺寸和垂直尺寸等于序列参数集水平尺寸 SpsHorizontalSize[LayerId]和序列参数集垂直尺寸 SpsVerticalSize[LayerId]的 1/2，当前图像的水平编码尺寸和垂直编码尺寸等于序列参数集变分辨率水平编码尺寸 SpsRprPicWidthInLuma[LayerId]和序列参数集变分辨率垂直编码尺寸 SpsRprPicHeightInLuma[LayerId]。

T/AI 109.8-2025

值为‘0’表示当前图像的水平尺寸和垂直尺寸等于序列参数集水平尺寸SpsHorizontalSize[LayerId]和序列参数集垂直尺寸 SpsVerticalSize[LayerId]，当前图像的水平编码尺寸和垂直编码尺寸等于序列参数集水平编码尺寸 SpsPicWidthInLuma[LayerId]和序列参数集垂直编码尺寸 SpsPicHeightInLuma[LayerId]。

PictureRprFlag 的值等于 picture_rpr_flag。如果位流中不存在 picture_rpr_flag，PictureRprFlag 的值为 0。

当前图像的水平尺寸 PicHorizontalSize、垂直尺寸 PicVerticalSize、水平编码尺寸 PicWidthInLuma、垂直编码尺寸 PicHeightInLuma，及图像中各个片以 LCU 为单位的宽度、高度和位置按以下方法获得：

```
PicWidthInLuma = PictureRprFlag ? SpsRprPicWidthInLuma[LayerId] : SpsPicWidthInLuma[LayerId]
PicHeightInLuma = PictureRprFlag ? SpsRprPicHeightInLuma[LayerId] : SpsPicHeightInLuma[LayerId]
PicHorizontalSize = PictureRprFlag ? (SpsHorizontalSize[LayerId] >> 1) : SpsHorizontalSize[LayerId]
PicVerticalSize = PictureRprFlag ? (SpsVerticalSize[LayerId] >> 1) : SpsVerticalSize[LayerId]
tempW = PatchWidth[LayerId]
tempH = PatchHeight[LayerId]
LcuSize = 1 << LcuSizeInBit
PictureWidthInLcu = (PicWidthInLuma + LcuSize -1) / LcuSize
PictureHeightInLcu = (PicHeightInLuma + LcuSize -1) / LcuSize
if (tempW> PictureWidthInLcu)
    tempW = PictureWidthInLcu
if (tempH> PictureHeightInLcu)
    tempH = PictureHeightInLcu
if (tempW < Min(2, PictureWidthInLcu))
    tempW= Min(2, PictureWidthInLcu)
NumPatchColumns[LayerId] = PictureWidthInLcu / tempW
NumPatchRows[LayerId] = PictureHeightInLcu / tempH
for (i = 0; i < NumPatchRows[LayerId]; i++) {
    for (j = 0; j < NumPatchColumns[LayerId]; j++) {
        PatchSizeInLcu[i][j]->Width = PatchWidth[LayerId]
        PatchSizeInLcu[i][j]->Height = PatchHeight[LayerId]
        PatchSizeInLcu[i][j]->PosX = PatchWidth[LayerId] * j
        PatchSizeInLcu[i][j]->PosY= PatchHeight[LayerId] * i
    }
}
temp = PictureHeightInLcu % tempH
if (temp != 0) {
    for (j = 0; j < NumPatchColumns[LayerId]; j++) {
        PatchSizeInLcu[NumPatchRows[LayerId]][j]->Width = tempW
        PatchSizeInLcu[NumPatchRows[LayerId]][j]->Height = temp
        PatchSizeInLcu[NumPatchRows[LayerId]][j]->PosX = tempW * j
        PatchSizeInLcu[NumPatchRows[LayerId]][j]->PosY = tempH * NumPatchRows[LayerId]
    }
```

T/AI 109.8-2025

```
        NumPatchRows[LayerId]++
}
temp = PictureWidthInLcu % tempW
if (temp != 0) {
        for (j = 0; j < NumPatchRows[LayerId]; j++)
                PatchSizeInLcu[j][NumPatchColumns[LayerId]-1]->Width += temp
}
```

MinPatchWidthInPixel 的值等于 Min($2^{LcuSizeInBit+1}$,HorizontalSize[LayerId])。任何片以像素为单位的宽度不应小于 MinPatchWidthInPixel 的值。

PicHorizontalSize、PicVerticalSize 与图像边界的关系见图 9。图 9 中，实线表示图像可显示区域边界，其宽度和高度分别由 PicHorizontalSize 和 PicVerticalSize 决定；虚线表示图像边界，其宽度和高度分别由 PicWidthInLuma 和 PicHeightInLuma 决定。

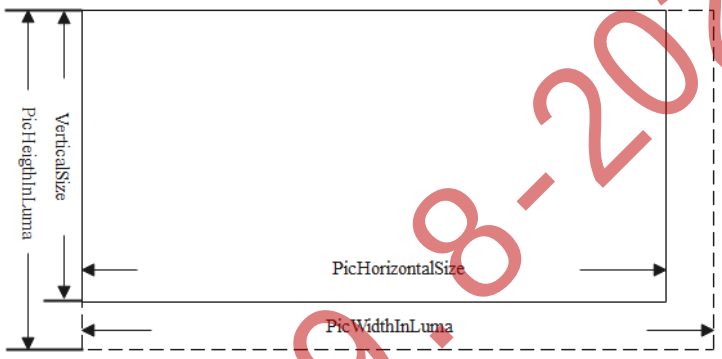


图9 图像边界示意图

图像时域运动信息预测标志 picture_tmvp_enable_flag

二值变量。值为‘1’表示当前图像可使用时域运动信息预测；值为‘0’表示当前图像不应使用时域运动信息预测。PictureTmvpEnableFlag 的值等于 picture_tmvp_enable_flag 的值。如果位流中不存在 picture_tmvp_enable_flag，PictureTmvpEnableFlag 的值为 TmvpEnableFlag 的值。

若当前图像为 P 图像，且参考图像队列 0 中参考索引值为 0 的参考图像的水平尺寸或垂直尺寸与当前图像的水平尺寸或垂直尺寸不同，或参考图像队列 0 中参考索引值为 0 的参考图像的 LayerId 与当前图像的 LayerId 不同，PictureTmvpEnableFlag 的值应为 0。若当前图像为 B 图像，且参考图像队列 1 中参考索引值为 0 的参考图像的水平尺寸或垂直尺寸与当前图像的水平尺寸或垂直尺寸不同，或参考图像队列 1 中参考索引值为 0 的参考图像的 LayerId 与当前图像的 LayerId 不同 PictureTmvpEnableFlag 的值应为 0。

表69 图像的类型

picture_coding_type值	PictureCodingType值	图像类型
00	0	禁止
01	1	I图像
10	2	P图像
11	3	B图像

T/AI 109.8-2025

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参数集之后且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参数集之后且random_access_decodable_flag的值为‘1’的图像。其中，对应的序列参数集指的是位流中在当前图像之前最近的一个序列参数集。RandomAccessDecodableFlag的值等于random_access_decodable_flag的值。如果当前图像的图像参数集中不存在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，则在其对应的序列参数集发生随机访问时，当前图像可能无法正确解码。

序列参数集后的第一幅编码图像应是I图像或RL图像。解码顺序在该I图像或RL图像之后，显示顺序在该I图像或RL图像之前的图像称为该序列参数集对应的前置图像，只有前置图像的RandomAccessDecodableFlag的值可以为0，其余图像的RandomAccessDecodableFlag的值应为1。对于一个序列集参数，如果存在RandomAccessDecodableFlag 的值为0的前置图像，则该序列集参数后的第一个图像不应为IDR图像。并且对于同一个序列参数集对应的前置图像，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。

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27 MHz系统时钟导出的90 kHz时钟周期数来表示。如果编码视频序列中某一幅图像的BbvDelay等于BbvDelayMax，那么整个编码视频序列中所有图像的BbvDelay均应等于BbvDelayMax。见附录H。

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70。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70 时间编码（time_code）

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

知识图像索引 library_picture_index

T/AI 109.8-2025

说明当前图像的知识图像索引，解析过程见 7.2。取值范围是 0～511。LibraryPictureIndex 的值等于 library_picture_index 的值。同一显示图像集所参考的所有知识图像中任意两幅不同的知识图像所对应的知识图像索引不应相同。

知识图像帧片个数 library_patch_num_minus1

说明当前知识图像所包含的知识图像编码片的个数,取值范围 0～255。知识图像编码片的个数等于 library_patch_num_minus1 的值加 1。知识图像编码片的个数应等于 NumPatchColumns[LayerId]乘以 NumPatchRows[LayerId]的值。

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，以解码图像为单位，解析过程见 7.2。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。

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7.2。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 H）。

引用参考图像队列配置集标志 ref_pic_list_set_flag[0]、ref_pic_list_set_flag[1]

二值变量。值为‘1’表示通过序列参数集获得当前图像的参考图像队列 0（或参考图像队列 1）的参考图像队列配置集；值为‘0’表示通过图像参数集获得当前图像的参考图像队列 0（或参考图像队列 1）的参考图像队列配置集。

如果 NumRefPicListSet[LayerId][0]（或 NumRefPicListSet[LayerId][1]）的值为 0，ref_pic_list_set_flag[0]（或 ref_pic_list_set_flag[1]）的值应为‘0’。RefPicListSetFlag[0]的值等于 ref_pic_list_set_flag[0]的值。如果 Rpl1IndexExistFlag[LayerId]的值为 0，RefPicListSetFlag[1]的值等于 ref_pic_list_set_flag[0]的值；否则 RefPicListSetFlag[1]的值等于 ref_pic_list_set_flag[1]的值。

引用参考图像队列配置集索引 ref_pic_list_set_index[0]、ref_pic_list_set_index[1]

表示当前图像的参考图像队列 0（或参考图像队列 1）的参考图像队列配置集索引值。该值由 Ceil(Log(NumRefPicListSet[LayerId][i]))位表示，取值范围是 0～(NumRefPicListSet[LayerId][i]-1)（i 等于 0 或 1）。

如果位流中不存在 ref_pic_list_set_index[0]，ref_pic_list_set_index[0]的值应为‘0’。如果位流中不存在 ref_pic_list_set_index[1]且 Rpl1IndexExistFlag[LayerId]的值为 0，ref_pic_list_set_index[1]的值等于 ref_pic_list_set_index[0]的值。如果位流中不存在 ref_pic_list_set_index[1]且 Rpl1IndexExistFlag[LayerId]的值为 1，ref_pic_list_set_index[1]的值应为‘0’。

如果 ref_pic_list_set_flag[i]的值为‘1’，则 RplsIndex[i]的值等于 ref_pic_list_set_index[i]的值；否则，RplsIndex[i]的值等于 NumRefPicListSet[LayerId][i]的值（i 等于 0 或 1）。

活跃参考图像数重载标志 num_ref_active_override_flag

二值变量。值为‘1’表示位流中存在 num_ref_active_minus1[0]或 num_ref_active_minus1[1]；值为‘0’表示位流中不存在 num_ref_active_minus1[0]和 num_ref_active_minus1[1]。

活跃参考图像数 num_ref_active_minus1[0]、num_ref_active_minus1[1]

T/AI 109.8-2025

表示解码当前图像时，参考图像队列 0（或参考图像队列 1）的参考索引的最大值。解析过程见 7.2。取值范围是 0～14。NumRefActive[i]的值等于 num_ref_active_minus1[i]加 1（i 等于 0 或 1）。如果位流中不存在 num_ref_active_minus1[0]，则 NumRefActive[0]的值等于 0；如果位流中不存在 num_ref_active_minus1[1]，则 NumRefActive[1]的值等于 0。

当前图像解码的参考图像队列 i 的参考索引的最大值等于 num_ref_active_minus1[i]的值（i 等于 0 或 1）。如果当前图像是 I 图像，NumRefActive[0]和 NumRefActive[1]的值均为 0。如果当前图像是 P 图像且 NumRefActiveOverrideFlag 的值为 0，NumRefActive[0]的值等于 NumRefDefaultActive[LayerId][0]+1，NumRefActive[1]的值为 0。如果当前图像是 B 图像且 NumRefActiveOverrideFlag 的值为 0，NumRefActive[i]的值等于 NumRefDefaultActive[LayerId][i]+1（i 等于 0 或 1）。

如果 num_ref_active_override_flag 的值为‘0’，则 NumRefActive[i]的值等于 NumRefDefaultActive[LayerId][i]的值加 1（i 等于 0 或 1）。NumRefActive[i]的值应小于或等于 NumOfRefPic[LayerId][i][RplsIndex[i]]的值（i 等于 0 或 1）。

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))。

图像级权限保护允许标志 picture_privacy_enable_flag

二值变量。值为‘1’表示当前图像可使用权限保护；值为‘0’表示当前图像不应使用权限保护。PicturePrivacyEnableFlag 的值等于 picture_privacy_enable_flag 的值。如果位流中不存在 picture_privacy_enable_flag，PicturePrivacyEnableFlag 的值为 0。

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。如果位流中不存在 alpha_c_offset，AlphaCOffset 的值为 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。如果位流中不存在 beta_offset，BetaOffset 的值为 0。

图像级去块滤波垂直调整允许标志 picture_dbr_v_enable_flag

二值变量。值为‘1’表示当前图像可使用去块滤波垂直调整；值为‘0’表示当前图像不应使用去块滤波垂直调整。PictureDbrVEnableFlag 的值等于 picture_dbr_v_enable_flag 的值。如果位流中不存在 picture_dbr_v_enable_flag，PictureDbrVEnableFlag 的值为 0。

图像级增强去块滤波垂直调整允许标志 picture_alt_dbr_v_enable_flag

二值变量。值为‘1’表示当前图像应使用增强去块滤波垂直调整；值为‘0’表示当前图像不应使用增强去块滤波垂直调整。PictureAltDbrVEnableFlag 的值等于 picture_alt_dbr_v_enable_flag 的值。如果位流中不存在 picture_alt_dbr_v_enable_flag，PictureAltDbrVEnableFlag 的值为 0。

去块滤波垂直调整阈值 dbr_v_threshold_minus1

用于确定当前图像去块滤波垂直调整的阈值，取值范围是 0～1。DbrVThreshold 的值等于 dbr_v_threshold_minus1 的值加 1。如果位流中不存在 dbr_v_threshold_minus1，DbrVThreshold 的值为 0。

去块滤波垂直调整偏移值 0 dbr_v_offset0_minus1

用于确定当前图像去块滤波垂直调整的偏移值 0，取值范围是 0～3。DbrVOffset0 的值等于 dbr_v_offset0_minus1 的值加 1 后的负值。如果位流中不存在 dbr_v_offset0_minus1，DbrVOffset0 的值为 0。

T/AI 109.8-2025

去块滤波垂直调整偏移值 1 dbr_v_offset1_minus1

用于确定当前图像去块滤波垂直调整的偏移值 1，取值范围是 0～3。DbrVOffset1 的值等于 dbr_v_offset1_minus1 的值加 1。如果位流中不存在 dbr_v_offset1_minus1，DbrVOffset1 的值为 0。

增强去块滤波垂直调整偏移值 0 dbr_v_alt_offset0_minus1

用于确定当前图像增强去块滤波垂直调整的偏移值 0，取值范围是 0～3。DbrVAltOffset0 的值等于 dbr_v_alt_offset0_minus1 的值加 1 后的负值。如果位流中不存在 dbr_v_alt_offset0_minus1，DbrVAltOffset0 的值为 0。

增强去块滤波垂直调整偏移值 1 dbr_v_alt_offset1_minus1

用于确定当前图像增强去块滤波垂直调整的偏移值 1，取值范围是 0～3。DbrVAltOffset1 的值等于 dbr_v_alt_offset1_minus1 的值加 1。如果位流中不存在 dbr_v_alt_offset1_minus1，DbrVAltOffset1 的值为 0。

图像级去块滤波水平调整允许标志 picture_dbr_h_enable_flag

二值变量。值为‘1’表示当前图像可使用去块滤波水平调整；值为‘0’表示当前图像不应使用去块滤波水平调整。PictureDbrHEnableFlag 的值等于 picture_dbr_h_enable_flag 的值。如果位流中不存在 picture_dbr_h_enable_flag，PictureDbrHEnableFlag 的值为 0。

图像级增强去块滤波水平调整允许标志 picture_alt_dbr_h_enable_flag

二值变量。值为‘1’表示当前图像应使用增强去块滤波水平调整；值为‘0’表示当前图像不应使用增强去块滤波水平调整。PictureAltDbrHEnableFlag 的值等于 picture_alt_dbr_h_enable_flag 的值。如果位流中不存在 picture_alt_dbr_h_enable_flag，PictureAltDbrHEnableFlag 的值为 0。

去块滤波水平调整阈值 dbr_h_threshold_minus1

用于确定当前图像去块滤波水平调整的阈值，取值范围是 0～1。DbrHThreshold 的值等于 dbr_h_threshold_minus1 的值加 1。如果位流中不存在 dbr_h_threshold_minus1，DbrHThreshold 的值为 0。

去块滤波水平调整偏移值 0 dbr_h_offset0_minus1

用于确定当前图像去块滤波水平调整的偏移值 0，取值范围是 0～3。DbrHOffset0 的值等于 dbr_h_offset0_minus1 的值加 1 后的负值。如果位流中不存在 dbr_h_offset0_minus1，DbrHOffset0 的值为 0。

去块滤波水平调整偏移值 1 dbr_h_offset1_minus1

用于确定当前图像去块滤波水平调整的偏移值 1，取值范围是 0～3。DbrHOffset1 的值等于 dbr_h_offset1_minus1 的值加 1。如果位流中不存在 dbr_h_offset1_minus1，DbrHOffset1 的值为 0。

增强去块滤波水平调整偏移值 0 dbr_h_alt_offset0_minus1

用于确定当前图像增强去块滤波水平调整的偏移值 0，取值范围是 0～3。DbrHAltOffset0 的值等于 dbr_h_alt_offset0_minus1 的值加 1 后的负值。如果位流中不存在 dbr_h_alt_offset0_minus1，DbrHAltOffset0 的值为 0。

增强去块滤波水平调整偏移值 1 dbr_h_alt_offset1_minus1

用于确定当前图像增强去块滤波水平调整的偏移值 1，取值范围是 0～3。DbrHAltOffset1 的值等于 dbr_h_alt_offset1_minus1 的值加 1。如果位流中不存在 dbr_h_alt_offset1_minus1，DbrHAltOffset1 的值为 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参数集中不存在 chroma_quant_param_delta_cb 和

T/AI 109.8-2025

chroma_quant_param_delta_cr；值为‘0’表示当前图像的图像参数集中存在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

色度编码块量化参数相对于 CurrentQp 的增量，取值范围-16～16。解析过程见 7.2，解码过程见 8.6.2。如果当前图像的图像参数集中不存在 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’。

图像级自适应修正滤波允许标志 picture_alf_enable_flag[compIndex]

二值变量。值为‘1’表示当前图像的第 compIndex 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 compIndex 个分量不应使用自适应修正滤波。PictureAlfEnableFlag[compIndex]的值等于 picture_alf_enable_flag[compIndex]的值。

图像级跨多分量预测参数索引 picture_pmc_param_index

1 位无符号整数。用于确定当前图像的跨多分量预测参数。PicturePmcParamIndex 的值等于 picture_pmc_param_index。如果位流中不存在 picture_pmc_param_index，PicturePmcParamIndex 的值为 0。

图像级高级运动矢量表达模式运动矢量偏移量集合索引 picture_umve_step_set_index

1 位无符号整数。用于确定高级运动矢量表达模式可使用的运动矢量偏移量。PictureUmveStepSetIndex 的值等于 picture_umve_step_set_index。如果位流中不存在 picture_umve_step_set_index，PictureUmveStepSetIndex 的值为 0。

图像级重叠块运动补偿加权系数集合索引 picture_obmc_blending_set_index

1 位无符号整数。用于确定重叠块运动补偿加权系数。PictureObmcBlendingSetIndex 的值等于 picture_obmc_blending_set_index。如果位流中不存在 picture_obmc_blending_set_index，PictureObmcBlendingSetIndex 的值为 0。

图像级角度加权预测权重索引 picture_awp_refine_index

1 位无符号整数。用于确定角度加权预测模式的参考权重。PictureAwpRefineIndex 的值等于 picture_awp_refine_index 的值。如果位流中不存在 picture_awp_refine_index，PictureAwpRefineIndex 的值为 0。

图像级跳过模式帧间预测修正允许标志 picture_inter_pc_skip_flag

二值变量。值为‘1’表示当前图像使用跳过模式的编码单元可使用帧间预测修正；值为‘0’表示当前图像使用跳过模式的编码单元不应使用帧间预测修正。PictureInterPcSkipFlag 的值等于 picture_inter_pc_skip_flag 的值。如果位流中不存在 picture_inter_pc_skip_flag，PictureInterPcSkipFlag 的值为 0。

图像级变换跳过允许标志 picture_ts_enable_flag

二值变量。值为‘1’表示当前图像可使用变换跳过方法；值为‘0’表示当前图像不应使用变换跳过方法。PictureTsEnableFlag 的值等于 picture_ts_enable_flag 的值。如果位流中不存在 picture_ts_enable_flag，PictureTsEnableFlag 的值为 0。

图像级神经网络滤波允许标志 picture_nn_filter_enable_flag[compIdx]

二值变量。值为‘1’表示当前图像的第 compIdx 个分量可使用神经网络滤波；值为‘0’则表示当前图像的第 compIdx 个分量不应使用神经网络滤波。PictureNnFilterEnableFlag[compIdx]的值等于 picture_nn_filter_enable_flag[compIdx]的值。如果位流中不存在 picture_nn_filter_enable_flag[compIdx]，PictureNnFilterEnableFlag[compIdx]的值为 0。解码

T/AI 109.8-2025

处理应该忽略这些位。

图像级选择性滤波自适应标志 picture_nn_filter_adaptive_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的神经网络滤波；值为‘0’则表示当前图像的第 compIdx 个分量不应使用自适应的神经网络滤波。PictureNnFilterAdaptiveFlag[compIdx]的值等于 picture_nn_filter_adaptive_flag[compIdx]的值。如果位流中不存在 picture_nn_filter_adaptive_flag[compIdx]，PictureNnFilterAdaptiveFlag[compIdx]的值为 0。解码处理应该忽略这些位。

图像级神级网络滤波模型索引 picture_nn_filter_set_index[compIdx]

表示当前图像的第 compIdx 个分量使用神经网络滤波的模型索引。解析过程见 7.2，取值范围 0～NumOfNnFilter-1。PictureNnFilterSetIndex[compIdx]的值等于 picture_nn_filter_set_index[compIdx]的值。如果位流中不存在 picture_nn_filter_set_index[compIdx]，PictureNnFilterSetIndex[compIdx]的值为-1。解码处理应该忽略这些位。

6.2.10 片 RBSP 语义

本条中 compIndex 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。

片索引 patch_index

8 位无符号整数，表示 patch 的索引号，取值范围是 0x00～0x7F。

按以下过程导出片的行索引和列索引：

```
PatchIndexX = patch_index % NumPatchColumns[LayerId]
PatchIndexY = patch_index / NumPatchColumns[LayerId]
LcuRow = PatchSizeInLcu[PatchIndexY][PatchIndexX]->PosY
LcuColumn = PatchSizeInLcu[PatchIndexY][PatchIndexX]->PosX
```

固定片量化因子标志 fixed_patch_qp_flag

二值变量。值为‘1’表示在该片内量化因子不变；值为‘0’表示在该片内量化因子可变。FixedPatchQpFlag 的值等于 fixed_patch_qp_flag 的值。

片量化因子 patch_qp

7 位无符号整数。给出片的量化因子，取值范围是 0～(63 + 8 × (BitDepth - 8))。PatchQp 的值等于 patch_qp 的值。如果 patch_qp 不存在，PatchQp 的值等于 picture_qp 的值。

片样值偏移补偿允许标志 patch_sao_enable_flag[compIndex]

二值变量。PatchSaoEnableFlag[compIndex]的值等于 patch_sao_enable_flag[compIndex]的值。PatachSaoEnableFlag[compIndex]的值为 1 表示在该片内第 compIndex 个分量可使用样值偏移补偿；值为 0 则表示在该片内第 compIndex 个分量不应使用样值偏移补偿。如果位流中不存在 patch_sao_enable_flag[compIndex]，PatchSaoEnableFlag[compIndex]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’。

最大编码单元量化参数增量 lcu_qp_delta

给出当前最大编码单元的量化参数相对预测量化参数的增量。解析过程见 7.3。LcuQpDelta 的值等于 lcu_qp_delta 的值。LcuQpDelta 的取值范围应是(-32 - 4×(BitDepth-8))～(31 + 4×(BitDepth-8))。

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 7.3。SaoMergeTypeIndex 的值等于

T/AI 109.8-2025

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片，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片，则 SaoMergeUpAvai 的值为 1；否则 SaoMergeUpAvai 的值为 0。

样值偏移补偿模式 sao_mode[compIndex]

样值偏移补偿模式的索引值。解析过程见 7.3。由 sao_mode 的值查表 71 得到 SaoMode 和 MaxOffsetNumber 的值。

表71 样值偏移补偿模式和最大偏移数

sao_mode	SaoMode	MaxOffsetNumber	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[compIndex][j]

区间模式偏移值的绝对值，取值范围是 0～7。解析过程见 7.3。SaoIntervalOffsetAbs[compIndex][j]的值等于 sao_interval_offset_abs[compIndex][j]的值。

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

区间模式偏移值的符号位。解析过程见 7.3。如果 sao_interval_offset_sign[compIndex][j]的值为‘0’，则 SaoIntervalOffsetSign[compIndex][j]的值等于 1；否则 SaoIntervalOffsetSign[compIndex][j]的值等于-1。如果位流中不存在 sao_interval_offset_sign[compIndex][j]，SaoIntervalOffsetSign[compIndex][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[compIndex]

区间模式起始偏移子区间的位置，取值范围是 0～31。解析过程见 7.3。SaoIntervalStartPos[compIndex]的值等于 sao_interval_start_pos[compIndex]的值。

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[compIndex]

区间模式起始偏移子区间的位置差，取值范围是 0～14。解析过程见 7.3。SaoIntervalDeltaPosMinus2[compIndex]的值等于 sao_interval_delta_pos_minus2[compIndex]的值。

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

边缘模式的偏移值。解析过程见 7.3。SaoEdgeOffset[compIndex][j]的值等于 sao_edge_offset[compIndex][j]的值。sao_edge_offset[compIndex][0]的取值范围是-1～6，sao_edge_offset[compIndex][1]的取值范围是 0～1，sao_edge_offset[compIndex][2]的取值范围是-1～0，sao_edge_offset[compIndex][3]的取值范围是-6～1。

样值偏移补偿边缘模式类型 sao_edge_type[compIndex]

边缘模式的类型。解析过程见 7.3。SaoEdgeType[compIndex]的值等于 sao_edge_type[compIndex]的值。

最大编码单元增强样值偏移补偿允许标志 esao_lcu_enable_flag[compIndex][LcuIndex]

二值变量。值为‘1’表示第 LcuIndex 个最大编码单元 compIndex 分量的样本应使用增强样值偏移补偿；值为‘0’表示第 LcuIndex 个最大编码单元 compIndex 分量的样本不应使用增强样值偏移补偿。解析过程见 7.3。EsaoLcuEnableFlag[compIndex][LcuIndex]的值等于

T/AI 109.8-2025

esao_lcu_enable_flag[compIndex][LcuIndex]的值。如果位流中不存在esao_lcu_enable_flag[compIndex][LcuIndex]，EsaoLcuEnableFlag[compIndex][LcuIndex]的值等于PictureEsaoEnableFlag[compIndex]的值。

最大编码单元增强样值偏移补偿组索引 esao_lcu_set_index[compIndex][LcuIndex]

表示第LcuIndex个最大编码单元compIndex分量的样本应使用的增强样值偏移补偿系数组，解析过程见7.3。esao_lcu_set_index[compIndex][LcuIndex]的取值范围是0～1。EsaoLcuSetIndex[compIndex][LcuIndex]的值等于esao_lcu_set_index[compIndex][LcuIndex]的值。如果位流中不存在esao_lcu_set_index[compIndex][LcuIndex]，EsaoLcuSetIndex[compIndex][LcuIndex]的值为0。

最大编码单元跨分量样值偏移补偿允许标志 ccsao_lcu_enable_flag[compIndex][LcuIndex]

二值变量。值为‘1’表示第LcuIndex个最大编码单元compIndex分量的样本应使用跨分量样值偏移补偿；值为‘0’表示第LcuIndex个最大编码单元compIndex分量的样本不应使用跨分量样值偏移补偿。解析过程见7.3。CcsaoLcuEnableFlag[compIndex][LcuIndex]的值等于ccsao_lcu_enable_flag[compIndex][LcuIndex]的值。如果位流中不存在ccsao_lcu_enable_flag[compIndex][LcuIndex]，CcsaoLcuEnableFlag[compIndex][LcuIndex]的值等于PictureCcsaoEnableFlag[compIndex]的值。

最大编码单元跨分量样值偏移补偿组索引 ccsao_lcu_set_index[compIndex][LcuIndex]

表示第LcuIndex个最大编码单元compIndex分量的样本应使用的跨分量样值偏移补偿组，解析过程见7.3。CcsaoLcuSetIndex[compIndex][LcuIndex]的值等于ccaso_lcu_set_index[compIndex][LcuIndex]的值。ccsao_lcu_set_index[compIndex][LcuIndex]的取值范围是0～3。如果位流中不存在ccsao_lcu_set_index[compIndex][LcuIndex]，CcsaoLcuSetIndex[compIndex][LcuIndex]的值为0。

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

二值变量。值为‘1’表示第LcuIndex个最大编码单元compIndex分量的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compIndex分量的样本不应使用自适应修正滤波。AlfLcuEnableFlag[compIndex][LcuIndex]的值等于alf_lcu_enable_flag[compIndex][LcuIndex]的值。

最大编码单元神经网络滤波允许标志 nn_filter_lcu_enable_flag[compIdx][LcuIdx]

二值变量。值为‘1’表示第LcuIdx个最大编码单元compIdx分量的样本应使用神经网络滤波；值为‘0’表示第LcuIdx个最大编码单元compIdx分量的样本不应使用神经网络滤波。解析过程见7.3。NnFilterLcuEnableFlag[compIdx][LcuIdx]的值等于nn_filter_lcu_enable_flag[compIdx][LcuIdx]的值。如果位流中不存在nn_filter_lcu_enable_flag[compIdx][LcuIdx]，NnFilterLcuEnableFlag[compIdx][LcuIdx]值为0。解码处理应该忽略这些位。

最大编码单元神经网络滤波模型索引 nn_filter_lcu_set_index[compIdx][LcuIdx]

表示第LcuIdx个最大编码单元compIdx分量的样本使用的神经网络滤波的模型索引，解析过程见7.3，取值范围是0～NumOfNnFilter-1。NnFilterLCUSetIndex[compIdx][LcuIdx]的值等于nn_filter_lcu_set_index[compIdx][LcuIdx]的值。如果位流中不存在nn_filter_lcu_set_index[compIdx][LcuIdx]，NnFilterLCUSetIndex[compIdx][LcuIdx]值为-1。解码处理应该忽略这些位。

高级熵编码最大编码单元填充位 aec_lcu_stuffing_bit

填充位。片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

T/AI 109.8-2025

7.3。

6.2.11 编码树语义

四叉树划分标志 qt_split_flag

二值变量。值为‘1’表示应使用划分过程进行四叉树划分；值为‘0’表示不应进行四叉树划分。解析过程见 7.3。QtSplitFlag 的值等于 qt_split_flag。如果位流中不存在 qt_split_flag，QtSplitFlag 的值等于 allowSplitQt 的值。

编码单元预测模式限制 root_cu_constraint

二值变量。值为‘1’表示当前编码树节点覆盖的编码单元的预测模式限制是‘PRED_Intra_Only’；值为‘0’表示当前编码树节点覆盖的编码单元的预测模式限制是‘PRED_Inter_Only’。如果位流中不存在 root_cu_constraint，当前编码单元的预测模式限制应与父节点一致。

二叉树扩展四叉树划分标志 bet_split_flag

二值变量。值为‘1’表示应使用划分过程进行二叉树扩展四叉树划分；值为‘0’表示不应进行二叉树扩展四叉树划分。解析过程见 7.3。BetSplitFlag 的值等于 bet_split_flag。如果位流中不存在 bet_split_flag，BetSplitFlag 等于 allowNoSplit 的值取反。

二叉树扩展四叉树划分类型标志 bet_split_type_flag

二值变量。值为‘0’表示进行二叉树扩展四叉树划分时应使用二叉树划分；值为‘1’表示进行二叉树扩展四叉树划分时应使用扩展四叉树。解析过程见 7.3。BetSplitTypeFlag 的值等于 bet_split_type_flag。如果位流中不存在 bet_split_type_flag，BetSplitTypeFlag 的值按以下方式确定：

——如果 BetSplitFlag 的值为 0，则 BetSplitTypeFlag 等于 0；

——否则，如果 allowSplitBtVer 的值为 1 或 allowSplitBtHor 的值为 1，则 BetSplitTypeFlag 等于 0；

——否则，如果 allowSplitBtVer 和 allowSplitBtHor 的值均为 0，则 BetSplitTypeFlag 等于 1。

二叉树扩展四叉树划分方向标志 bet_split_dir_flag

二值变量。值为‘1’表示进行二叉树扩展四叉树划分时应使用垂直划分；值为‘0’表示进行二叉树扩展四叉树划分时应使用水平划分。解析过程见 7.3。BetSplitDirFlag 的值等于 bet_split_dir_flag。如果位流中不存在 bet_split_dir_flag，BetSplitDirFlag 的值按以下方式确定：

——如果 BetSplitFlag 的值为 0，则 BetSplitDirFlag 等于 0；

——否则，如果 BetSplitTypeFlag 的值为 0，则 BetSplitDirFlag 等于 allowSplitBtVer；

——否则，如果 BetSplitTypeFlag 的值为 1，则 BetSplitDirFlag 等于 allowSplitEqtVer。

由 QtSplitFlag、BetSplitFlag、BetSplitTypeFlag 和 BetSplitDirFlag 查表 72 得到 BlockSplitMode。

表72 BlockSplitMode

QtSplitFlag	BetSplitFlag	BetSplitTypeFlag	BetSplitDirFlag	BlockSplitMode
0	0	0	0	NO_SPLIT
1	0	0	0	SPLIT_QT
0	1	0	1	SPLIT_BT_VER

T/AI 109.8-2025

表 72 BlockSplitMode（续）

QtSplitFlag	BetSplitFlag	BetSplitTypeFlag	BetSplitDirFlag	BlockSplitMode
0	1	0	0	SPLIT_BT_HOR
0	1	1	1	SPLIT_EQT_VER
0	1	1	0	SPLIT_EQT_HOR

如果至少满足以下条件之一，则 ChildSizeOccur4 的值为 1；否则，ChildSizeOccur4 的值为 0：

——BlockSplitMode 的值为‘SPLIT_QT’且当前节点的宽度或高度等于 8；

——BlockSplitMode 的值为‘SPLIT_BT_VER’且当前节点的宽度等于 8；

——BlockSplitMode 的值为‘SPLIT_BT_HOR’且当前节点的高度等于 8；

——BlockSplitMode 的值为‘SPLIT_EQT_VER’且当前节点的宽度等于 16 或高度等于 8；

——BlockSplitMode 的值为‘SPLIT_EQT_HOR’且当前节点的高度等于 16 或宽度等于 8。

编码单元权限等级 privacy_level

二值变量。值为‘1’表示当前编码单元内所有样本点的权限等级为 1；值为‘0’表示当前编码单元内所有样本点的权限等级为 0。解析过程见 7.3。PrivacyLevel 的值等于 privacy_level 的值。如果位流中不存在 privacy_level，PrivacyLevel 的值等于 0。若一个色度编码单元对应大于一个的亮度编码单元，这些亮度编码单元内的 PrivacyLevel 的值应该均相等。

6.2.12 编码单元语义

跳过模式标志 skip_flag

二值变量。值为‘1’表示当前编码单元使用跳过模式；值为‘0’表示不是跳过模式。解析过程见 7.3。SkipFlag 的值等于 skip_flag 的值。如果位流不存在 skip_flag，SkipFlag 的值为 0。

高级预测模式标志 advanced_pred_flag

二值变量。值为‘1’表示当前编码单元使用高级运动矢量表达模式或角加权预测模式或增强运动矢量预测模式；值为‘0’表示不是高级运动矢量表达模式或角加权预测模式或增强运动矢量预测模式。解析过程见 7.3。AdvancedPredFlag 的值等于 advanced_pred_flag 的值。如果位流中不存在 advanced_pred_flag，AdvancedPredFlag 的值为 0。

增强时域运动矢量预测模式标志 etmvp_flag

二值变量。值为‘1’表示当前编码单元使用增强时域运动矢量预测模式；值为‘0’表示当前编码单元不是增强时域运动矢量预测模式。解析过程见 7.3。EtmvpFlag 的值等于 etmvp_flag 的值。如果位流中不存在 etmvp_flag，EtmvpFlag 的值如下：

```
EtmvpFlag = AdvancedPredFlag && AllowEtmvp
```

角度加权预测模式标志 awp_flag

二值变量。值为‘1’表示当前编码单元是角度加权预测模式；值为‘0’表示当前编码单元不是角度加权预测模式。解析过程见 7.3。AwpFlag 的值等于 awp_flag 的值。如果位流中不存在 awp_flag， AwpFlag 的值如下：

```
AwpFlag = AdvancedPredFlag && (! EtmvpFlag) && AllowAwp
```

UmveFlag 的值如下：

```
UmveFlag = AdvancedPredFlag && (! EtmvpFlag) && (! AwpFlag)
```

仿射模式标志 affine_flag

二值变量。值为‘1’表示当前编码单元是仿射模式；值为‘0’表示不是仿射模式。解析过程见

T/AI 109.8-2025

7.3。AffineFlag 的值等于 affine_flag 的值。如果位流中不存在 affine_flag，AffineFlag 的值为 0。

仿射高级运动矢量表达模式标志 affine_umve_flag

二值变量。值为‘1’表示当前编码单元是仿射高级运动矢量表达模式；值为‘0’表示不是仿射高级运动矢量表达模式。解析过程见 7.3。AffineUmveFlag 的值等于 affine_umve_flag 的值。如果位流中不存在 affine_umve_flag，AffineUmveFlag 的值为 0。

直接模式标志 direct_flag

二值变量。值为‘1’表示当前编码单元是直接模式；值为‘0’表示不是直接模式。解析过程见 7.3。directFlag 的值等于 direct_flag 的值。如果位流中不存在 direct_flag，directFlag 的值为 0。

帧间预测滤波标志 inter_pf_flag

二值变量。值为‘1’表示当前编码单元应使用帧间预测滤波；值为‘0’表示不应使用帧间预测滤波。解析过程见 7.3。InterPfFlag 的值等于 inter_pf_flag 的值。如果位流中不存在 inter_pf_flag，InterPfFlag 的值为 0。

帧间预测滤波索引 inter_pf_index

帧间预测滤波的索引值。解析过程见 7.3。InterPfIndex 的值等于 inter_pf_index 的值。InterPfIndex 的值应为 0 或 1。

帧间预测修正标志 inter_pc_flag

二值变量。值为‘1’表示当前编码单元应使用帧间预测修正；值为‘0’表示不应使用帧间预测修正。解析过程见 7.3。InterPcFlag 的值等于 inter_pc_flag 的值。如果位流中不存在 inter_pc_flag，InterPcFlag 的值为 0。

帧间预测修正索引 inter_pc_index

帧间预测修正索引值。解析过程见 7.3。InterPcIndex 的值等于 inter_pc_index 的值。InterPcIndex 的值应为 0、1 或 2。

帧内编码单元标志 intra_cu_flag

二值变量。值为‘1’表示当前编码单元的预测类型是普通帧内预测或块复制帧内预测或串复制帧内预测；值为‘0’表示当前编码单元的预测类型是帧间预测。解析过程见 7.3。IntraCuFlag 的值等于 intra_cu_flag。如果位流中不存在 intra_cu_flag，则：

——如果当前图像是 I 图像，或当前图像不是 I 图像且当前编码单元的预测模式限制是‘PRED_Intra_Only’，则 IntraCuFlag 的值为 1；

——如果当前编码单元只包含色度编码块且 PriorCuMode 的值等于 1，则 IntraCuFlag 的值为 1；

——否则，IntraCuFlag 的值为 0。

基础运动矢量索引 umve_mv_index

高级运动矢量表达模式中使用的基础运动矢量的索引值。解析过程见 7.3。解码过程见 8.6.7.8.5。UmveMvIndex 的值等于 umve_mv_index 的值。UmveMvIndex 的值应为 0 或 1。

运动矢量偏移量索引 umve_step_index

高级运动矢量表达模式中使用的运动矢量偏移量的索引值。解析过程见 7.3。解码过程见 8.6.7.8.5。UmveStepIndex 的值等于 umve_step_index 的值。

运动矢量方向索引 umve_dir_index

高级运动矢量表达模式中使用的运动矢量方向的索引值。解析过程见 7.3。解码过程见 8.6.7.8.5。UmveDirIndex 的值等于 umve_dir_index 的值。

仿射运动矢量索引 cu_affine_cand_index

T/AI 109.8-2025

跳过模式、直接模式或仿射高级运动矢量表达模式下的仿射模式索引值。解析过程见 7.3。解码过程见 8.6.7。AffineCandIndex 的值等于 cu_affine_cand_index 的值。如果位流中不存在 cu_affine_cand_index，AffineCandIndex 的值为 0。

增强时域运动矢量预测模式索引 cu_etmvp_cand_index

增强时域运动矢量预测模式索引值。解析过程见 7.3。解码过程见 8.6.7。EtmvpCandIndex 的值等于 cu_etmvp_cand_idex 的值。

角度加权预测模式索引 awp_index

角度加权预测模式索引值。解析过程见 7.3。如果当前图像为 B 图像，AwpIndex 的值等于 awp_index 的值；否则，根据 awp_index 的值以及当前编码单元的宽度 W 和高度 H 得到 AwpIndex 的值。

```
AwpIndex = awpWxH[awp_index] << 1
```

其中，W 和 H 为 8、16、32、64，awpWxH 的规定如下：

```
awp8x8 = {8,13,14,15,16,22}
awp8x16 = {4,5,6,12,13,14,15,18,20,21,22,24}
awp8x32 = {0,1,2,4,5,6,8,9,10,12,13,14,15,16,17,18,20,21,22,24,25,26}
awp8x64 = {0,1,2,4,5,6,8,9,10,12,13,14,15,16,17,18,20,21,22,24,25,26}
awp16x8 = {3,4,6,12,13,14,15,18,20,24,26,27}
awp16x16 = {2,3,4,5,6,8,12,13,14,15,16,18,20,21,22,24,26,27}
awp16x32 = {0,1,2,3,4,5,6,8,9,10,12,13,14,15,16,17,18,20,21,22,24,25,26,27}
awp16x64 = {0,1,2,3,4,5,6,8,9,10,12,13,14,15,16,17,18,20,21,22,24,25,26,27}
awp32x8 = {0,2,3,4,6,7,8,10,11,12,13,14,15,16,18,19,20,22,24,26,27}
awp32x16 = {0,1,2,3,4,6,7,8,10,11,12,13,14,15,16,17,18,19,20,21,22,24,25,26,27}
awp32x32 = {0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,24,25,26,27}
awp32x64 = {0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,24,25,26,27}
awp64x8 = {0,2,3,4,6,7,8,10,11,12,13,14,15,16,18,19,20,22,23,24,26,27}
awp64x16 = {0,1,2,3,4,6,7,8,10,11,12,13,14,15,16,18,19,20,22,23,24,25,26,27}
awp64x32 = {0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,26,27}
awp64x64 = {0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,25,26,27}
```

如果位流中不存在 awp_index，AwpIndex 的值为 0。

角度加权预测第一运动矢量修正标志 awp_mvr_cand_flag0

二值变量。值为‘1’表示应修正角度加权预测第一运动矢量；值为‘0’表示不应修正角度加权预测第一运动矢量。解析过程见 7.3。AwpMvrCandFlag0 的值等于 awp_mvr_cand_flag0 的值。如果位流中不存在 awp_mvr_cand_flag0，AwpMvrCandFlag0 的值为 0。

角度加权预测第一运动矢量修正步长索引值 awp_mvr_cand_step0

角度加权预测第一运动矢量修正步长索引值。解析过程见 7.3。解码过程见 8.6.7。AwpMvrCandStep0 的值等于 awp_mvr_cand_step0 的值。如果位流中不存在 awp_mvr_cand_step0，AwpMvrCandStep0 的值为 0。

角度加权预测第一运动矢量修正方向索引值 awp_mvr_cand_dir0

角度加权预测第一运动矢量修正方向索引值。解析过程见 7.3。解码过程见 8.6.7。AwpMvrCandDir0 的值等于 awp_mvr_cand_dir0 的值。如果位流中不存在 awp_mvr_cand_dir0，AwpMvrCandDir0 的值为 0。

角度加权预测第二运动矢量修正标志 awp_mvr_cand_flag1

T/AI 109.8-2025

二值变量。值为‘1’表示应修正角度加权预测第二运动矢量；值为‘0’表示不应修正角度加权预测第二运动矢量。解析过程见 7.3。AwpMvrCandFlag1 的值等于 awp_mvr_cand_flag1 的值。如果位流中不存在 awp_mvr_cand_flag1，AwpMvrCandFlag1 的值为 0。

角度加权预测第二运动矢量修正步长索引值 awp_mvr_cand_step1

角度加权预测第二运动矢量修正步长索引值。解析过程见 7.3。解码过程见 8.6.7。AwpMvrCandStep1 的值等于 awp_mvr_cand_step1 的值。如果位流中不存在 awp_mvr_cand_step1，AwpMvrCandStep1 的值为 0。

角度加权预测第二运动矢量修正方向索引值 awp_mvr_cand_dir1

角度加权预测第二运动矢量修正方向索引值。解析过程见 7.3。解码过程见 8.6.7。AwpMvrCandDir1 的值等于 awp_mvr_cand_dir1 的值。如果位流中不存在 awp_mvr_cand_dir1，AwpMvrCandDir1 的值为 0。

角度加权预测模式第一运动信息索引 awp_cand_index0

角度加权预测模式第一运动信息索引值。解析过程见 7.3。解码过程见 8.6.7。AwpCandIndex0 的值等于 awp_cand_index0 的值。如果位流中不存在 awp_cand_index0，AwpCandIndex0 的值为 0。

角度加权预测模式第二运动信息索引 awp_cand_index1

角度加权预测模式第二运动信息索引值。解析过程见 7.3。解码过程见 8.6.7。AwpCandIndex1 的值等于 awp_cand_index1 的值。如果(AwpMvrCandFlag0 == 0 && AwpMvrCandFlag1 == 0) || (AwpMvrCandFlag0 == 1 && AwpMvrCandFlag1 == 1 && AwpMvrCandStep0 == AwpMvrCandStep1 && AwpMvrCandDir0 == AwpMvrCandDir1) && (AwpCandIndex1 >= AwpCandIndex0)，AwpCandIndex1 等于 AwpCandIndex1+1。如果位流中不存在 awp_cand_indxe1，AwpCandIndex1 的值为 0。

仿射运动矢量偏移量索引 affine_umve_step_index0，affine_umve_step_index1

仿射高级运动矢量表达模式中控制点 0 和控制点 1 使用的运动矢量偏移量的索引值。解析过程见 7.3。解码过程见 8.6.7。AffineUmveStepIndex0 和 AffineUmveStepIndex1 的值分别等于 affine_umve_step_index0 和 affine_umve_step_index1 的值。

仿射运动矢量方向索引 affine_umve_dir_index0，affine_umve_dir_index1

仿射高级运动矢量表达模式中控制点 0 和控制点 1 使用的运动矢量方向的索引值。解析过程见 7.3。解码过程见 8.6.7。AffineUmveDirIndex0 和 AffineUmveDirIndex1 的值分别等于 affine_umve_dir_index0 和 affine_umve_dir_index1 的值。

空域角度加权预测模式标志 sawp_flag

二值变量。值为‘1’表示当前编码单元是空域角度加权预测模式；值为‘0’表示当前编码单元不是空域角度加权预测模式。解析过程见 7.3。SawpFlag 的值等于 sawp_flag 的值。如果位流中不存在 sawp_flag，SawpFlag 的值为 0。

空域角度加权预测模式索引 sawp_index

空域角度加权预测模式索引值。解析过程见 7.3。SawpIndex 的值等于 sawp_index 的值。如果位流中不存在 sawp_index，SawpIndex 的值为 0。

空域角度加权预测模式第一预测模式索引 sawp_pred_mode0_index

空域角度加权预测模式第一预测模式的索引值。解析过程见 7.3。解码过程见 8.6.6。sawp_pred_mode0_index 的取值范围是 0～25。SawpPredMode0Index 的值等于 sawp_pred_mode0_index 的值。

空域角度加权预测模式第二预测模式索引 sawp_pred_mode1_index

空域角度加权预测模式第二预测模式的索引值。解析过程见 7.3。解码过程见 8.6.6。sawp_pred_mode1_index 的取值范围是 0～25。SawpPredMode1Index 的值等于 sawp_pred_mode1_index

T/AI 109.8-2025

的值。

衍生模式划分标志 dt_split_flag

二值变量。值为‘1’表示应进行衍生模式划分；值为‘0’表示不应进行衍生模式划分。解析过程见 7.3。如果位流中不存在 dt_split_flag，DtSplitFlag 的值为 0。

衍生模式划分方向 dt_split_dir

二值变量。值为‘1’表示进行水平衍生模式划分；值为‘0’表示进行垂直衍生模式划分。解析过程见 7.3。DtSplitDir 的值等于 dt_split_dir 的值。

水平四叉衍生模式划分标志 dt_split_hqt_flag

二值变量。值为‘1’表示应进行水平四叉衍生模式划分；值为‘0’表示不应进行水平四叉衍生模式划分。解析过程见 7.3。DtSplitHqtFlag 的值等于 dt_split_hqt_flag 的值。

垂直四叉衍生模式划分标志 dt_split_vqt_flag

二值变量。值为‘1’表示应进行垂直四叉衍生模式划分；值为‘0’表示不应进行垂直四叉衍生模式划分。解析过程见 7.3。DtSplitVqtFlag 的值等于 dt_split_vqt_flag 的值。

水平非对称衍生模式标志 dt_split_hadt_flag

二值变量。值为‘0’表示应进行‘SIZE_2MxnU’模式划分；值为‘1’表示应进行‘SIZE_2MxnD’模式划分。解析过程见 7.3。DtSplitHadtFlag 的值等于 dt_split_hadt_flag 的值。

垂直非对称衍生模式标志 dt_split_vadt_flag

二值变量。值为‘0’表示应进行‘SIZE_nLx2N’模式划分；值为‘1’表示应进行‘SIZE_nRx2N’模式划分。解析过程见 7.3。DtSplitVadtFlag 的值等于 dt_split_vadt_flag 的值。

仿射自适应运动矢量精度索引 affine_amvr_index

编码单元的仿射运动矢量精度。解析过程见 7.3，解码过程见 8.6.7。AffineAmvrIndex 的值等于 affine_amvr_index 的值。如果位流中不存在 affine_amvr_index，AffineAmvrIndex 的值为 0。

自适应运动矢量精度索引 amvr_index

编码单元的运动矢量精度。解析过程见 7.3，解码过程见 8.6.7。AmvrIndex 的值等于 amvr_index 的值。如果位流中不存在 amvr_index，AmvrIndex 的值为 0。

编码单元子类型索引 cu_subtype_index

当前编码单元的跳过模式或直接模式的运动信息索引值。解析过程见 7.3。CuSubtypeIndex 的值等于 cu_subtype_index 的值。如果位流中不存在 cu_subtype_index，CuSubtypeIndex 的值为 0。如果当前图像是 P 图像，cu_subtype_index 值的取值范围是 0～1 + Max(NumOfMvapCand,NumOfHmvpCand)。如果当前图像是 B 图像，cu_subtype_index 值的取值范围是 0～3 + Max(NumOfMvapCand,NumOfHmvpCand)。

串复制或块复制帧内预测标志 isc_ibc_cu_flag

二值变量。值为‘1’表示当前编码单元是串复制帧内预测模式或块复制帧内预测模式；值为‘0’表示当前编码单元既不是串复制帧内预测模式也不是块复制帧内预测模式。解析过程见 7.3。IscIbcCuFlag 的值等于 isc_ibc_cu_flag 的值。如果位流中不存在 isc_ibc_cu_flag，IscIbcCuFlag 的值为 0。

块复制帧内模式标志 ibc_cu_flag

二值变量。值为‘1’表示当前编码单元是块复制帧内预测模式；值为‘0’表示当前编码单元不是块复制帧内预测模式。解析过程见 7.3。IbcCuFlag 的值等于 ibc_cu_flag。如果位流中不存在 ibc_cu_flag，IbcCuFlag 的值为 0。

IscCuFlag 的值为 1 表示当前编码单元是串复制帧内预测模式；值为 0 表示当前编码单元不是串复制帧内预测模式。IscCuFlag 的值如下：

T/AI 109.8-2025

IscCuFlag = IscIbcCuFlag && (! IbcCuFlag)

串复制帧内预测子模式标志 isc_subtype_flag

二值变量。解析过程见 7.3。如果位流中存在 isc_subtype_flag，IscSubtypeFlag 的值等于 isc_subtype_flag 的值。如果位流中不存在 isc_subtype_flag，IscSubtypeFlag 的值按以下方法确定：

IscSubtypeFlag = PictureIscEvsUbvsEnableFlag ? 1 : 0

IscSubtypeFlag 值为‘1’表示当前编码单元是串复制帧内预测非普通串子模式；值为‘0’表示当前编码单元是串复制帧内预测普通串子模式。

预测块矢量类别索引 cbvp_index

当前编码单元预测块矢量索引值。解析过程见 7.3。解码过程见 8.6.6。CbvpIndex 的值等于 cbvp_index 的值。如果位流中不存在 cbvp_index，CbvpIndex 的值为 0。

自适应块矢量精度索引 abvr_index

用于确定编码单元的块矢量精度。解析过程见 7.3，解码过程见 8.6.6。如果 abvr_index 值为‘0’，AbvrIndex 的值等于 0。如果 abvr_index 值为‘1’，AbvrIndex 的值等于 2。如果位流中不存在 abvr_index，AbvrIndex 的值为 0。

块矢量水平分量差绝对值 mv_diff_x_abs_bv

块矢量垂直分量差绝对值 mv_diff_y_abs_bv

块矢量差值的绝对值。MvDiffXAbsBv 等于 mv_diff_x_abs_bv 的值，MvDiffYAbsBv 等于 mv_diff_y_abs_bv 的值。

块矢量水平分量差符号值 mv_diff_x_sign_bv

块矢量垂直分量差符号值 mv_diff_y_sign_bv

块矢量差值的符号位。MvDiffXSignbV 的值等于 mv_diff_x_sign_bv 的值，MvDiffYSignBv 的值等于 mv_diff_y_sign_bv。如果位流中不存在 mv_diff_x_sign_bv 或 mv_diff_y_sign_bv，MvDiffXSignBv 或 MvDiffYSignBv 的值为 0。

MvDiffXBv 和 MvDiffYBv 的值如下，取值范围是-32768～32767。

$$MvDiffXBv = (-1)^{MvDiffXSignBv} * MvDiffXAbsBv$$

$$MvDiffYBv = (-1)^{MvDiffYSignBv} * MvDiffYAbsBv$$

预测参考模式索引 inter_pred_ref_mode_index

用于确定当前预测单元使用的预测参考模式。解析过程见 7.3。解码过程见 8.6.3。InterPredRefModeIndex 等于 inter_pred_ref_mode_index 的值。如果位流中不存在 inter_pred_ref_mode_index，或当前图像是 P 图像，InterPredRefModeIndex 的值为 0。

对称运动矢量差标志 smvd_flag

二值变量。值为‘1’表示当前预测单元应使用对称运动矢量差模式；值为‘0’表示当前编码单元不应使用对称运动矢量差模式。解析过程见 7.3。SmvdFlag 的值等于 smvd_flag 的值。如果位流中不存在 smvd_flag，SmvdFlag 的值为 0。

如果 DistanceIndex-DistanceIndexL0 等于 DistanceIndexL1-DistanceIndex，SmvdApplyFlag 的值为 1；否则，SmvdApplyFlag 的值为 0。DistanceIndexL0 是当前图像的参考图像队列 0 中第 0 幅图像的距离索引，DistanceIndexL1 是当前图像的参考图像队列 1 中第 0 幅图像的距离索引。

运动矢量精度扩展模式标识 extend_mvr_flag

二值变量。值为‘1’表示当前预测单元应使用运动矢量扩展精度模式。值为‘0’表示当前预测单元不应使用运动矢量扩展精度模式。解析过程见 7.3。解码过程见 8.6.7.6。ExtendMvrFlag 的值等于 extend_mvr_flag 的值。

帧内亮度预测模式索引 intra_luma_pred_mode_index

T/AI 109.8-2025

亮度编码块的帧内预测模式。解析过程见 7.3。解码过程见 8.6.6。intra_luma_pred_mode_index 的取值范围是 0～33。IntraLumaPredModeIndex 的值等于 intra_luma_pred_mode_index 的值。

如果 NumOfIntraPredBlock 的值大于 1，从符合本文件的位流中解码得到的 IntraLumaPredMode 的值不应为 33。

亮度帧内预测扩展模式索引 luma_eipm_index

用于确定当前编码单元的亮度帧内预测扩展模式。解析过程见 7.3。解码过程见 8.6.6.3。LumaEipmIndex 的值等于 luma_eipm_index 的值。如果位流中不存在 luma_eipm_index，LumaEipmIndex 的值为 0。

帧内色度预测模式索引 intra_chroma_pred_mode_index

用于确定 4:2:0 格式下当前编码单元中两个色度编码块的帧内预测模式。解析过程见 7.3。解码过程见 8.6.6.3。IntraChromaPredModeIndex 的值等于 intra_chroma_pred_mode_index 的值。如果位流中不存在 intra_chroma_pred_mode_index，IntraChromaPredModeIndex 的值等于-1。如果位流中不存在 intra_chroma_pred_mode_index，IntraChromaPredMode 的值为-1。

如果 isRedundant（见 8.6.6.3）的值为 1，intra_chroma_pred_mode_index 的值不应为((TscpmEnableFlag || PmcEnableFlag) ? 5 : 4)。

如果当前编码单元只包含色度编码块，若 IsPrivacyFlag 的值为 0 且 UserPermission 的值为 1 且 PrivacyModeIndex 的值为 2 且 PrivacyLevel 的值为 1，PriorCuMode 的值等于 PriorCuModeLuma 的值，否则，PriorCuMode 的值由当前色度编码块右下角色度样本位置上的亮度样本所在的编码单元的预测类型决定。如果该编码单元的预测类型是普通帧内预测，PriorCuMode 的值为 1；否则，如果编码单元的预测类型是块复制帧内预测或串复制帧内预测，PriorCuMode 的值为 2；否则，如果该编码单元的预测类型是帧间预测，PriorCuMode 的值为 0。

如果当前编码单元只包含色度编码块且 PriorCuMode 为 1，当前编码单元的预测类型是普通帧内预测，则令其色度编码块对应的普通亮度帧内预测模式 IntraLumaPredMode 等于当前色度编码块右下角色度样本位置上的亮度样本所在的空域帧内预测信息存储单元的帧内亮度预测模式。

如果当前编码单元只包含色度编码块且 PriorCuMode 为 2，当前编码单元的预测类型是普通帧内预测，则令其色度编码块对应的普通亮度帧内预测模式 IntraLumaPredMode 等于‘Intra_Luma_DC’。

如果当前编码单元包含亮度编码块和色度编码块，当前色度编码块对应的普通亮度帧内预测模式 IntraLumaPredMode 等于编码单元中 PredBlockOrder 的值为 0 的预测块的普通亮度预测模式。

如果当前编码单元只包含色度编码块且 PriorCuMode 为 0，当前编码单元的预测类型是帧间预测，则令当前色度编码块的运动信息等于当前色度编码块右下角色度样本位置上的亮度样本所在的空域运动信息存储单元的运动信息。

色度帧内预测扩展模式索引 chroma_eipm_index

用于确定当前编码单元的色度帧内预测扩展模式。解析过程见 7.3。解码过程见 8.6.6.3。ChromaEipmIndex 的值等于 chroma_eipm_index。如果位流中不存在 chroma_eipm_index，ChromaEipmIndex 的值为 0。

色度帧内跨多分量预测模式索引 chroma_pmc_index

用于确定当前编码单元的色度帧内跨多分量预测模式。解析过程见 7.3。解码过程见 8.6.6.3。ChromaPmcIndex 的值等于 chroma_pmc_index。如果位流中不存在 chroma_pmc_index，ChromaPmcIndex 的值等于 PmcEnableFlag 的值。

帧内色度跨多分量预测参数索引 chroma_pmc_param_index

用于确定当前编码单元的色度帧内跨多分量预测参数。解析过程见 7.3。解码过程见 8.6.6.3。ChromaPmcParamIndex 的值等于 chroma_pmc_param_index。如果位流中不存在

T/AI 109.8-2025

chroma_pmc_param_index，ChromaPmcParamIndex 的值为 0。

帧内预测滤波标志 intra_pf_flag

二值变量。值为‘1’表示当前编码单元应使用帧内预测滤波；值为‘0’表示当前编码单元不应使用帧内预测滤波。解析过程见 7.3。IntraPfFlag 的值等于 intra_pf_flag 的值。如果位流中不存在 intra_pf_flag，IntraPfFlag 的值为 0。

改进型帧内预测标志 iip_flag

二值变量。值为‘1’表示当前编码单元应使用改进型帧内预测；值为‘0’表示当前编码单元不应使用改进型帧内预测。解析过程见 7.3。IipFlag 的值等于 iip_flag 的值。如果位流中不存在 iip_flag，IipFlag 的值为 0。

L0 预测单元参考索引 pu_reference_index_l0

预测单元的 L0 参考索引。解析过程见 7.3。RefIndexL0 的值等于 pu_reference_index_l0 的值。如果位流中不存在 pu_reference_index_l0，则 RefIndexL0 的值为 0。pu_reference_index_l0 的值应小于 NumRefActive[0]的值。

L0 运动矢量水平分量差绝对值 mv_diff_x_abs_l0

L0 运动矢量垂直分量差绝对值 mv_diff_y_abs_l0

来自参考图像队列 0 的运动矢量差值的绝对值。解析过程见 7.3。MvDiffXAbsL0 的值等于 mv_diff_x_abs_l0 的值，MvDiffYAbsL0 的值等于 mv_diff_y_abs_l0 的值。

L0 运动矢量水平分量差符号值 mv_diff_x_sign_l0

L0 运动矢量垂直分量差符号值 mv_diff_y_sign_l0

来自参考图像队列 0 的运动矢量差值的符号位。解析过程见 7.3。MvDiffXSignL0 的值等于 mv_diff_x_sign_l0 的值，MvDiffYSignL0 的值等于 mv_diff_y_sign_l0。如果位流中不存在 mv_diff_x_sign_l0 或 mv_diff_y_sign_l0，MvDiffXSignL0 或 MvDiffYSignL0 的值为 0。

MvDiffXL0 和 MvDiffYL0 的计算如下，取值范围是-32768～32767。

$$MvDiffXL0 = (-1)^{MvDiffXSignL0} * MvDiffXAbsL0$$

$$MvDiffYL0 = (-1)^{MvDiffYSignL0} * MvDiffYAbsL0$$

仿射帧间模式 L0 运动矢量水平分量差绝对值 mv_diff_x_abs_l0_affine

仿射帧间模式 L0 运动矢量垂直分量差绝对值 mv_diff_y_abs_l0_affine

来自参考图像队列 0 的运动矢量差值的绝对值。解析过程见 7.3。MvDiffXAbsL0Affine 的值等于 mv_diff_x_abs_l0_affine 的值，MvDiffYAbsAffineAffine 的值等于 mv_diff_y_abs_l0_affine 的值。

仿射帧间模式 L0 运动矢量水平分量差符号值 mv_diff_x_sign_l0_affine

仿射帧间模式 L0 运动矢量垂直分量差符号值 mv_diff_y_sign_l0_affine

来自参考图像队列 0 的运动矢量差值的符号位。解析过程见 7.3。MvDiffXSignL0Affine 的值等于 mv_diff_x_sign_l0_affine 的值，MvDiffYSignL0Affine 的值等于 mv_diff_y_sign_l0_affine。如果位流中不存在 mv_diff_x_sign_l0_affine 或 mv_diff_y_sign_l0_affine，MvDiffXSignL0Affine 或 MvDiffYSignL0Affine 的值为 0。

MvDiffXL0Affine 和 MvDiffYL0Affine 的计算如下，取值范围是-32768～32767。

$$MvDiffXL0Affine = (-1)^{MvDiffXSignL0Affine} * MvDiffXAbsL0Affine$$

$$MvDiffYL0Affine = (-1)^{MvDiffYSignL0Affine} * MvDiffYAbsL0Affine$$

L1 预测单元参考索引 pu_reference_index_l1

预测单元的 L1 参考索引。解析过程见 7.3。RefIndexL1 的值等于 pu_reference_index_l1 的值。如果位流中不存在 pu_reference_index_l1，RefIndexL1 的值为 0。pu_reference_index_l1 的值应小于 NumRefActive[1]的值。

T/AI 109.8-2025

L1 运动矢量水平分量差绝对值 mv_diff_x_abs_l1

L1 运动矢量垂直分量差绝对值 mv_diff_y_abs_l1

来自参考图像队列 1 的运动矢量差值的绝对值。解析过程见 7.3。MvDiffXAbsL1 的值等于 mv_diff_x_abs_l1 的值，MvDiffYAbsL1 的值等于 mv_diff_y_abs_l1 的值。

L1 运动矢量水平分量差符号值 mv_diff_x_sign_l1

L1 运动矢量垂直分量差符号值 mv_diff_y_sign_l1

来自参考图像队列 1 的运动矢量差值的符号位。解析过程见 7.3。MvDiffXSignL1 的值等于 mv_diff_x_sign_l1 的值，MvDiffYSignL1 的值等于 mv_diff_y_sign_l1。如果位流中不存在 mv_diff_x_sign_l1 或 mv_diff_y_sign_l1，MvDiffXSignL1 或 MvDiffYSignL1 的值为 0。

如果 SmvdFlag 的值为 0，MvDiffXL1 和 MvDiffYL1 的计算如下，取值范围是-32768～32767。

$$MvDiffXL1 = (-1)^{MvDiffXSignL1} * MvDiffXAbsL1$$
$$MvDiffYL1 = (-1)^{MvDiffYSignL1} * MvDiffYAbsL1$$

如果 SmvdFlag 的值为 1，MvDiffXL1 和 MvDiffYL1 的计算如下：

$$MvDiffXL1 = Clip3(-32768, 32767, -MvDiffXL0)$$
$$MvDiffYL1 = Clip3(-32768, 32767, -MvDiffYL0)$$

仿射帧间模式 L1 运动矢量水平分量差绝对值 mv_diff_x_abs_l1_affine

仿射帧间模式 L1 运动矢量垂直分量差绝对值 mv_diff_y_abs_l1_affine

来自参考图像队列 1 的运动矢量差值的绝对值。解析过程见 7.3。MvDiffXAbsL1Affine 的值等于 mv_diff_x_abs_l1_affine 的值，MvDiffYAbsAffineAffine 的值等于 mv_diff_y_abs_l1_affine 的值。

仿射帧间模式 L1 运动矢量水平分量差符号值 mv_diff_x_sign_l1_affine

仿射帧间模式 L1 运动矢量垂直分量差符号值 mv_diff_y_sign_l1_affine

来自参考图像队列 1 的运动矢量差值的符号位。解析过程见 7.3。MvDiffXSignL1Affine 的值等于 mv_diff_x_sign_l1_affine 的值，MvDiffYSignL1Affine 的值等于 mv_diff_y_sign_l1_affine。如果位流中不存在 mv_diff_x_sign_l1_affine 或 mv_diff_y_sign_l1_affine，MvDiffXSignL1Affine 或 MvDiffYSignL1Affine 的值为 0。

MvDiffXL1Affine 和 MvDiffYL1Affine 的计算如下，取值范围是-32768～32767。

$$MvDiffXL1Affine = (-1)^{MvDiffXSignL1Affine} * MvDiffXAbsL1Affine$$
$$MvDiffYL1Affine = (-1)^{MvDiffYSignL1Affine} * MvDiffYAbsL1Affine$$

双向梯度修正标志 bgc_flag

二值变量。解析过程见7.3。值为‘1’表示当前编码单元应使用双向梯度修正；值为‘0’表示不应使用双向梯度修正。BgcFlag的值等于bgc_flag的值。如果位流中不存在bgc_flag，BgcFlag的值为0。

双向梯度修正索引 bgc_index

双向梯度修正的索引值。解析过程见 7.3。BgcIndex 的值等于 bgc_index 的值。如果位流中不存在 bgc_index，则 BgcIndex 的值为 0。BgcIndex 的值应为 0 或 1。

变换块系数标志 ctp_zero_flag

二值变量。解析过程见 7.3。CtpZeroFlag 的值等于 ctp_zero_flag 的值。如果位流中不存在 ctp_zero_flag，CtpZeroFlag 的值为 0。

基于位置的变换块标志 pbt_cu_flag

二值变量。解析过程见 7.3。PbtCuFlag 的值等于 pbt_cu_flag 的值。如果位流中不存在 pbt_cu_flag，PbtCuFlag 的值为 0。

编码单元量化参数增量绝对值 cu_qp_delta_abs

T/AI 109.8-2025

当前编码单元量化参数相对于预测量化参数的增量的绝对值。解析过程见7.3。CuQpDeltaAbs的值等于cu_qp_delta_abs的值。

编码单元量化参数增量符号值 cu_qp_delta_sign

当前编码单元量化参数相对于预测量化参数的增量的符号位。解析过程见7.3。CuQpDeltaSign的值等于cu_qp_delta_sign的值。CuQpDeltaAbs的取值范围应是0～(32 + 4×(BitDepth-8))。

CuDeltaQp 的计算如下：

$$CuDeltaQp = (-1)^{CuQpDeltaSign} * CuQpDeltaAbs$$

CuDeltaQp 的取值范围为(-32 - 4×(BitDepth-8))～(31+ 4×(BitDepth-8))。如果位流中不存在 cu_qp_delta_abs，CuDeltaQp 的值为 0。

Cb 变换块编码模板 ctp_u

Cr 变换块编码模板 ctp_v

ctp_u 和 ctp_v 分别确定 Cb 变换块和 Cr 变换块中是否包含编码数据。解析过程见 7.3。如果位流中不存在 ctp_u，ctp_u 的值为 0；如果位流中不存在 ctp_v，ctp_v 的值为 0。

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 8.6.5）是否包含编码数据。解析过程见 7.3。如果位流中不存在 ctp_y[0]且同时满足以下条件，ctp_y[0]的值为 1；否则，ctp_y[0]的值为 0。

——CtpZeroFlag 的值为 0；
——PbtCuFlag 的值为 0；
——IntraCuFlag 的值为 0；
——SkipFlag 的值为 0；
——IscCuFlag 的值为 0；
——ctp_u 和 ctp_v 的值均为 0；
——当前编码单元包含亮度分量。

扩展变换跳过标志 enhanced_ts_flag

解析过程见 7.3。值为‘1’表示应使用扩展变换跳过；值为‘0’表示不应使用扩展变换跳过。EnhancedTsFlag 的值等于 enhanced_ts_flag 的值。如果位流中不存在 enhanced_ts_flag，EnhancedTsFlag 的值为 0。

子块变换标志 sbt_cu_flag

二值变量。值为‘1’表示当前编码单元应使用子块变换，值为‘0’表示当前编码单元不应使用子块变换。解析过程见 7.3。SbtCuFlag 的值等于 sbt_cu_flag 的值。如果位流中不存在 sbt_cu_flag，SbtCuFlag 的值为 0。

子块变换大小标志 sbt_quad_flag

二值变量。值为‘1’表示当前亮度变换块是亮度编码块的 1/4 大小，值为‘0’表示当前亮度变换块是亮度编码块的 1/2 大小。解析过程见 7.3。SbtQuadFlag 的值等于 sbt_quad_flag 的值。如果位流中不存在 sbt_quad_flag，SbtQuadFlag 的值为 0。

子块变换方向标志 sbt_dir_flag

二值变量。值为‘1’表示当前亮度编码块的宽度等于亮度变换块的宽度，值为‘0’表示当前亮度编码块的高度等于亮度变换块的高度。解析过程见 7.3。SbtDirFlag 的值等于 sbt_dir_flag 的值。如果位流中不存在 sbt_dir_flag，则当 SbtQuadFlag 为 1 时，SbtDirFlag 的值等于 SbtHorQuad 的值；当 SbtQuadFlag 为 0 时，SbtDirFlag 的值等于 SbtHorHalf 的值。

色度二次变换标志 chroma_st_flag

T/AI 109.8-2025

二值变量。值为‘1’表示当前色度变换块应使用二次变换，值为‘0’表示当前色度变换块不应使用二次变换。解析过程见 7.3。ChromaStFlag 的值等于 chroma_st_flag 的值。如果位流中不存在 chroma_st_flag，ChromaStFlag 的值为 0。

变换跳过标志 ts_cu_flag

二值变量。解析过程见 7.3。TsCuFlag 的值等于 ts_cu_flag 的值。如果位流中不存在 ts_cu_flag，TsCuFlag 的值为 0。

6.2.13 变换块语义

扫描表 ScanCGInBlk、ScanCoeffInCG、InvScanCoeffInBlk 和 ScanOrder 的规定见附录 A。

扫描区域右端横坐标 scan_region_x

扫描区域下端纵坐标 scan_region_y

用于确定系数扫描区域，解析过程见 7.3。scan_region_x 和 scan_region_y 的值应小于或等于 31。ScanRegionX 的值等于 scan_region_x 的值，ScanRegionY 的值等于 scan_region_y 的值。系数扫描区域是以(0,0)为左上角，以(ScanRegionX,ScanRegionY)为右下角的矩形区域。

系数非零标志 significant_flag

用于确定当前系数是否为非零系数，解析过程见 7.3。significant_flag 值为‘0’表示当前系数为零系数；值为‘1’表示当前系数为非零系数。SignificantFlag 的值等于 significant_flag 的值。

系数绝对值大于 1 标志 coeff_abs_level_greater1_flag

用于确定当前系数的绝对值是否大于 1，解析过程见 7.3。coeff_abs_level_greater1_flag 的值为‘0’表示当前系数的绝对值为 1；值为‘1’表示当前系数的绝对值大于 1。CoeffAbsLevelGt1Flag 的值等于 coeff_abs_level_greater1_flag 的值。如果位流中不存在 coeff_abs_level_greater1_flag，CoeffAbsLevelGt1Flag 的值等于 0。

系数绝对值大于 2 标志 coeff_abs_level_greater2_flag

用于确定当前系数的绝对值是否大于 2，解析过程见 7.3。coeff_abs_level_greater2_flag 的值为‘0’表示当前系数的绝对值为 2；值为‘1’表示当前系数的绝对值大于 2。CoeffAbsLevelGt2Flag 的值等于 coeff_abs_level_greater2_flag 的值。如果位流中不存在 coeff_abs_level_greater2_flag，CoeffAbsLevelGt2Flag 的值等于 0。

系数绝对值大于 4 标志 coeff_abs_level_greater4_flag

用于确定当前系数的绝对值是否大于 4，解析过程见 7.3。coeff_abs_level_greater4_flag 的值为‘0’表示当前系数的绝对值小于等于 4；值为‘1’表示当前系数的绝对值大于 4。CoeffAbsLevelGt4Flag 的值等于 coeff_abs_level_greater4_flag 的值。如果位流中不存在 coeff_abs_level_greater4_flag，CoeffAbsLevelGt4Flag 的值等于 0。

系数绝对值大于 8 标志 coeff_abs_level_greater8_flag

用于确定当前系数的绝对值是否大于 8，解析过程见 7.3。coeff_abs_level_greater8_flag 的值为‘0’表示当前系数的绝对值小于等于 8；值为‘1’表示当前系数的绝对值大于 8。CoeffAbsLevelGt8Flag 的值等于 coeff_abs_level_greater8_flag 的值。如果位流中不存在 coeff_abs_level_greater8_flag，CoeffAbsLevelGt8Flag 的值等于 0。

系数绝对值剩余值 coeff_abs_level_remaining

用于确定当前系数的绝对值的剩余值。解析过程见 7.3。CoeffAbsLevelRem 的值等于 coeff_abs_level_remaining 的值。

系数游程 coeff_run

T/AI 109.8-2025

用于确定游程的长度，解析过程见 7.3。coeff_run 的值不应大于(NumOfCoeff - ScanPosOffset - 1)，其中 NumOfCoeff 是当前变换块系数的个数，ScanPosOffset 是上一个非零系数的位置。

系数幅值 coeff_level_minus1

用于确定非零量化系数的幅值，解析过程见 7.3。

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 7.3。coeff_sign 值为‘0’表示当前系数是正数；值为‘1’表示当前系数是负数。CoeffSign 的值等于 coeff_sign 的值。

系数结束符 coeff_last

用于确定是否为非零量化系数块中最后一个非零系数，解析过程见 7.3。

增强二次变换标志 est_tu_flag

解析过程见 7.3。est_tu_flag 的值为‘1’表示当前变换块应使用增强的二次变换；值为‘0’表示当前变换块不应使用增强的二次变换。EstTuFlag 的值等于 est_tu_flag 的值。如果位流中不存在 est_tu_flag，EstTuFlag 的值为 0。

熵编码脉冲编码调制模式填充位 aec_ipcm_stuffing_bit

填充位。值应为‘1’，解析过程见 7.3。

高级熵编码字节对齐填充位 aec_byte_alignment_bit0

填充位。值应为‘0’。

脉冲编码调制模式系数 pcm_coeff

n 位无符号整数，n 的值等于 SamplePrecision 的值。用于确定脉冲调制编码模式的系数。

6.2.14 自适应修正滤波参数语义

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～FilterNum-1。AlfFilterNumMinus1 的值等于 alf_filter_num_minus1 的值。

图像亮度分量自适应修正滤波区域顺序索引 alf_region_order_index

alf_region_order_index 表示当前图像亮度分量自适应修正滤波区域顺序的索引值。AlfRegionOrderIndex 的值等于 alf_region_order_scan_index 的值。

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～FilterNum-1。当位流中不存在 alf_region_distance[i]时，如果 i 等于 0，alf_region_distance[i]的值为 0；如果 i 不等于 0 且 AlfFilterNumMinus1 的值为 FilterNum-1，alf_region_distance[i]的值为 1。

alf_region_distance[i]（i=0～AlfFilterNumMinus1）之和应小于或等于 FilterNum-1。

图像亮度分量样本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 个自适应修正滤波滤波器的第 j 个系数。AlfCoeffLuma[i][j]的值等于 alf_coeff_luma[i][j]的值。

如果 EalfEnableFlag 的值等于 0，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。

如果 EalfEnableFlag 的值等于 1，AlfCoeffLuma[i][j]（j=0～13）的取值范围应为-64～63，且：

T/AI 109.8-2025

——如果 AlfLumaShift[i]的值等于 5，AlfCoeffLuma[i][14]的取值范围应为-1824～1859；

——否则，如果 AlfLumaShift[i]的值等于 6，则 AlfCoeffLuma[i][14]的取值范围应为-1856～1827；

——否则，如果 AlfLumaShift[i]的值等于 7，则 AlfCoeffLuma[i][14]的取值范围应为-1920～1763。

图像亮度分量样本自适应修正滤波器移位 alf_luma_shift_num_minus6[i]

alf_luma_shift_num_minus6[i]的值加 6 表示亮度分量第 i 个自适应滤波器的移位值。AlfLumaShift[i]的值等于 alf_luma_shift_num_minus6[i]的值加 6。

alf_luma_shift_num_minus6[i]的取值范围应为-1～1。

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。

如果 EalfEnableFlag 的值等于 0，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。

如果 EalfEnableFlag 的值等于 1，AlfCoeffChroma[i][j]（i=0～1，j=0～13）的取值范围应为-64～63，且：

——如果 AlfChromaShift[i]的值等于 5，AlfCoeffChroma[i][14]（i=0～1）的取值范围应为-1824～1859；

——如果 AlfChromaShift[i]的值等于 6，AlfCoeffChroma[i][14]（i=0～1）的取值范围应为-1856～1827；

——如果 AlfChromaShift[i]的值等于 7，AlfCoeffChroma[i][14]（i=0～1）的取值范围应为-1920～1763。

图像色度分量样本自适应修正滤波器移位 alf_chroma_shift_num_minus6[i]

alf_chroma_shift_num_minus6[0]的值加 6 表示 Cb 分量自适应滤波器的移位值，AlfChromaShift[0]的值等于 alf_chroma_shift_num_minus6[0]的值加 6。alf_chroma_shift_num_minus6[1]的值加 6 表示 Cr 分量自适应滤波器的移位值，AlfChromaShift[1]的值等于 alf_chroma_shift_num_minus6[1]的值加 6。alf_chroma_shift_num_minus6[0]和 alf_chroma_shift_num_minus6[1]的取值范围应为-1～1。

6.2.15 增强样值偏移补偿参数语义

本条中 compIndex 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。

图像增强样值偏移补偿允许标志 picture_esao_enable_flag[compIndex]

二值变量。值为‘1’表示当前图像的 compIndex 分量可使用增强样值偏移补偿；值为‘0’表示当前图像的 compIndex 分量不应使用增强样值偏移补偿。PictureEsaoEnableFlag[compIndex]的值等于 picture_esao_enable_flag[compIndex]的值。

图像增强样值偏移补偿最大编码单元控制标志 picture_esao_lcu_control_flag[compIndex]

二值变量。值为‘0’表示当前图像的 compIndex 分量的所有最大编码单元均根据 PictureEsaoEnableFlag[compIndex]的值决定是否使用增强样值偏移补偿；值为‘1’表示当前图像的 compIndex 分量的每一个最大编码单元单独决定是否使用增强样值偏移补偿。PictureEsaoLcuControlFlag[compIndex]的值等于 picture_esao_lcu_control_flag[compIndex]的值。如果位流中不存在 picture_esao_lcu_control_flag[compIndex]，

T/AI 109.8-2025

PictureEsaoLcuControlFlag[compIndex]的值位0。

图像增强样值偏移补偿偏移值系数 picture_esao_set_minus1[compIndex]

1位无符号整数。表示当前图像的第compIndex个分量的偏移值组数，取值范围是0～1。PictureEsaoSetNum[compIndex]的值等于picture_esao_set_num_minus1[compIndex]的值加1。如果位流中不存在picture_esao_set_num_minus1[compIndex]，PictureEsaoSetNum[compIndex]的值为1。

图像增强样值偏移补偿自适应参数 picture_esao_adaptive_param_minus1[setIndex]

4位无符号整数。用于确定当前图像的compIndex分量的增强样值偏移补偿自适应参数值。取值范围是0～15。PictureEsaoAdaptiveParam[setIndex]的值等于picture_esao_adaptive_param_minus1[setIndex]的值加1。

增强样值偏移补偿亮度参数类型 esao_luma_param_type[setIndex]

1位无符号整数。确定当前图像亮度分量的增强样值偏移补偿参数类型。EsaoLumaParamType[setIndex]的值等于esao_luma_param_type[setIndex]的值。

图像增强样值偏移补偿自适应参数索引值 picture_esao_adaptive_param_index[compIndex][setIndex]

7位无符号整数。当前图像compIndex分量的增强样值偏移补偿自适应参数索引值。取值范围是0～95。PictureEsaoAdaptiveParamIndex[compIndex][setIndex]的值等于picture_esao_adaptive_param_index[compIndex][setIndex]的值。

增强样值偏移补偿色度子带模式标志 esao_chroma_band_mode_flag[compIndex][setIndex]

1位无符号整数。值为‘1‘表示当前图像的compIndex分量应使用子带模式；值为‘0’表示当前图像的compIndex分量不应使用子带模式。EsaoChromaBandModeFlag[compIndex][setIndex]的值等于esao_chroma_band_mode_flag[compIndex][setIndex]的值。

增强样值偏移补偿色度起始子带 esao_chroma_start_band[compIndex][setIndex]

表示当前图像的compIndex分量的起始子带。解析过程见7.2。EsaoChromaStartBand[compIndex][setIndex]的值等于esao_chroma_start_band[compIndex][setIndex]的值。

增强样值偏移补偿色度子带数 esao_chroma_band_num[compIndex][setIndex]

表示当前图像的compIndex分量的子带个数。解析过程见7.2。EsaoChromaBandNum[compIndex][setIndex]的值等于esao_chroma_band_num[compIndex][setIndex]的值。

增强样值偏移补偿系数 esao_filter_coeff[compIndex][setIndex][i]

表示当前图像的compIndex分量的增强样值偏移补偿系数。解析过程见7.2。取值范围是-15～15。EsaoFilterCoeff[compIndex][setIndex][i]的值等于esao_filter_coeff[compIndex][setIndex][i]的值。如果位流中不存在esao_filter_coeff[compIndex][setIndex][i]，EsaoFilterCoeff[compIndex][setIndex][i]的值为0。

6.2.16 跨分量样值偏移补偿参数语义

本条中compIndex等于1表示Cb分量，等于2表示Cr分量。

图像跨分量样值偏移补偿允许标志 picture_ccsao_enable_flag[compIndex]

二值变量。值为‘1’表示当前图像的compIndex分量可使用跨分量样值偏移补偿；值为‘0’表示当前图像的compIndex分量不应使用跨分量样值偏移补偿。PictureCcsaoEnableFlag[compIndex]的值等于picture_ccsao_enable_flag[compIndex]的值。如果位流中不存在

T/AI 109.8-2025

picture_ccsao_enable_flag[compIndex]，PictureCcsaoEnableFlag[compIndex]的值为 0。

图像跨分量样值偏移补偿最大编码单元控制标志 picture_ccsao_lcu_control_flag[compIndex]

二值变量。值为‘0’表示当前图像的 compIndex 分量的所有最大编码单元均根据 PictureCcsaoEnableFlag[compIndex]的值决定是否使用跨分量样值偏移补偿；值为‘1’表示当前图像的 compIndex 分量的每一个最大编码单元单独决定是否使用跨分量样值偏移补偿。PictureCcsaoLcuControlFlag[compIndex]的值等于 picture_ccsao_lcu_control_flag[compIndex]的值。如果位流中不存在 picture_ccsao_lcu_control_flag[compIndex]，PictureCcsaoLcuControlFlag[compIndex]的值为 0。

图像跨分量样值偏移补偿偏移值组数 picture_ccsao_set_num_minus1[compIndex]

2 位无符号整数。表示当前图像的第 compIndex 个分量的偏移值组数，取值范围是 0～3。PictureCcsaoSetNum[compIndex]的值等于 picture_ccsao_set_num_minus1[compIndex]的值加 1。如果位流中不存在 picture_ccsao_set_num_minus1[compIndex]，PictureCcsaoSetNum[compIndex]的值为 1。

图像跨分量样值偏移补偿候选索引 picture_ccsao_cand_index[compIndex][setIndex]

4 位无符号整数。用于确定当前图像跨分量样值偏移补偿所使用的亮度样本的位置，取值范围是 0～8。PictureCcsaoCandIndex[compIndex][setIndex]的值等于 picture_ccsao_cand_index[compIndex][setIndex]的值。

图像跨分量样值偏移补偿亮度条带个数 picture_ccsao_luma_band_num_minus1[compIndex][setIndex]

4 位无符号整数。当前图像的 compIndex 分量选择的补偿亮度条带个数，取值范围是 0～15。PictureCcsaoLumaBandNum[compIndex][setIndex]的值等于 picture_ccsao_luma_band_num_minus1[compIndex][setIndex]的值加 1。

图像跨分量样值偏移补偿色度条带个数 picture_ccsao_chroma_band_num_minus1[compIndex][setIndex]

1 位无符号整数。当前图像的 compIndex 分量选择的补偿色度条带个数，取值范围是 0～1。PictureCcsaoChromaBandNum[compIndex][setIndex]的值等于 picture_ccsao_chroma_band_num_minus1[compIndex][setIndex]的值加 1。PictureCcsaoClassNum[compIndex][setIndex]的值等于 PictureCcsaoLumaBandNum[compIndex][setIndex]的值乘 PictureCcsaoChromaBandNum[compIndex][setIndex]的值。

跨分量样值偏移补偿偏移值 ccsao_offset[compIndex][setIndex][i]

表示当前图像的 compIndex 分量的跨分量样值偏移补偿偏移值，取值范围是-15～15。解析过程见 8.2。CcsaoOffset[compIndex][setIndex][i]的值等于 ccsao_offset[compIndex][setIndex][i]的值。

图像跨分量样值偏移补偿模式 picture_ccsao_mode[compIndex][setIndex]

1 位无符号整数。表示当前图像的 compIndex 分量的跨分量样值偏移补偿模式。值为 0 表示当前图像的跨分量样值偏移补偿模式为区间模式；值为 1 表示当前图像的跨分量样值偏移补偿模式为边缘模式。PictureCcsaoMode[compIndex][setIndex]的值等于 picture_ccsao_mode[compIndex][setIndex]的值。

图像跨分量样值偏移补偿边缘模式类型 picture_ccsao_edge_type[compIndex][setIndex]

2 位无符号整数。表示当前图像的 compIndex 分量的跨分量样值偏移补偿边缘模式的类型，取值范围是 0～3。PictureCcsaoEdgeType[compIndex][setIndex]的值等于

T/AI 109.8-2025

picture_ccsao_edge_type[compIndex][setIndex]的值。

图像跨分量样值偏移补偿边缘模式条带索引

picture_ccsao_edge_band_num_index[compIndex][setIndex]

2 位无符号整数。表示当前图像的 compIndex 分量的跨分量样值偏移补偿边缘模式选择的补偿像素类型以及条带个数，取值范围是 0～3。PictureCcsaoEdgeBandNumIndex[compIndex][setIndex]的值等于 picture_ccsao_edge_band_num_index[compIndex][setIndex]的值。

按以下方式导出 PictureCcsaoClassNum[compIndex][setIndex]：

```
if (PictureCcsaoEdgeBandNumIndex[compIndex][setIndex] == 0 ||
    PictureCcsaoEdgeBandNumIndex[compIndex][setIndex] == 1) {
    PictureCcsaoEdgeBandNum[compIndex][setIndex] = PictureCcsaoEdgeBandNumIndex[compIndex][setIndex] + 1
}
else {
    PictureCcsaoEdgeBandNum[compIndex][setIndex] = PictureCcsaoEdgeBandNumIndex[compIndex][setIndex] - 1
}
PictureCcsaoClassNum[compIndex][setIndex] = PictureCcsaoEdgeBandNum[compIndex][setIndex] * 16
```

图像跨分量样值偏移补偿边缘模式阈值索引

picture_ccsao_edge_thr_index[compIndex][setIndex]

3 位无符号整数。表示当前图像的 compIndex 分量的跨分量样值偏移补偿边缘模式的阈值索引，取值范围是 0～7。PictureCcsaoEdgeThr[compIndex][setIndex]的值等于(picture_ccsao_edge_thr_index[compIndex][setIndex]+1)*2。

6.2.17 串复制帧内预测语义

串复制帧内预测的匹配类型 isc_match_type[i]

解析过程见 7.3。值为‘1’表示当前编码单元的第 i 部分是一个匹配串；值为‘0’表示第 i 部分是一个不完全匹配串。IscMatchType[i]等于 isc_match_type[i]的值。如果位流中不存在 isc_match_type[i]，IscMatchType[i]的值为 0。

剩余样本数量 isc_next_remaining_pixel_in_cu[i]

完成当前编码单元第 i 部分的解码之后当前编码单元中剩余的尚未完成解码的像素的数目。解析过程见 7.3。IscNextRemainingPixelInCu[i]的值等于 isc_next_remaining_pixel_in_cu[i]的值。

样本匹配类型标志 isc_pixel_match_type[i][j]

解析过程见 7.3。值为‘1’表示当前编码单元的第 i 部分的第 j 个像素是一个匹配样本；值为‘0’表示是一个未匹配样本。IscPixelMatchType[i][j]等于 isc_pixel_match_type[i][j]的值。如果位流中不存在 isc_pixel_match_type[i][j]，IscPixelMatchType[i][j]的值为 1。

串复制帧内预测未匹配样本亮度分量值 isc_unmatched_pixel_y[i][j]

串复制帧内预测未匹配样本色度 Cb 分量值 isc_unmatched_pixel_cb[i][j]

串复制帧内预测未匹配样本色度 Cr 分量值 isc_unmatched_pixel_cr[i][j]

当前编码单元第 i 部分的第 j 个未匹配像素的亮度分量、色度 Cb 分量和色度 Cr 分量的值。解析过程见 7.3。IscUnmatchedPixelY[i][j]、IscUnmatchedPixelCb[i][j]和 IscUnmatchedPixelCr[i][j]分别等于 isc_unmatched_pixel_y[i][j]、isc_unmatched_pixel_cb[i][j]和 isc_unmatched_pixel_cr[i][j]的值。

串矢量上方标志 isc_sv_above_flag[i]

解析过程见 7.3。值为‘1’表示当前编码单元第 i 部分的串矢量是(0,-1)；值为‘0’表示串矢量

T/AI 109.8-2025

不是(0,-1)。IscSvAboveFlag[i]等于 isc_sv_above_flag[i]的值。如果 isc_sv_above_flag[i]的值等于 0，则 isc_sv_y_abs 的值不应等于 1 且 isc_sv_y_sign 的值不应等于 1 且 isc_sv_x_non_zero_flag 的值不应等于 0。

串矢量历史标志 isc_sv_recent_flag[i]

解析过程见 7.3。值为‘1’表示应从历史帧内复制信息表中导出当前编码单元第 i 部分的串矢量；值为‘0’表示不应从历史帧内复制信息表中导出串矢量。IscSvRecentFlag[i]等于 isc_sv_recent_flag[i]的值。如果位流中不存在 isc_sv_recent_flag[i]，IscSvRecentFlag[i]的值为 0。

如果 IscSvAboveFlag[i]和 IscSvRecentFlag[i]的值均为 0，当前编码单元第 i 部分的串矢量的 X 分量的值由 IscSvXNonZeroFlag[i]、IscSvXSign[i]和 IscSvXAbs[i]确定，Y 分量的值由 IscSvYSign[i]和 IscSvYAbs[i]确定。

串矢量历史索引 isc_sv_recent_index[i]

当前编码单元第 i 部分的串矢量在历史帧内复制信息表中的索引值。解析过程见 7.3。解码过程见 8.6.6.5.2。IscSvRecentIndex[i]的值等于 isc_sv_recent_index[i]的值。

串矢量 Y 分量绝对值 isc_sv_y_abs[i]

当前编码单元第 i 部分的串矢量的 Y 分量的绝对值。解析过程见 7.3。IscSvYAbs[i]等于 isc_sv_y_abs[i]的值。

串矢量 Y 分量符号位 isc_sv_y_sign[i]

当前编码单元第 i 部分的串矢量的 Y 分量的符号位。值为‘1’表示 Y 分量的值为负数；值为‘0’表示 Y 分量的值为正数。解析过程见 7.3。IscSvYSign[i]等于 isc_sv_y_sign[i]的值。如果位流中不存在 isc_sv_y_sign[i]，则 IscSvYSign[i]的值为 1。

串矢量 X 分量符号位 isc_sv_x_sign[i]

当前编码单元第 i 部分的串矢量的 X 分量的符号位。值为‘1’表示 X 分量的值为负数；值为‘0’表示 X 分量的值为正数。解析过程见 7.3。IscSvXSign[i]等于 isc_sv_x_sign[i]的值。如果位流中不存在 isc_sv_x_sign[i]，IscSvXSign[i]的值为 1。

串矢量 X 分量绝对值 isc_sv_x_abs_minus1[i]

当前编码单元第 i 部分的串矢量的 X 分量的绝对值减 1。解析过程见 7.3。IscSvXAbs[i]等于 isc_sv_x_abs_minus1[i]的值加 1。如果位流中不存在 isc_sv_x_abs_minus1[i]，IscSvXAbs[i]等于 0。

串矢量 X 分量标志 isc_sv_x_non_zero_flag[i]

解析过程见 7.3。值为‘1’表示当前编码单元第 i 部分的串矢量的 X 分量的值不等于 0；值为‘0’表示 X 分量的值等于 0。IscSvXNonZeroFlag[i]等于 isc_sv_x_non_zero_flag [i]的值。如果位流中不存在 isc_sv_x_non_zero_flag[i]，IscSvXNonZeroFlag[i]的值为 1。

串复制帧内预测未匹配样本串存在标志 isc_ups_present_flag

二值变量。解析过程见 7.3。值为‘1’表示当前编码单元中存在未匹配样本串；值为‘0’表示当前编码单元中不存在未匹配样本串。IscUpsPresentFlag 的值等于 isc_ups_present_flag 的值。如果位流中不存在 isc_ups_present_flag，IscUpsPresentFlag 的值为 0。

新增点矢量数 isc_num_of_new_pv

当前编码单元中新增点矢量的个数。解析过程见 7.3。IscNumOfNewPv 的值等于 isc_num_of_new_pv 的值。如果位流中不存在 isc_num_of_new_pv，IscNumOfNewPv 的值为 0。

复用点矢量数 isc_num_of_reused_pv

表示当前编码单元中复用历史点预测历史运动信息表 PrevPpInfoList 中的点矢量数。解析过程见 7.3。IscNumOfReusedPv 的值等于 isc_num_of_reused_pv 的值。如果位流中不存在

T/AI 109.8-2025

isc_num_of_reused_pv，IscNumOfReusedPv 的值为 0。

历史点预测信息表中连续排列的不复用点矢量数 isc_prev_pv_not_reused_run

表示在从历史点预测信息表 PrevPpInfoList 中选择复用的点矢量时，连续排列的不复用的点矢量数。解析过程见 7.3。IscPrevPvNotReusedRun 的值等于 isc_prev_pv_not_reused_run 的值。如果位流中不存在 isc_prev_pv_not_reused_run，IscPrevPvNotReusedRun 的值为 0。

未匹配样本串长度最大值 isc_ups_max_length_minus1

isc_ups_max_length_minus1 的值加 1 表示当前编码单元中所有未匹配样本串长度的最大值。解析过程见 7.3。IscUpsMaxLengthMinus1 的值等于 isc_ups_max_length_minus1 的值。如果位流中不存在 isc_ups_max_length_minus1，IscUpsMaxLengthMinus1 的值为 0。

等值串或未匹配样本串数 isc_evs_ups_num_minus1

用于确定连续的等值串或未匹配样本串的个数。解析过程见 7.3。IscEvsUPSNumMinus1 的值等于 isc_evs_ups_num_minus1 的值。如果位流中不存在 isc_evs_ups_num_minus1，IscEvsUPSNumMinus1 的值为 0。

串长度第一部分 str_length_minus1_prefix

串长度的第一部分。解析过程见 7.3。StrLengthMinus1Prefix 等于 str_length_minus1_prefix 的值。如果位流中不存在 str_length_minus1_prefix，StrLengthMinus1Prefix 的值为 0。

串长度第二部分 str_length_minus1_infix

串长度的第二部分。解析过程见 7.3。StrLengthMinus1Infix 等于 str_length_minus1_infix 的值。如果位流中不存在 str_length_minus1_infix，StrLengthMinus1Infix 的值为 0。

串长度第三部分 str_length_minus1_suffix

串长度的第三部分。解析过程见 7.3。StrLengthMinus1Suffix 等于 str_length_minus1_suffix 的值。如果位流中不存在 str_length_minus1_suffix，StrLengthMinus1Suffix 的值为 0。

点矢量在点预测信息表中索引值的第一部分 pv_address_prefix

点矢量索引值的第一部分，解析过程见 7.3。PvAdressPrefix 等于 pv_address_prefix 的值。如果位流中不存在 pv_address_prefix，PvAdressPrefix 的值为 0。

点矢量在点预测信息表中索引值的第二部分 pv_address_infix

点矢量索引值的第二部分，解析过程见 7.3。PvAdressInfix 等于 pv_address_infix 的值。如果位流中不存在 pv_address_infix，PvAdressInfix 的值为 0。

点矢量在点预测信息表中索引值的第三部分 pv_address_suffix

点矢量索引值的第三部分，解析过程见 7.3。PvAdressSuffix 等于 pv_address_suffix 的值。如果位流中不存在 pv_address_suffix，PvAdressSuffix 的值为 0。

新增点矢量对应常现位置像素亮度分量值 isc_fopixel_y

新增点矢量对应常现位置像素 Cb 分量值 isc_fopixel_cb

新增点矢量对应常现位置像素 Cr 分量值 isc_fopixel_cr

新增点矢量对应的常现位置上像素的亮度分量、色度 Cb 分量和色度 Cr 分量的值。解析过程见 7.3。

串复制帧内预测未匹配样本亮度分量值 isc_nos_up_y[i]

串复制帧内预测未匹配样本色度 Cb 分量值 isc_nos_up_cb[i]

串复制帧内预测未匹配样本色度 Cr 分量值 isc_nos_up_cr[i]

非普通串子模式中的当前编码单元第 i 部分的未匹配像素的亮度分量、色度 Cb 分量和色度 Cr 分量的值。解析过程见 7.3。IscNosUpY[i]、IscNosUpCb[i]和 IscNosUpCr[i]的值分别等于 isc_nos_up_y[i]、isc_nos_up_cb[i]和 isc_nos_up_cr[i]的值。

T/AI 109.8-2025

6.2.18 扩展数据 RBSP 语义

6.2.18.1 扩展数据 RBSP 语义通则

extension_id

8 位无符号整数，用于区分不同的扩展数据类型。有效的 extension_id 不应等于 0x80。如果 profile_id 的值为 0x30,位流中不应存在 extension_id 为 0xE3 的扩展数据。

extension_length

表示 extension_length 之后的 extension_data 的长度，以字节为单位。如果 extension_id 小于或等于 192，extension_length 长度为 1 字节；如果 extension_id 大于 192 且小于或等于 224，extension_length 长度为 2 字节；如果 extension_id 大于 224，extension_length 长度为 4 字节。

扩展数据单元中的信息在图像解码过程中不是必要的。

surveillance_extension_stop_byte 应等于 0x80。

6.2.18.2 绝对时间信息扩展语义

extension_id

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

extension_length

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

hour_bits

表示小时信息。hour_bits 取值应为 0～23。

minute_bits

表示分钟信息。minute_bits 取值应为 0～59。

second_bits

表示秒信息。second_bits 取值应为 0～59。

second_fraction_bits

表示秒的分数信息，以 1/16384 秒为单位。second_fraction_bits 取值应为 0～16383。

ref_date_flag

二值变量。值为‘1’表示绝对时间信息扩展中包含绝对日期参考信息；值为‘0’表示绝对时间信息扩展中不包含绝对日期参考信息。

year_minus2000_bits

取值应为 0～127，加 2000 表示年份信息 Year。计算如下：

Year = year_minus2000_bits + 2000

month_bits

表示月份信息。month_bits 取值应为 1～12。

day_bits

表示日期信息。date_bits 取值应为 1～31。

tz_bits

表示基于 UTC 的时区编号，取值范围[-12,12]，其中 0 标识中时区，[-12,-1]依次标识西 12 至西 1 时区，[1,12]则依次标识东 1 至东 12 时区。

6.2.18.3 地理信息扩展语义

extension_id

T/AI 109.8-2025

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

longitude_type

1 位无符号整数。0 表示东经；1 表示西经。

longitude_degree

8 位无符号整数，表示经度的度数，取值应为 0～90。

longitude_fraction_bits

表示度的分数信息，以 1/1048576 度为单位，取值应为 0～1048575。

latitude_type

1 位无符号整数。0 表示北纬；1 表示南纬。

latitude_degree

8 位无符号整数，表示纬度的度数，取值应为 0～90。

latitude_fraction_bits

表示度的分数信息，以 1/1048576 度为单位，取值应为 0～1048575。

height

15 位有符号整数，表示高度，单位为米。

speed

8 位有符号整数，表示速度，单位为米/秒。

yaw_degree

表示方向角度数，0 表示方向为正北，角度数沿顺时针方向递增。

6.2.18.4 OSD 信息扩展语义

OSD 信息扩展语义应符合 GB/T 25724-2017 中 5.2.4.7.4 的规定。

6.2.18.5 视频结构化信息扩展语义

extension_id

8 位无符号整数，extension_id 应等于 0xE1。

extension_length

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

subtype

8 位无符号整数，为后续扩展预留，取值应为 0。

analysis_num

视频结构化信息条目数量，包括图像分析规则和图像分析结果的数量总和。

analysis_id[i]

第 i 项图像分析规则或图像分析结果的分析功能标识，其定义见表 73。

表73 图像分析结果功能标识表

analysis_id	定义
0x01	图像分析规则
0x02	人脸属性分析
0x03	人员属性分析
0x04	机动车特征分析
0x05	非机动车特征分析

T/AI 109.8-2025

表 73 图像分析结果功能标识表（续）

analysis_id	定义
0x06	任意物体特征分析
0x07	实时分析结果
0x08	周界报警目标信息
0x09	检测结果框信息
0x0A	伴随结果框的文字信息
0x0B	伴随设备序列号和设备唯一标识码信息
0x0C	经加密的伴随设备序列号和设备唯一标识码信息
0x0D	智能分割信息
0x0E-0x3F	保留
0x40-0xFF	自定义

data_length[i]

analysis_data[i]的长度，以字节为单位。

analysis_data[i][j]

第 i 项图像分析规则或图像分析结果的第 j 个字节。

其中，analysis_id 为 0x01 至 0x0C 对应的 analysis_data 语义应符合 T/BAX 0007-2025 中 5.3.5.2 至 5.2.5.13 描述。analysis_id 为 0x0D 表示智能分割信息，其对应的 analysis_data 语法及语义见附录 B。

6.2.18.6 特征张量图信息扩展语义

特征张量图数据编码采用本文件规范的视频编码器，等价于小分辨率的灰度图复用本文件规范的视频编码器进行编码，且其采用视频编码器的档次和级别不高于视频编码器的档次和级别。特征张量图编码模式示例见附录 K。

extension_id

8 位无符号整数，extension_id 应等于 0xE3。

extension_length

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

ftm_subtype_index

5 位无符号整数，表示第 ftm_subtype_index 路特征张量图。存在多路特征张量图时，同一路特征张量图位流应该使用相同的 ftm_subtype_index。

ftm_ra_flag

1 位无符号整数，表示当前特征张量图在特征位流中是否为随机访问点。1 表示当前特征张量图是随机访问点，0 表示当前特征张量图不是随机访问点。

ftm_type

8 位无符号整数，表示当前特征张量图支持的智能分析算法类别。0x01 表示支持泛化智能分析算法。智能分析算法标识定义见表 74。

T/AI 109.8-2025

表74 智能分析算法标识表

ftm_type	定义
0x01	泛化智能分析算法
0x02	目标跟踪
0x03	目标检测
0x04	实例分割
0x05	姿态估计
其它值	保留

ftm_width

16 位无符号整数，表示特征张量图的宽度。

ftm_height

16 位无符号整数，表示特征张量图的高度。

ftm_depth

4 位无符号整数，表示特征张量图像素值的位深。ftm_depth 为 10 表示位深为 10bit。

ftm_ipack_mode

4 位无符号整数，表示在特征重排阶段所用的重排算法。重排算法标识定义见表 75。

表75 重排算法标识表

ftm_ipack_mode	定义
0x01	水平扫描
0x02	ZigZag 扫描
0x03	垂直扫描
0x04	对角线扫描
0x05	拉普拉斯扫描
其它值	保留

ftm_iquant_mode

4 位无符号整数，表示在特征反量化阶段的反量化算法。ftm_iquant_mode 为 10 表示进行 10 位线性反量化；ftm_iquant_mode 为 8 表示进行 8 位线性反量化。

ft_num

4 位无符号整数，表示重建后的三维特征张量的数量。例如，特征张量图重建得到 3 个三维特征张量，分别是 P1（C1，H1，W1），P2（C2，H2，W2），P3（C3，H3，W3）。

ft_channel[i]

16 位无符号整数，表示通过特征张量图重建的第 i 个三维特征张量的通道数量。

ft_height[i]

16 位无符号整数，表示通过特征张量图重建的第 i 个三维特征张量的通道高度。

ft_width[i]

16 位无符号整数，表示通过特征张量图重建的第 i 个三维特征张量的通道宽度。

ftm_lag_picture_order_index

6 位无符号整数，表示与视频帧相比，特征张量图滞后的帧数。

T/AI 109.8-2025

ftm_iquant_max

32 位无符号整数，表示特征张量图量化的最大像素值。

ftm_iquant_min

32 位无符号整数，表示特征张量图量化的最小像素值。

ftm_data_length

32 位无符号整数，表示 ftm_data 的长度，以字节为单位。

ftm_data

n 位无符号整数，n = ftm_data_length*8。ftm_data 为特征张量图的特征位流，可以提取出来，并用本文件规范的视频解码器进行解码。

ftm_data 应与 ftm_ra_flag 保持一致，当 ftm_ra_flag 为 1 时，ftm_data 应为随机访问点。

6.2.19 认证数据 RBSP 语义

认证数据位置标识 for_current_ras_idc

二值变量。值为‘1’表示该认证数据中摘要列表中所有摘要对应的访问单元都在当前随机访问片段内。如果为‘0’，指示该认证数据中摘要列表中所有摘要对应的访问单元都不在当前随机访问片段内。共同签名的所有访问单元应位于同一个随机访问片段内。

知识图像认证数据标志位 auth_is_library_flag

二值变量。值为‘1’表示该认证数据为知识图像的签名数据；值为‘0’表示该认证数据为显示图像的签名数据。AuthIsLibraryFlag 的值等于 auth_is_library_flag 的值。

认证知识图像索引 authenticaion_library_picture_index

n 位无符号整数。表示当前认证数据所作用的知识位流对应知识图像的索引，解析过程见 7.2。取值范围是 0～511。如果 authenticaion_library_picture_index 在位流中不存在，默认其值等于 0。

认证数据标识 authentication_data_id

1 位无符号整数。取值范围 0～1，该认证数据的标识。

authentication_data_id 取值为 0 或 1 的认证数据 NAL 单元应在编码视频序列中交替出现，即一个编码视频序列中依解码次序相邻的两个认证数据 NAL 单元的 authentication_data_id 取值不应相同。

认证摘要列表标识 authentication_hash_list_flag

二值变量。值为‘1’表示该认证数据携带生成该认证数据中签名的访问单元的摘要列表。值为‘0’表示该认证数据不携带生成该认证数据中签名的访问单元的摘要列表。

认证摘要数量 authentication_hash_number_minus1

8 位无符号整数，取值范围 0～255。authentication_hash_number_minus1 加 1 表示认证摘要数据的数量。

认证摘要数据 authentication_hash

二进制数据。按照对应安全参数集中 hash_type 对应的摘要算法对访问单元做摘要得到的摘要数据，长度为表 65 中所列的摘要算法 hash_type 对应的摘要数据长度 hash_size。认证数据 NAL 单元中携带的摘要列表中的摘要的排列先后次序应与其中各个层单元的位流次序相同。

签名数据的长度 authentication_data_length_minus1

8 位无符号整数。加 1 表示签名数据的长度，以字节为单位，取值应为 0～255。

签名数据的字节数 authentication_data[i]

8 位无符号整数。一个签名数据的第 i 个字节。

T/AI 109.8-2025

认证数据NAL单元应位于访问单元中除流结尾NAL单元、编码视频序列结尾NAL单元之外的所有其它类型的NAL单元之后。对应同一安全参数集的认证数据NAL单元在位流中的次序应与其对应的访问单元的位流次序相同，即第一认证数据NAL单元位于第二认证数据NAL单元前，则与第一认证数据NAL单元关联的任一个访问单元位于与第二认证数据NAL单元关联的任一个访问单元之前。

显示图像的认证数据NAL单元应位于此次认证的最后一个访问单元中或之后、下一个认证数据NAL单元之前及下下一个不是显示知识图像的随机访问点访问单元之前。每个RAS中的最后一个认证数据允许放入下一个RAS中，此情况下，设置认证数据中的for_current_ras_idc为0。认证数据和最后一个参与认证的访问单元中的层单元间隔不能超过HashPeriodInDoi（等于此次认证对应的安全参数集中的hash_period_in_doi_minus1+1）个访问单元（不含非显示知识图像访问单元）。显示图像的认证数据可位于最后一个显示图像编码片所在的访问单元中；也可位于下一个RAS的显示图像编码片的访问单元中。

知识图像的认证数据NAL单元应位于此次认证的最后一个访问单元中或之后、下一个认证数据NAL单元之前及下下一个随机访问点访问单元之前。

显示知识图像的认证数据和显示知识图像访问单元间隔应不超过HashPeriodInDoi（等于此次认证对应的安全参数集中的hash_period_in_doi_minus1+1）个访问单元。显示知识图像的认证数据可位于最后一个显示知识图像编码片所在的访问单元中；也可位于下一个RAS的显示图像编码片的访问单元中。

非显示知识图像的认证数据位于最后一个非显示知识图像编码片所在的访问单元中。

6.2.20 访问单元分界 RBSP 语义

访问单元分界 RBSP 表示两个访问单元间的分界。访问单元分界 RBSP 的内容为空。

6.2.21 编码视频序列结尾 RBSP 语义

编码视频序列结尾 RBSP 表示一个编码视频序列的结束。编码视频序列结尾 RBSP 的内容为空。

6.2.22 流结尾 RBSP 语义

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

6.2.23 RBSP 尾位语义

RBSP 停止位 rbsp_stop_one_bit

位串‘1’。

RBSP 对齐补零位 rbsp_alignment_zero_bit

位串‘0’。

7 视频解析过程

7.1 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++)
```

T/AI 109.8-2025

```
    b = read_bits(1)
CodeNum = 2^(leadingZeroBits+k) − 2^k + read_bits(leadingZeroBits + k)
```

表 76 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76 k 阶指数哥伦布码表

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

T/AI 109.8-2025

7.2 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：

——ue(v)：语法元素的值等于 CodeNum；

——se(v)：根据表 77 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77 se(v)与 CodeNum 的映射关系

CodeNum值	语法元素值
0	0
1	1

表 77 se(v)与 CodeNum 的映射关系（续）

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

7.3 ae(v)的解析过程

7.3.1 概述

对片进行解析前，如果 PicturePrivacyEnableFlag 不等于 0 且 UserPermission 不等于 0，熵编码解码器个数为 2，否则熵编码解码器个数为 1，首先初始化所有二元符号模型和所有熵编码解码器，见 7.3.2。然后从位流中依次解析得到二元符号串，见 7.3.3。最后根据二元符号串得到 ae(v)描述的语法元素的值，见 7.3.4。

如果 IsPrivacyFlag 的值为 0，使用第 1 套熵编码解码器，否则使用第 2 套熵编码解码器。对每个熵编码解码器遵循以下初始化方法：

如果存在以下任意一种情况，则初始化熵编码解码器，见 7.3.2.2

——当前编码单元只包含亮度分量且 IntraLumaPredMode 等于‘Intra_Luma_PCM’，或当前编码单元包含亮度分量和色度分量且 IntraLumaPredMode 等于‘Intra_Luma_PCM’且 IntraChromaPredMode 不等于‘Intra_Chroma_PCM’，在解析完亮度编码块的 pcm_coeff 后，初始化熵编码解码器；

——当前编码单元只包含色度分量且 IntraChromaPredMode 等于‘Intra_Chroma_PCM’，或当前编码单元包含亮度分量和色度分量且 IntraLumaPredMode 等于‘Intra_Luma_PCM’且 IntraChromaPredMode 等于‘Intra_Chroma_PCM’，在解析完色度编码块的 pcm_coeff 后，初始化熵编码解码器。

7.3.2 初始化

7.3.2.1 初始化二元符号模型

解码器应初始化 ctxArray 中的每个二元符号模型。

如果 MaecEnableFlag 的值为 0，则一个二元符号模型包括变量 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。

否则，一个二元符号模型包括变量 mps、cycno、counterThr1、counterThr2、lgPmps0、lgPmps1、baseWin、diffWin0 和 diffWin1。mps 的位宽为 1 位，cycno 的位宽为 5 位，lgPmps0 和 lgPms1 的位宽为 10 位。mps 和 cycno 的值应初始化为 0；lgPmps0 和 lgPms1 的值应初始化为 1023。

二元符号模型中，diffWin0 和 diffWin1 的值均应为 1，baseWin 的值应为 5，counterThr1 和 counterThr2 的值由二元符号所在当前图像的图像类型得到：

——如果当前图像是 I 图像，counterThr1 的值为 0，counterThr2 的值为 8；

T/AI 109.8-2025

——如果当前图像是 P 图像或者 B 图像，counterThr1 的值为 3，counterThr2 的值为 16。

7.3.2.2 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
```

7.3.3 二元符号串解析

7.3.3.1 概述

解析二元符号串的步骤如下：

a) 设二元符号的索引号 binIndex 的值为-1，二元符号串为空；

b) binIndex 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 binIndex 不为 0 的二元符号；
- sao_interval_offset_abs 中 binIndex 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；
- ccsao_lcu_set_index 的二元符号；
- mv_diff_x_sign_l0 、 mv_diff_y_sign_l0 、 mv_diff_x_sign_l0_affine 、 mv_diff_y_sign_l0_affine 、 mv_diff_x_sign_l1 、 mv_diff_y_sign_l1 、 mv_diff_x_sign_l1_affine、mv_diff_y_sign_l1_affine、mv_diff_x_sign_bv 或 mv_diff_y_sign_bv 的二元符号；
- mv_diff_x_abs_l0 、 mv_diff_y_abs_l0 、 mv_diff_x_abs_l0_affine 、 mv_diff_y_abs_l0_affine 、 mv_diff_x_abs_l1 、 mv_diff_y_abs_l1 、 mv_diff_x_abs_l1_affine 或 mv_diff_y_abs_l1_affine 中 binIndex 大于或等于 3 的二元符号；
- mv_diff_x_abs_bv 或 mv_diff_y_abs_bv 或 isc_sv_y_abs[i]中 binIndex 等于 6 且前一位二元符号等于 0 的二元符号；
- mv_diff_x_abs_bv 或 mv_diff_y_abs_bv 或 isc_sv_y_abs[i]中 binIndex 大于或等

T/AI 109.8-2025

于 7 的二元符号；

- coeff_sign 的二元符号；
- coeff_run 中 binIndex 大于或等于 16 的二元符号；
- umve_mv_index 中 binIndex 不为 0 的二元符号；
- umve_step_index 中 binIndex 不为 0 的二元符号；
- sawp_index 的二元符号；
- sawp_pred_mode0_index 中 binIndex 不为 0 的二元符号；
- sawp_pred_mode1_index 中 binIndex 不为 0 的二元符号；
- awp_index 的二元符号；
- awp_mvr_cand_step0 中 binIndex 不为 0 的二元符号；
- awp_mvr_cand_step1 中 binIndex 不为 0 的二元符号；
- awp_cand_index0 中 binIndex 不为 0 的二元符号；
- awp_cand_index1 中 binIndex 不为 0 的二元符号；
- chroma_eipm_index 中 binIndex 不为 0 的二元符号；
- affine_umve_step_index0 中 binIndex 不为 0 的二元符号；
- affine_umve_step_index1 中 binIndex 不为 0 的二元符号；
- coeff_level_minus1 中 binIndex 大于或等于 8 的二元符号；
- scan_region_x 中第二部分的二元符号（第一、二部分的规定见 7.3.4.20）；
- scan_region_y 中第二部分的二元符号（第一、二部分的规定见 7.3.4.20）；
- coeff_abs_level_remaining 中的二元符号；
- isc_next_remaining_pixel_in_cu[i]的第二和第三部分的二元符号（第一、二、三部分的规定见 7.3.4.24）；
- cu_qp_delta_sign 的二元符号；
- esao_lcu_set_index 的二元符号；
- isc_sv_recent_index[i]中 binIndex 大于或等于 3 的二元符号；
- isc_sv_x_sign[i]或 isc_sv_y_sign[i]的二元符号；
- isc_sv_x_abs_minus1[i]的二元符号；
- isc_unmatched_pixel_y[i][j] 、 isc_unmatched_pixel_cb[i][j] 或 isc_unmatched_pixel_cr[i][j]的二元符号；
- isc_nos_up_y[i]、isc_nos_up_cb[i]或 isc_nos_up_cr[i]的二元符号；
- isc_num_of_reused_pv 的二元符号；
- isc_num_of_new_pv 的二元符号；
- isc_prev_pv_not_reused_run 的二元符号；
- str_length_minus1_prefix 中 binIndex 大于或等于 5 的二元符号；
- str_length_minus1_infix 或 str_length_minus1_suffix 的二元符号；
- pv_address_prefix、pv_address_infix 或 pv_address_suffix 的二元符号；
- isc_fopixel_y、isc_fopixel_cb、isc_fopixel_cr 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；
- aec_ipcm_stuffing_bit 的二元符号。

T/AI 109.8-2025

3) 否则，置 BypassFlag 和 StuffingBitFlag 的值为 0，根据 binIndex 得到每个二元符号对应的唯一的 ctxIndex，并根据 ctxIndex 导出二元符号模型 ctx（见 7.3.3.2）。

c) 如果当前二元符号是 coeff_last，置 CtxWeight 的值为 1；否则，置 CtxWeight 的值为 0；

d) 解析当前二元符号（见 7.3.3.3）；

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 7.3.4 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。

7.3.3.2 导出二元符号模型

7.3.3.2.1 导出二元符号模型

如果 ctxIndexIncW 不存在，二元符号模型 ctx 等于 ctxArray[ctxIndex]，其中 ctxArray 是保存二元符号模型的数组，ctxIndex 是数组的索引值；否则，二元符号模型 ctx 和 ctxW 分别等于 ctxArray[ctxIndex]和 ctxArray[ctxIndexW]，其中 ctxArray 是保存二元符号模型的数组，ctxIndex 和 ctxIndexW 是数组的索引值。IsPrivacyFlag 表示当前语法元素解析时使用的熵编码解码器索引减 1，语法元素的每个二元符号的 ctxIndex 等于 ctxIndexInc 加 ctxIndexStart，ctxIndexW 等于 ctxIndexIncW 加 ctxIndexStart。各语法元素对应的 ctxIndexStart 及每个二元符号对应的 ctxIndexInc 见表 78，ctxIndexIncW 见 7.3.3.2.27。

表78 语法元素对应的 ctxIndexStart 和 ctxIndexInc

IsPrivacyFlag	语法元素	ctxIndexInc	ctxIndexStart	ctx数量
0	lcu_qp_delta	见7.3.3.2.2	0	4
0	sao_merge_type_index	binIndex+SaoMergeLeftAvai+SaoMergeUpAvai-1	4	3
0	sao_mode	0	7	1
0	sao_interval_offset_abs	0	8	1
0	alf_lcu_enable_flag	0	9	1
0	qt_split_flag	见7.3.3.2.3	10	4
0	bet_split_flag	见7.3.3.2.4	14	9
0	bet_split_type_flag	见7.3.3.2.5	23	3
0	bet_split_dir_flag	见7.3.3.2.6	26	5
0	root_cu_constraint	0	31	1
0	privacy_level	0	32	1
0	intra_chroma_pred_mode_index	见7.3.3.2.7	33	3
0	chroma_eipm_index	0	36	1
0	chroma_pmc_index	0	37	1
0	chroma_pmc_param_index	0	38	1
0	ctp_u	0	39	1
0	ctp_v	0	40	1
0	chroma_st_flag	0	41	1

T/AI 109.8-2025

表 78 语法元素对应的 ctxIndexStart 和 ctxIndexInc（续）

IsPrivacyFlag	语法元素	ctxIndexInc	ctxIndexStart	ctx数量
0	ts_cu_flag	0	42	1
0	skip_flag	见7.3.3.2.8	43	4
0	advanced_pred_flag	0	47	1
0	affine_flag	0	48	1
0	affine_umve_flag	0	49	1
0	direct_flag	见7.3.3.2.9	50	2
0	inter_pf_flag	0	52	1
0	inter_pf_index	0	53	1
0	inter_pc_flag	0	54	1
0	inter_pc_index	binIndex	55	2
0	intra_cu_flag	见7.3.3.2.10	57	6
0	sawp_flag	0	63	1
0	dt_split_flag	0	64	1
0	dt_split_dir	0	65	1
0	dt_split_hqt_flag	0	66	1
0	dt_split_vqt_flag	0	67	1
0	dt_split_hadt_flag	0	68	1
0	dt_split_vadt_flag	0	69	1
0	umve_mv_index	0	70	1
0	umve_step_index	0	71	1
0	umve_dir_index	binIndex	72	2
0	cu_affine_cand_index	binIndex	74	4
0	etmvp_flag	0	78	1
0	cu_etmvp_cand_index	binIndex	79	4
0	awp_flag	0	83	1
0	awp_mvr_cand_flag0 awp_mvr_cand_flag1	0	84	1
0	awp_mvr_cand_step0 awp_mvr_cand_step1	0	85	1
0	awp_mvr_cand_dir0 awp_mvr_cand_dir1	binIndex	86	2
0	awp_cand_index0	0	88	1
0	awp_cand_index1	见7.3.3.2.11	89	2
0	affine_umve_step_index0 affine_umve_step_index1	0	91	1
0	affine_umve_dir_index0 affine_umve_dir_index1	binIndex	92	2
0	cu_subtype_index	binIndex	94	11

T/AI 109.8-2025

表 78 语法元素对应的 ctxIndexStart 和 ctxIndexInc（续）

IsPrivacyFlag	语法元素	ctxIndexInc	ctxIndexStart	ctx数量
0	isc_ibc_cu_flag	0	105	1
0	ibc_cu_flag	见7.3.3.2.12	106	3
0	cbvp_index	binIndex	109	6
0	abvr_index	binIndex	115	1
0	extend_mvr_flag	0	116	1
0	affine_amvr_index	binIndex	117	2
0	amvr_index	binIndex	119	4
0	inter_pred_ref_mode_index	见7.3.3.2.13	123	3
0	smvd_flag	0	126	1
0	pu_reference_index_l0 pu_reference_index_l1	见7.3.3.2.14	127	3
0	mv_diff_x_abs_l0 mv_diff_x_abs_l0_affine mv_diff_x_abs_l1 mv_diff_x_abs_l1_affine	见7.3.3.2.15	130	3
0	mv_diff_y_abs_l0 mv_diff_y_abs_l0_affine mv_diff_y_abs_l1 mv_diff_y_abs_l1_affine	见7.3.3.2.15	133	3
0	mv_diff_x_abs_bv	见7.3.3.2.16	136	7
0	mv_diff_y_abs_bv	见7.3.3.2.17	143	8
0	bgc_flag	0	151	1
0	bgc_index	0	152	1
0	intra_luma_pred_mode_index sawp_pred_mode0_index sawp_pred_mode1_index	见7.3.3.2.18	153	7
0	luma_eipm_index	0	160	1
0	intra_pf_flag	0	161	1
0	iip_flag	0	162	1
0	ctp_zero_flag	见7.3.3.2.19	163	2
0	pbt_cu_flag	0	165	1
0	ctp_y[i]	0	166	1
0	cu_qp_delta_abs	见7.3.3.2.20	167	4
0	enhanced_ts_flag	0	171	1
0	sbt_cu_flag	见7.3.3.2.21	172	2
0	sbt_quad_flag	0	174	1
0	sbt_dir_flag	见7.3.3.2.22	175	3
0	scan_region_x	见7.3.3.2.23	178	28

T/AI 109.8-2025

表 78 语法元素对应的 ctxIndexStart 和 ctxIndexInc（续）

IsPrivacyFlag	语法元素	ctxIndexInc	ctxIndexStart	ctx数量
0	scan_region_y	见7.3.3.2.23	206	28
0	significant_flag	见7.3.3.2.24	234	60
0	coeff_abs_level_greater1_flag coeff_abs_level_greater2_flag coeff_abs_level_greater4_flag coeff_abs_level_greater8_flag	见7.3.3.2.25	294	22
0	coeff_run	见7.3.3.2.26	316	24
0	coeff_level_minus1	见7.3.3.2.26	340	24
0	coeff_last	见7.3.3.2.27	364	34
0	est_tu_flag	0	398	1
0	esao_lcu_enable_flag	0	399	1
0	ccsao_lcu_enable_flag	0	400	1
0	isc_match_type[i]	0	401	1
0	isc_pixel_match_type[i][j]	0	402	1
0	isc_next_remaining_pixel_in_cu[i]	binIndex	403	3
0	isc_sv_above_flag[i]	0	406	1
0	isc_sv_recent_flag[i]	0	407	1
0	isc_sv_recent_index[i]	binIndex	408	3
0	isc_sv_y_abs[i]	见7.3.3.2.16	411	7
0	isc_sv_x_non_zero_flag[i]	0	418	1
0	isc_subtype_flag	0	419	1
0	isc_ups_present_flag	0	420	1
0	isc_evs_ups_num_minus1	0	421	1
0	isc_ups_max_length_minus1	Min(binIndex,3)	422	4
0	str_length_minus1_prefix	(binIndex+1)>>1	426	3
0	nn_filter_lcu_enable_flag	compIdx	429	3
0	nn_filter_lcu_set_index	compIdx*3+Min(binIdx,2)	432	9
1	intra_chroma_pred_mode_index	见7.3.3.2.7	500	3
1	chroma_eipm_index	0	503	1
1	chroma_pmc_index	0	504	1
1	chroma_pmc_param_index	0	505	1
1	ctp_u	0	506	1
1	ctp_v	0	507	1
1	chroma_st_flag	0	508	1
1	ts_cu_flag	0	509	1
1	skip_flag	见7.3.3.2.8	510	4
1	advanced_pred_flag	0	514	1

T/AI 109.8-2025

表 78 语法元素对应的 ctxIndexStart 和 ctxIndexInc（续）

IsPrivacyFlag	语法元素	ctxIndexInc	ctxIndexStart	ctx数量
1	affine_flag	0	515	1
1	affine_umve_flag	0	516	1
1	direct_flag	见7.3.3.2.9	517	2
1	inter_pf_flag	0	519	1
1	inter_pf_index	0	520	1
1	inter_pc_flag	0	521	1
1	inter_pc_index	binIndex	522	2
1	intra_cu_flag	见7.3.3.2.10	524	6
1	sawp_flag	0	530	1
1	dt_split_flag	0	531	1
1	dt_split_dir	0	532	1
1	dt_split_hqt_flag	0	533	1
1	dt_split_vqt_flag	0	534	1
1	dt_split_hadt_flag	0	535	1
1	dt_split_vadt_flag	0	536	1
1	umve_mv_index	0	537	1
1	umve_step_index	0	538	1
1	umve_dir_index	binIndex	539	2
1	cu_affine_cand_index	binIndex	541	4
1	etmvp_flag	0	545	1
1	cu_etmvp_cand_index	binIndex	546	4
1	awp_flag	0	550	1
1	awp_mvr_cand_flag0 awp_mvr_cand_flag1	0	551	1
1	awp_mvr_cand_step0 awp_mvr_cand_step1	0	552	1
1	awp_mvr_cand_dir0 awp_mvr_cand_dir1	binIndex	553	2
1	awp_cand_index0	0	555	1
1	awp_cand_index1	见7.3.3.2.11	556	2
1	affine_umve_step_index0 affine_umve_step_index1	0	558	1
1	affine_umve_dir_index0 affine_umve_dir_index1	binIndex	559	2
1	cu_subtype_index	binIndex	561	11
1	isc_ibc_cu_flag	0	572	1
1	ibc_cu_flag	见7.3.3.2.12	573	3
1	cbvp_index	binIndex	576	6

T/AI 109.8-2025

表 78 语法元素对应的 ctxIndexStart 和 ctxIndexInc（续）

IsPrivacyFlag	语法元素	ctxIndexInc	ctxIndexStart	ctx数量
1	abvr_index	binIndex	582	1
1	extend_mvr_flag	0	583	1
1	affine_amvr_index	binIndex	584	2
1	amvr_index	binIndex	586	4
1	inter_pred_ref_mode_index	见7.3.3.2.13	590	3
1	smvd_flag	0	593	1
1	pu_reference_index_l0 pu_reference_index_l1	见7.3.3.2.14	594	3
1	mv_diff_x_abs_l0 mv_diff_x_abs_l0_affine mv_diff_x_abs_l1 mv_diff_x_abs_l1_affine	见7.3.3.2.15	597	3
1	mv_diff_y_abs_l0 mv_diff_y_abs_l0_affine mv_diff_y_abs_l1 mv_diff_y_abs_l1_affine	见7.3.3.2.15	600	3
1	mv_diff_x_abs_bv	见7.3.3.2.16	603	7
1	mv_diff_y_abs_bv	见7.3.3.2.17	610	8
1	bgc_flag	0	618	1
1	bgc_index	0	619	1
1	intra_luma_pred_mode_index sawp_pred_mode0_index sawp_pred_mode1_index	见7.3.3.2.18	620	7
1	luma_eipm_index	0	627	1
1	intra_pf_flag	0	628	1
1	iip_flag	0	629	1
1	ctp_zero_flag	见7.3.3.2.19	630	2
1	pbt_cu_flag	0	632	1
1	ctp_y[i]	0	633	1
1	cu_qp_delta_abs	见7.3.3.2.20	634	4
1	enhanced_ts_flag	0	638	1
1	sbt_cu_flag	见7.3.3.2.21	639	2
1	sbt_quad_flag	0	641	1
1	sbt_dir_flag	见7.3.3.2.22	642	3
1	scan_region_x	见7.3.3.2.23	645	28
1	scan_region_y	见7.3.3.2.23	673	28
1	significant_flag	见7.3.3.2.24	701	60

T/AI 109.8-2025

表 78 语法元素对应的 ctxIndexStart 和 ctxIndexInc（续）

IsPrivacyFlag	语法元素	ctxIndexInc	ctxIndexStart	ctx数量
1	coeff_abs_level_greater1_flag coeff_abs_level_greater2_flag coeff_abs_level_greater4_flag coeff_abs_level_greater8_flag	见7.3.3.2.25	761	22
1	coeff_run	见7.3.3.2.26	783	24
1	coeff_level_minus1	见7.3.3.2.26	807	24
1	coeff_last	见7.3.3.2.27	831	34
1	est_tu_flag	0	865	1
1	esao_lcu_enable_flag	0	866	1
1	ccsao_lcu_enable_flag	0	867	1
1	isc_match_type[i]	0	868	1
1	isc_pixel_match_type[i][j]	0	869	1
1	isc_next_remaining_pixel_in_cu[i]	binIndex	870	3
1	isc_sv_above_flag[i]	0	873	1
1	isc_sv_recent_flag[i]	0	874	1
1	isc_sv_recent_index[i]	binIndex	875	3
1	isc_sv_y_abs[i]	见7.3.3.2.16	878	7
1	isc_sv_x_non_zero_flag[i]	0	885	1
1	isc_subtype_flag	0	886	1
1	isc_ups_present_flag	0	887	1
1	isc_evs_ups_num_minus1	0	888	1
1	isc_ups_max_length_minus1	Min(binIndex,3)	889	4
1	str_length_minus1_prefix	(binIndex+1)>>1	893	3
1	nn_filter_lcu_enable_flag	compIdx	896	3
1	nn_filter_lcu_set_index	compIdx*3+Min(binIdx,2)	899	9

T/AI 109.8-2025

7.3.3.2.2 确定 lcu_qp_delta 的 ctxIndexInc

根据以下方法确定 lcu_qp_delta 的 ctxIndexInc：

——如果 binIndex 和 PreviousDeltaQP 均等于 0，则 ctxIndexInc 等于 0；

——否则，如果 binIndex 等于 0 且 PreviousDeltaQP 不等于 0，则 ctxIndexInc 等于 1；

——否则，如果 binIndex 等于 1，则 ctxIndexInc 等于 2；

——否则，ctxIndexInc 等于 3。

7.3.3.2.3 确定 qt_split_flag 的 ctxIndexInc

根据以下方法确定 qt_split_flag 的 ctxIndexInc（当前亮度编码块 E 与其左边亮度编码块 A、上边亮度编码块 B 的关系见 8.6.4）：

——如果当前图像为帧内预测图像且 E 的宽度为 128，则 ctxIndexInc 等于 3；
——否则，如果 A“存在”且 A 的高度小于 E 的高度，且 B“存在”且 B 的宽度小于 E 的宽度，则 ctxIndexInc 等于 2；
——否则，如果 A“存在”且 A 的高度小于 E 的高度，或 B“存在”且 B 的宽度小于 E 的宽度，则 ctxIndexInc 等于 1；
——否则，ctxIndexInc 等于 0。

7.3.3.2.4 确定 bet_split_flag 的 ctxIndexInc

根据以下方法确定 bet_split_flag 的 ctxIndexInc（当前亮度编码块 E 与其左边亮度编码块 A、上边亮度编码块 B 的关系见 8.6.4）：
——如果 A“存在”且 A 的高度小于 E 的高度，且 B“存在”且 B 的宽度小于 E 的宽度，则 ctxIndexInc 等于 2；
——否则，如果 A“存在”且 A 的高度小于 E 的高度，或块 B“存在”且块 B 的宽度小于 E 的宽度，则 ctxIndexInc 等于 1；
——否则，ctxIndexInc 等于 0；
根据以下方法进一步确定 bet_split_flag 的 ctxIndexInc：
——如果 E 的宽度乘以 E 的高度的积大于 1024，则 ctxIndexInc 不变；
——否则，如果 E 的宽度乘以 E 的高度的积大于 256，则 ctxIndexInc 增加 3；
——否则，ctxIndexInc 增加 6。

7.3.3.2.5 确定 bet_split_type_flag 的 ctxIndexInc

根据以下方法确定 bet_split_type_flag 的 ctxIndexInc（当前亮度编码块 E 与其左边亮度编码块 A、上边亮度编码块 B 的关系见 8.6.4）：
——如果 A“存在”且 A 的高度小于 E 的高度，且 B“存在”且 B 的宽度小于 E 的宽度，则 ctxIndexInc 等于 2；
——否则，如果 A“存在”且 A 的高度小于 E 的高度，或 B“存在”且 B 的宽度小于 E 的宽度，则 ctxIndexInc 等于 1；
——否则，ctxIndexInc 等于 0。

7.3.3.2.6 确定 bet_split_dir_flag 的 ctxIndexInc

根据以下方法确定 bet_split_dir_flag 的 ctxIndexInc（E 是当前亮度编码块）：
——如果 E 的宽度为 128 且高度为 64，则 ctxIndexInc 等于 4；
——否则，如果 E 的宽度为 64 且高度为 128，则 ctxIndexInc 等于 3；
——否则，如果 E 的高度大于 E 的宽度，则 ctxIndexInc 等于 2；
——否则，如果 E 的宽度大于 E 的高度，则 ctxIndexInc 等于 1；
——否则，ctxIndexInc 等于 0。

7.3.3.2.7 确定 intra_chroma_pred_mode_index 的 ctxIndexInc

根据以下方法确定 intra_chroma_pred_mode_index 的 ctxIndexInc：
——如果 binIndex 等于 0，则 ctxIndexInc 等于 0；
——否则，如果 TscpmEnableFlag 的值等于 1 且 binIndex 等于 1，则 ctxIndexInc 等于 2；
——否则，如果 PmcEnableFlag 的值等于 1 且 binIndex 等于 1，则 ctxIndexInc 等于 2；

T/AI 109.8-2025

——否则，ctxIndexInc 等于 1。

7.3.3.2.8 确定 skip_flag 的 ctxIndexInc

根据以下方法确定 skip_flag 的 ctxIndexInc（当前预测块 E 与其左边预测块 A、上边预测块 B 的关系见 8.6.4）：

——如果当前预测块 E 的宽度乘以当前预测块 E 的高度的积小于 64，ctxIndexInc 等于 3；

——否则，如果当前预测块 E 的左边预测块 A“可用”且 A 是跳过模式，且当前预测块 E 的上边预测块 B“可用”且 B 是跳过模式，则 ctxIndexInc 等于 2；

——否则，如果当前预测块 E 的左边预测块 A“可用”且 A 是跳过模式，或当前预测块 E 的上边预测块 B“可用”且 B 是跳过模式，则 ctxIndexInc 等于 1；

——否则，ctxIndexInc 等于 0。

7.3.3.2.9 确定 direct_flag 的 ctxIndexInc

根据以下方法确定 direct_flag 的 ctxIndexInc：

——如果当前预测块 E 的宽度乘以当前预测块 E 的高度的积小于 64，或当前预测块 E 的宽度大于 64，或当前预测块 E 的高度大于 64，则 ctxIndexInc 等于 1；

——否则，ctxIndexInc 等于 0。

7.3.3.2.10 确定 intra_cu_flag 的 ctxIndexInc

根据以下方法确定 intra_cu_flag 的 ctxIndexInc（当前预测块 E 与其左边预测块 A、上边预测块 B 的关系见 8.6.4）：

——如果当前预测块 E 的宽度大于 64，或当前预测块 E 高度大于 64，则 ctxIndexInc 等于 5；

——否则，如果当前预测块 E 的左边预测块 A“可用”且预测块 A 是帧内预测模式，且当前预测块 E 的上边预测块 B“可用”且预测块 B 是帧内预测模式，则 ctxIndexInc 等于 2；

——否则，如果当前预测块 E 的左边预测块 A“可用”且预测块 A 是帧内预测模式，或当前预测块 E 的上边预测块 B“可用”且预测块 B 是帧内预测模式，则 ctxIndexInc 等于 1；

——否则，ctxIndexInc 等于 0。

如果 ctxIndexInc 等于 0，根据以下方法进一步确定 intra_cu_flag 的 ctxIndexInc：

——如果当前预测块 E 的宽度乘以当前预测块 E 的高度的积大于 256，则 ctxIndexInc 等于 0；

——否则，如果当前预测块 E 的宽度乘以当前预测块 E 的高度的积大于 64，则 ctxIndexInc 等于 3；

——否则，ctxIndexInc 等于 4。

7.3.3.2.11 确定 awp_cand_index1 的 ctxIndexInc

根据以下方法确定 awp_cand_index1 的 ctxIndexInc：

——如果 AwpMvrCandFlag0 和 AwpMvrCandFlag1 均为 0，则 ctxIndexInc 等于 1；

——否则，如果 AwpMvrCandFlag0 和 AwpMvrCandFlag1 均为 1，且 AwpMvrCandStep0 等于 AwpMvrCandStep1，且 AwpMvrCandDir0 等于 AwpMvrCandDir1，则 ctxIndexInc 等于 1；

——否则，ctxIndexInc 等于 0。

7.3.3.2.12 确定 ibc_cu_flag 的 ctxIndexInc

根据以下方法确定 ibc_cu_flag 的 ctxIndexInc（当前预测块 E 与其左边预测块 A、上边预测块 B

T/AI 109.8-2025

的关系见 8.6.4）：

——如果当前预测块 E 的左边预测块 A“可用”且 A 是块复制帧内预测模式，且当前预测块 E 的上边预测块 B“可用”且 B 是块复制帧内预测模式，则 ctxIndexInc 等于 2；

——否则，如果当前预测块 E 的左边预测块 A“可用”且 A 是块复制帧内预测模式，或当前预测块 E 的上边预测块 B“可用”且 B 是块复制帧内预测模式，则 ctxIndexInc 等于 1；

——否则，ctxIndexInc 等于 0。

7.3.3.2.13 确定 inter_pred_ref_mode_index 的 ctxIndexInc

根据以下方法确定 inter_pred_ref_mode_index 的 ctxIndexInc：

——如果该语法元素的 binIndex 为 0，且当前预测块 E 的宽度乘以当前预测块 E 的高度的积小于 64，则 ctxIndexInc 等于 2；

——否则，如果该语法元素的 binIndex 为 0，且当前预测块 E 的宽度乘以当前预测块 E 的高度的积大于等于 64，则 ctxIndexInc 等于 0；

——否则（该语法元素的 binIndex 为 1），ctxIndexInc 等于 1。

7.3.3.2.14 确定 pu_reference_index_l0 或 pu_reference_index_l1 的 ctxIndexInc

根据以下方法确定 pu_reference_index 的 ctxIndexInc：

——如果 binIndex 等于 0，则 ctxIndexInc 等于 0；

——否则，如果 binIndex 等于 1，则 ctxIndexInc 等于 1；

——否则，ctxIndexInc 等于 2。

7.3.3.2.15 确定 mv_diff_x_abs_l0、mv_diff_x_abs_l0_affine、mv_diff_x_abs_l1、mv_diff_x_abs_l1_affine、mv_diff_y_abs_l0、mv_diff_y_abs_l0_affine、mv_diff_y_abs_l1 或 mv_diff_y_abs_l1_affine 的 ctxIndexInc

根据以下方法确定 mv_diff_x_abs_l0、mv_diff_x_abs_l0_affine、mv_diff_x_abs_l1、mv_diff_x_abs_l1_affine、mv_diff_y_abs_l0、mv_diff_y_abs_l0_affine、mv_diff_y_abs_l1 或 mv_diff_y_abs_l1_affine 的 ctxIndexInc：

——如果 binIndex 等于 0，则 ctxIndexInc 等于 0；

——否则，如果 binIndex 等于 1，则 ctxIndexInc 等于 1；

——否则，如果 binIndex 等于 2，则 ctxIndexInc 等于 2；

——否则，BypassFlag 的值为 1。

7.3.3.2.16 确定 mv_diff_x_abs_bv 或 isc_sv_y_abs[i]的 ctxIndexInc

根据以下方法确定 mv_diff_x_abs_bv 或 isc_sv_y_abs[i]的 ctxIndexInc：

——如果 binIndex 小于等于 5，则 ctxIndexInc 等于 binIndex；

——否则，如果 binIndex 等于 6，且前一个 bin 为 1，则 ctxIndexInc 等于 6；

——否则，BypassFlag 的值为 1。

7.3.3.2.17 确定 mv_diff_y_abs_bv 的 ctxIndexInc

根据以下方法确定 mv_diff_y_abs_bv 的 ctxIndexInc：

——如果 mv_diff_x_abs_bv 等于 0 且 binIndex 等于 0，则 ctxIndexInc 等于 7；

——否则，如果 binIndex 小于等于 5，则 ctxIndexInc 等于 binIndex；

T/AI 109.8-2025

——否则，如果 binIndex 等于 6，且前一个 bin 为 1，则 ctxIndexInc 等于 6；
——否则，BypassFlag 的值为 1。

7.3.3.2.18 确定 intra_luma_pred_mode_index、sawp_pred_mode0_index 或 sawp_pred_mode1_index 的 ctxIndexInc

根据以下方法确定 intra_luma_pred_mode_index 的 ctxIndexInc：
——根据该语法元素 binIndex 和当前已解析的二元符号串和 binIndex，查表 79 得到 ctxIndexInc。表 79 中，x_i(i=1～4)的值为‘0’或‘1’。

表79 前缀二元符号串与 ctxIndexInc 的关系

binIndex	当前已解析的二元符号串	ctxIndexInc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

根据以下方法确定 sawp_pred_mode0_index 或 sawp_pred_mode1_index 的 ctxIndexInc：
——如果 binIndex 等于 0，则 ctxIndexInc 等于 0；
——否则，BypassFlag 的值为 1。

7.3.3.2.19 确定 ctp_zero_flag 的 ctxIndexInc

根据以下方法确定 ctp_zero_flag 的 ctxIndexInc：
——如果当前预测块 E 的宽度大于 64，或当前预测块 E 的高度大于 64，则 ctxIndexInc 等于 1；
——否则，则 ctxIndexInc 等于 0。

7.3.3.2.20 确定 cu_qp_delta_abs 的 ctxIndexInc

根据以下方法确定 cu_qp_delta_abs 的 ctxIndexInc：
——如果 binIndex 为 0，且 IsPrivacyFlag 为 0，则 ctxIndexInc = min(NumDeltaQp, 2)；
——否则，如果binIndex为0，且IsPrivacyFlag不为0，则ctxIndexInc = min(NumDeltaQpPrivacy, 2)；
——否则，ctxIndexInc 等于 3。

7.3.3.2.21 确定 sbt_cu_flag 的 ctxIndexInc

根据以下方法确定 sbt_cu_flag 的 ctxIndexInc：
——如果当前亮度编码块的宽度和高度的乘积大于或等于 256，则 ctxIndexInc 等于 0；
——否则，ctxIndexInc 等于 1。

7.3.3.2.22 确定 sbt_dir_flag 的 ctxIndexInc

T/AI 109.8-2025

根据以下方法确定 sbt_dir_flag 的 ctxIndexInc：

——如果当前亮度编码块的宽度等于高度，则 ctxIndexInc 等于 0；

——否则，如果当前亮度编码块的宽度小于高度，则 ctxIndexInc 等于 1；

——否则，ctxIndexInc 等于 2。

7.3.3.2.23 确定 scan_region_x 或 scan_region_y 的 ctxIndexInc

根据以下步骤确定 scan_region_x 或 scan_region_y 的 ctxIndexInc：

——令 ctxIndexInc 的值为 0；

——当前块的宽度和高度是 blockWidth 和 blockHeight。如果需要确定的是 scan_region_x，M 等于 blockWidth；如果需要确定的是 scan_region_y，M 等于 blockHeight；

——如果当前块是亮度编码块，则：

```
offset = prefix_ctx[logM]
shift = (logM+1)>>2
```

其中 prefix_ctx 的规定见附录 L。

——如果当前块是色度编码块，则：

```
offset = 25
shift = Clip3(0, 2, (M >> 3))
```

——执行以下操作：

a) 如果 binIndex 等于 0，则 ctxIndexInc = offset；

b) 否则，如果 binIndex 大于 0，且前一个 bin 为 1，且 binIndex 小于 group_index[Min(M, 32)-1]，则 ctxIndexInc = offset + (binIndex>>shift)。其中 group_index 的规定见附录 L。

7.3.3.2.24 确定 significant_flag 的 ctxIndexInc

根据以下步骤依次执行，确定 significant_flag 的 ctxIndexInc：

——令 ctxIndexInc 的值为 0；

——如果当前块是亮度编码块，则：

```
scanArea = (ScanRegionX + 1) * (ScanRegionY + 1)
ctxIndexInc = ctxIndexInc + scanArea <= 4 ? 0 : 17 << (scanArea <= 16 ? 0 : 1)
```

——如果当前块是色度编码块，则：

```
ctxIndexInc = ctxIndexInc + 51
```

——扫描由左上角坐标(0, 0)和右下角坐标(ScanRegionX, ScanRegionY)确定的矩形区域；

——如果 PosX 不等于 ScanRegionX 或 PosY 不等于 ScanRegionY ：

a) 如果当前块是亮度编码块，且 PosX 不为 0 或 PosY 不为 0：

- 如果当前块预测类型是普通帧内预测：

```
i = (PosX<= ScanRegionX/2) ? (PosX <= ScanRegionX/4 ? 0 : 1) : ( PosX <= ScanRegionX * 3/4 ? 2 :
        3 )
j = (PosY <= ScanRegionY/2) ? (PosY <= ScanRegionY/4 ? 0 : 1) : ( PosY <= ScanRegionY * 3/4 ? 2 :
        3 )
s = i + j
if (s <= 1)
    ctxIndexInc = ctxIndexInc + 4
else if (s > 2)
```

T/AI 109.8-2025

```
        ctxIndexInc = ctxIndexInc + 8
    else
        ctxIndexInc = ctxIndexInc + 12
```

- 否则：

```
if ((PosX <= ScanRegionX / 2) && (PosY <= ScanRegionY / 2))
        ctxIndexInc = ctxIndexInc + 4
    else
        ctxIndexInc = ctxIndexInc + 8
```

- 如果当前块是色度编码块，且 PosX 不为 0 或 PosY 不为 0，则 ctxIndexInc = ctxIndexInc + 4。

b) 令 Pre5Gt0[j]（j=0～4）中 Pre5Gt0[j]的值不为零的个数为为 num_gt0，ctxIndexInc 为：

```
ctxIndexInc = ctxIndexInc + Min(num_gt0, 3) + 1
```

7.3.3.2.25 确定 coeff_abs_level_greater1_flag，coeff_abs_level_greater2_flag，coeff_abs_level_greater4_flag 或 coeff_abs_level_greater8_flag 的 ctxIndexInc

根据以下步骤依次执行，确定 coeff_abs_level_greater1_flag、coeff_abs_level_greater2_flag、coeff_abs_level_greater4_flag 或 coeff_abs_level_greater8_flag 的 ctxIndexInc：

——令 ctxIndexInc 的值为 0；

——如果当前块是色度块，则 ctxIndexInc = ctxIndexInc + 17；

——扫描由左上角坐标(0,0)和右下角坐标(ScanRegionX,ScanRegionY)确定的矩形区域中的非零系数位置；

——如果(PosX,PosY)不是扫描到的第一个非零系数位置：

a) 如果当前块是亮度块，且 PosX 不等于 0 或 PosY 不等于 0：

- 如果当前块预测类型是普通帧内预测，则：

```
i =(PosX<= ScanRegionX/2) ? (PosX <= ScanRegionX/4 ? 0 : 1) : ( PosX <= ScanRegionX * 3/4 ? 2 : 3)
j= (PosY <= ScanRegionY/2) ? (PosY <= ScanRegionY/4 ? 0 : 1) : ( PosY <= ScanRegionY * 3/4 ? 2 : 3)
s = i + j
if (s <= 1)
        ctxIndexInc = ctxIndexInc + 4
    else if (s > 2)
        ctxIndexInc = ctxIndexInc + 8
    else
        ctxIndexInc = ctxIndexInc + 12
```

- 否则：

```
if ((PosX <= ScanRegionX / 2) && (PosY <= ScanRegionY / 2))
        ctxIndexInc = ctxIndexInc + 4
    else
        ctxIndexInc = ctxIndexInc + 8
```

b) 令 Pre5GtX[j]（j=0～4）中 Pre5GtX[j]的绝对值大于 k 的个数为 num_gtX（k=1,2,4,8）。

- 如果需要确定的是 coeff_abs_level_greater1_flag 的 ctxIndexInc，k 等于 1；

T/AI 109.8-2025

- 如果需要确定的是 coeff_abs_level_greater2_flag 的 ctxIndexInc，k 等于 2；
- 如果需要确定的是 coeff_abs_level_greater4_flag 的 ctxIndexInc，k 等于 4；
- 如果需要确定的是 coeff_abs_level_greater8_flag 的 ctxIndexInc，k 等于 8；
- ctxIndexInc 为：

```
ctxIndexInc = ctxIndexInc + Min(num_gtX, 3) + 1
```

7.3.3.2.26 确定 coeff_run 或 coeff_level_minus1 的 ctxIndexInc

根据以下方法确定 coeff_level_minus1 或 coeff_run 的 ctxIndexInc：

——令 ctxIndexInc 的值为 0；

——当前系数块上一个解码出的 level 值记为 prev_level，如果没有上一个解码出的 level 值，则 prev_level = 6，ctxIndexInc = ctxIndexInc + min(prev_level-1, 5) × 2；

——如果当前块是色度编码块，则 ctxIndexInc = ctxIndexInc + 12；

——如果 binIndex 大于或等于 1，则 ctxIndexInc = ctxIndexInc + 1。

7.3.3.2.27 确定 coeff_last 的 ctxIndexInc 和 ctxIndexIncW

当前系数块上一个解码出的 level 值记为 prev_level，如果没有上一个解码出的 level 值，则 prev_level 的值等于 6。

根据以下方法确定 coeff_last 第一个上下文的 ctxIndexInc：

——令 ctxIndexInc 的值为 min(prev_level-1,5)；

——如果当前块是色度编码块，则 ctxIndexInc = ctxIndexInc + 6。

根据以下方法确定 coeff_last 第二个上下文的 ctxIndexIncW：

——令 ctxIndexIncW 的值为 12 + Floor(Log(ScanPosOffset +1))；

——如果当前块是色度编码块，则 ctxIndexIncW = ctxIndexIncW + 12。

7.3.3.3 二元符号解析

7.3.3.3.1 解析过程

二元符号的解析过程如下：

第 1 步，解析二元符号值 binVal。

a) 如果 BypassFlag 为 1，执行 decode_bypass 过程（见 7.3.3.3.4）；

b) 否则，如果 StuffingBitFlag 为 1，执行 decode_aec_stuffing_bit 过程（见 7.3.3.3.3）；

c) 否则，令 cFlag 等于 1，执行 decode_decision 过程（见 7.3.3.3.2）。

第 2 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’。

7.3.3.3.2 decode_decision

decode_decision 过程的输入是 bFlag、cFlag、rS1、rT1、valueS、valueT、valueD 以及上下文模型。如果 CtxWeight 的值为 0，输入的上下文模型为 ctx；否则输入的上下文模型为 ctx 和 ctxW。decode_decision 过程的输出是二元符号值 binVal。decode_decision 过程如下：

```
decode_decision( ) {
    if (CtxWeight == 0) {
        predMps = ctx->mps
        if (MaecEnableFlag)
```

T/AI 109.8-2025

```
            lgPmps = ((ctx->lgPmps0 + ctx->lgPmps1 + 1) >> 3
        else
            lgPmps = ctx->lgPmps >> 2
    }
    else {
        if (MaecEnableFlag) {
            lgPmps1 = ((ctx->lgPmps0 + ctx->lgPmps1 + 1) >> 1
            lgPmps2 = ((ctxW->lgPmps0 + ctxW->lgPmps1 + 1) >> 1
        }
        else {
            lgPmps1 = ctx->lgPmps
            lgPmps2 = ctxW->lgPmps
        }
        if (ctx->mps == ctxW->mps ) {
            predMps = ctx->mps
            lgPmps = (lgPmps1 + lgPmps2) >> 1
        }
        else {
            if (lgPmps1 < lgPmps2) {
                predMps = ctx->mps
                lgPmps = 1023 - ((lgPmps2 - lgPmps1) >> 1 )
            }
            else {
                predMps = ctxW->mps
                lgPmps = 1023 - (lgPmps1 - lgPmps2) >> 1 )
            }
        }
        lgPmps = 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
    }
```

T/AI 109.8-2025

```
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) {
    if (CtxWeight == 0)
        ctx = update_ctx( binVal, ctx )
    else {
        ctx = update_ctx(binVal, ctx)
        ctxW = update_ctx(binVal, ctxW)
    }
}
return(binVal)
```

T/AI 109.8-2025

```
}
```

7.3.3.3.3 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。ctx0 是二元符号模型，令 ctx0->lgPmps 等于 4，ctx0->mps 等于 0。令 cFlag 等于 0，ctx 等于 ctx0，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M的描述方式实现。

7.3.3.3.4 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。ctx1 是二元符号模型，令 ctx1->lgPmps 等于 1024，ctx1->mps 等于 0。令 cFlag 等于 0，ctx 等于 ctx1，带入 decode_decision 过程实现 decode_bypass 过程。

注：decode_bypass过程也可参考附录M的描述方式实现。

7.3.3.3.5 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 过程如下：

```
update_ctx( ) {
    if (MaecEnableFlag) {
        a = baseWin - diffWin0
        b = baseWin + diffWin1
        cwr = a
        if (ctx->cycno < counterThr1)
            cwr = a - 1
        else if (ctx->cycno < counterThr2)
            cwr = a
        longUpdate = (ctx->cycno < counterThr2) ? 0 : 1
        if (binVal != ctx->mps) {
            if (ctx->cycno < counterThr2)
                ctx->cycno = ctx->cycno + 1
            else
                ctx->cycno = counterThr2
        }
        else if (ctx->cycno == 0)
            ctx->cycno = 1
        if (binVal == ctx->mps) {
            ctx->lgPmps0 = ctx->lgPmps0 - (ctx->lgPmps0 >> cwr) - (ctx->lgPmps0 >> (cwr+2))
            ctx->lgPmps1 = longUpdate ? (ctx->lgPmps1 - (ctx->lgPmps1 >> b) - (ctx->lgPmps1 >> (b+2))) :
ctx->lgPmps0
        }
```

T/AI 109.8-2025

```
            else {
                ctx->lgPmps0 = ctx->lgPmps0 + cwr2LGS[cwr]
                ctx->lgPmps1 = longUpdate ? (ctx->lgPmps1 + cwr2LGS[b]) : ctx->lgPmps0
                if ( ctx->lgPmps0 > 1023 || ctx->lgPmps1 > 1023) {
                    ctx->mps = ! (ctx->mps)
                }
                if (ctx->lgPmps0 > 1023)
                    ctx->lgPmps0 = 2047 - ctx->lgPmps0
                if (ctx->lgPmps1 > 1023)
                    ctx->lgPmps1 = 2047 - ctx->lgPmps1
            }
        }
        else {
            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 {
                ctx->lgPmps = ctx->lgPmps + cwr2LGS[cwr]
                if (ctx->lgPmps > 1023)
                    ctx->lgPmps = 2047 - ctx->lgPmps
                    ctx->mps = ! (ctx->mps)
            }
        }
        return(ctx)
}
```

其中 diffWin0、diffWin1 和 baseWin 见 7.3.2.1，cwr2LGS[10]的值是{427, 427, 427, 197, 95, 46, 23, 12, 6, 3}。

T/AI 109.8-2025

7.3.4 反二值化方法

7.3.4.1 概述

本条规定语法元素的反二值化方法，见表 80。

表80 语法元素的反二值化方法

语法元素	反二值化方法
lcu_qp_delta	见 7.3.4.5
sao_merge_type_index	见 7.3.4.6
sao_mode	见 7.3.4.2，maxVal=2，sao_mode 的值等于 synElVal
sao_interval_offset_abs	见 7.3.4.2，maxVal=7，sao_interval_offset_abs 的值等于 synElVal
sao_interval_offset_sign	见 7.3.4.4，sao_interval_offset_sign 的值等于 synElVal
sao_interval_start_pos	见 7.3.4.7
sao_interval_delta_pos_minus2	见 7.3.4.8
sao_edge_offset[compIndex][j]（j = 0～3）	见 7.3.4.9
sao_edge_type	见 7.3.4.10
alf_lcu_enable_flag	见 7.3.4.4，alf_lcu_enable_flag 的值等于 synElVal
aec_lcu_stuffing_bit	见 7.3.4.4，aec_lcu_stuffing_bit 的值等于 synElVal
qt_split_flag	见 7.3.4.4，qt_split_flag 的值等于 synElVal
bet_split_flag	见 7.3.4.4，bet_split_flag 的值等于 synElVal
bet_split_type_flag	见 7.3.4.4，bet_split_type_flag 的值等于 synElVal
bet_split_dir_flag	见 7.3.4.4，bet_split_dir_flag 的值等于 synElVal
root_cu_constraint	见 7.3.4.4，root_cu_constraint 的值等于 synElVal
intra_chroma_pred_mode_index	见 7.3.4.2，maxVal=((TscpmEnableFlag || PmcEnableFlag) ? 5 : 4)，intra_chroma_pred_mode_index 的值等于 synElVal
chroma_eipm_index	见 7.3.4.2，maxVal=3，chroma_eipm_index 的值等于 synElVal
chroma_pmc_index	见 7.3.4.4，chroma_pmc_index 的值等于 synElVal
chroma_pmc_param_index	见 7.3.4.4，chroma_pmc_param_index 的值等于 synElVal
ctp_u	见 7.3.4.4，ctp_u 的值等于 synElVal
ctp_v	见 7.3.4.4，ctp_v 的值等于 synElVal
skip_flag	见 7.3.4.4，skip_flag 的值等于 synElVal
privacy_level	见 7.3.4.4，privacy_level 的值等于 synElVal
advanced_pred_flag	见 7.3.4.4，advanced_pred_flag 的值等于 synElVal
awp_flag	见 7.3.4.4，awp_flag 的值等于 synElVal
etmvp_flag	见 7.3.4.4，etmvp_flag 的值等于 synElVal
cu_etmvp_cand_index	见 7.3.4.2，maxVal=4，cu_etmvp_cand_index 的值等于 synElVal
affine_flag	见 7.3.4.4，affine_flag 的值等于 synElVal
affine_umve_flag	见 7.3.4.4，affine_umve_flag 的值等于 synElVal
direct_flag	见 7.3.4.4，direct_flag 的值等于 synElVal
inter_pf_flag	见 7.3.4.4，inter_pf_flag 的值等于 synElVal

T/AI 109.8-2025

表 80 语法元素的反二值化方法（续）

语法元素	反二值化方法
inter_pf_index	见 7.3.4.4，inter_pf_index 的值等于 synElVal
inter_pc_flag	见 7.3.4.4，inter_pc_flag 的值等于 synElVal
inter_pc_index	见 7.3.4.2，maxVal=2，inter_pc_index 的值等于 synElVal
intra_cu_flag	见 7.3.4.4，intra_cu_flag 的值等于 synElVal
sawp_flag	见 7.3.4.4，sawp_flag 的值等于 synElVal
sawp_index	见 7.3.4.15
sawp_pred_mode0_index	见 7.3.4.12
sawp_pred_mode1_index	如果 sawp_pred_mode0_index 的值大于 1，见 7.3.4.12；否则，见 7.3.4.13
dt_split_flag	见 7.3.4.4，dt_split_flag 的值等于 synElVal
dt_split_dir	见 7.3.4.4，dt_split_dir 的值等于 synElVal
dt_split_hqt_flag	见 7.3.4.4，dt_split_hqt_flag 的值等于 synElVal
dt_split_vqt_flag	见 7.3.4.4，dt_split_vqt_flag 的值等于 synElVal
dt_split_hadt_flag	见 7.3.4.4，dt_split_hadt_flag 的值等于 synElVal
dt_split_vadt_flag	见 7.3.4.4，dt_split_vadt_flag 的值等于 synElVal
umve_mv_index	见 7.3.4.2，maxVal=1，umve_mv_index 的值等于 synElVal
umve_step_index	如果 PictureUmveStepSetIndex 的值为 0，见 7.3.4.2，maxVal=4，umve_step_index 的值等于 synElVal；否则，见 7.3.4.11
umve_dir_index	见 7.3.4.14
cu_affine_cand_index	见 7.3.4.2，maxVal=4，cu_affine_cand_index 的值等于 synElVal
awp_index	见 7.3.4.15
awp_mvr_cand_flag0	见 7.3.4.4，awp_mvr_cand_flag0 的值等于 synElVal
awp_mvr_cand_flag1	见 7.3.4.4，awp_mvr_cand_flag1 的值等于 synElVal
awp_mvr_cand_step0	见 7.3.4.2，maxVal=4，awp_mvr_cand_step0 的值等于 synElVal
awp_mvr_cand_step1	见 7.3.4.2，maxVal=4，awp_mvr_cand_step1 的值等于 synElVal
awp_mvr_cand_dir0	见 7.3.4.14
awp_mvr_cand_dir1	见 7.3.4.14
awp_cand_index0	见 7.3.4.2，maxVal=4，awp_cand_index0 的值等于 synElVal
awp_cand_index1	见 7.3.4.2，maxVal=(AwpMvrCandFlag0 == 0 && AwpMvrCandFlag1 == 0) || (AwpMvrCandFlag0 == 1 && AwpMvrCandFlag1 == 1 && AwpMvrCandStep0 == AwpMvrCandStep1 && AwpMvrCandDir0 == AwpMvrCandDir1)? 3 : 4，awp_cand_index1 的值等于 synElVal
affine_umve_step_index0	见 7.3.4.2，maxVal=4，affine_umve_step_index0 的值等于 synElVal
affine_umve_step_index1	见 7.3.4.2，maxVal=4，affine_umve_step_index1 的值等于 synElVal
affine_umve_dir_index0	见 7.3.4.14
affine_umve_dir_index1	见 7.3.4.14
cu_subtype_index	见 7.3.4.2，maxVal=((PictureCodingType == 2) ? (1: 3) + Max(NumOfMvapCand,NumOfHmvpCand))，cu_subtype_index 的值等于 synElVal

T/AI 109.8-2025

表 80 语法元素的反二值化方法（续）

语法元素	反二值化方法
isc_ibc_cu_flag	见 7.3.4.4，isc_ibc_cu_flag 的值等于 synElVal
ibc_cu_flag	见 7.3.4.4，ibc_cu_flag 的值等于 synElVal
isc_subtype_flag	见 7.3.4.4，isc_subtype_flag 的值等于 synElVal
cbvp_index	见 7.3.4.2，maxVal=6，cbvp_index 的值等于 synElVal
abvr_index	见 7.3.4.2，maxVal=1，abvr_index 的值等于 synElVal
extend_mvr_flag	见 7.3.4.4，extend_mvr_flag 的值等于 synElVal
affine_amvr_index	见 7.3.4.2， maxVal=2，affine_amvr_index 的值等于 synElVal
amvr_index	见 7.3.4.2，maxVal=4，amvr_index 的值等于 synElVal
inter_pred_ref_mode_index	见 7.3.4.16
smvd_flag	见 7.3.4.4，smvd_flag 的值等于 synElVal
pu_reference_index_l0	见 7.3.4.2，maxVal=NumRefActive[0]-1，pu_reference_index_l0 的值等于 synElVal
pu_reference_index_l1	见 7.3.4.2，maxVal=NumRefActive[1]-1，pu_reference_index_l1 的值等于 synElVal
mv_diff_x_abs_l0 mv_diff_x_abs_l0_affine mv_diff_x_abs_l1 mv_diff_x_abs_l1_affine	见 7.3.4.17
mv_diff_x_sign_l0 mv_diff_x_sign_l0_affine mv_diff_x_sign_l1 mv_diff_x_sign_l1_affine mv_diff_x_sign_bv	见 7.3.4.4，mv_diff_x_sign_l0、mv_diff_x_sign_l0_affine、mv_diff_x_sign_l1、mv_diff_x_sign_l1_affine 或 mv_diff_x_sign_bv 的值等于 synElVal
mv_diff_y_abs_l0 mv_diff_y_abs_l0_affine mv_diff_y_abs_l1 mv_diff_y_abs_l1_affine	见 7.3.4.17
mv_diff_y_sign_l0 mv_diff_y_sign_l0_affine mv_diff_y_sign_l1 mv_diff_y_sign_l1_affine mv_diff_y_sign_bv	见 7.3.4.4，mv_diff_y_sign_l0、mv_diff_y_sign_l0_affine、mv_diff_y_sign_l1、mv_diff_y_sign_l1_affine 或 mv_diff_y_sign_bv 的值等于 synElVal
mv_diff_x_abs_bv、mv_diff_y_abs_bv	见 7.3.4.18
bgc_flag	见 7.3.4.4，bgc_flag 的值等于 synElVal
bgc_index	见 7.3.4.4，bgc_index 的值等于 synElVal
intra_luma_pred_mode_index	见 7.3.4.19
luma_eipm_index	见 7.3.4.4，luma_eipm_index 的值等于 synElVal
intra_pf_flag	见 7.3.4.4，intra_pf_flag 的值等于 synElVal

T/AI 109.8-2025

表 80 语法元素的反二值化方法（续）

语法元素	反二值化方法
iip_flag	见 7.3.4.4，iip_flag 的值等于 synElVal
ctp_zero_flag	见 7.3.4.4，ctp_zero_flag 的值等于 synElVal
pbt_cu_flag	见 7.3.4.4，pbt_cu_flag 的值等于 synElVal
ctp_y[i]	见 7.3.4.4，cty_y[i]的值等于 synElVal
cu_qp_delta_abs	见 7.3.4.3，cu_qp_delta_abs 的值等于 synElVal
cu_qp_delta_sign	见 7.3.4.4，cu_qp_delta_sign 的值等于 synElVal
enhanced_ts_flag	见 7.3.4.4，enhanced_ts_flag 的值等于 synElVal
sbt_cu_flag	见 7.3.4.4，sbt_cu_flag 的值等于 synElVal
sbt_quad_flag	见 7.3.4.4，sbt_quad_flag 的值等于 synElVal
sbt_dir_flag	见 7.3.4.4，sbt_dir_flag 的值等于 synElVal
chroma_st_flag	见 7.3.4.4，chroma_st_flag 的值等于 synElVal
ts_cu_flag	见 7.3.4.4，ts_cu_flag 的值等于 synElVal
scan_region_x	见 7.3.4.20，scan_region_x 的值等于 synElVal
scan_region_y	见 7.3.4.20，scan_region_y 的值等于 synElVal
significant_flag	见 7.3.4.4，significant_flag 的值等于 synElVal
coeff_abs_level_greater1_flag	见 7.3.4.4，coeff_abs_level_greater1_flag 的值等于 synElVal
coeff_abs_level_greater2_flag	见 7.3.4.4，coeff_abs_level_greater2_flag 的值等于 synElVal
coeff_abs_level_greater4_flag	见 7.3.4.4，coeff_abs_level_greater4_flag 的值等于 synElVal
coeff_abs_level_greater8_flag	见 7.3.4.4，coeff_abs_level_greater8_flag 的值等于 synElVal
coeff_abs_level_remaining	见 7.3.4.21，coeff_abs_level_remaining 的值等于 synElVal
coeff_run	见 7.3.4.22，TH=16，coeff_run 的值等于 synElVal
coeff_level_minus1	见 7.3.4.22，TH=8，coeff_level_minus1 的值等于 synElVal
coeff_sign	见 7.3.4.4，coeff_sign 的值等于 synElVal
coeff_last	见 7.3.4.4，coeff_last 的值等于 synElVal
est_tu_flag	见 7.3.4.4，est_tu_flag 的值等于 synElVal
aec_ipcm_stuffing_bit	见 7.3.4.4，aec_ipcm_stuffing_bit 的值等于 synElVal
esao_lcu_enable_flag	见 7.3.4.4，esao_lcu_enable_flag 的值等于 synElVal
esao_lcu_set_index	见 7.3.4.2，maxVal=1，esao_lcu_set_index 的值等于 synElVal
ccsao_lcu_enable_flag	见 7.3.4.4，ccsao_lcu_enable_flag 的值等于 synElVal
ccsao_lcu_set_index	见 7.3.4.2，maxVal=PictureCcsaoSetNum-1，ccsao_lcu_set_index 的值等于 synElVal
isc_match_type[i]	见 7.3.4.4，isc_match_type[i]的值等于 synElVal
isc_sv_above_flag[i]	见 7.3.4.4，isc_sv_above_flag[i]的值等于 synElVal
isc_sv_recent_flag[i]	见 7.3.4.4，isc_sv_recent_flag[i]的值等于 synElVal
isc_sv_x_non_zero_flag[i]	见 7.3.4.4，isc_sv_x_non_zero_flag[i]的值等于 synElVal
isc_sv_x_sign[i]	见 7.3.4.4，isc_sv_x_sign[i]的值等于 synElVal
isc_sv_y_sign[i]	见 7.3.4.4，isc_sv_y_sign[i]的值等于 synElVal
isc_sv_x_abs_minus1[i]	见 7.3.4.23，isc_sv_x_abs_minus1[i]的值等于 synElVal

T/AI 109.8-2025

表 80 语法元素的反二值化方法（续）

语法元素	反二值化方法
isc_sv_y_abs[i]	见 7.3.4.18
isc_unmatched_pixel_y[i][j]	见表 103，len=BitDepth，isc_unmatched_pixel_y[i][j]的值等于 synElVal
isc_unmatched_pixel_cb[i][j]	见表 103，len=BitDepth，isc_unmatched_pixel_cb[i][j]的值等于 synElVal
isc_unmatched_pixel_cr[i][j]	见表 103，len=BitDepth，isc_unmatched_pixel_cr[i][j]的值等于 synElVal
isc_nos_up_y[i]	见表 103，len=BitDepth，isc_nos_up_y[i]的值等于 synElVal
isc_nos_up_cb[i]	见表 103，len=BitDepth，isc_nos_up_cb[i]的值等于 synElVal
isc_nos_up_cr[i]	见表 103，len=BitDepth，isc_nos_up_cr[i]的值等于 synElVal
isc_sv_recent_index[i]	见 7.3.4.2，maxVal=11，isc_sv_recent_index[i]的值等于 synElVal
isc_next_remaining_pixel_in_cu[i]	见 7.3.4.24，isc_next_remaining_pixel_in_cu[i]的值等于 synElVal
isc_pixel_match_type[i][j]	见 7.3.4.4，isc_pixel_match_type[i][j]的值等于 synElVal
isc_ups_present_flag	见 7.3.4.4，isc_ups_present_flag 的值等于 synElVal
isc_num_of_new_pv	见表 76，k=0，isc_num_of_new_pv 的值等于 CodeNum
isc_num_of_reused_pv	见表 76，k=0，isc_num_of_reused_pv 的值等于 CodeNum
isc_prev_pv_not_reused_run	见表 76，k=0，isc_prev_pv_not_reused_run 的值等于 CodeNum
isc_ups_max_length_minus1	见 7.3.4.2，maxVal = NumTotalPixel，isc_ups_max_length_minus1 的值等于 synElVal
isc_evs_ups_num_minus1	见 7.3.4.2，maxVal = NumTotalPixel - NumCodedPixel + FirstStringNumFlag - 1，isc_evs_ups_num_minus1 的值等于 synElVal
str_length_minus1_prefix	见 7.3.4.2，maxVal = maxValPrefix，str_length_minus1_prefix 的值等于 synElVal
str_length_minus1_infix	见 103，len= Ceil(Log(maxValInfix+1))-1，str_length_minus1_infix 的值等于 synElVal
str_length_minus1_suffix	见 103，len=1，str_length_minus1_suffix 的值等于 synElVal
pv_address_prefix	见 103，len= Ceil(Log(MaxValPvAddress+1))-1，pv_address_prefix 的值等于 synElVal
pv_address_infix	见表 103，len=1，pv_address_infix 的值等于 synElVal
pv_address_suffix	见表 103，len=1，pv_address_suffix 的值等于 synElVal
isc_fopixel_y	见表 103，len=BitDepth，isc_fopixel_y 的值等于 synElVal
isc_fopixel_cb	见表 103，len=BitDepth，isc_fopixel_cb 的值等于 synElVal
isc_fopixel_cr	见表 103，len=BitDepth，isc_fopixel_cr 的值等于 synElVal
nn_filter_lcu_enable_flag	见 7.3.4.4，nn_filter_lcu_enable_flag 的值等于 synElVal
nn_filter_lcu_set_index	见 7.3.4.2maxVal = NumOfNnFilter-1，nn_filter_lcu_set_index 的值等于 synElVal

T/AI 109.8-2025

7.3.4.2 采用截断一元码的反二值化方法

由二元符号串根据表 81 得到 synElVal 的值。

表81 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
binIndex	0	1	2	3	4	5	…	maxVal-1

7.3.4.3 采用一元码的反二值化方法

由二元符号串根据表 82 得到 synElVal 的值。

表82 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
…						
binIndex	0	1	2	3	4	5

7.3.4.4 采用标记位的反二值化方法

由二元符号串根据表 83 得到 synElVal 的值。

表83 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
binIndex	0

T/AI 109.8-2025

7.3.4.5 lcu_qp_delta 的反二值化方法

首先由二元符号串根据 7.3.4.3 得到 synElVal 的值，然后根据以下方法得到 lcu_qp_delta 的值：
——如果 synElVal 的值能被 2 整除，则 cu_qp_delta = -(synElVal / 2)；
——否则，cu_qp_delta = (synElVal+1) / 2。

7.3.4.6 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 84 得到 sao_merge_type_index 的值。

表84 sao_merge_type_index 的值与二元符号串的关系

SaoMergeLeftAvai值	SaoMergeUpAvai值	sao_merge_type_index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
binIndex			0	1

7.3.4.7 sao_interval_start_pos 的反二值化方法

由二元符号串查表 85 得到 sao_interval_start_pos 的值。

表85 sao_interval_start_pos 的值与二元符号串的关系

sao_interval_start_pos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
binIndex	0	1	2	3	4

T/AI 109.8-2025

7.3.4.8 sao_interval_delta_pos_minus2 的反二值化方法

由二元符号串查表 86 得到 sao_interval_delta_pos_minus2 的值。

表86 sao_interval_delta_pos_minus2 的值与二元符号串的关系

sao_interval_delta_pos_minus2值	二元符号串							
0	1	0						
1	1	1						
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					
binIndex	0	1	2	3	4	5	6	7

7.3.4.9 sao_edge_offset[compIndex][j]的反二值化方法

由二元符号串查表 87 得到 sao_edge_offset[compIndex][0]或 sao_edge_offset[compIndex][3]的值；由二元符号串查表 88 得到 sao_edge_offset[compIndex][1]或 sao_edge_offset[compIndex][2]的值。

表87 sao_edge_offset[compIndex][0]和 sao_edge_offset[compIndex][3]的值与二元符号串的关系

sao_edge_offset[compIndex][0]	sao_edge_offset[compIndex][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
binIndex		0	1	2	3	4	5	6

T/AI 109.8-2025

表88 sao_edge_offset[compIndex][1]和sao_edge_offset[compIndex][2]的值与二元符号串的关系

sao_edge_offset[compIndex][1]	sao_edge_offset[compIndex][2]	二元符号串
0	0	1
1	-1	0
binIndex		0

7.3.4.10 sao_edge_type 的反二值化方法

由二元符号串查表 89 得到 sao_edge_type 的值。

表89 sao_edge_type 的值与二元符号串的关系

sao_edge_type值	二元符号串	
0	0	0
1	1	0
2	0	1
3	1	1
binIndex	0	1

7.3.4.11 umve_step_index 的反二值化方法

如果 PictureUmveStepSetIndex 的值为 1，由二元符号串查表 90 得到 umve_step_index 的值。

表90 umve_step_index 的值与二元符号串的关系

umve_step_index值	二元符号串			
0	0	0	0	
1	0	0	1	
2	0	1	1	
3	0	1	0	
4	1	0		
5	1	1	0	
6	1	1	1	0
7	1	1	1	1
binIndex	0	1	2	3

7.3.4.12 sawp_pred_mode0_index 和 sawp_pred_mode1_index 的反二值化方法

由二元符号串查表 91 得到 sawp_pred_mode0_index 或 sawp_pred_mode1_index 的值。

T/AI 109.8-2025

表91 sawp_pred_mode0_index 或 sawp_pred_mode1_index 的值与二元符号串的关系

sawp_pred_mode0_index或sawp_pred_mode1_index值	二元符号串					
0	1	0				
1	1	1				
2	0	0	0	0	0	0
3	0	0	0	0	0	1
4	0	1	0	0	0	
5	0	0	0	0	1	0
6	0	0	0	0	1	1
7	0	1	0	0	1	
8	0	0	0	1	0	0
9	0	0	0	1	0	1
10	0	1	0	1	0	
11	0	0	0	1	1	0
12	0	0	0	1	1	1
13	0	1	0	1	1	
14	0	0	1	0	0	0
15	0	0	1	0	0	1
16	0	1	1	0	0	
17	0	0	1	0	1	0
18	0	0	1	0	1	1
19	0	1	1	0	1	
20	0	0	1	1	0	0
21	0	0	1	1	0	1
22	0	1	1	1	0	
23	0	0	1	1	1	0
24	0	0	1	1	1	1
25	0	1	1	1	1	
binIndex	0	1	2	3	4	5

T/AI 109.8-2025

7.3.4.13 sawp_pred_mode1_index 的反二值化方法

由二元符号串查表 92 得到 sawp_pred_mode1_index 的值。

表92 sawp_pred_mode1_index 的值与二元符号串的关系

sawp_pred_mode1_index值	二元符号串					
0	1					
1	0	0	0	0	0	0
2	0	0	0	0	0	1
3	0	1	0	0	0	

表 92 sawp_pred_mode1_index 的值与二元符号串的关系（续）

sawp_pred_mode1_index值	二元符号串					
4	0	0	0	0	1	0
5	0	0	0	0	1	1
6	0	1	0	0	1	
7	0	0	0	1	0	0
8	0	0	0	1	0	1
9	0	1	0	1	0	
10	0	0	0	1	1	0
11	0	0	0	1	1	1
12	0	1	0	1	1	
13	0	0	1	0	0	0
14	0	0	1	0	0	1
15	0	1	1	0	0	
16	0	0	1	0	1	0
17	0	0	1	0	1	1
18	0	1	1	0	1	
19	0	0	1	1	0	0
20	0	0	1	1	0	1
21	0	1	1	1	0	
22	0	0	1	1	1	0
23	0	0	1	1	1	1
24	0	1	1	1	1	
binIndex	0	1	2	3	4	5

7.3.4.14 umve_dir_index、awp_mvr_cand_dir0 、awp_mvr_cand_dir1、affine_umve_dir_index0 和 affine_umve_dir_index1 的反二值化方法

由二元符号串查表 93 得到 synElVal 的值。umve_dir_index、awp_mvr_cand_dir0、awp_mvr_cand_dir1、awp_mvr_cand_dir1、affine_umve_dir_index0 和 affine_umve_dir_index1 的值等于 synElVal。

表93 synElVal 与二元符号串的关系

synElVal值	二元符号串	
0	0	0
1	0	1
2	1	0
3	1	1
binIndex	0	1

T/AI 109.8-2025

7.3.4.15 awp_index 和 sawp_index 的反二值化方法

第 1 步，如果当前图像是 B 图像或 SawpFlag 的值等于 1，maxValue 的值为 56；否则，根据当前编码单元的宽度 W 和高度 H 查表 94 确定 maxValue 的值。

表94 W 和 H 的值与 maxValue 的关系

W	H	maxValue
8	8	6
	16	12
	32	22
	64	22
16	8	12
	16	18
	32	24
	64	24
32	8	21
	16	25
	32	27
	64	27
64	8	22
	16	24
	32	27
	64	28

第 2 步，计算 stage、num0、num1 和 thd 的值。

```
stage = Ceil(Log(maxValue)) - 1
num0 = 1 << stage
num1 = 1 << (stage + 1)
thd = (num0 << 1) - maxValue
```

第 3 步，由二元符号串查表 95 得到 synElVal 的值。其中，0～(thd-1)的二元符号串为值为 0～thd-1 对应的长度为 stage 的定长码，thd～(maxValue-1)的二元符号串为值为(thd << 1)～num1 对应的长度为(stage+1)的定长码。

表95 synElVal 的值与二元符号串的关系

synElVal值	二元符号串			
0	0	...	0	
...	...	...	...	
thd - 1	...	...	...	
thd	...	...	...	0
thd + 1	...	...	...	1
...	...	...	...	...

T/AI 109.8-2025

表 95 synElVal 的值与二元符号串的关系（续）

synElVal值	二元符号串			
maxVlaue - 2	1	...	1	0
maxVlaue - 1	1	...	1	1
binIndex	0	...	stage - 1	stage

第 4 步，awp_index 和 sawp_index 的值等于 synElVal。

7.3.4.16 inter_pred_ref_mode_index 的反二值化方法

由二元符号串查表 96 得到 inter_pred_ref_mode_index 的值。

表96 inter_pred_ref_mode_index 的值与二元符号串的关系

inter_pred_ref_mode_index值	二元符号串	
0	0	0
1	0	1
2	1	
binIndex	0	1

7.3.4.17 mv_diff_x_abs_l0、mv_diff_x_abs_l0_affine、mv_diff_x_abs_l1、mv_diff_x_abs_l1_affine、mv_diff_y_abs_l0、mv_diff_y_abs_l0_affine、mv_diff_y_abs_l1 和 mv_diff_y_abs_l1_affine 的的反二值化方法

由二元符号串查表 97 得到 synElVal 的值。表 97 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 76）；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_l0、mv_diff_x_abs_l0_affine、mv_diff_x_abs_l1、mv_diff_x_abs_l1_affine、mv_diff_y_abs_l0、mv_diff_y_abs_l0_affine、mv_diff_y_abs_l1 或 mv_diff_y_abs_l1_affine 的值等于 synElVal。

表97 synElVal 与二元符号串的关系

synElVal值	二元符号串										
0	0										
1	1	0									
2	1	1	0								
3	1	1	1	0	1						
4	1	1	1	1	1						
5	1	1	1	0	0	1	0				
6	1	1	1	1	0	1	0				
7	1	1	1	0	0	1	1				

T/AI 109.8-2025

表 97 synElVal 与二元符号串的关系（续）

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

7.3.4.18 mv_diff_x_abs_bv、mv_diff_y_abs_bv 和 isc_sv_y_abs[i]的反二值化方法

由二元符号串查表 98 得到 synElVal 的值。表 98 中，如果 synElVal 的值大于或等于 5 且小于 9，二元符号的前 6 位是‘111110’，后续位是(synElVal-5)对应的长度为 2 的定长码（见表 103）。如果 synElVal 的值大于或等于 9 且小于 17，二元符号的前 7 位是‘1111110’，后续位是(synElVal-9)对应的长度为 3 的定长码（见表 103）。如果 synElVal 的值大于或等于 17 且 synElVal 的值为奇数，二元符号串的前 8 位是‘11111110’，后续位是(synElVal-17)/2 对应的 2 阶指数哥伦布码（见表 76）；如果 synElVal 的值大于 17 且 synElVal 的值为偶数，二元符号串的前 8 位是‘11111111’，后续位是(synElVal-17)/2 对应的 2 阶指数哥伦布码。

mv_diff_x_abs_bv、mv_diff_y_abs_bv 或 isc_sv_y_abs[i]的值等于 synElVal。

表98 synElVal 与二元符号串的关系

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

T/AI 109.8-2025

表 98 synElVal 与二元符号串的关系（续）

synElVal值	二元符号串										
15	1	1	1	1	1	1	0	1	1	0	
16	1	1	1	1	1	1	0	1	1	1	
17	1	1	1	1	1	1	1	0	1	0	0
18	1	1	1	1	1	1	1	1	1	0	0
19	1	1	1	1	1	1	1	0	1	0	1
20	1	1	1	1	1	1	1	1	1	0	1
…											
binIndex	0	1	2	3	4	5	6	7	8	9	10

7.3.4.19 intra_luma_pred_mode_index 的反二值化方法

由二元符号串查表 99 得到 intra_luma_pred_mode_index 的值。

表99 intra_luma_pred_mode_index 的值与二元符号串的关系

intra_luma_pred_mode_index值	二元符号串					
0	1	0				
1	1	1				
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
33	0	1	1	1	1	1
binIndex	0	1	2	3	4	5

7.3.4.20 scan_region_x 或 scan_region_y 的反二值化方法

令当前块的宽度和高度是 blockWidth 和 blockHeight。如果要确定的语法元素是 scan_region_x，M 等于 blockWidth；如果要确定的语法元素是 scan_region_y，M 等于 blockHeight。

按以下方法得到 scan_region_x 或 scan_region_y 的值 synElVal（包括第一部分的值 synElVal1 和第二部分的值 synElVal2）。其中 val1[binIndex]和 val2[binIndex]分别是第一部分和第二部分的 binIndex 对应的二元符号值，group_index 和 min_in_group 的规定见附录 L。

```
count = 0
for (binIndex=0; binIndex<group_index[Min(M, 32)-1]; binIndex++) {
    if (val1[binIndex] == 0)
        break
    count++
}
```

T/AI 109.8-2025

```
if (count <= 3) {
    synElVal1 = count
    synElVal = synElVal1
}
else {
    synElVal1 = min_in_group[count]
    synElVal2 = 0
    binIndex = 0
    for (i=((count-2)>>1)-1; i>=0; i--) {
        synElVal_2 += val2[binIndex] << i
        binIndex++
    }
    synElVal = synElVal1 + synElVal2
}
```

7.3.4.21 coeff_abs_level_remaining 的反二值化方法

coeff_abs_level_remaining 的反二值化方法如下：

——如果 EscapeDataPresent 等于 1 且满足下面任一条件，使用 0 阶指数哥伦布码由表 100 得到 synElVal 的值：

- PictureTsEnableFlag 的值为 1 且 CoeffAbsLevelGt8Flag 的值为 1；
- PictureTsEnableFlag 的值为 0。

——否则，如果 CoeffAbsLevelGt2Flag 的值为 1 且 CoeffAbsLevelGt4Flag 的值为 0，使用长度为 len=1 的定长码由表 103 得到 synElVal 的值；

——否则，如果 CoeffAbsLevelGt4Flag 的值为 1 且 CoeffAbsLevelGt8Flag 的值为 0，使用长度为 len=2 的定长码由表 103 得到 synElVal 的值。

表100 synElVal 与二元符号串的关系

synElVal 值	二元符号串							
0	1							
1	0	1	0					
2	0	1	1					
3	0	0	1	0	0			
4	0	0	1	0	1			
5	0	0	1	1	0			
6	0	0	1	1	1			
7	0	0	0	1	0	0	0	
8	0	0	0	1	0	0	1	
9	0	0	0	1	0	1	0	
binIndex	0	1	2	3	4	5	6	…

7.3.4.22 coeff_run 和 coeff_level_minus1 的反二值化方法

T/AI 109.8-2025

由二元符号串查表 101 得到 synElVal 的值。如果 synElVal 的值小于 TH，使用截断一元码且 maxVal 等于 TH；如果 synElVal 的值大于或等于 TH，则二元符号串的前 TH 位均为 0，后续位为(synElVal-TH)对应的 0 阶指数哥伦布码。

表101 coeff_run 的值与二元符号串的关系

synElVal值	二元符号串																				
0	1																				
1	0	1																			
2	0	0	1																		
3	0	0	0	1																	
4	0	0	0	0	1																
5	0	0	0	0	0	1															
6	0	0	0	0	0	0	1														
7	0	0	0	0	0	0	0	1													
8	0	0	0	0	0	0	0	0	1												
9	0	0	0	0	0	0	0	0	0	1											
10	0	0	0	0	0	0	0	0	0	0	1										
11	0	0	0	0	0	0	0	0	0	0	0	1									
12	0	0	0	0	0	0	0	0	0	0	0	0	1								
13	0	0	0	0	0	0	0	0	0	0	0	0	0	1							
14	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1						
15	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1					
16	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1				
17	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0		
18	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1		
19	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0
20	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1
21	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0
22	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1
…																					
binIndex	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21

T/AI 109.8-2025

7.3.4.23 isc_sv_x_abs_minus1[i]的反二值化方法

由二元符号串查表 102 得到 synElVal 的值。表 102 中，二元符号串是 synElVal 的值对应的 5 阶指数哥伦布码（见表 76）。

表102 synElVal 与二元符号串的关系

synElVal 值	二元符号串										
0	1	0	0	0	0	0					
1	1	0	0	0	0	1					
2	1	0	0	0	1	0					
3	1	0	0	0	1	1					
4	1	0	0	1	0	0					
5	1	0	0	1	0	1					
6	1	0	0	1	1	0					
7	1	0	0	1	1	1					
8	1	0	1	0	0	0					
9	1	0	1	0	0	1					
10	1	0	1	0	1	0					
11	1	0	1	0	1	1					
12	1	0	1	1	0	0					
13	1	0	1	1	0	1					
14	1	0	1	1	1	0					
15	1	0	1	1	1	1					
16	1	1	0	0	0	0					
…											
31	1	1	1	1	1	1					
32	0	1	0	0	0	0	0	0			
…											
binIndex	0	1	2	3	4	5	6	7	8	9	…

T/AI 109.8-2025

7.3.4.24 isc_next_remaining_pixel_in_cu[i]的反二值化方法

isc_next_remaining_pixel_in_cu[i]二元符号串由三部分组成。

第 1 步，进行第一部分的反二值化。

a) 计算 maxValPrefix：

```
if (CurRemainingPixelQuarter <= 5)
    maxValPrefix = 1
else if (CurRemainingPixelQuarter <= 21)
    maxValPrefix = 2
else
    maxValPrefix = 3
```

b) 由第一部分和 maxVal=maxValPrefix 查表 81 得到 synElVal 的值 a。如果 a 等于 0，位流中不存在第二部分和第三部分，isc_next_remaining_pixel_in_cu[i]的值为 0，结束反二值化；否则（即 a 大于 0），继续执行后续步骤。

第 2 步，进行第二部分的反二值化。

a) 计算 maxValInfix：

```
if (a == 1) {
    d = 1
    maxValInfix = Min(CurRemainingPixelQuarter - 2, 3)
}
else if (a == 2) {
    d = 5
    maxValInfix = Min(CurRemainingPixelQuarter - 6, 15)
}
else {
    d = 21
    maxValInfix = Min(CurRemainingPixelQuarter - 22, 255)
}
```

b) 令 n=Ceil(Log(maxValInfix + 1))，由第二部分和 len=n-1 查表 103 得到 synElVal 的值 b。如果 len 的值小于 1，二元符号串为空，b 的值为 0。

表103 synElVal 与二元符号串的关系（长度为 len 的定长码）

synElVal 值	二元符号串				
0	0	0	…	0	0
1	0	0	…	0	1
2	0	0	…	1	0
3	0	0	…	1	1
…			…		
$2^{len}-4$	1	1	…	0	0
$2^{len}-3$	1	1	…	1	0
$2^{len}-2$	1	1	…	1	0
$2^{len}-1$	1	1	…	1	1
binIndex	0	1	…	len-2	len-1

第 3 步，进行第三部分的反二值化。

a) 如果 b 小于 2^n-maxValInfix-1，或 maxValInfix 为 0，则 k 等于 0；否则，k 等于 1；

b) 由第三部分和 len=k 查表 103 得到 synElVal 的值 c。如果 len 的值小于 1，二元符号串为空，c 的值为 0。

第 4 步，根据 d、b、c、k、n 和 maxValInfix 计算 isc_next_remaining_pixel_in_cu[i]的值。

```
isc_next_remaining_pixel_in_cu[i] = d + (b << k) + c - (2^n - maxValInfix - 1) * k
```

8 视频解码过程

8.1 NAL 位流解码

本过程的输入为位流，该位流由一系列位流 NAL 单元语法结构组成。

本过程的输出为一系列的 NAL 单元。

T/AI 109.8-2025

在解码过程开始时，解码器把其当前位置初始化为位流的起始位置。然后提取并丢弃每一个 leading_zero_8bits 语法元素（如果存在的话），并相应移动当前位置，直到当前位置紧接的四个字节为 0x00000001。

解码器此时重复执行下述按步骤的过程，对位流中每一个 NAL 单元语法结构进行提取与解码，直到位流中最后一个 NAL 单元也已经解码：

a) 当位流里的紧接的四个字节构成四字节序列 0x00000001，对位流中下一个字节（为 zero_byte 语法元素）进行提取并丢弃时，位流的当前位置设为紧接被丢弃的字节的字节位置；

b) 提取与丢弃位流中下一个三字节序列（为 start_code_prefix_one_3bytes），且位流当前位置设为此紧接被丢弃的 3 字节序列的字节的位置；

c) NumBytesInNALunit 设为自当前字节位置起至满足下述任一条件的位置的最后一个字节，且包括最后一个字节的编号；

1) 一个三字节序列的排列等于 0x000000；

2) 一个三字节序列的排列等于 0x000001；

3) 位流的结束。

d) 该 NumBytesInNALunit 字节从位流中移除，位流的当前位置前移 NumBytesInNALunit 字节。这个字节序列为 nal_unit(NumBytesInNALunit)，并用 NAL 单元解码过程进行解码；

e) 当位流中的当前位置不为位流的结尾，且位流中一个字节不是等于 0x00000001 开始的四字节序列，解码器提取并丢弃每一个 trailing_zero_8bits 语法元素，并相应移动当前位置，直到位流里的当前位置接下的四个字节构成四字节的序列 0x00000001 或已至位流的结尾。

注：位流的结束的判断方法不在本文件中规定。

8.2 序列解码

NAL 单元解码过程的输入为 NAL 单元，输出为封装在 NAL 单元中的 RBSP 语法结构。该过程为从 NAL 单元中提取出 RBSP 语法结构。当 UserPermission 的值等于 0 时，不处理 nal_unit_type 的值为 19 的 NAL 单元。如果 encryption_idc 等于 1，则从 NAL 单元中提取 RBSP 语法结构时需对加密的 RBSP 进行解密处理，得到未加密的 RBSP。该解密过程不在本文件中规定。

对 NAL 单元中的 RBSP 语法结构按照如下的方式进行解码：

——nal_unit_type 的值为 1、2、4、12、14、17 和 18 时 NAL 单元的解码过程，见 8.4；

——nal_unit_type 的值为 1、2、4、12、14、17 和 18 时 NAL 单元的帧内预测过程，见 8.8；

——nal_unit_type 的值为 1 和 14 时 NAL 单元中的的帧间预测过程，见 8.9；

——nal_unit_type 的值为 1、2、4、12、14、17 和 18 时 NAL 单元中的树形编码单元在去块效应滤波前变换系数的解码过程和图像重建过程，见 8.6；

——nal_unit_type 的值为 1、2、4、12、14、17 和 18 时 NAL 单元的重建图像的去块效应滤波过程，见 8.11；

——nal_unit_type 的值为 1、2、4、12、14、17 和 18 时 NAL 单元的重建图像的样值偏移补偿过程，见 8.12；

——nal_unit_type 的值为 1、2、4、12、14、17 和 18 时 NAL 单元的重建图像的自适应修正滤波过程，见 8.13；

——nal_unit_type 的值为 3、7 和 8 时 NAL 单元中的 RBSP 分别为图像头、序列参数集和图像参数集。有效的图像头、序列参数集和图像参数集用于其他 NAL 单元的解码过程。

注：UserPermission 的值可由外部装置输入确定，否则 UserPermission 的值等于 0。

序列解码过程如下：

T/AI 109.8-2025

第 1 步，将 InitialFlag 的值初始化为 0。

第 2 步，解码序列参数集 NAL 单元：

a) 步骤 2.1，如果 InitialFlag 的值为 1，或者下一个图像不是 IDR 图像，直接执行步骤 2.2；否则：
   1) 将 DOIPrev[i]和 DOICycleCnt[i]的值均初始化为 0（i=0～NumOfLayers-1）；
   2) 将 InitialFlag 的值置为 1；
   3) 将 LibraryBufferEmpty[i]的值初始化为 1（i=0～NumOfLayers-1）；
   4) 清空解码图像缓冲区。
b) 步骤 2.2，根据序列参数集中得到的 WeightQuantEnableFlag 和 LoadSeqWeightQuantDataFlag 的值，按附录 I 或表 15（见 6.1.5）确定序列级的 4×4 加权量化矩阵 $WeightQuantMatrix_{4x4}$ 和 8×8 加权量化矩阵 $WeightQuantMatrix_{8x8}$。

第 3 步，解码图像参数集 NAL 单元，并根据 8.3.7 确定 4×4 加权量化矩阵 $WeightQuantMatrix_{4x4}$ 和 8×8 加权量化矩阵 $WeightQuantMatrix_{8x8}$。

第 4 步，依次解码编码图像，直到遇到序列参数集 NAL 单元、图像参数集 NAL 单元、编码视频序列结尾 NAL 单元或流结尾 NAL 单元。

第 5 步，执行以下操作：

a) 如果遇到序列参数集 NAL 单元，继续执行第 2 步；
b) 否则，如果遇到图像参数集 NAL 单元，继续执行第 3 步；
c) 否则，如果遇到编码视频序列结尾 NAL 单元或流结尾 NAL 单元，则按照 POI 从小到大的顺序依次输出解码图像缓冲区中每幅“未输出”的显示图像和显示知识图像，直到所有显示图像和显示知识图像均已输出。同时将 InitialFlag 的值置为 0。

8.3 图像解码

8.3.1 概述

记 curLayerId 等于当前图像的 LayerId。图像的解码过程如下：

——解码图像头（见 8.3.2）；
——导出参考图像队列信息（见 8.3.3）；
——更新解码图像缓冲区（见 8.3.4）；
——构建参考图像队列（见 8.3.5）；
——解码当前图像的各个片（见 8.4），得到补偿后样本；
——如果 DeblockingFilterDisableFlag 的值为0，对补偿后样本进行去块效应滤波操作（见 8.11），得到滤波后样本；如果 DeblockingFilterDisableFlag 的值为 1，直接将补偿后样本作为滤波后样本；
——如果 SaoEnableFlag 的值为 1，对滤波后样本进行样值偏移补偿操作（见 8.12.1），得到偏移后样本；如果 SaoEnableFlag 的值为 0 且 EsaoEnableFlag 的值为 1，对滤波后样本进行增强样值偏移补偿（见 8.12.2），得到偏移后样本；如果 SaoEnableFlag 和 EsaoEnableFlag 的值均为 0，直接将滤波后样本作为偏移后样本；
——如果 CcsaoEnableFlag 的值为 1，对偏移后样本进行跨分量样值偏移补偿操作（见 8.12.3），得到跨分量偏移后样本；如果 CcsaoEnableFlag 的值为 0，直接将偏移后样本作为跨分量偏移后样本；
——如果 AlfEnableFlag 的值为 1，对跨分量偏移后样本进行自适应修正滤波操作（见 8.13），得

T/AI 109.8-2025

到重建样本；如果 AlfEnableFlag 的值为 0，直接将跨分量偏移后样本作为重建样本；

——重建样本构成当前图像的解码图像；

——将当前图像的解码图像移入解码图像缓冲区，置该图像为“未输出”和“被参考”。如果 LibraryPictureModeIndex 为 0 或为 1，且 LibraryStreamFlag 和 LibraryBufferEmpty[curLayerId]都为 1，则置 LibraryBufferEmpty[curLayerId]为 0；

——输出解码显示图像（见 8.3.6）；

——存储当前解码图像的运动信息（见 8.6.7.10）和图像级量化参数；

——如果 PrivacyModeIndex 的值为 2 或 3，则存储当前解码图像的权限信息（见 8.6.7.11）。

8.3.2 图像头解码

记 curLayerId 等于当前图像的 LayerId。当前图像的图像头解码过程如下：

——解析图像类型：

a) 如果当前图像头 RBSP 中 PictureCodingType 等于 1，当前图像是 I 图像；

b) 如果当前图像头 RBSP 中 PictureCodingType 等于 2，当前图像是 P 图像；

c) 如果当前图像头 RBSP 中 PictureCodingType 等于 3，当前图像是 B 图像。

——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 FixedPictureQpFlag；

——如果当前图像的 LayerId 大于 0 且 SpsIndependentLayerFlag[curLayerId]等于 0，并且在当前访问单元中有另一个图片 picA 的 LayerId 等于 RefLayerId[curLayerId],当前图像的 POI 应等于 picA 的 POI；

——如果 LibraryStreamFlag 的值等于 0 且 DOI 小于 DOIPrev[curLayerId]，则解码图像缓冲区中的所有 LayerId 等于 curLayerId 的显示图像和显示知识图像的解码顺序索引都减去 256，DOICycleCnt[curLayerId] 加 1。如果 LibraryStreamFlag 的值等于 1，且 LibraryPictureModeIndex 的值等于 1 且 DOI 小于 DOIPrev[curLayerId]，则解码图像缓冲区中的所有 LayerId 等于 curLayerId 的显示图像和显示知识图像的解码顺序索引都减去 256，DOICycleCnt[curLayerId]加 1；

——如果 LibraryStreamFlag 的值等于 0，或者 LibraryStreamFlag 的值等于 1 且 LibraryPictureModeIndex 的值等于 1，则 DOIPrev[curLayerId]等于 DOI；

——POI 等于 DOI+PictureOutputDelay-OutputReorderDelay+256×DOICycleCnt[curLayerId]。解码图像缓冲区中 LayerId 等于 curLayerId 的显示图像或显示知识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；

——距离索引 DistanceIndex 等于 POI 乘 2。

符合本文件的位流应满足以下要求：如果当前图像是显示图像或显示知识图像，则当前图像的 DOI 不应与解码图像缓冲区中 LayerId 等于 curLayerId 的任意显示图像或显示知识图像的 DOI 相同，且当前图像的 POI 不应与解码图像缓冲区中 LayerId 等于 curLayerId 的任意显示图像或显示知识图像的 POI 相同。

8.3.3 导出参考图像队列信息

P 图像解码时只应使用参考图像队列 0。B 图像解码时可同时使用参考图像队列 0 和参考图像队列 1。

记 curLayerId 等于当前图像的 LayerId。导出当前图像的参考图像队列 i（i 等于 0 或 1）的信息 RefPicDoiList[curLayerId][i]的方法如下：

T/AI 109.8-2025

——初始化 LibraryPictureExistFlag 和 j 为 0，doiBase 为当前图像的 DOI；

——进行 NumOfRefPic[curLayerId][i][RplsIndex[i]]次以下操作：

a) 如果 LibraryIndexFlag[curLayerId][i][RplsIndex[i]][j] 的值等于 1，则令 RefPicDoiList[curLayerId][i][j] 等于 ReferencedLibraryPictureIndex[curLayerId][i][RplsIndex[i]][j]，LibraryPictureExistFlag 等于 1，距离索引 DistanceIndex 等于当前图像的 POI 减 1 并乘 2，j 等于 j+1；

b) 如果 LibraryIndexFlag[curLayerId][i][RplsIndex[i]][j] 的值等于 0，则令 RefPicDoiList[curLayerId][i][j] 等于 doiBase-DeltaDoi[curLayerId][i][RplsIndex[i]][j]，并令 doiBase 等于 RefPicDoiList[curLayerId][i][j]。如果 RefPicDoiList[curLayerId][i][j]等于当前图像的 DOI，则距离索引 DistanceIndex 等于当前图像的 POI 减 1 并乘 2，否则，距离索引 DistanceIndex 等于 DOI 为 RefPicDoiList[curLayerId][i][j]的图像的 POI 乘 2，j 等于 j+1。

符合本文件的位流应满足以下要求：

——对于参考图像队列 i（i 等于 0 或 1），NumRefActive[i]的值应小于或等于 NumOfRefPic[curLayerId][i][RplsIndex[i]]的值；

——在参考图像队列 i（i 等于 0 或 1）中，对于任意 k 和 m（k 不等于 m），如果 LibraryIndexFlag[curLayerId][i][RplsIndex[i]][k] 等于 LibraryIndexFlag[curLayerId][i][RplsIndex[i]][m]，则 RefPicDoiList[curLayerId][i][k]不应等于 RefPicDoiList[curLayerId][i][m]；

——在参考图像队列 i(i 等于 0 或 1)中，对于任意 k，如果 SpsIndependentLayerFlag[curLayerId]等于 1，则 RefPicDoiList[curLayerId][i][k]不应等于当前图像的 DOI；

——如果当前图像是 RL 图像（记为 RL0）：

a) 对于 j 小于 NumRefActive[i] 的所有 LibraryIndexFlag[curLayerId][i][RplsIndex[i]][j]均应为 1；

b) 如果位流中在 RL0 对应的序列参数集之前存在另一幅 RL 图像（记为 RL1），且 RL1 是位流中紧跟另一个序列参数集之后的 RL 图像中与 RL0 距离最近的 RL 图像，则应满足以下任一条件：

- RL0 所参考的每一幅知识图像均为 RL1 所参考的知识图像之一；
- RL0 所参考的知识图像最多只有 NumOfUpdatedLibrary 幅不同于 RL1 所参考的知识图像，且 RL0 与 RL1 的解码时间间隔至少为 MinLibraryUpdatePeriod 秒。

8.3.4 更新解码图像缓冲区

记 curLayerId 等于当前图像的 LayerId。完成当前图像的图像头解码及参考图像队列构建之后，在解码当前图像之前应先更新解码图像缓冲区。如果当前图像是知识图像且 LibraryPictureModeIndex 为 2，则不应更新解码图像缓冲区。

如果当前图像是显示图像，执行以下操作：

a) 对解码图像缓冲区中所有与当前图像具有相同 LayerId 的显示图像，如果该图像既不在参考图像队列 0 中又不在参考图像队列 1 中，则将该图像标记为“不被参考”；否则，如果该图像在参考图像队列 0 或参考图像队列 1 中，将该图像标记为“被参考”。如果一副图像被标记为“不被参考”，则后续该图像不应再被标记为“被参考”；

T/AI 109.8-2025

b) 对解码图像缓冲区中与当前图像具有相同 DOI 的显示图像，如果该图像在参考图像队列 0 或参考图像队列 1 中，将该图像标记为“被参考”；否则，维持原标记不变。

对解码图像缓冲区依次进行以下操作：

——移出图像：

a) 如果当前图像是显示图像,移出解码图像缓冲区中所有被标记为“不被参考”且“已输出”的显示图像；

b) 如果解码图像缓冲区中存在与当前图像的 LayerId 相同的知识图像，且当前图像的 LibraryPictureExistFlag 等于 1，且当前图像所参考的知识图像的知识图像索引与解码图像缓冲区中的知识图像的知识图像索引不相同，或者当前图像是知识图像，则移出解码图像缓冲区中与当前图像的 LayerId 相同的知识图像并置 LibraryBufferEmpty[curLayerId]为 1。

——移入图像：

如果 LibraryPictureModeIndex 为 2，且 LibraryBufferEmpty[curLayerId]为 1，且当前图像的 LibraryPictureExistFlag 等于 1，则将对应的知识图像从外部移入解码图像缓冲区并置 LibraryBufferEmpty[curLayerId]为 0。

符合本文件的位流应满足以下要求：

——在参考图像队列 i（i 等于 0 或 1）中最前面的 NumRefActive[i]幅参考图像的 temporal_id 应小于或等于当前图像的 temporal_id；

——在参考图像队列 i（i 等于 0 或 1）中最前面的 NumRefActive[i]幅参考图像的 LayerId 应小于或等于当前图像的 LayerId；

——解码过程中，当前解码图像、“未输出”显示图像与知识图像的总数不应超过 MaxDpbSize 的值。

8.3.5 构建参考图像队列

记 curLayerId 等于当前图像的 LayerId。构建当前图像的参考图像队列 i（i 等于 0 或 1）的方法如下：

——初始化 j 为 0；

——进行 NumOfRefPic[curLayerId][i][RplsIndex[i]]次以下操作：

a) 如果 LibraryIndexFlag[curLayerId][i][RplsIndex[i]][j]的值等于 1,将知识图像索引等于 RefPicDoiList[curLayerId][i][j]且 LayerId 等于 curLayerId 的知识图像从解码图像缓冲区移入参考图像队列 i 中第 j 个位置。符合本文件的位流应满足，该知识图像应存在于解码图像缓冲区中；

b) 否则,如果 LibraryIndexFlag[curLayerId][i][RplsIndex[i]][j]的值等于 0:

1) 如果 RefPicDoiList[curLayerId][i][j]等于当前图像的 DOI，将解码图像缓冲区中解码顺序索引等于 RefPicDoiList[curLayerId][i][j] 且 LayerId 等于 RefLayerId[curLayerId]的显示图像移入参考图像队列 i 中第 j 个位置。符合本文件的位流应满足，该显示图像应存在于解码图像缓冲区中。

2) 否则，将解码图像缓冲区中解码顺序索引等于 RefPicDoiList[curLayerId][i][j]且 LayerId 等于 curLayerId 的显示图像移入参考图像队列 i 中第 j 个位置。符合本文件的位流应满足，该显示图像应存在于解码图像缓冲区中；

c) j 等于 j+1。

T/AI 109.8-2025

8.3.6 输出解码图像

执行以下步骤输出解码图像，

——如果当前序列的 LibraryStreamFlag 的值等于 1，且 LibraryPictureModeIndex 的值等于 2，则输出当前解码图像和对应的 LibraryPictureIndex 到外部的知识图像缓冲区；

——否则：

a) 标记“可输出”图像：

- 在解码图像缓冲区中查找被标记为“未输出”且解码顺序索引与图像输出延迟之和小于或等于 DOI 的显示图像，如果存在则标记该图像为“可输出”；
- 在解码图像缓冲区中查找被标记为“未输出”且解码顺序索引与图像输出延迟之和小于或等于 DOI 的显示知识图像，如果存在且 LibraryDisplayFlag 的值等于 1 则标记该图像为“可输出”；
- 在解码图像缓冲区中查找非显示知识图像，如果存在则标记该图像为“不可输出”；
- 在解码图像缓冲区中查找显示知识图像，如果存在且 LibraryDisplayFlag 的值等于 0 则标记该图像为“不可输出”；
- 如果 PictureOutputDelay 的值为 0，则标记当前解码图像为“可输出”。

b) 如果存在“可输出”的图像，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”。

8.3.7 确定加权量化矩阵

8.3.7.1 概述

本条规定确定当前图像的加权量化矩阵的方法。

如果 WeightQuantEnableFlag 的值为 0 或 PictureWeightQuantEnableFlag 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 = 0～$M_1$-1，j = 0～$M_2$-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 PictureWeightQuantEnableFlag 的值为 1，先根据 8.3.7.2 确定 4×4 和 8×8 加权量化矩阵，再根据 8.3.7.3 映射导出 M_1×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。

完成上述操作后，根据 8.3.7.4 修正加权量化矩阵 $WeightQuantMatrix_{M1xM2}$。

8.3.7.2 确定 4×4 和 8×8 加权量化矩阵

如果 picture_weight_quant_data_index 的值为‘00’，则根据 LoadSeqWeightQuantDataFlag 的值，按附录 I 或表 15（见 6.1.5）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 picture_weight_quant_data_index 的值为‘10’，则根据表 15（见 6.1.5）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 picture_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：

第 1 步，确定当前图像的 wqP（wqP 的元素的取值范围应是 1～255）。

T/AI 109.8-2025

a) 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）；

b) 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）；

c) 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）。

第 2 步，根据 WeightQuantModel 确定 8×8 加权量化矩阵 wqM8x8。

a) 如果 WeightQuantModel 的值为 0，则 wqM8x8 的规定为：

```
wqM8x8 = {
{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]}}
```

b) 如果 WeightQuantModel 的值为 1，则 wqM8x8 的规定为：

```
wqM8x8 = {
{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]}}
```

c) 如果 WeightQuantModel 的值为 2，则 wqM8x8 的规定为：

```
wqM8x8 = {
{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]}}
```

第 3 步，根据 WeightQuantModel 确定 4×4 加权量化矩阵 wqM4x4。

a) 如果 WeightQuantModel 的值为 0，则 wqM4x4 的规定为：

T/AI 109.8-2025

```
wqM4x4 = {
{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]}}
```

b) 如果 WeightQuantModel 的值为 1，则 wqM4x4 的规定为：

```
wqM4x4 = {
{wqP[0], wqP[4], wqP[4], wqP[5]}
{wqP[3], wqP[2], wqP[2], wqP[5]}
{wqP[3], wqP[2], wqP[1], wqP[5]}
{wqP[5], wqP[5], wqP[5], wqP[5]}}
```

c) 如果 WeightQuantModel 的值为 2，则 wqM4x4 的规定为：

```
wqM4x4 = {
{wqP[0], wqP[4], wqP[3], wqP[5]}
{wqP[4], wqP[3], wqP[2], wqP[5]}
{wqP[3], wqP[2], wqP[1], wqP[5]}
{wqP[5], wqP[5], wqP[5], wqP[5]}}
```

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。

8.3.7.3 M1×M2 加权量化矩阵的导出方法

如果 M_1 或者 M_2 中至少有一个大于或等于 8，按以下方法确定 $M_1 \times M_2$ 变换块所使用的加权量化矩阵 $WeightQuantMatrix_{M1xM2}$：

a) 根据 8.3.7.2 确定 8×8 变换块所使用的加权量化矩阵 $WeightQuantMatrix_{8x8}$；

b) 确定 $M_1 \times M_2$ 变换块所使用的加权量化矩阵 $WeightQuantMatrix_{M1xM2}$ 为 $WeightQuantMatrix_{M1xM2}[x][y] = WeightQuantMatrix_{8x8}[x>>shift][y>>shift]$，$x = 0 \sim M_1-1$，$y = 0 \sim M_2-1$，其中 shift 等于 $Max(Log(M_1), Log(M_2))-3$。

8.3.7.4 M1×M2 加权量化矩阵的修正方法

对加权量化矩阵 $WeightQuantMatrix_{M1xM2}$ 的元素 $WeightQuantMatrix_{M1xM2}[i][j]$（$i=0 \sim M_1-1$，$j=0 \sim M_2-1$），如果 i 的值大于或等于 32，或 j 的值大于或等于 32，则将 $WeightQuantMatrix_{M1xM2}[i][j]$ 置为 0；否则，$WeightQuantMatrix_{M1xM2}[i][j]$ 的值保持不变。

8.4 片解码

片解码过程如下：

第 1 步，计算 LcuIndex。

```
LcuIndex = LcuRow * PictureWidthInLcu + LcuColumn
```

第 2 步，如果 FixedPictureQpFlag 等于 0，预测量化参数 PreviousQp 等于 PatchQp，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 FixedPatchQpFlag。

第 3 步，以 LCU 为单位计算当前片的边界：

```
PatchLeftInLcu = LcuIndex % PictureWidthInLcu
```

T/AI 109.8-2025

```
PatchRightInLcu = PatchLeftInLcu + PatchSizeInLcu[PatchIndexY][PatchIndexX]->Width
PatchAboveInLcu = LcuIndex / PictureWidthInLcu
PatchBelowInLcu = PatchAboveInLcu + PatchSizeInLcu[PatchIndexY][PatchIndexX]->Height
```

第 4 步，依次解码各个最大编码单元（见 8.5）。

8.5 最大编码单元解码

按片内的光栅扫描顺序依次解码最大编码单元，过程如下：

第 1 步，如果当前最大编码单元是片中当前行的第一个最大编码单元，将历史运动信息表的候选项数 CntHmvp 和历史帧内复制信息表的候选项数 CntIntraHmvp 和历史帧内复制信息表是否为空的标志 HbvpEmptyFlag 的值均初始化为 0；将串复制帧内预测模式的非普通串子模式的历史点预测信息表 PrevPpInfoList[28][2]中所有元素的值初始化为-1，将 PrevPvBufSize 的值初始化为 0，将 LcuRx0 和 LcuRy0 分别设为 x0 和 y0。

第 2 步，解码当前最大编码单元的编码树，依次解码编码树的各编码单元（见 8.6）。在解码编码树的节点时，按照 8.6.6.5.4 更新等值串参考图像缓冲区。完成当前最大编码单元的解码后，按以下方法更新 LcuIndex。更新后，如果 LcuIndex / PictureWidthInLcu 的值大于或等于 PatchBelowInLcu，则结束当前片内所有最大编码单元的解码。

```
if ((LcuIndex + 1) % PictureWidthInLcu>= PatchRightInLcu)
    LcuIndex += (PictureWidthInLcu - PatchSizeInLcu[PatchIndexY][PatchIndexX]->Width +1)
else
    LcuIndex++
```

8.6 编码单元解码

8.6.1 概述

编码单元解码首先初始化帧内预测模式频数表（FimcFrequencyList），ModeFimc[0]和 ModeFimc[1]是频数表中的高频模式。

```
if ((x0 % (1 << LcuSizeInBit) == 0) && (y0 % (1 << (LcuSizeInBit - 1)) == 0)) {
    ModeFimc[0] = 'Intra_Luma_Vertical'
    ModeFimc[1] = 'Intra_Luma_Horizontal'
    for (i=0; i<4; i++)
        FimcFrequencyList[i] = 0
}
```

编码单元解码分为预测样本解码和残差样本解码。

如果 PrivacyModeIndex 的值等于 2 或 3，存储编码单元的权限信息（见 8.6.7.11.1）。

如果 PrivacyModeIndex 的值等于 3 且编码单元 PrivacyLevel 的值等于 1 且 UserPermission 的值等于 0，则当前编码单元跳过预测样本解码和残差样本解码过程，当前编码单元所有重建值等于 PaddingPixel，PaddingPixel 的值为 0。

否则，按照以下分别进行预测样本解码和残差样本解码操作。

预测样本解码包括：确定编码单元类型和相关信息（见 8.6.3）；对预测类型为普通帧内预测的编码单元分别导出其所包含的所有帧内预测块的普通帧内预测模式（见 8.6.6.3）并进行普通帧内预测（见 8.8.1），对预测类型为块复制帧内预测的编码单元分别导出其所包含的块矢量信息（见 8.6.6.4）并进

T/AI 109.8-2025

行块复制帧内预测（见 8.8.2），对预测类型为串复制帧内预测的编码单元分别导出其所包含的串复制帧内预测信息（见 8.6.6.5）并进行串复制帧内预测（见 8.8.3），对预测类型为帧间预测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 8.6.7）并进行帧间预测（见 8.9）；存储编码单元的运动信息（见 8.6.7.10.2），存储编码单元的空域帧内预测信息（见 8.6.6.8）。

残差样本解码包括：确定量化参数（见 8.6.2）；确定编码单元划分为变换块的方式（见 8.6.5）；依次解码各个变换块（见 8.7）。如果当前编码单元的亮度编码块包含多个变换块，则首先将编码单元的残差样值矩阵 ResidueMatrix 中的值置为 0，然后将解码得到的 $M_1 \times M_2$ 大小的亮度变换块放置在编码单元的残差样值矩阵 ResidueMatrix 中以(blockX,blockY)为左上角，宽度为 M_1，高度为 M_2 的区域内。

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 8.10）。

8.6.2 确定量化参数

本条确定量化参数 QP_X（X 为 Y、Cb 或 Cr）。

第 1 步，确定当前编码单元的量化参数 CurrentQp，其取值范围应是 0～(63 + 8 × (BitDepth − 8))。

——如果 FixedQP 为 0 且 CuDeltaQpFlag 为 1，并且当前编码单元左上角的坐标等于(CuQpGroupX,CuQpGroupY)，将预测量化参数 PreviousCuQp 初始化为当前编码单元左边包含亮度分量的编码单元 A 的亮度量化参数 QP_Y。如果编码单元 A“不存在”或编码单元 A 未解码或编码单元 A 的权限大于 0，则 PreviousCuQp 的值等于 PatchQp。将预测量化参数 PreviousCuQpPrivacy 初始化为当前编码单元左边包含亮度分量的编码单元 A 的亮度量化参数 QP_Y。如果编码单元 A“不存在”或编码单元 A 未解码，则 PreviousCuQpPrivacy 的值等于 PatchQp；

——如果IsPrivacyFlag等于0，predCuQp等于PreviousCuQp；否则，predCuQp等于PreviousCuQpPrivacy；

——如果 FixedQP 为 1 或 CuDeltaQpFlag 为 0，则 CurrentQp = ((PreviousQp + LcuQpDelta + 64 + 8 × (BitDepth − 8)) % (64 + 8 × (BitDepth − 8)))；

——否则，如果 FixedQP 为 0 且 CuDeltaQpFlag 为 1 且当前编码单元仅包含色度分量，则 CurrentQp 等于当前编码单元右下角 4×4 子块对应的亮度编码单元的量化参数；

——否则，如果 FixedQP 为 0 且 CuDeltaQpFlag 为 1 且 CuCtp 为 0，则 CurrentQp 等于 predCuQp；

——否则，CurrentQp = (predCuQp + CuDeltaQp + 64 + 8 × (BitDepth − 8)) % (64 + 8 * (BitDepth − 8))；

——如果 IsPrivacyFlag 等于 0，将 PreviousCuQp 的值设置为 CurrentQp；否则，将 PreviousCuQpPrivacy 的值设置为 CurrentQp。

PreviousQp 的值等于上一个解码的最大编码单元的量化参数 QP_Y。如果上一个解码的最大编码单元“不可用”或 FixedQP 等于 1，则 PreviousQp 的值等于 PatchQp。如果上一个解码的最大编码单元与当前编码单元不属于同一个片，则上一个解码的最大编码单元“不可用”。

第 2 步，亮度的量化参数 QP_Y 等于其所在编码单元的 CurrentQp。QP_Y 的取值范围应是 0～[63 + 8 × (BitDepth − 8)]。

第 3 步，确定色度的量化参数 QP_{Cb} 和 QP_{Cr}。

——计算 xCb 和 xCr，其取值范围应是-16～63。其中 IntraChromaPredMode 见 8.6.6.3；

```
xCb = Clip3(-16, 63, CurrentQp − 8 * (BitDepth − 8) + chroma_quant_param_delta_cb)
xCr = Clip3(-16, 63, CurrentQp − 8 * (BitDepth − 8) + chroma_quant_param_delta_cr)
```

T/AI 109.8-2025

```
if (IntraChromaPredMode == 'Intra_Chroma_PMC' || IntraChromaPredMode == 'Intra_Chroma_PMC_LT' ||
    IntraChromaPredMode == 'Intra_Chroma_PMC_T' || IntraChromaPredMode == 'Intra_Chroma_PMC_L' ||
    IntraChromaPredMode == 'Intra_Chroma_EPMC' || IntraChromaPredMode == 'Intra_Chroma_EPMC_LT' ||
    IntraChromaPredMode == 'Intra_Chroma_EPMC_T' || IntraChromaPredMode == 'Intra_Chroma_EPMC_L' ||
    IntraChromaPredMode == 'Intra_Chroma_EPMC2' || IntraChromaPredMode == 'Intra_Chroma_EPMC2_LT' ||
    IntraChromaPredMode == 'Intra_Chroma_EPMC2_T' || IntraChromaPredMode == 'Intra_Chroma_EPMC2_L')
  xCr = Clip3(-16, 63, CurrentQp - 8 * (BitDepth - 8) + chroma_quant_param_delta_cr + 1)
```

——分别以 xCb 和 xCr 为索引查表 104 得到 QP'_{cb} 和 QP'_{cr}，并计算 Cb、Cr 色度编码块的量化参数 QP_{cb} 和 QP_{cr}。

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

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

表104 色度量化参数与 xCb、xCr 的映射关系

xCb、xCr值	QP'_{cb}和QP'_{cr}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

T/AI 109.8-2025

8.6.3 编码单元类型和相关信息

编码单元的分量模式包括‘COMPONENT_Luma’（仅包含亮度编码块）、‘COMPONENT_Chroma’（仅包

含色度编码块）或‘COMPONENT_Luma_Chroma’（可包含亮度编码块和色度编码块）。

编码单元的预测模式限制包括‘PRED_Intra_Only’（可使用普通帧内预测、块复制帧内预测或串复制帧内预测）、‘PRED_Inter_Only’（仅使用帧间预测）或‘PRED_No_Constraint’（可使用普通帧内预测、块复制帧内预测、串复制帧内预测和帧间预测）。

如果 IntraCuFlag 的值为 1，预测划分尺寸（PartSize）、预测划分方式（SplitMode）、预测参考模式（PredRefMode）和编码单元类型（CuType）由 DtSplitFlag、DtSplitDir、DtSplitHqtFlag、DtSplitHadtFlag、DtSplitVqtFlag、DtSplitVadtFlag、块复制帧内模式标志（IbcCuFlag）和串复制帧内模式标志（IscCuFlag）查表 105 得到。

如果 IntraCuFlag 的值为 1 且当前编码单元的分量模式不是‘COMPONENT_Chroma’，帧内亮度预测块数（NumOfIntraPredBlock）由 DtSplitFlag、DtSplitDir、DtSplitHqtFlag、DtSplitHadtFlag、DtSplitVqtFlag、DtSplitVadtFlag、块复制帧内模式标志（IbcCuFlag）和串复制帧内模式标志（IscCuFlag）查表 105 得到。

表105　编码单元类型和相关信息（IntraCuFlag 的值为 1）

DtSplitFlag	DtSplitDir	DtSplitHqtFlag	DtSplitHadtFlag	DtSplitVqtFlag	DtSplitVadtFlag	块复制帧内模式标志	串复制帧内预测标志	预测划分尺寸	预测划分方式	帧内亮度预测块数	预测参考模式	编码单元类型
0	-	-	-	-	-	0	0	SIZE_2Mx2N	I_NO_SPLIT	1	PRED_I	I_2M_2N
1	1	1	-	-	-	-	-	SIZE_2MxhN	HOR_tN	4	PRED_I	I_2M_hN
1	1	0	0	-	-	-	-	SIZE_2MxnU	HOR_UP	2	PRED_I	I_2M_nU
1	1	0	1	-	-	-	-	SIZE_2MxnD	HOR_DOWN	2	PRED_I	I_2M_nD
1	0	-	-	1	-	-	-	SIZE_hMx2N	VER_tN	4	PRED_I	I_hM_2N
1	0	-	-	0	0	-	-	SIZE_nLx2N	VER_LEFT	2	PRED_I	I_nL_2N
1	0	-	-	0	1	-	-	SIZE_nRx2N	VER_RIGHT	2	PRED_I	I_nR_2N
0	-	-	-	-	-	1	0	SIZE_2Mx2N	I_NO_SPLIT	1	PRED_I	IBC_2M_2N
0	-	-	-	-	-	0	1	SIZE_2Mx2N	I_NO_SPLIT	1	PRED_I	ISC_2M_2N

如果 IntraCuFlag 的值为 0，预测划分尺寸（PartSize）、预测划分方式（SplitMode）和编码单

T/AI 109.8-2025

元类型（CuType）由图像类型（PictureCodingType）、跳过模式标志（SkipFlag）和直接模式标志（DirectFlag）查表 106 得到。

表106 编码单元类型和相关信息（IntraCuFlag 的值为 0）

图像类型	跳过模式标志	直接模式标志	预测划分尺寸	预测划分方式	编码单元类型
2	1	0	SIZE_2Mx2N	NO_SPLIT	P_Skip
	0	1	SIZE_2Mx2N	NO_SPLIT	P_Direct
	0	0	SIZE_2Mx2N	NO_SPLIT	P_2M_2N
3	1	0	SIZE_2Mx2N	NO_SPLIT	B_Skip
	0	1	SIZE_2Mx2N	NO_SPLIT	B_Direct
	0	0	SIZE_2Mx2N	NO_SPLIT	B_2M_2N

如果编码单元类型是‘P_Skip’、‘P_Direct’、‘B_Skip’或‘B_Direct’，先按 8.6.7.8.2 导出有效角度预测模式数（ValidMvapModeNum），再根据编码单元类型（CuType）、编码单元子类型索引（CuSubTypeIndex）、UMVE 模式标志（UmveFlag）、ETMVP 模式标志（EtmvpFlag）、仿射模式标志（AffineFlag）、AWP 模式标志（AwpFlag）、子块时域运动信息标志（SbTmvpFlag）和预测参考模式索引（InterPredRefModeIndex）查表 107 得到编码单元子类型（CuSubType）和预测参考模式（predRefMode）。

表 107 中，如果 SbTmvpEnableFlag 的值为 1 且当前编码单元的宽度和高度均大于或等于 16，则 SbTmvpFlag 的值为 1；否则，SbTmvpFlag 的值为 0。符合本文件的位流应满足 UmveFlag、EtmvpFlag、AffineFlag 和 AwpFlag 中最多可有一个的值为 1。

表 107 中，如果当前图像是 P 图像：

——如果当前编码单元类型是‘P_Skip’且 CuSubTypeIndex 大于或等于 2：

a) 如果 ValidMvapModeNum 大于 0，且 CuSubTypeIndex 小于 2+ValidMvapModeNum，则编码单元子类型是‘P_Skip_Mvap’；

b) 否则，编码单元子类型是‘P_Skip_Hmvp’。

——如果当前编码单元类型是‘P_Direct’且 CuSubTypeIndex 大于或等于 2：

a) 如果 ValidMvapModeNum 大于 0，且 CuSubTypeIndex 小于 2+ValidMvapModeNum，编码单元子类型是‘P_Direct_Mvap’；

b) 否则，编码单元子类型是‘P_Direct_Hmvp’。

表 107 中，如果当前图像是 B 图像：

——如果当前编码单元类型是‘B_Skip’且 CuSubTypeIndex 大于或等于 4：

a) 如果 ValidMvapModeNum 大于 0，且 CuSubTypeIndex 小于 4+ValidMvapModeNum，则编码单元子类型是‘B_Skip_Mvap’；

b) 否则，编码单元子类型是‘B_Skip_Hmvp’。

——如果当前编码单元类型是‘B_Direct’且 CuSubTypeIndex 大于或等于 4：

a) 如果 ValidMvapModeNum 大于 0，且 CuSubTypeIndex 小于 4+ValidMvapModeNum，则编码单元子类型是‘B_Direct_Mvap’；

b) 否则，编码单元子类型是‘B_Direct_Hmvp’。

T/AI 109.8-2025

表107 编码单元子类型和预测参考模式

编码单元类型	编码单元子类型索引	UMVE 模式标志	ETMVP 模式标志	仿射模式标志	AWP 模式标志	子块时域运动信息标志	预测参考模式索引	编码单元子类型	预测参考模式
P_Skip	0	0	0	0	0	0	-	P_Skip_Temporal	PRED_List0
	0	0	0	0	0	1	-	P_Skip_SbTemporal	PRED_List0
	1	0	0	0	0	-	-	P_Skip_Spatial_list0	PRED_List0
	2 ～ 1 + Max(NumOfMvapCand, NumOfHmvpCand)	0	0	0	0	-	-	P_Skip_Mvap 或 P_Skip_Hmvp	PRED_List0
	-	1	0	0	0	-	-	P_Skip_Umve	PRED_List0
	-	0	1	0	0	-	-	P_Skip_Etmvp	PRED_List0
	-	0	0	1	0	-	-	P_Skip_Affine	PRED_List0
	-	0	0	0	1	-	-	P_Skip_Awp	由 8.6.7.8.7 得到
P_Direct	0	0	0	0	0	0	-	P_Direct_Temporal	PRED_List0

T/AI 109.8-2025

表 107 编码单元子类型和预测参考模式（续）

编码单元类型	编码单元子类型索引	UMVE 模式标志	ETMVP 模式标志	仿射模式标志	AWP 模式标志	子块时域运动信息标志	预测参考模式索引	编码单元子类型	预测参考模式
	0	0	0	0	0	1	-	P_Direct_SbTemporal	PRED_List0
	1	0	0	0	0	-	-	P_Direct_Spatial_List0	PRED_List0
	2 ～ 1 + Max(NumOfMvapCand, NumOfHmvpCand)	0	0	0	0	-	-	P_Direct_Mvap 或 P_Direct_Hmvp	PRED_List0
	-	1	0	0	0	-	-	P_Direct_Umve	PRED_List0
	-	0	1	0	0	-	-	P_Direct_Etmvp	PRED_List0
	-	0	0	1	0	-	-	P_Direct_Affine	PRED_List0
	-	0	0	0	1	-	-	P_Direct_Awp	由 8.6.7.8.7 得到
P_2M_2N	-	-	-	0	-	-	-	P_Inter	PRED_List0
	-	-	-	1	-	-	-	P_Inter_Affine	PRED_List0

T/AI 109.8-2025

表 107 编码单元子类型和预测参考模式（续）

编码单元类型	编码单元子类型索引	UMVE 模式标志	ETMVP 模式标志	仿射模式标志	AWP 模式标志	子块时域运动信息标志	预测参考模式索引	编码单元子类型	预测参考模式
B_Skip	0	0	0	0	0	0	-	B_Skip_Temporal	PRED_List01
	0	0	0	0	0	1	-	B_Skip_SbTemporal	PRED_List01
	1	0	0	0	0	-	-	B_Skip_Spatial_list01	PRED_List01
	2	0	0	0	0	-	-	B_Skip_Spatial_list1	PRED_List1
	3	0	0	0	0	-	-	B_Skip_Spatial_list0	PRED_List0
	4 ～ 3 + Max(NumOfMvapCand, NumOfHmvpCand)	0	0	0	0	-	-	B_Skip_Mvap 或 B_Skip_Hmvp	由 8.6.7.8.4 和 8.6.7.8.7.4 得到
	-	1	0	0	0	-	-	B_Skip_Umve	由 8.6.7.8.5 得到
	-	0	1	0	0	-	-	B_Skip_Etmvp	由 8.6.7.8.10 得到

T/AI 109.8-2025

表 107 编码单元子类型和预测参考模式（续）

编码单元类型	编码单元子类型索引	UMVE 模式标志	ETMVP 模式标志	仿射模式标志	AWP 模式标志	子块时域运动信息标志	预测参考模式索引	编码单元子类型	预测参考模式
	-	0	0	1	0	-	-	B_Skip_Affine	由 8.6.7.8.9 得到
	-	0	0	0	1	-	-	B_Skip_Awp	由 8.6.7.8.7 得到
B_Direct	0	0	0	0	0	0	-	B_Direct_Temporal	PRED_List01
	0	0	0	0	0	1	-	B_Direct_SbTemporal	PRED_List01
	1	0	0	0	0	-	-	B_Direct_Spatial_list01	PRED_List01
	2	0	0	0	0	-	-	B_Direct_Spatial_list1	PRED_List1
	3	0	0	0	0	-	-	B_Direct_Spatial_list0	PRED_List0

T/AI 109.8-2025

表 107 编码单元子类型和预测参考模式（续）

编码单元类型	编码单元子类型索引	UMVE 模式标志	ETMVP 模式标志	仿射模式标志	AWP 模式标志	子块时域运动信息标志	预测参考模式索引	编码单元子类型	预测参考模式
	4 ～ 3 + Max(NumOfMvapCand, NumOfHmvpCand)	0	0	0	0	-	-	B_Direct_Mvap 或 B_Direct_Hmvp	由 8.6.7.8.4 和 8.6.7.8.7.4 得到
	-	1	0	0	0	-	-	B_Direct_Umve	由 8.6.7.8.5 得到
	-	0	1	0	0	-	-	B_Direct_Etmvp	由 8.6.7.8.10 得到
	-	0	0	1	0	-	-	B_Direct_Affine	由 8.6.7.8.9 得到
	-	0	0	0	1	-	-	B_Skip_Awp	由 8.6.7.8.7 得到
B_2M_2N	-	-	-	0	-	-	0	B_Inter	PRED_List0
	-	-	-	0	-	-	1	B_Inter	PRED_List1
	-	-	-	0	-	-	2	B_Inter	PRED_List01
	-	-	-	1	-	-	0	B_Inter_Affine	PRED_List0
	-	-	-	1	-	-	1	B_Inter_Affine	PRED_List1

T/AI 109.8-2025

表 107 编码单元子类型和预测参考模式（续）

编码单元类型	编码单元子类型索引	UMVE 模式标志	ETMVP 模式标志	仿射模式标志	AWP 模式标志	子块时域运动信息标志	预测参考模式索引	编码单元子类型	预测参考模式
	-	-	-	1	-	-	2	B_Inter_Affine	PRED_List01

表 105 和表 106 中的预测划分方式（SplitMode）表示编码单元划分为帧内预测块或帧间预测单元的方式：

——如果帧内预测(包括普通帧内预测、块复制帧内预测和串复制帧内预测)划分方式是‘NO_SPLIT’，1 个 2M×2N 的编码单元划分为 1 个 2M×2N 的亮度预测块、1 个 M×N 的 Cb 预测块和 1 个 M×N 的 Cr 预测块（见图 10）；

——如果普通帧内预测划分方式是‘HOR_tN’，1 个 2M×2N 的编码单元划分为 4 个 2M×0.5N 的亮度预测块、1 个 M×N 的 Cb 预测块和 1 个 M×N 的 Cr 预测块（见图 11）；

——如果普通帧内预测划分方式是‘VER_tN’，1 个 2M×2N 的编码单元划分为 4 个 0.5M×2N 的亮度预测块、1 个 M×N 的 Cb 预测块和 1 个 M×N 的 Cr 预测块（见图 12）；

——如果普通帧内预测划分方式是‘HOR_UP’，1 个 2M×2N 的编码单元划分为 1 个 2M×0.5N 的亮度预测块、1 个 2M×1.5N 的亮度预测块、1 个 M×N 的 Cb 预测块和 1 个 M×N 的 Cr 预测块（见图 13）；

——如果普通帧内预测划分方式是‘HOR_DOWN’，1 个 2M×2N 的编码单元划分为 1 个 2M×1.5N 的亮度预测块、1 个 2M×0.5N 的亮度预测块、1 个 M×N 的 Cb 预测块和 1 个 M×N 的 Cr 预测块（见图 14）；

——如果普通帧内预测划分方式是‘VER_LEFT’，1 个 2M×2N 的编码单元划分为 1 个 0.5M×2N 的亮度预测块、1 个 1.5M×2N 的亮度预测块、1 个 M×N 的 Cb 预测块和 1 个 M×N 的 Cr 预测块（见图 15）；

——如果普通帧内预测划分方式是‘VER_RIGHT’，1 个 2M×2N 的编码单元划分为 1 个 1.5M×2N 的亮度预测块、1 个 0.5M×2N 的亮度预测块、1 个 M×N 的 Cb 预测块和 1 个 M×N 的 Cr 预测块（见图 16）；

——如果帧间预测划分方式是‘NO_SPLIT’，1 个 2M×2N 的编码单元不进行划分（见图 17）。

图 10 到图 16 中矩形里的数字表示各预测块的 PredBlockOrder。图 17 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_2M_2N’、‘I_2M_hN’、‘I_2M_nU’、‘I_2M_nD’、‘I_hM_2N’、‘I_nL_2N’或‘I_nR_2N’，则其预测类型为普通帧内预测，其所包含的预测块的预测类型为普通帧内预测；否则，如果编码单元类型为‘IBC_2M_2N’，则其预测类型为块复制帧内预测，其所包含的预测块的预测类型为块复制帧内预测；否则，如果编码单元类型为‘ISC_2M_2N’，则其预测类型为串复制帧内预测，其所包含的预测块的预测类型为串复制帧内预测；否则该编码单元的编码单元的预测类型为帧间预测，其所包含的预测单元和预测块的预测类型为帧间预测。

T/AI 109.8-2025

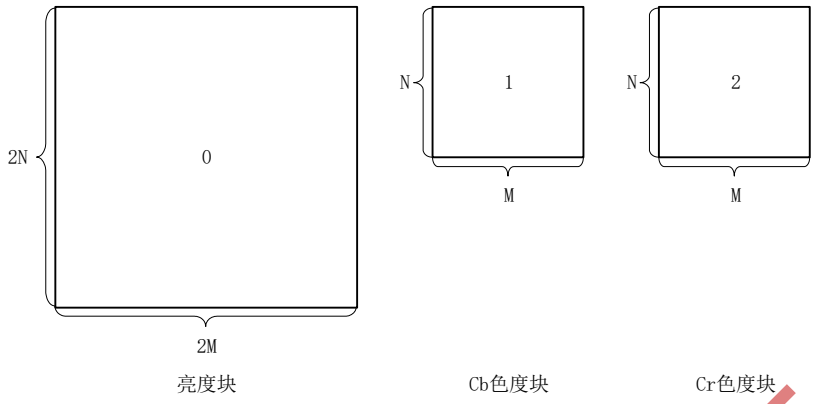


图10　帧内编码单元划分为预测块（预测划分方式‘NO_SPLIT’）

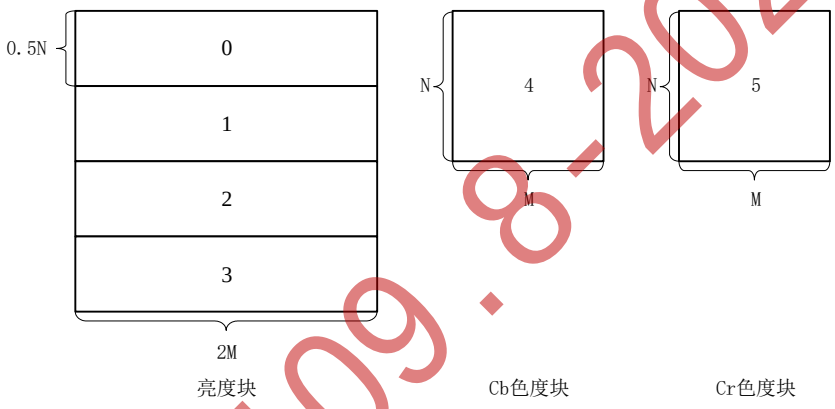


图11　普通帧内编码单元划分为预测块（预测划分方式‘HOR_tN’）

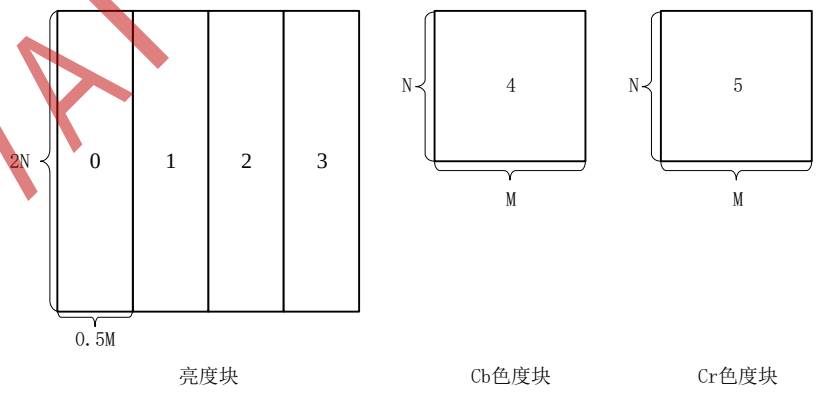


图12　普通帧内编码单元划分为预测块（预测划分方式‘VER_tN’）

T/AI 109.8-2025

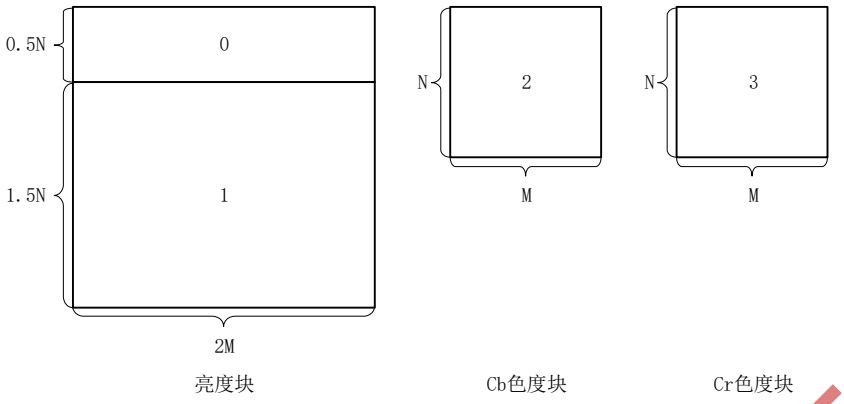


图13　普通帧内编码单元划分为预测块（预测划分方式‘HOR_UP’）

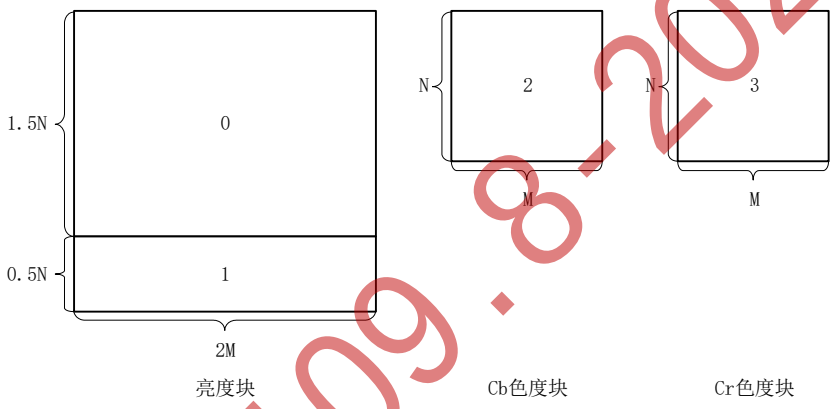


图14　普通帧内编码单元划分为预测块（预测划分方式‘HOR_DOWN’）

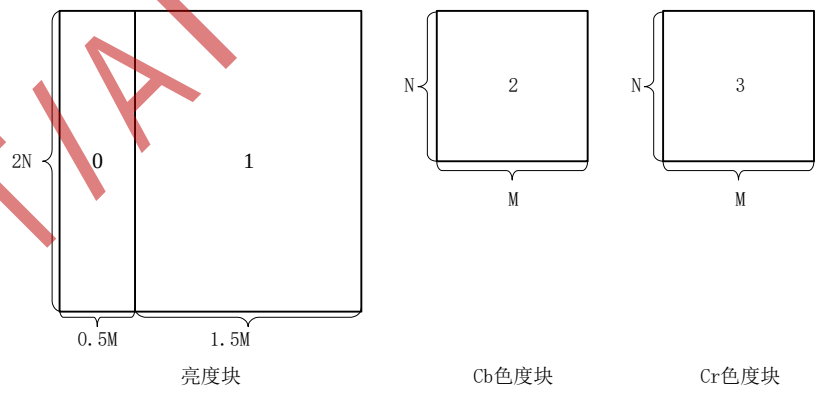


图15　普通帧内编码单元划分为预测块（预测划分方式‘VER_LEFT’）

T/AI 109.8-2025

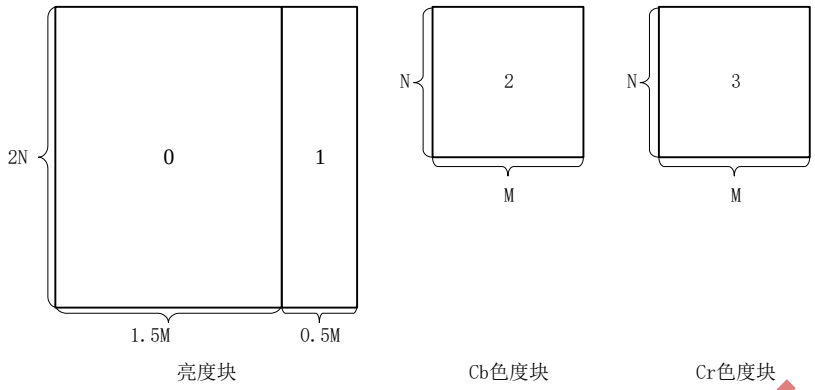


图16　普通帧内编码单元划分为预测块（预测划分方式‘VER_RIGHT’）

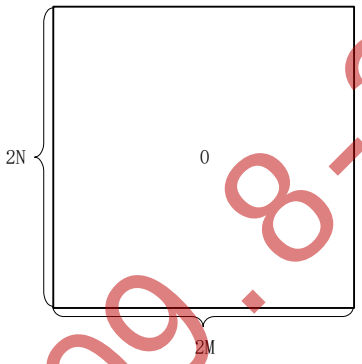


图17　帧间编码单元划分为预测单元（预测划分方式‘NO_SPLIT’）

8.6.4　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 18。

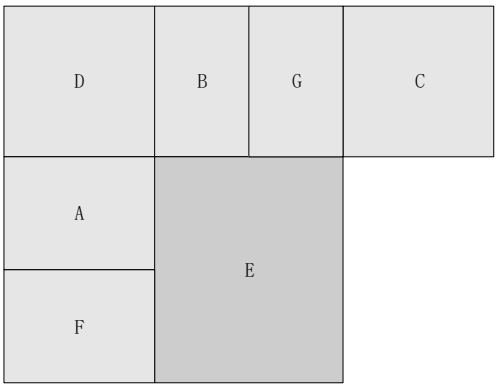


图18　块 E 和相邻块的空间位置关系

T/AI 109.8-2025

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 属于同一片；否则相邻块“不存在”。

如果此块符合以下某一种情况时，此块“不可用”，否则此块“可用”。

——如果块“不存在”；

——如果块尚未解码；

——如果 PicturePrivacyEnableFlag 的值等于 1 且相邻块权限储存单元中权限值大于 IsPrivacyFlag。

如果图像样本所在的块“不可用”，则此样本“不可用”；否则此样本“可用”。

8.6.5 确定编码单元划分为变换块的方式

确定 TransformSplitDirection 的方法如下：

——如果 PbtCuFlag 的值为 1，则 TransformSplitDirection 的值为 1；

——否则，如果当前编码单元是普通帧内编码单元且预测划分方式是‘HOR_tN’、‘HOR_UP’或‘HOR_DOWN’，则 TransformSplitDirection 的值为 2；

——否则，如果当前编码单元是普通帧内编码单元且预测划分方式是‘VER_tN’、‘VER_LEFT’或‘VER_RIGHT’，则 TransformSplitDirection 的值为 3：

——否则，TransformSplitDirection 的值为 0。

当前编码单元划分为变换块的方式如下：

——如果 TransformSplitDirection 的值为 0，则当前编码单元划分为 3 个变换块（1 个矩形亮度变换块和 2 个矩形色度变换块，见图 19），NumOfTransBlocks 的值为 3；

——否则，如果 TransformSplitDirection 的值为 1，则当前编码单元划分为 6 个变换块（4 个矩形亮度变换块和 2 个矩形色度变换块，见图 20），NumOfTransBlocks 的值为 6；

——否则，如果 TransformSplitDirection 的值为 2，当前编码单元划分为 6 个变换块（4 个矩形亮度变换块和 2 个矩形色度变换块，见图 21），NumOfTransBlocks 的值为 6；

——否则，如果 TransformSplitDirection 的值为 3，则当前编码单元划分为 6 个变换块（4 个矩形亮度变换块和 2 个矩形色度变换块，见图 22），NumOfTransBlocks 的值为 6。

图 19 到图 22 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’，则当前 $M_1 \times M_2$ 变换块应按 8.7.3 规定的方法进行反变换：

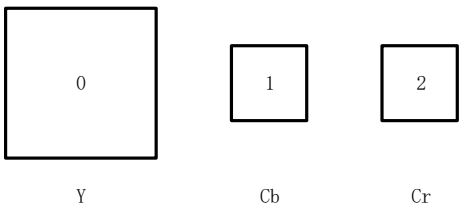


图19 编码单元划分为 1 个矩形亮度变换块和 2 个矩形色度变换块（4:2:0 格式）

T/AI 109.8-2025

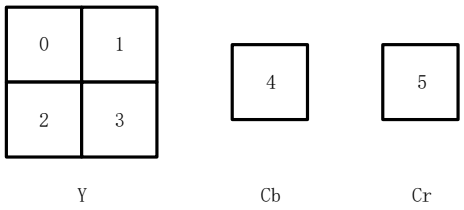


图20　编码单元划分为 4 个矩形亮度变换块和 2 个矩形色度变换块（4:2:0 格式）

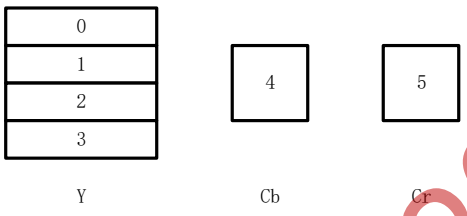


图21　编码单元划分为 4 个矩形亮度变换块和 2 个矩形色度变换块（4:2:0 格式）

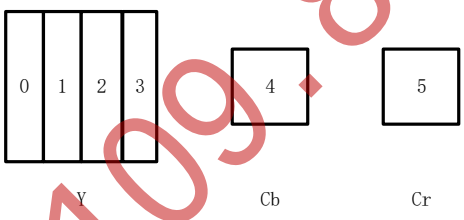


图22　编码单元划分为 4 个矩形亮度变换块和 2 个矩形色度变换块（4:2:0 格式）

T/AI 109.8-2025

8.6.6　帧内预测模式

8.6.6.1　概述

如果编码单元的帧内预测类型是普通帧内预测模式，按 8.6.6.3 导出当前编码单元的每一个预测块的亮度预测模式和色度预测模式。

如果编码单元的帧内预测类型是块复制帧内预测，按 8.6.6.4 导出当前编码单元的块矢量。

如果编码单元的帧内预测类型是串复制帧内预测，按 8.6.6.5 导出当前编码单元的全部串矢量和未匹配像素。

块复制帧内预测和串复制帧内预测可使用历史帧内复制信息表 IntraHmvpCandList 中的历史运动信息，包括位移矢量信息、位置信息、尺寸信息和重复次数。

8.6.6.2　获得相邻块空域帧内预测信息存储单元

当前预测单元的亮度预测块 E 的相邻亮度预测块 X（X 为 A、B、C、D、F 或 G）的空域帧内预测信息存储单元如下（见图 18）：

——相邻块 A 的空域帧内预测信息存储单元是样本（x_0-1, y_0）所对应的空域帧内预测信息存储单元；

——相邻块 B 的空域帧内预测信息存储单元是样本（x_0, y_0-1）所对应的空域帧内预测信息存储单元；

——相邻块 C 的空域帧内预测信息存储单元是样本(x_1+1, y_0-1)所对应的空域帧内预测信息存储单元；

——相邻块 D 的空域帧内预测信息存储单元是样本(x_0-1, y_0-1)所对应的空域帧内预测信息存储单元；

——相邻块 F 的空域帧内预测信息存储单元是样本(x_0-1, y_1)所对应的空域帧内预测信息存储单元；

——相邻块 G 的空域帧内预测信息存储单元是样本(x_1, y_0-1)所对应的空域帧内预测信息存储单元。

8.6.6.3 普通帧内预测模式

8.6.6.3.1 概述

如果当前分量是亮度分量且 SawpFlag 的值为 0，按 8.6.6.3.2 导出当前预测块的亮度预测模式。如果当前分量是亮度分量且 SawpFlag 的值为 1，按 8.6.6.3.3 导出当前预测块的亮度预测模式。按 8.6.6.8 存储空域帧内亮度预测模式。

如果当前分量是色度分量，按 8.6.6.3.4 导出当前预测块的色度预测模式。

8.6.6.3.2 普通亮度帧内预测模式

如果当前预测块 E 是亮度块，第 1 步计算当前预测块预测模式的预测值 predIntraPredMode0 和 predIntraPredMode1 的值。

——如果 PictureFimcEnableFlag 的值为 1，计算如下：

```
predIntraPredMode0 = Min(ModeFimc[0], ModeFimc[1])
predIntraPredMode1 = Max(ModeFimc[0], ModeFimc[1])
```

——如果 PictureFimcEnableFlag 的值为 0，按以下方法计算：

a) 如果左边预测块 A“可用”且为普通帧内预测，则将 A 的 IntraLumaPredMode 赋值给 intraPredModeA；否则 intraPredModeA 等于 0；

b) 如果上边预测块 B“可用”且为普通帧内预测，则将 B 的 IntraLumaPredMode 赋值给 intraPredModeB；否则 intraPredModeB 等于 0；

c) 如果 intraPredModeA 不等于 intraPredModeB，则 predIntraPredMode0 等于 Min(intraPredModeA, intraPredModeB)，predIntraPredMode1 等于 Max(intraPredModeA, intraPredModeB)；

d) 否则，如果 intraPredModeA 等于 0，则 predIntraPredMode0 等于 0，predIntraPredMode1 等于 2；

e) 否则，则 predIntraPredMode0 等于 0，predIntraPredMode1 等于 intraPredModeA。

第 2 步，计算 IntraLumaPredMode 的值。

——如果 IntraLumaPredModeIndex 的值为 0，则 IntraLumaPredMode 等于 predIntraPredMode0；

——否则，如果 IntraLumaPredModeIndex 的值为 1，则 IntraLumaPredMode 等于 predIntraPredMode1；

——否则，IntraLumaPredMode 的值等于(IntraLumaPredModeIndex + LumaEipmIndex × 32)。

第 3 步，检查是否需要修改 IntraLumaPredMode 的值。

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

T/AI 109.8-2025

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

——否则，IntraLumaPredMode 的值不变。

第 4 步，如果 PictureFimcEnableFlag 的值为 1，根据当前预测单元的亮度帧内预测模式 IntraLumaPredMode 按 8.6.6.6 更新帧内预测模式频数表。

第 5 步，根据 IntraLumaPredMode 的值，查表 108 得到亮度预测块的帧内预测模式。

表108　亮度预测块帧内预测模式

IntraLumaPredMode	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
33	Intra_Luma_PCM
34～65	Intra_Luma_Angular

亮度预测块帧内预测模式见图 23。

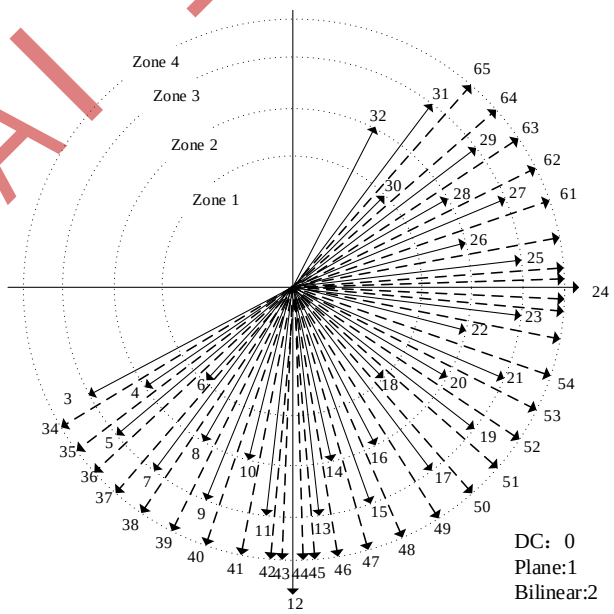


图23　亮度预测块帧内预测模式

T/AI 109.8-2025

8.6.6.3.3 空域角度加权预测亮度帧内预测模式

如果当前预测块 E 是亮度块，令 modAngNum 等于 SawpIndex%8。

第 1 步，计算当前预测块预测模式的预测值 predIntraPredMode[i]的值，i 的取值范围是 0～1。

a) 对 intraPredModeA 执行以下操作：
   1) 如果左边预测块 A“可用”且为普通帧内预测，则将 A 的 IntraLumaPredMode 赋值给 intraPredModeA；否则，intraPredModeA 的值等于-1；
   2) 对 intraPredModeA 进行模式有效性处理；
   3) 如果 intraPredModeA 等于-1，则 intraPredModeA 等于表 109 中 modAngNum 对应的候选模式 0。
b) 对 intraPredModeB 执行以下操作：
   1) 如果上边预测块 B“可用”且为普通帧内预测，则将 B 的 IntraLumaPredMode 赋值给 intraPredModeB；否则，intraPredModeB 的值等于-1；
   2) 对 intraPredModeB 进行模式有效性处理；
   3) 如果 intraPredModeB 等于-1，则 intraPredModeB 等于表 109 中 modAngNum 对应的候选模式 0。
c) 如果 intraPredModeA 等于 intraPredModeB，执行以下操作：
   1) 如果 intraPredModeA 不等于表 109 中 modAngNum 对应的候选模式 0，则 intraPredModeA 等于表 109 中 modAngNum 对应的候选模式 0；
   2) 否则，intraPredModeA 等于表 109 中 modAngNum 对应的候选模式 1。
d) predIntraPredMode[0]等于 Min(intraPredModeA,intraPredModeB)，predIntraPredMode[1]等于 Max(intraPredModeA,intraPredModeB)。

第 2 步，计算 SawpPredMode0 的值。

```
if (SawpPredMode0Index < 2)
    SawpPredMode0 = predIntraPredMode[SawpPredMode0Index]
else {
    SawpPredMode0Index += 3
    SawpPredMode0 = (SawpPredMode0Index + (SawpPredMode0Index >= predIntraPredMode[0] ? 1 : 0) + ((SawpPredMode0Index + 1) >= predIntraPredMode[1] ? 1 : 0))
}
```

第 3 步，计算 SawpPredMode1 的值。

```
if (SawpPredMode0Index < 2) {
    if (SawpPredMode1Index >= SawpPredMode0Index)
        SawpPredMode1Index = SawpPredMode1Index + 1
}
if (SawpPredMode1Index < 2)
    SawpPredMode1 = predIntraPredMode[SawpPredMode1Index]
else {
    SawpPredMode1Index += 3
    SawpPredMode1 = (SawpPredMode1Index + (SawpPredMode1Index >= predIntraPredMode[0] ? 1 : 0) + ((SawpPredMode1Index + 1) >= predIntraPredMode[1] ? 1 : 0))
}
```

T/AI 109.8-2025

本条对 intraPredModeX 进行以下模式有效性处理，其中 X 为 A 或 B：

```
if (intraPredModeX <=4 || (intraPredModeX >= 31 && intraPredModeX <= 34 ) || intraPredModeX == 65)
    intraPredModeX = -1
else if (intraPredModeX > 34) {
    if (intraPredModeX < 44)
        intraPredModeX = intraPredModeX - 30
    else if (intraPredModeX < 58)
        intraPredModeX = intraPredModeX - 33
    else
        intraPredModeX = intraPredModeX - 34
}
```

表109 modAngNum 对应的候选模式

modAngNum	候选模式0	候选模式1
0	30	6
1	27	24
2	24	12
3	21	24
4	18	24
5	15	12
6	12	24
7	9	12

8.6.6.3.4 普通色度帧内预测模式

如果当前预测块 E 是色度编码块，第 1 步计算 isRedundant 的值。

——如果 SawpFlag 等于 1，则 isRedundant 等于 0；

——否则，如果 E 对应的普通亮度预测模式 IntraLumaPredMode 等于 0、2、12 或 24，则 isRedundant 等于 1；

——否则，isRedundant 等于 0。

第 2 步，计算 IntraChromaPredMode 的值。

```
if ((TscpmEnableFlag || PmcEnableFlag) && (IntraChromaPredModeIndex == 1))
    IntraChromaPredMode = 5 + ChromaEipmIndex + 4 * ChromaPmcIndex + 4 * ChromaPmcParamIndex * (1 + PicturePmcParamIndex)
else {
    if ((TscpmEnableFlag || PmcEnableFlag) && (IntraChromaPredModeIndex != 0))
        IntraChromaPredModeIndex = IntraChromaPredModeIndex - 1
    if ( !isRedundant)
        IntraChromaPredMode = IntraChromaPredModeIndex
    else {
        if (IntraLumaPredMode == 0)
```

T/AI 109.8-2025

```
            predIntraChromaPredMode = 1
        else if (IntraLumaPredMode == 2)
            predIntraChromaPredMode = 4
        else if (IntraLumaPredMode == 12)
            predIntraChromaPredMode = 3
        else if (IntraLumaPredMode == 24)
            predIntraChromaPredMode = 2
        if (IntraChromaPredModeIndex == 0)
            IntraChromaPredMode = 0
        else if (IntraChromaPredModeIndex < predIntraChromaPredMode)
            IntraChromaPredMode = IntraChromaPredModeIndex
        else
            IntraChromaPredMode = IntraChromaPredModeIndex + 1
    }
}
```

第 3 步，根据 IntraChromaPredMode 的值，查表 110 得到色度预测块的帧内预测模式。

表110　色度预测块帧内预测模式

IntraChromaPredMode	帧内预测模式
0	Intra_Chroma_DM（IntraLumaPredMode的值不等于33）
0	Intra_Chroma_PCM（IntraLumaPredMode的值等于33）
1	Intra_Chroma_DC
2	Intra_Chroma_Horizontal
3	Intra_Chroma_Vertical
4	Intra_Chroma_Bilinear
5	Intra_Chroma_TSCPM
6	Intra_Chroma_TSCPM_LT
7	Intra_Chroma_TSCPM_L
8	Intra_Chroma_TSCPM_T
9	Intra_Chroma_PMC
10	Intra_Chroma_PMC_LT
11	Intra_Chroma_PMC_L
12	Intra_Chroma_PMC_T
13	Intra_Chroma_EPMC
14	Intra_Chroma_EPMC_LT
15	Intra_Chroma_EPMC_L
16	Intra_Chroma_EPMC_T
17	Intra_Chroma_EPMC2
18	Intra_Chroma_EPMC2_LT
19	Intra_Chroma_EPMC2_L
20	Intra_Chroma_EPMC2_T

T/AI 109.8-2025

8.6.6.4 块复制帧内预测模式

8.6.6.4.1 概述

先按 8.6.6.4.2 得到块矢量的预测值 MvPredXBv 和 MvPredYBv，再按 8.6.6.4.3 解码得到块矢量 bvE。完成当前预测单元的解码后，令 HbvpEmptyFlag 的值为 1，按 8.6.6.4.4 更新历史帧内复制信息表。

符合本文件的位流应满足 8.6.6.4.5 规定的限制。

当前块复制帧内预测单元的宽度和高度分别为 widthCur 和 heightCur，当前预测单元左上角位置的坐标是(xCur,yCur)。历史运动信息 IntraHmvpCandList[X]（X=0～NumOfIntraHmvpCand-1）的位移矢量是 intraMvCandX，位置是(xCandX,yCandX)，尺寸是 sizeCandX，重复次数是 cntCandX。

如果 AbvrIndex 的值为 0，块矢量的基本单位是 1 个整数样本；如果 AbvrIndex 的值为 2，块矢量的基本单位是 4 个整数样本。

8.6.6.4.2 块矢量预测

8.6.6.4.2.1 导出块矢量预测值

本条规定导出块矢量预测值 MvPredXBv 和 MvPredYBv 的方法。

第 1 步，构建历史帧内复制信息类别列表 blockMotionClass[Y]（Y=0～6）。

a) 历史帧内复制信息候选项数 NumAllowedHbvpCand = Min(CntIntraHmvp,NumOfIntraHmvpCand)。如果 NumAllowedHbvpCand 的值小于 2，不进行块矢量预测，立即结束本条的预测过程；否则继续执行下面的步骤；

b) 将 cbvpValidClass[Y]（Y = 0～6）初始化为 0，根据历史帧内复制信息分类方法对 IntraHmvpCandList[X]（X = NumAllowedHbvpCand-1～0）进行分类：

1) 如果 CbvpIndex 等于 0，执行 8.6.6.4.2.2 的方法 a)。如果 cbvpValidClass[0]等于 1，结束分类过程；
2) 如果 CbvpIndex 等于 1，依次执行 8.6.6.4.2.2 的方法 a)和 b)。如果 cbvpValidClass[1]等于 1，结束分类过程；
3) 如果 CbvpIndex 等于 2，依次执行 8.6.6.4.2.2 的方法 a)、b)和 g)。如果 cbvpValidClass[2]等于 1，结束分类过程；
4) 如果 CbvpIndex 等于 3，依次执行 8.6.6.4.2.2 的方法 a)、b)和 f)。如果 cbvpValidClass[3]等于 1，结束分类过程；
5) 如果 CbvpIndex 等于 4，依次执行 8.6.6.4.2.2 的方法 a)、b)和 c)。如果 cbvpValidClass[4]等于 1，结束分类过程；
6) 如果 CbvpIndex 等于 5，依次执行 8.6.6.4.2.2 的方法 a)、b)和 d)。如果 cbvpValidClass[5]等于 1，结束分类过程；
7) 如果 CbvpIndex 等于 6，依次执行 8.6.6.4.2.2 的方法 a)、b)和 e)。如果 cbvpValidClass[6]等于 1，结束分类过程。

第 2 步，令 Y 等于 CbvpIndex，按以下方法导出候选类列表 CbvpCandidateList。

a) 如果 cbvpValidClass[Y]等于 1，CbvpCandidate 等于 blockMotionClass[Y]；

b) 否则，基于空域邻近块，导出空域邻近块块矢量候选 BlockMotionSpatialCandidate，如果 BlockMotionSpatialCandidate 为零矢量，则 BlockMotionSpatialCandidate 不存在。

T/AI 109.8-2025

如果图像类型为 I 图像，且左上角坐标为(xCur-8,yCur+heightCur/2)的 4×4 子块所在的编码单元已重建，且与当前编码单元属于同一个最大编码单元，且是串复制帧内预测模式或块复制帧内预测模式，则 BlockMotionSpatialCandidateA 等于该 4×4 子块对应的空域帧内预测信息存储单元中的串矢量或块矢量。

否则，如果左上角坐标为(xCur-4,yCur+heightCur/2)的 4×4 子块所在的编码单元已重建且是串复制帧内预测模式，以该 4×4 子块中左下角样本的串矢量对应参考子串的第一个样本和最后一个样本在编码单元中的坐标分别为(x1,y1)和(x2,y2)，该参考子串的长度为 StrLen，该参考子串所在编码单元的宽度为 widthRef。如果该参考子串不满足条件(((y1 == y2) || (x1 == x2) || (StrLen==(y2 - y1 + 1) × widthRef)))，则 BlockMotionSpatialCandidateA 等于该 4×4 子块对应的空域帧内预测信息存储单元中的串矢量。

如果图像类型为 I 图像，且左上角坐标为(xCur+widthCur/2,yCur-8)的 4×4 子块所在的编码单元已重建，且与当前编码单元属于同一个最大编码单元，且是串复制帧内预测模式或块复制帧内预测模式，则 BlockMotionSpatialCandidateB 等于以该 4×4 子块对应的空域帧内预测信息存储单元中的串矢量或块矢量。

否则，如果左上角坐标为(xCur+widthCur/2,yCur-4)的 4×4 子块所在的编码单元已重建且是串复制帧内预测模式，以该 4×4 子块中左下角样本的串矢量对应参考子串的第一个样本和最后一个样本在编码单元中的坐标分别为(x1,y1)和(x2,y2)，该参考子串的长度为 StrLen，该参考子串所在编码单元的宽度为 widthRef。若该参考子串不满足条件(((y1 == y2) || (x1 == x2) || (StrLen == (y2 - y1 + 1) × widthRef)))，则 BlockMotionSpatialCandidateB 等于该 4×4 子块对应的空域帧内预测信息存储单元中的串矢量。

c) 按以下步骤基于空域邻近块块矢量候选 BlockMotionSpatialCandidate 导出候选类 CbvpCandidate；

如果 CbvpIndex 为偶数，且 BlockMotionSpatialCandidateA 存在，则 CbvpCandidate 的块矢量等于 BlockMotionSpatialCandidateA；

如果 CbvpIndex 为奇数，且 BlockMotionSpatialCandidateB 存在，则 CbvpCandidate 的块矢量等于 BlockMotionSpatialCandidateB。

d) 如果对应的空域邻近块块矢量候选 BlockMotionSpatialCandidate 也不存在，按以下步骤基于历史帧内复制信息表 IntraHmvpCandList 导出候选类 CbvpCandidate：

如果 CbvpIndex 小于 NumAllowedHbvpCand，则 CbvpCandidate 的块矢量等于 IntraHmvpCandList[NumAllowedHbvpCand%(CbvpIndex+1)]的块矢量；

否则，CbvpCandidate 的块矢量等于 IntraHmvpCandList[0]的块矢量。

第 3 步，MvPredXBv 和 MvPredYBv 分别等于 CbvpCandidate 的 X 分量和 Y 分量。

8.6.6.4.2.2 历史帧内复制信息分类方法

块矢量预测过程中对历史帧内复制信息 IntraHmvpCandList[X]分类的方法如下：

a) 如果 sizeCandX 大于 32，且 cbvpValidClass[0]等于 0，将 IntraHmvpCandList[X]加入 blockMotionClass[0]，并令 cbvpValidClass[0]等于 1；

b) 否则，如果 cntCandX 大于或等于 3，且 cbvpValidClass[1]等于 0，将 IntraHmvpCandList[X]加入 blockMotionClass[1]，并令 cbvpValidClass[1]等于 1；

T/AI 109.8-2025

c) 否则，如果 xCandX 小于 xCur，且 yCandX 小于 yCur，且 cbvpValidClass[4]等于 0，将 IntraHmvpCandList[X]加入 blockMotionClass[4]，并令 cbvpValidClass[4]等于 1；

d) 否则，如果 xCandX 大于或等于 xCur + widthCur，且 yCandX 小于 yCur，且 cbvpValidClass[5]等于 0，将 IntraHmvpCandList[X]加入 blockMotionClass[5]，并令 cbvpValidClass[5]等于 1；

e) 否则，如果 yCandX 大于或等于 yCur + heightCur，且 xCandX 小于 xCur，且 cbvpValidClass[6]等于 0，将 IntraHmvpCandList[X]加入 blockMotionClass[6]，并令 cbvpValidClass[6]等于 1；

f) 否则，如果 yCandX 小于 yCur，且 xCandX 大于或等于 xCur，且 xCandX 小于 xCur + widthCur，且 cbvpValidClass[3]等于 0，将 IntraHmvpCandList[X]加入 blockMotionClass[3]，并令 cbvpValidClass[3]等于 1；

g) 否则，如果 xCandX 小于 xCur，且 yCandX 大于或等于 yCur，且 yCandX 小于 yCur + heightCur，且 cbvpValidClass[2]等于 0，将 IntraHmvpCandList[X]加入 blockMotionClass[2]，并令 cbvpValidClass[2]等于 1。

8.6.6.4.3 块矢量解码

块矢量解码过程如下：

```
if (NumAllowedHbvpCand < 2) {
    MvDiffXBv = MvDiffXBv << (AbvrIndex + 2)
    MvDiffYBv = MvDiffYBv << (AbvrIndex + 2)
    bvE->x = Clip3(-32768, 32767, MvDiffXBv)
    bvE->y = Clip3(-32768, 32767, MvDiffYBv)
}
else {
    MvDiffXBv = MvDiffXBv << (AbvrIndex + 2)
    MvDiffYBv = MvDiffYBv << (AbvrIndex + 2)
    MvPredXBv = Rounding(MvPredXBv, AbvrIndex + 2) << (AbvrIndex + 2)
    MvPredYBv = Rounding(MvPredYBv, AbvrIndex + 2) << (AbvrIndex + 2)
    bvE->x = Clip3(-32768, 32767, MvDiffXBv + MvPredXBv)
    nbvE->y = Clip3(-32768, 32767, MvDiffYBv + MvPredYBv)
}
```

8.6.6.4.4 块复制预测模式下更新历史帧内复制信息表

位移矢量是块矢量 bvE；位置是当前预测块左上角位置的坐标(xCur, yCur)；尺寸是当前预测块宽度与高度的积，即 widthCur×heightCur。当前预测块的重复次数应初始化为 0。

如果 NumOfIntraHmvpCand 的值大于 0，根据当前块复制帧内预测块的运动信息，按 8.6.6.7 更新历史帧内复制信息表 IntraHmvpCandList；否则，不执行更新操作。

8.6.6.4.5 块复制帧内预测的限制

从符合本文件的位流中解码得到的块矢量指向的参考块应限制在当前最大编码单元或左边 N 个最大编码单元的范围内。参考块的所有参考样本均应与当前块的样本属于同一个片，且已完全重建。N 的值如下：

T/AI 109.8-2025

```
N = (1 << ((7 - LcuSizeInBit) << 1)) - (LcuSizeInBit < 7 ? 1 : 0)
```

令当前块左上角位置的坐标为(xCur, yCur)，参考块亮度分量左上角位置的坐标为(xRefTL, yRefTL)，参考块亮度分量右下角位置的坐标为(xRefBR, yRefBR)。坐标(xRefTL, yRefTL)和(xRefBR, yRefBR)应满足以下条件：

```
vSize = (LcuSize >= 64) ? 64 : LcuSize
xRefTL / vSize == xRefBR / vSize
yRefTL / vSize == yRefBR / vSize
```

当块矢量指向的参考块的左上角落在左边相邻最大编码单元，且最大编码单元的尺寸为 128×128 时，参考块左上角右移 128 像素后的位置所在的 64×64 区域的左上角应尚未重建，且坐标(xCur, yCur)应满足以下条件：

```
xCur != ((xRefTL + 128) / 64) * 64
yCur != (yRefTL / 64) * 64
```

8.6.6.5 串复制帧内预测模式

8.6.6.5.1 概述

串复制帧内预测模式有两种子模式：普通串子模式、非普通串子模式。

非普通串子模式的编码单元由等值串、单位基矢量串和未匹配样本组成。一个等值串的所有样本的值均等于同一个常现位置上的样本的值。点矢量(pvX, pvY) = (x, y) - (LcuRx0, LcuRy0)指向常现位置，其中(x, y)和(LcuRx0, LcuRy0)分别是常现位置和当前最大编码单元行左上角样本在图像中的坐标。单位基矢量串的串矢量(svX, svY)为(0, -1)。

串复制帧内预测模式的编码单元按往返扫描顺序划分为 IscPartNum 个部分，依次对各部分进行处理。

在普通串子模式中，如果 IscMatchType[i](i=0～IscPartNum-1)的值为 1，或 IscMatchType[i]的值为 0 且 NumMatchedPixel[i]大于 0，则按 8.6.6.5.2 解码得到串矢量；否则，直接从位流中得到未匹配样本。完成当前预测单元的解码后，如果 IscMatchType[i]的值为 1 且 IscSvAboveFlag[i]的值不为 1，则令 HbvpEmptyFlag 的值为 1，按 8.6.6.5.3 更新历史帧内复制信息表。

在非普通串子模式中，如果 StringType[i]的值等于 0（即第 i 部分是单位基矢量串），则按 8.6.6.5.2 解码得到串矢量；否则，如果 StringType[i]的值等于 1（即第 i 部分是等值串），则从点预测信息表中得到点矢量；否则，直接从位流中得到未匹配像素。完成当前预测单元的解码后，按 8.6.6.5.5 更新等值串历史点预测信息表。

符合本文件的位流应满足 8.6.6.5.6 规定的限制。

8.6.6.5.2 串矢量解码

如果 IscSubtypeFlag 的值等于 1，则单位基矢量串的串矢量 Sv[i]为：

```
Sv[i]->x = 0
Sv[i]->y = -1
```

否则，按以下方法解码得到第 i 部分的串矢量 Sv[i]。

——如果 IscSvAboveFlag[i]等于 1，Sv[i]等于(0, -1)。

——否则，如果 IscSvRecentFlag[i]等于 1，Sv[i]等于 IntraHmvpCandList[CntIntraHmvp-1-IscSvRecentIndex[i]]的位移矢量。

——否则，Sv[i]为：

T/AI 109.8-2025

```
Sv[i]->x = (-1)^IscSvXSign[i] * IscSvXAbs[i]
Sv[i]->y = (-1)^IscSvYSign[i] * IscSvYAbs[i]
```

8.6.6.5.3 串复制预测模式下更新历史帧内复制信息表

位移矢量是当前预测块第 i 部分的串矢量 Sv[i]；位置是当前预测块第 i 部分的第一个样本的位置坐标(xi,yi)；尺寸是当前预测块第 i 部分的串长度 StrLen[i]。当前预测块第 i 部分的重复次数应初始化为 0。

如果 NumOfIntraHmvpCand 的值大于 0，根据当前串复制预测块的运动信息，按以下方法更新历史帧内复制信息表 IntraHmvpCandList；否则，不执行更新操作。

a) 初始化 strPos 为 0；
b) i=0～IscPartNum-1，执行步骤 c)和步骤 d)；
c) 如果 IscMatchType[i]的值为 1：
   1) 计算该部分的第一个样本的坐标：

```
xi = TravScan[Log(CuWidth)-2][Log(CuHeight)-2][strPos][0]
yi = TravScan[Log(CuWidth)-2][Log(CuHeight)-2][strPos][1]
```

   2) 如果 Sv[i]不等于(0,-1)，按 8.6.6.7 更新历史帧内复制信息表；
   3) strPos 加 StrLen[i]。
d) 否则，跳过历史帧内复制信息表的更新，strPos 加 4。

8.6.6.5.4 串复制预测模式下更新等值串参考样本缓冲区

令vSize的值等于Min(LcuSize,64)，width和height分别为当前编码树节点的宽度和高度，如果PictureIscEvsUbvsEnableFlag的值等于1，且x0%vSize的值等于0，且y0%vSize的值等于0，且width的值等于vSize，且height的值等于vSize，按以下方法更新等值串参考样本缓冲区EvsDpb。

第1步，将cnt的值初始化为0，令tmpEvsDpb[m][n]中的值全部等于0，其中m的值为0～2，n的值为0～27。

第2步，令i=0～PrevPvBufSize-1，执行以下操作。

a) 检查历史点预测信息表 PrevPpInfoList 的第 i 个点矢量(pvX[i],pvY[i])是否满足以下条件：
   1) 点矢量指向的参考像素限制在当前最大编码单元或左边 N 个最大编码单元的范围内，N 的大小由最大编码单元的大小决定；

```
N = (1 << ((7 - log2_lcu_size_minus2 + 2) << 1)) - (((log2_lcu_size_minus2 + 2) < 7) ? 1 : 0)
```

   2) 令参考像素亮度分量 A 的位置坐标为(xRefA,yRefA)。当点矢量指向的参考像素落在相邻最大编码单元，且最大编码单元的尺寸为 128×128 时，该亮度分量 A 右移 128 像素后的位置所在的 64×64 区域的左上角应尚未重建，且上述坐标应满足以下条件；

```
xCur != ((xRefA + 128) / 64) * 64
yCur != (yRefA / 64) * 64
```

   3) 点矢量指向的参考像素位于已完成重建的区域,且应与当前编码单元中的样本处于同一个片内。
b) 如果 PrevPPInfoList[i]满足条件，PrevEvsDpbIndex[i]的值等于 28，PrevEvsDpbReactivatedYonly[i]的值等于 0；
c) 否则，按以下方法将 PrevPPInfoList[i]对应的参考像素存入 tmpEvsDpb。

```
if (PrevEvsDpbIndex[i] < 28 ) {
```

T/AI 109.8-2025

```
        tmpEvsDpb[0][cnt] = EvsDpb[0][PrevEvsDpbIndex[i]]
        if (PrevFopYonly[k] == 0) {
            tmpEvsDpb[1][cnt] = EvsDpb[1][PrevEvsDpbIndex[i]]
            tmpEvsDpb[2][cnt] = EvsDpb[2][PrevEvsDpbIndex[i]]
        }
    }
    else {
        tmpEvsDpb[0][cnt] = LcuRowBufY[pvX[i]][pvY[i]]
        if (PrevFopYonly[k] == 0) {
            tmpEvsDpb[1][cnt] = LcuRowBufU[pvX[0]>>1][pvY[1]>>1]
            tmpEvsDpb[2][cnt] = LcuRowBufV[pvX[0]>>1][pvY[1]>>1]
        }
    }
    PrevEvsDpbIndex[i] = cnt
    cnt = cnt + 1
```

第3步，令EvsDpb等于tmpEvsDpb。

8.6.6.5.5 串复制预测模式下更新历史点预测信息表

完成当前编码单元的解码后，如果当前编码单元采用串复制帧内预测非普通串子模式且(IscNumOfNewPv+IscNumofReusedPv)不等于0，则根据当前编码单元的点预测信息表PpInfoList更新历史点预测信息表PrevPpInfoList，并更新列表PrevFopYonly、PrevEvsDpbReactivatedYonly、PrevCompLumaFreqOccurPos和PrevEvsDpbIndex；否则，不执行本条规定的操作。

令PrevPvBufSize等于历史点预测信息表中点矢量的总数，PvBufSize等于当前编码单元的点预测信息表中点矢量的总数，tmpPvBuf[i]、tmpFlag[i]、tmpEvsDpbIndex[i]、tmpEvsDpbReactivatedYonly[i]和tmpCompLumaFreqOccurPos[i]（i=0～27）是点预测信息临时缓冲区，按以下方式执行更新操作：

```
tmpIndex = 0;
for (k=0; k<PrevPvBufSize; k++) {
    if (PrevPpInfoList[k][0] != -1 && PrevPpInfoList[k][1] != -1) {
        tmpPvBuf[tmpIndex][0] = PrevPpInfoList[k][0]
        tmpPvBuf[tmpIndex][1] = PrevPpInfoList[k][1]
        tmpFlag[tmpIndex]= PrevFopYonly[k]
        tmpEvsDpbIndex[tmpIndex]= PrevEvsDpbIndex[k]
        tmpEvsDpbReactivatedYonly[tmpIndex] = PrevEvsDpbReactivatedYonly[k]
        tmpCompLumaFreqOccurPos[tmpIndex] = PrevCompLumaFreqOccurPos[k]
        tmpIndex++
    }
}
PrevPvBufSize = Min(28, PvNum + tmpIndex)
for (k=0; k<PrevPvBufSize; k++) {
    if (k <PvBufSize) {
        PrevPpInfoList[k][0] = PpInfoList[k][0]
        PrevPpInfoList[k][1] = PpInfoList[k][1]
```

T/AI 109.8-2025

```
            PrevFopYonly[k] = FopYonly[k]
            PrevEvsDpbIndex[k] = EvsDpbIndex[k]
            PrevCompLumaFreqOccurPos[k] = CompLumaFreqOccurPos[k]
            PrevEvsDpbReactivatedYonly[k] = EvsDpbReactivatedYonly[k]
        }
        else {
            PrevPpInfoList[k][0] = tmpPvBuf[k-PvNum][0]
            PrevPpInfoList[k][1] = tmpPvBuf[k-PvNum][1]
            PrevFopYonly[k] = tmpFlag[k-PvNum]
            PrevEvsDpbIndex[k] = tmpEvsDpbIndex[k-PvNum]
            PrevCompLumaFreqOccurPos[k] = tmpCompLumaFreqOccurPos[k-PvNum]
            PrevEvsDpbReactivatedYonly[k] = tmpEvsDpbReactivatedYonly[k-PvNum]
        }
}
```

8.6.6.5.6 串复制帧内预测的限制

从符合本文件的位流中解码得到的串矢量指向的参考串或点矢量指向的常现位置中任意参考像素应限制在当前最大编码单元或左边 N 个最大编码单元的范围内。N 的值如下：

```
N = (1 << ((7 - LcuSizeInBit) << 1)) - (LcuSizeInBit < 7 ? 1 : 0)
```

如果 IscSubtypeFlag 的值为 0，令当前块左上角位置的坐标为(xCur, yCur)，参考串任意亮度分量 A 和 B 的位置坐标分别为(xRefA, yRefA)和(xRefB, yRefB)。坐标(xRefA, yRefA)和(xRefB, yRefB)应满足以下条件：

```
vSize = (LcuSize >= 64 ? 64 : LcuSize)
xRefA / vSize == xRefB / vSize
yRefA / vSize == yRefB / vSize
```

如果 IscSubtypeFlag 的值为 0，且当参考串任意亮度分量 A 落在左边相邻最大编码单元，且最大编码单元的尺寸为 128×128 时，该亮度分量 A 右移 128 像素后的位置所在的 64×64 区域的左上角应尚未重建，且坐标(xCur, yCur)应满足以下条件：

```
xCur != ((xRefA + 128) / 64) * 64
yCur != (yRefA / 64) * 64
```

如果 IscSubtypeFlag 的值为 0，当前串的参考串中的所有参考样本均应与当前串中的样本处于同一个片内，且已完全重建。如果串矢量 Y 分量的值大于或等于 0，则串矢量指向的参考串中任意参考像素不应位于当前串内。

如果 IscSubtypeFlag 的值为 1，令当前块左上角位置的坐标为(xCur, yCur)，常现位置亮度分量 A 的位置坐标为(xRef, yRef)。坐标(xCur, yCur)和(xRef, yRef)应满足以下条件：

```
xCur != ((xRef + 128) / 64) * 64
yCur != (yRef / 64) * 64
```

令 numOfP 等于 IscNumOfNewPv+IscNumofReusedPv，如果 IscSubtypeFlag 的值为 1：

——如果当前编码单元的宽度乘以高度的积大于 64，则 numOfP 的值不应大于 15；

——否则，如果当前编码单元的宽度乘以高度的积大于 32（即为 8×8、4×16 或 16×4），则 numOfP 的值不应大于 10；

——否则，如果当前编码单元的宽度乘以高度的积大于 16（即为 8×4 或 4×8），则 numOfP 的值不

T/AI 109.8-2025

应大于 5；

——否则（即为 4×4），则 numOfP 的值不应大于 2。

当前编码单元使用的点矢量均存放在点预测信息表 PpInfoList[15][2]中。PpInfoList[15][2]中的点矢量由两部分组成，一部分来自历史点预测信息表 PrevPpInfoList[28][2]，历史点预测信息表中的点矢量的数量不应大于 28；另一部分是当前编码单元中新出现的点矢量。

如果当前编码单元使用普通串子模式，则匹配串的个数、至少包含一个匹配样本的不完全匹配串的个数、未匹配样本的个数，以及 IscPartNumSplit 四者之和应小于或等于当前编码单元样本数的四分之一；否则，如果当前编码单元使用非普通串子模式，则等值串的个数、单位基矢量串的个数、未匹配样本的个数，以及 IscPartNumSplit 四者之和应小于或等于当前编码单元样本数的四分之一。其中普通串子模式的 IscPartNumSplit 按 8.8.3.2 得到，等值串及单位基矢量串模式的 IscPartNumSplit 按 8.8.3.3 得到。

8.6.6.6 更新帧内预测模式频数表

如果 IsPrivacyFlag 的值为 1，则跳过更新帧内预测模式频数表。否则按照以下方法更新帧内预测模式频数表。

如果 IntraLumaPredMode 的值为‘Intra_Luma_DC’、‘Intra_Luma_Bilinear’、‘Intra_Luma_Vertical’或‘Intra_Luma_Horizontal’，则按以下方法更新帧内模式频数表 FimcFrequencyList 以及高频模式 ModeFimc[0]和 ModeFimc[1]。

FimcFrequencyList[0]、FimcFrequencyList[1]、FimcFrequencyList[2]和 FimcFrequencyList[3]分别对应‘Intra_Luma_DC’、‘Intra_Luma_Bilinear’、‘Intra_Luma_Vertical’和‘Intra_Luma_Horizontal’模式的频数。IntraLumaPredMode、ModeFimc[0]和 ModeFimc[1]分别对应帧内预测模式频数表的索引为 IntraLumaPredModeIndex、modeIndex0 和 modeIndex1。

```
FimcFrequencyList[IntraLumaPredModeIndex] = FimcFrequencyList[IntraLumaPredModeIndex] + 1
freqCurr = FimcFrequencyList[IntraLumaPredModeIndex]
mode0 = ModeFimc[0]
freq0 = FimcFrequencyList[modeIndex0]
mode1 = ModeFimc[1]
freq1 = FimcFrequencyList[modeIndex1]
if (freqCurr >= freq0) {
    ModeFimc[0] = IntraLumaPredMode
    if (mode0 != IntraLumaPredMode)
        ModeFimc[1] = mode0
}
else if (freqCurr >= freq1)
    ModeFimc[1] = IntraLumaPredMode
```

8.6.6.7 更新历史帧内复制信息表

如果IsPrivacyFlag的值为1，则跳过更新历史帧内复制信息表。否则按照以下方法更新历史帧内复制信息表。

表中的历史帧内复制信息记为IntraHmvpCandList[X]，包括位移矢量intraMvCandX，位置(xCandX,yCandX)，尺寸sizeCandX和重复次数cntCandX。

T/AI 109.8-2025

令intraCur是当前块的帧内复制信息，包括位移矢量intraMvCur，位置(xCur,yCur)，尺寸sizeCur和重复次数cntCur。更新历史帧内复制信息表的方法如下：

a) 将 X 和 cntCur 均初始化为 0；
b) 如果 CntIntraHmvp 等于 0，则 IntraHmvpCandList[CntIntraHmvp]为当前预测单元的帧内预测运动信息，CntIntraHmvp 加 1，结束本条的更新过程；
c) 否则，根据 intraMvCur 和 intraMvCandX 是否相等判断当前预测块的帧内预测运动信息和 IntraHmvpCandList[X]是否相同：
   1) 如果 intraMvCur 和 intraMvCandX 相同，执行步骤 d)；否则，X 加 1；
   2) 如果 X 小于 CntIntraHmvp，执行步骤 c)；否则，执行步骤 e)。
d) cntCur 等于 cntCandX 的值加 1。如果 sizeCur 小于 sizeCandX，则 xCur、yCur 和 sizeCur 分别等于 xCandx、yCandx 和 sizeCandX；
e) 如果 X 小于 CntIntraHmvp，则：
   1) i 从 X 到 CntIntraHmvp-2，令 IntraHmvpCandList[i]等于 IntraHmvpCandList[i+1]；
   2) IntraHmvpCandList[CntIntraHmvp-1]等于当前预测单元的帧内预测运动信息。
f) 否则，如果 X 等于 CntIntraHmvp 且 CntIntraHmvp 等于 NumOfIntraHmvpCand，则：
   1) i 从 0 到 CntIntraHmvp-1，令 IntraHmvpCandList [i]等于 IntraHmvpCandList[i+1]；
   2) IntraHmvpCandList[CntIntraHmvp-1]等于当前预测单元的帧内预测运动信息。
g) 否则，如果 X 等于 CntIntraHmvp 且 CntIntraHmvp 小于 NumOfIntraHmvpCand，则 IntraHmvpCandList[CntIntraHmvp]等于当前预测单元的帧内预测运动信息，CntIntraHmvp 加 1。

8.6.6.8 帧内预测信息存储

8.6.6.8.1 导出空域角度加权预测帧内亮度预测模式存储方式

令W和H是当前编码单元的宽度和高度，(x,y)是当前编码单元中的每个4×4块的左上角像素位置，pos_x等于((x >> 2) << 2) + 2，pos_y等于((y >> 2) << 2) + 2。按以下方法导出每个4×4块的帧内亮度预测模式存储方式SawpModeType：

a) 计算 stepIndex、angleIndex 和 angelAreaIndex 的值；

```
stepIndex = (SawpIndex >> 3) - 3
modAngNum = SawpIndex % 8
if (modAngNum == 2)
    angleIndex = 7
else if (modAngNum == 6)
    angleIndex = 8
else
    angleIndex = modAngNum % 2
angleAreaIndex = modAngNum >> 1
```

b) 根据 angleAreaIndex 查表 111 得到 tP 和 fP，并计算当前编码单元中每个 4×4 块的帧内亮度预测模式存储方式 SawpModeType。

```
if (tP >= fP)
    SawpModeType = 0
else
```

T/AI 109.8-2025

SawpModeType = 1

表111 angleAreaIndex 和 fP、tP 的对应关系

angleAreaIndex	fP	tP
0	H + (W >> angleIndex) - 2 + stepIndex ×(((H + (W >> angleIndex)) >>2) -1)	(pos_y << 1) + ((pos_x << 1) >> angleIndex)
1	H + (W >> angleIndex) + stepIndex ×(((H + (W >> angleIndex)) >> 2) -1) - ((W << 1) >> angleIndex)	(pos_y << 1) - ((pos_x << 1) >> angleIndex)
2	W + (H >> angleIndex) + stepIndex ×(((W + (H >> angleIndex)) >> 2) -1) - ((H << 1) >> angleIndex)	(pos_x << 1) - ((pos_y << 1) >> angleIndex)
3	W + (H >> angleIndex) - 2 + stepIndex ×(((W + (H >> angleIndex)) >> 2) - 1)	(pos_x << 1) + ((pos_y << 1) >> angleIndex)

8.6.6.8.2 空域帧内预测信息存储

首先，确定样本(x,y)对应的空域帧内预测信息存储单元的左上角像素点位置(x_0,y_0)：

$x_0 = (x >> 2) << 2$

$y_0 = (y >> 2) << 2$

其次，设 X 是(x_0,y_0)亮度样本所在的预测单元，按以下方法将帧内亮度预测模式存储到样本(x,y)对应的空域帧内亮度预测模式存储单元：

a) 该空域帧内预测信息存储单元的预测类型为当前编码单元的预测类型；

b) 如果(x0,y0)亮度样本所在的预测单元是普通帧内预测且 SawpFlag 为 0，则该空域帧内亮度预测模式存储单元的帧内亮度预测模式等于 IntraLumaPredMode；

c) 否则，如果(x0,y0)亮度样本所在的预测单元是普通帧内预测且 SawpFlag 为 1，按 8.6.6.8.1 导出空域角度加权预测帧内亮度预测模式存储方式 SawpModeType；

1) 如果 SawpModeType 等于 0，则该空域帧内亮度预测模式存储单元的帧内亮度预测模式等于 SawpPredMode0；

2) 否则（SawpModeType 等于 1），则该空域帧内亮度预测模式存储单元的帧内亮度预测模式等于 SawpPredMode1。

d) 否则，如果当前编码单元的预测类型为块复制帧内预测，该空域帧内预测信息存储单元的块矢量等于当前编码单元的块矢量；

e) 否则，如果当前编码单元的预测类型为串复制帧内预测，该空域帧内预测信息存储单元的串矢量等于坐标为(x_0,y_0+3)样本对应的串矢量。当坐标为(x_0,y_0+3)样本所在的串不为匹配串时，对应的串矢量为(0,0)。

8.6.7 运动信息

8.6.7.1 概述

本条规定了导出预测单元运动信息的方法。

T/AI 109.8-2025

如果当前预测单元包含亮度分量，执行以下处理。

第 1 步，确定 MvExistL0、MvExistL1 的值。

——如果同时满足以下条件，则当前预测单元的 MvExistL0 的值为 1，否则 MvExistL0 的值为 0。

a) 当前预测单元所在编码单元的编码单元类型是‘B_2M_2N’或‘P_2M_2N’；

b) 当前预测单元的预测参考模式是‘PRED_List0’或‘PRED_List01’。

——如果同时满足以下条件，则当前预测单元的 MvExistL1 的值为 1，否则 MvExistL1 的值为 0。

a) 当前预测单元所在编码单元的编码单元类型是‘B_2M_2N’或‘P_2M_2N’；

b) 当前预测单元的预测参考模式是‘PRED_List1’或‘PRED_List01’。

第 2 步，导出运动信息。

——如果当前预测单元所在编码单元的编码单元类型是‘P_Skip’或‘P_Direct’或‘B_Skip’或‘B_Direct’，按 8.6.7.8 导出运动信息；

——否则：

a) 如果当前预测单元的 MvExistL0 为 1，先按 8.6.7.6 得到运动矢量预测值，再按 8.6.7.7 解码得到 L0 运动矢量；

b) 如果当前预测单元的 MvExistL1 为 1，先按 8.6.7.6 得到运动矢量预测值，再按 8.6.7.7 解码得到 L1 运动矢量。

——按 8.6.7.12 导出相邻块单向运动信息矩阵。

第 3 步，按 8.6.7.9 用当前预测单元的运动信息更新历史运动信息表。

如果当前预测单元的预测参考模式是‘PRED_List01’，或 AffineSubblockSizeFlag 的值为 1，则该预测单元的各个 8×8 子块运动矢量可不同；否则该预测单元的各个 4×4 子块运动矢量可不同。

运动矢量的基本单位如下。

——如果当前编码单元不是仿射模式，根据 AmvrIndex 的值决定运动矢量的基本单位：

a) 如果 AmvrIndex 的值为 0，运动矢量的基本单位是 1/4 个整数样本；

b) 如果 AmvrIndex 的值为 1，运动矢量的基本单位是 1/2 个整数样本；

c) 如果 AmvrIndex 的值为 2，运动矢量的基本单位是 1 个整数样本；

d) 如果 AmvrIndex 的值为 3，运动矢量的基本单位是 2 个整数样本；

e) 如果 AmvrIndex 的值为 4，运动矢量的基本单位是 4 个整数样本。

——否则（编码单元是仿射模式），根据 AffineAmvrIndex 的值决定运动矢量的基本单位：

a) 如果 AffineAmvrIndex 的值为 0，运动矢量的基本单位是 1/4 个整数样本；

b) 如果 AffineAmvrIndex 的值为 1，运动矢量的基本单位是 1 个整数样本；

c) 如果 AffineAmvrIndex 的值为 2，运动矢量的基本单位是 1/16 个整数样本。

8.6.7.2 判断运动信息的异同

如果两个运动信息满足下面一个或多个条件，则两个运动信息不同；否则两个运动信息相同：

——InterPredRefMode 不同；

——InterPredRefMode 均为 0，L0 运动矢量不相等或 L0 参考索引不相等；

——InterPredRefMode 均为 1，L1 运动矢量不相等或 L1 参考索引不相等；

——InterPredRefMode 均为 2，L0 运动矢量、L1 运动矢量、L0 参考索引或 L1 参考索引中有任意一个不相等。

8.6.7.3 判断参考帧的异同

如果两个参考帧满足下面一个或多个条件，则两个参考帧不同；否则两个参考帧相同：

T/AI 109.8-2025

——其中一个参考帧是显示图像，另一个参考帧是知识图像；
——两个参考帧都是显示图像，对应的参考图像 DOI 不相等；
——两个参考帧都是知识图像，对应的知识图像索引不相等。

8.6.7.4 导出仿射预测单元的子块信息和控制参数

8.6.7.4.1 概述

先按 8.6.7.4.2 导出子块仿射运动矢量组，如果 AsrEnableFlag 的值为 1，再按 8.6.7.4.3 导出仿射预测单元的子块偏差矩阵和仿射样本改善信息。

8.6.7.4.2 导出仿射预测单元的子块运动矢量组

如果仿射控制点运动矢量组中有 3 个运动矢量，运动矢量组表示为 mvsAffine(mv0,mv1,mv2)；否则（仿射控制点运动矢量组中有 2 个运动矢量），运动矢量组表示为 mvsAffine(mv0,mv1)。

第 1 步，计算 dHorX、dVerX、dHorY 和 dVerY。

——dHorX 和 dHorY 的计算如下：

```
dHorX = (mv1->x - mv0->x) << (7 - Log(width))
dHorY = (mv1->y - mv0->y) << (7 - Log(width))
```

——如果 mvsAffine 中有 3 个运动矢量，dVerX 和 dVerY 的计算如下：

```
dVerX = (mv2->x - mv0->x) << (7 - Log(height))
dVerY = (mv2->y - mv0->y) << (7 - Log(height))
```

——否则（mvsAffine 中有 2 个运动矢量），dVerX 和 dVerY 的计算如下：

```
dVerX = - dHorY
dVerY = dHorX
```

令(xE,yE)是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，当前预测单元的宽度和高度分别是 width 和 height，(x,y)是亮度子块左上角位置的坐标，子块的宽度和高度分别是 subWidth 和 subHeight，当前预测单元亮度预测块的左上角样本所在的子块是 A 子块，右上角样本所在的子块是 B 子块，左下角样本所在的子块是 C 子块，见图 24。

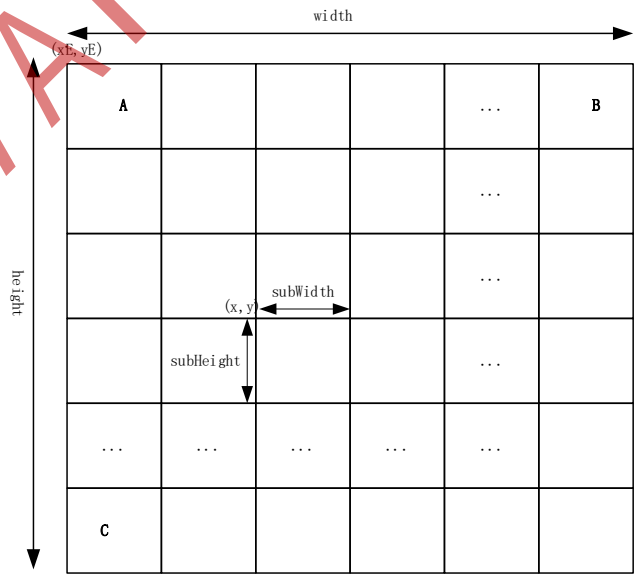


图24 样本的位置

T/AI 109.8-2025

第 2 步，计算亮度子块的运动矢量 mvE。

——如果当前预测单元的预测参考模式是‘Pred_List01’，或 AffineSubblockSizeFlag 等于 1，则 subWidth 和 subHeight 均等于 8，(x,y)是 8×8 子块左上角位置的坐标，计算每个 8×8 亮度子块的运动矢量：

a) 如果当前子块是 A，则 xPos 和 yPos 均等于 0；

b) 否则，如果当前子块是 B，则 xPos 等于 width，yPos 等于 0；

c) 否则，如果当前子块为 C 且 mvsAffine 中有 3 个运动矢量，则 xPos 等于 0，yPos 等于 height；

d) 否则，xPos 等于(x-xE)+4，yPos 等于(y-yE)+4；

e) 当前 8×8 子块的运动矢量 mvE：

```
mvE->x = Clip3(-131072, 131071, Rounding((mv0->x << 7) + dHorX * xPos + dVerX * yPos, 7))
mvE->y = Clip3(-131072, 131071, Rounding((mv0->y << 7) + dHorY * xPos + dVerY * yPos, 7))
```

——如果当前预测单元的预测参考模式是‘PRED_List0’或‘PRED_List1’且 AffineSubblockSizeFlag 等于 0，则 subWidth 和 subHeight 均等于 4，(x,y)是 4×4 子块左上角位置的坐标，计算每个 4×4 亮度子块的运动矢量：

a) 如果当前子块是 A，则 xPos 和 yPos 均等于 0；

b) 否则，如果当前子块是 B，则 xPos 等于 width，yPos 等于 0；

c) 否则，如果当前子块是 C 且 mvAffine 中有 3 个运动矢量，则 xPos 等于 0，yPos 等 height；

d) 否则，xPos 等于(x-xE)+2，yPos 等于(y-yE)+2；

e) 当前 4×4 子块的运动矢量 mvE：

```
mvE->x = Clip3(-131072, 131071, Rounding((mv0->x << 7) + dHorX * xPos + dVerX * yPos, 7))
mvE->y = Clip3(-131072, 131071, Rounding((mv0->y << 7) + dHorY * xPos + dVerY * yPos, 7))
```

符合本文件的位流中，按以下方法计算 MemoryAccess 的值：

——如果 mvAffine 中有 2 个运动矢量，计算 mv2：

```
mv2->x = Clip3(-131072, 131071, Rounding((mv0->x << 7) + dVerX*height, 7))
mv2->y = Clip3(-131072, 131071, Rounding((mv0->y << 7) + dVerY*height, 7))
```

——计算 mv3：

```
mv3->x = Clip3(-131072, 131071, Rounding((mv0->x << 7) + dHorX*width + dVerX*height, 7))
mv3->y = Clip3(-131072, 131071, Rounding((mv0->y << 7) + dHorY*width + dVerY*height, 7))
```

——计算 maxX、minX、maxY 和 minY：

```
maxX = Max((mv0->x >> 4), Max((mv1->x >> 4) + width, Max((mv2->x >> 4), (mv3->x >> 4) + width)))
minX = Min((mv0->x >> 4), Min((mv1->x >> 4) + width, Min((mv2->x >> 4), (mv3->x >> 4) + width)))
maxY = Max((mv0->y >> 4), Max((mv1->y >> 4), Max((mv2->y >> 4) + height, (mv3->y >> 4) + height)))
minY = Min((mv0->y >> 4), Min((mv1->y >> 4), Min((mv2->y >> 4) + height, (mv3->y >> 4) + height)))
```

——计算 MemoryAccess：

```
MemoryAccess = (maxX - minX + 7) * (maxY - minY + 7)
```

符合本文件的位流中MemoryAccess的值应满足以下条件：

```
MemoryAccess <= ((width + 7 + width / 4) * (height + 7 + height / 4))
```

8.6.7.4.3 导出仿射预测单元的子块偏差矩阵和控制参数

T/AI 109.8-2025

本条导出仿射样本改善信息asrInfo(ASRFlag,ASRHorFlag,ASRVerFlag,dMvA,dMvB,dMvC,dMvN)。

如果仿射控制点运动矢量组中有3个运动矢量，运动矢量组表示为mvsAffine(mv0,mv1,mv2)；否则（仿射控制点运动矢量组中有2个运动矢量），运动矢量组表示为mvsAffine(mv0,mv1)。

令当前预测单元的宽度和高度分别是width和height，(x,y)是亮度子块左上角位置的坐标，子块的宽度和高度分别是subWidth和subHeight，当前预测单元亮度预测块的左上角样本所在的子块是A子块，右上角样本所在的子块是B子块，左下角样本所在的子块是C子块，其他样本所在的子块是其它子块，见图24。

第 1 步，计算 dHorX、dVerX、dHorY 和 dVerY。

——dHorX 和 dHorY 的计算如下：

```
dHorX = (mv1->x - mv0->x) << (7 - Log(width))
dHorY = (mv1->y - mv0->y) << (7 - Log(width))
```

——如果 mvsAffine 中有 3 个运动矢量，dVerX 和 dVerY 的计算如下：

```
dVerX = (mv2->x - mv0->x) << (7 - Log(height))
dVerY = (mv2->y - mv0->y) << (7 - Log(height))
```

——否则（mvsAffine 中有 2 个运动矢量），dVerX 和 dVerY 的计算如下：

```
dVerX = - dHorY
dVerY = dHorX
```

第2步，导出子块偏差矩阵dMvX（X为A、B、C或N）。dMvX[i][j]是子块偏差矩阵的元素，其中(i,j)是每个亮度子块内部像素的坐标,i的取值范围是0～(subWidth-1),j的取值范围是0～(subHeight-1)。

——如果 mvAffine 中有 2 个运动矢量且 dHorX 和 dHorY 均等于 0，令 ASRFlag 等于 0；

——否则，如果 mvAffine 中有 3 个运动矢量，且 dHorX、dHorY、dVerX 和 dVerY 均等于 0，令 ASRFlag 等于 0；

——否则，

a) 令 ASRFlag 等于 1；

b) 如果当前预测单元的预测参考模式是‘PRED_List01’或 AffineSubblockSizeFlag 等于 1，则 subWidth 和 subHeight 均等于 8；

c) 如果当前预测单元的预测参考模式是‘PRED_List0’或‘PRED_List1’，且 AffineSubblockSizeFlag 等于 0，则 subWidth 和 subHeight 均等于 4；

d) 如果当前子块是 A，则当前子块的偏差矩阵是 dMvA：

```
dMvA[i][j][0] = dHorX * 4 * i + dVerX * 4 * j
dMvA[i][j][1] = dHorY * 4 * i + dVerY * 4 * j
```

e) 否则，如果当前子块是 B，则当前子块偏差矩阵是 dMvB：

```
dMvB[i][j][0] = dHorX * 4 * (i + 1 - subWidth) + dVerX * 4 * j
dMvB[i][j][1] = dHorY * 4 * (i + 1 - subWidth) + dVerY * 4 * j
```

f) 否则，如果当前子块为 C 且运动矢量组中有 3 个运动矢量,则当前子块偏差矩阵是 dMvC：

```
dMvC[i][j][0] = dHorX * 4 * i + dVerX * 4 * (j + 1 - subHeight)
dMvC[i][j][1] = dHorY * 4 * i + dVerY * 4 * (j + 1 - subHeight)
```

g) 否则，如果当前子块为 C 且运动矢量组中有 2 个运动矢量,则当前子块偏差矩阵是 dMvC：

```
dMvC[i][j][0] = dHorX * (2 + 4 * i - 2 * subWidth) + dVerX * (2 + 4 * j - 2 * subHeight)
dMvC[i][j][1] = dHorY × (2 + 4 * i - 2 * subWidth) + dVerY * (2 + 4 * j - 2 * subHeight)
```

h) 否则,当前子块偏差矩阵是 dMvN：

T/AI 109.8-2025

```
dMvN[i][j][0] = dHorX * (2 + 4 * i - 2 * subWidth) + dVerX * (2 + 4 * j - 2 * subHeight)
dMvN[i][j][1] = dHorY * (2 + 4 * i - 2 * subWidth) + dVerY * (2 + 4 * j - 2 * subHeight)
```

i) 计算 dMvHorAbsMax、dMvVerAbsMax 和 dMvThresh；

```
dMvHorAbsMax = Max(Max(Abs(dMvN[0][0][0]), Abs(dMvN[subWidth-1][0][0])),
    Max(Abs(dMvN[0][subHeight-1][0]), Abs(dMvN[subWidth-1][subHeight-1][0])))
dMvVerAbsMax = Max(Max(Abs(dMvN[0][0][1]), Abs(dMvN[subWidth-1][0][1])),
    Max(Abs(dMvN[0][subHeight-1][1]), Abs(dMvN[subWidth-1][subHeight-1][1])))
dMvThresh = ((1<<5) * 10)
```

j) 将 ASRHorFlag 和 ASRVerFlag 初始化为 1；

- 如果 dMvHorAbsMax 小于 dMvThresh 且 dMvVerAbsMax 小于 dMvThresh，ASRFlag、ASRHorFlag 和 ASRVerFlag 均等于 0；
- 否则，如果 dMvHorAbsMax 小于 dMvThresh，ASRHorFlag 为 0；
- 否则，如果 dMvVerAbsMax 小于 dMvThresh，ASRVerFlag 为 0。

k) 如果 ASRHorFlag 等于 1，导出亮度子块 X 的 dMvX[i][j]（X 为 A，B，C，N），i=0～(subWidth-1)，j=0～(subHeight-1)：

```
dMvX[i][j][0] = Clip3(-31, 31, Rounding(dMvX[i][j][0], 8))
```

l) 如果 ASRVerFlag 等于 1，导出亮度子块 X 的 dMvX[i][j]（X 为 A，B，C，N），i=0～(subWidth-1)，j=0～(subHeight-1)：

```
dMvX[i][j][1] = Clip3(-31, 31, Rounding(dMvX[i][j][1], 8))
```

8.6.7.5 获得相邻块空域运动信息存储单元

当前预测单元的亮度预测块 E 的相邻亮度预测块 X（X 为 A、B、C、D、F 或 G）的空域运动信息存储单元如下（见图 18）：

——相邻块 A 的空域运动信息存储单元是样本（x_0-1, y_0）所对应的空域运动信息存储单元；
——相邻块 B 的空域运动信息存储单元是样本（x_0, y_0-1）所对应的空域运动信息存储单元；
——相邻块 C 的空域运动信息存储单元是样本(x_1+1, y_0-1)所对应的空域运动信息存储单元；
——相邻块 D 的空域运动信息存储单元是样本(x_0-1, y_0-1)所对应的空域运动信息存储单元；
——相邻块 F 的空域运动信息存储单元是样本(x_0-1， y_1) 所对应的空域运动信息存储单元；
——相邻块 G 的空域运动信息存储单元是样本(x_1, y_0-1) 所对应的空域运动信息存储单元。

8.6.7.6 运动矢量预测

8.6.7.6.1 概述

如果 AffineFlag 和 ExtendMvrFlag 的值均为 0，运动矢量预测过程见 8.6.7.6.2；如果 AffineFlag 的值为 0 且 ExtendMvrFlag 的值为 1，运动矢量预测过程见 8.6.7.6.4。如果 AffineFlag 的值为 1，运动矢量预测过程见 8.6.7.6.3。

8.6.7.6.2 普通运动矢量预测

本条规定导出 mvX，referenceIndexX，BlockDistanceX（X 为 A、B、C 或 D）的方法。X 与当前预测单元的亮度预测块 E 的相邻关系见图 18。

第 1 步，执行以下操作：

T/AI 109.8-2025

a) 如果相邻亮度预测块 X（X 为 A、B、C 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；

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对应的参考图像队列相同的运动矢量。

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对应的参考图像队列相同的运动矢量，BlockDistanceX 等于 mvX 对应的 BlockDistance，referenceIndexX 等于 mvX 对应的 referenceIndex。

b) 如果 referenceIndexC 等于-1，则 mvC 等于 mvD，BlockDistanceC 等于 BlockDistanceD，referenceIndexC 等于 referenceIndexD。

第 2 步，计算待预测运动矢量预测值 MVEPred。

a) 步骤 2.1，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。其中 BlockDistanceE 是当前预测单元对应的 BlockDistance。

```
MVA->x = Clip3(-32768, 32767, Sign(mvA->x * BlockDistanceE * BlockDistanceA) * ((Abs(mvA->x * BlockDistanceE * (16384 / BlockDistanceA)) + 8192) >> 14))
MVA->y = Clip3(-32768, 32767, Sign(mvA->y * BlockDistanceE * BlockDistanceA) * ((Abs(mvA->y * BlockDistanceE * (16384 / BlockDistanceA)) + 8192) >> 14))
MVB->x = Clip3(-32768, 32767, Sign(mvB->x * BlockDistanceE * BlockDistanceB) * ((Abs(mvB->x * BlockDistanceE * (16384 / BlockDistanceB)) + 8192) >> 14))
MVB->y = Clip3(-32768, 32767, Sign(mvB->y * BlockDistanceE * BlockDistanceB) * ((Abs(mvB->y * BlockDistanceE * (16384 / BlockDistanceB)) + 8192) >> 14))
MVC->x = Clip3(-32768, 32767, Sign(mvC->x * BlockDistanceE * BlockDistanceC) * ((Abs(mvC->x * BlockDistanceE * (16384 / BlockDistanceC)) + 8192) >> 14))
MVC->y = Clip3(-32768, 32767, Sign(mvC->y * BlockDistanceE * BlockDistanceC) * ((Abs(mvC->y * BlockDistanceE * (16384 / BlockDistanceC)) + 8192) >> 14))
```

b) 步骤 2.2，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），结束第 2 步；否则执行步骤 2.3；

c) 步骤 2.3，计算 MVEPred 的 x 和 y 运动矢量分量。

```
if (((MVA->x < 0) && (MVB->x > 0) && (MVC->x > 0)) || ((MVA->x > 0) && (MVB->x < 0) && (MVC->x < 0)))
    MVEPred->x = (MVB->x + MVC->x) / 2
else if (((MVB->x < 0) && (MVA->x > 0) && (MVC->x > 0)) || ((MVB->x > 0) && (MVA->x < 0) && (MVC->x < 0)))
    MVEPred->x = (MVA->x + MVC->x) / 2
else if (((MVC->x < 0) && (MVA->x > 0) && (MVB->x > 0)) || ((MVC->x > 0) && (MVA->x < 0) && (MVB->x < 0)))
    MVEPred->x = (MVA->x + MVB->x) / 2
else if ((Abs(MVA->x - MVB->x) <= Abs(MVB->x - MVC->x)) && (Abs(MVA->x - MVB->x) <= Abs(MVC->x - MVA->x)))
```

T/AI 109.8-2025

```
        MVEPred->x = (MVA->x + MVB->x) / 2
else if ((Abs(MVB->x - MVC->x) <= Abs(MVA->x - MVB->x)) && (Abs(MVB->x - MVC->x) <= Abs(MVC->x
        - MVA->x)))
        MVEPred->x = (MVB->x + MVC->x) / 2
else
        MVEPred->x = (MVA->x + MVC->x) / 2

if (((MVA->y < 0) && (MVB->y > 0) && (MVC->y > 0)) || ((MVA->y > 0) && (MVB->y < 0) && (MVC->y <
        0)))
        MVEPred->y = (MVB->y + MVC->y) / 2
else if (((MVB->y < 0) && (MVA->y > 0) && (MVC->y > 0)) || ((MVB->y > 0) && (MVA->y < 0) && (MVC->y
        < 0)))
        MVEPred->y = (MVA->y + MVC->y) / 2
else if (((MVC->y < 0) && (MVA->y > 0) && (MVB->y > 0)) || ((MVC->y > 0) && (MVA->y < 0) && (MVB->y
        < 0)))
        MVEPred->y = (MVA->y + MVB->y) / 2
else if ((Abs(MVA->y - MVB->y) <= Abs(MVB->y - MVC->y)) && (Abs(MVA->y - MVB->y) <= Abs(MVC->y
        - MVA->y)))
        MVEPred->y = (MVA->y + MVB->y) / 2
else if ((Abs(MVB->y - MVC->y) <= Abs(MVA->y - MVB->y)) && (Abs(MVB->y - MVC->y) <= Abs(MVC->y
        - MVA->y)))
        MVEPred->y = (MVB->y + MVC->y) / 2
else
        MVEPred->y = (MVA->y + MVC->y) / 2
```

第 3 步，执行以下操作：

```
MVEPred->x = Clip3(-32768, 32767, Rounding( MVEPred->x, AmvrIndex ) << AmvrIndex)
MVEPred->y = Clip3(-32768, 32767, Rounding( MVEPred->y, AmvrIndex ) << AmvrIndex)
```

如果当前待预测运动矢量预测值 MVEPred 对应参考图像队列 0 中的图像，则 MVEPredL0 等于 MVEPred；否则，如果当前待预测运动矢量预测值 MVEPred 对应参考图像队列 1 中的图像，则 MVEPredL1 等于 MVEPred。

8.6.7.6.3 仿射运动矢量预测

本条规定导出 MvX，referenceIndexX，BlockDistanceX（X 为 A、B、C、D 或 G）的方法。X 与当前预测单元的亮度预测块 E 的相邻关系见图 18。

第 1 步，执行以下操作：

a) 如果相邻亮度预测块 X（X 为 A、B、C、D 或 G）满足以下条件之一，则 mvX 等于零矢量，BlockDistanceX 等于 1，referenceIndexX 等于-1；

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 不存在与当前待预测运动矢量对应的参考图像队列相同的运动矢量。

T/AI 109.8-2025

b) 否则，mvX 等于相邻亮度预测块 X 与当前待预测运动矢量对应的参考图像队列相同的运动矢量，BlockDistanceX 等于 X 所在的预测单元对应的 BlockDistance，referenceIndexX 等于 X 所在的预测单元对应的 referenceIndex。

第 2 步，计算当前待预测运动矢量预测值 MvEPred。

a) 如果相邻预测块 A 所在的预测单元的 referenceIndexA 不等于-1，根据 BlockDistanceA 和 BlockDistanceE 对 mvA 进行缩放得到 MvEPred；

```
MvEPred->x = Clip3(-32768, 32767, Sign(mvA->x * BlockDistanceE * BlockDistanceA) * ((Abs(mvA->x *
        BlockDistanceE * (16384 / BlockDistanceA)) + 8192) >> 14))
MvEPred->y = Clip3(-32768, 32767, Sign(mvA->y * BlockDistanceE * BlockDistanceA) * ((Abs(mvA->y *
        BlockDistanceE * (16384 / BlockDistanceA)) + 8192) >> 14))
```

b) 否则，如果相邻预测块 B 所在的预测单元的 referenceIndexB 不等于-1，根据 BlockDistanceB 和 BlockDistanceE 对 mvB 进行缩放得到 MvEPred；

```
MvEPred->x = Clip3(-32768, 32767, Sign(mvB->x * BlockDistanceE * BlockDistanceB) * ((Abs(mvB->x *
        BlockDistanceE * (16384 / BlockDistanceB)) + 8192) >> 14))
MvEPred->y = Clip3(-32768, 32767, Sign(mvB->y * BlockDistanceE * BlockDistanceB) * ((Abs(mvB_y *
        BlockDistanceE * (16384 / BlockDistanceB)) + 8192) >> 14))
```

c) 否则，如果相邻预测块 D 所在的预测单元的 referenceIndexD 不等于-1，根据 BlockDistanceD 和 BlockDistanceE 对 mvD 进行缩放得到 MvEPred；

```
MvEPred->x = Clip3(-32768, 32767, Sign(mvD->x * BlockDistanceE * BlockDistanceD) * ((Abs(mvD->x *
        BlockDistanceE * (16384 / BlockDistanceD)) + 8192) >> 14))
MvEPred->y = Clip3(-32768, 32767, Sign(mvD->y * BlockDistanceE * BlockDistanceD) * ((Abs(mvD->y *
        BlockDistanceE * (16384 / BlockDistanceD)) + 8192) >> 14))
```

d) 否则，MvEPred 为零矢量。

第 3 步，计算当前待预测运动矢量预测值 MvEPredAffine。

a) 如果相邻预测块 G 所在的预测单元的 referenceIndexG 不等于-1，根据 BlockDistanceG 和 BlockDistanceE 对 mvG 进行缩放得到 MvEPredAffine；

```
MvEPredAffine->x = Clip3(-32768, 32767, Sign(mvG->x * BlockDistanceE * BlockDistanceG) * ((Abs(mvG->x *
        BlockDistanceE * (16384 / BlockDistanceG)) + 8192) >> 14))
MvEPredAffine->y = Clip3(-32768, 32767, Sign(mvG->y × BlockDistanceE × BlockDistanceG) × ((Abs(mvG->y ×
        BlockDistanceE × (16384 / BlockDistanceG)) + 8192) >> 14))
```

b) 否则，如果相邻预测块 C 所在的预测单元的 referenceIndexC 不等于-1，根据 BlockDistanceC 和 BlockDistanceE 对 mvC 进行缩放得到 MvEPredAffine；

```
MvEPredAffine->x = Clip3(-32768, 32767, Sign(mvC->x * BlockDistanceE * BlockDistanceC) * ((Abs(mvC->x *
        BlockDistanceE * (16384 / BlockDistanceC)) + 8192) >> 14))
MvEPredAffine->y = Clip3(-32768, 32767, Sign(mvC->y * BlockDistanceE * BlockDistanceC) * ((Abs(mvC->y *
        BlockDistanceE * (16384 / BlockDistanceC)) + 8192) >> 14))
```

c) 否则，MvEPredAffine 为零矢量。

第 4 步，执行以下操作：

```
MvEPred->x = MvEPred->x << 2
MvEPred->y = MvEPred->y << 2
MvEPredAffine->x = MvEPredAffine->x << 2
MvEPredAffine->y = MvEPredAffine->y << 2
```

T/AI 109.8-2025

```
mvrIndex = (AffineAmvrIndex == 0) ? 2 : ((AffineAmvrIndex == 1) ? 4 : 0)
MvEPred->x = Clip3(-131072, 131071, Rounding( MvEPred->x, mvrIndex ) << mvrIndex)
MvEPred->y = Clip3(-131072, 131071, Rounding( MvEPred->y, mvrIndex ) << mvrIndex)
MvEPredAffine->x = Clip3(-131072, 131071, Rounding( MvEPredAffine->x, mvrIndex ) << mvrIndex)
MvEPredAffine->y = Clip3(-131072, 131071, Rounding( MvEPredAffine->y, mvrIndex ) << mvrIndex)
```

如果当前待预测运动矢量预测值 MVEPred、MVEPredAffine 对应参考图像队列 0 中的图像，则 MVEPredL0 等于 MVEPred，MVEPredL0Affine 等于 MVEPredAffine；否则，如果当前待预测运动矢量预测值 MVEPred、MVEPredAffine 对应参考图像队列 1 中的图像，则 MVEPredL1 等于 MVEPred，MVEPredL1Affine 等于 MVEPredAffine。

8.6.7.6.4 扩展精度运动矢量预测

第 1 步，得到运动信息。

a) 令 hmvpIndex 的值等于 AmvrIndex；

b) 如果 CntHmvp 的值大于 hmvpIndex 的值，则取历史运动信息表中第(CntHmvp – hmvpIndex)个运动信息，记为 motionInfo；

c) 否则，如果 CntHmvp 的值大于 0 且小于或等于 hmvpIndex，则取历史运动信息表中第 CntHmvp 个运动信息，记为 motionInfo；

d) 否则，如果当前图像是 P 图像，令 motionInfo 的 L0 运动矢量为零矢量，L0 参考索引值为 0，预测参考模式为‘PRED_List0’；

e) 否则，如果当前图像是 B 图像，令 motionInfo 的 L0 运动矢量为零矢量，L1 运动矢量为零矢量，L0 参考索引值为 0，L1 参考索引值为 0，预测参考模式为‘PRED_List01’；

f) 具有运动信息 motionInfo 的预测单元的 BlockDistance 分别记为 BlockDistanceL0 和 BlockDistanceL1，当前预测单元所在的图像与参考图像队列 0 和参考图像队列 1 中的参考图像对应的 BlockDistance 分别记为 BlockDistanceE0 和 BlockDistanceE1，motionInfo 的 L0 运动矢量和 L1 运动矢量分别记为 mvHmvp0 和 mvHmvp1，motionInfo 的 L0 参考索引和 L1 参考索引分别记为 hmvpRef0 和 hmvpRef1，motionInfo 的预测参考模式记为 hmvpInterMode。

第 2 步，计算当前待预测运动矢量预测值 MvEPred。

a) 由预测参考模式 hmvpInterMode、当前预测单元的预测参考模式查表 112 得到运动矢量 mv、距离索引 D1 和距离索引 D2。如果当前待预测运动矢量预测值 MVEPred 对应参考图像队列 0 中的图像，则 MVEPredL0 等于 MVEPred；否则，如果当前待预测运动矢量预测值 MVEPred 对应参考图像队列 1 中的图像，则 MVEPredL1 等于 MVEPred；

```
MvEPred->x = Clip3(-32768, 32767, Sign(mv->x * D1 * D2) * ((Abs(mv->x * D1 * (16384 / D2)) + 8192) >> 14))
MvEPred->y = Clip3(-32768, 32767, Sign(mv->y * D1 × D2) * ((Abs(mv->y * D1 * (16384 / D2)) + 8192) >> 14))
```

b) 执行以下操作：

```
MVEPred->x = Clip3(-32768, 32767, Rounding( MVEPred->x, AmvrIndex ) << AmvrIndex)
MVEPred->y = Clip3(-32768, 32767, Rounding( MVEPred->y, AmvrIndex ) << AmvrIndex)
```

T/AI 109.8-2025

表112 运动矢量、距离索引和预测参考模式

预测参考模式	当前预测单元 预测参考模式	运动 矢量	距离索引 D1	距离索引 D2	运动矢量 预测值
PRED_List0	PRED_List0	mvHmvp0	BlockDistanceE0	BlockDistanceL0	mvePredL0
	PRED_List1	mvHmvp0	BlockDistanceE1	BlockDistanceL0	mvePredL1
	PRED_List01	mvHmvp0	BlockDistanceE0	BlockDistanceL0	mvePredL0
		mvHmvp0	BlockDistanceE1	BlockDistanceL0	mvePredL1
PRED_List1	PRED_List0	mvHmvp1	BlockDistanceE0	BlockDistanceL1	mvePredL0
	PRED_List1	mvHmvp1	BlockDistanceE1	BlockDistanceL1	mvePredL1
	PRED_List01	mvHmvp1	BlockDistanceE0	BlockDistanceL1	mvePredL0
		mvHmvp1	BlockDistanceE1	BlockDistanceL1	mvePredL1
PRED_List01	PRED_List0	mvHmvp0	BlockDistanceE0	BlockDistanceL0	mvePredL0
	PRED_List1	mvHmvp1	BlockDistanceE1	BlockDistanceL1	mvePredL1
	PRED_List01	mvHmvp0	BlockDistanceE0	BlockDistanceL0	mvePredL0
		mvHmvp1	BlockDistanceE1	BlockDistanceL1	mvePredL1

如果当前待预测运动矢量预测值 MVEPred 对应参考图像队列 0 中的图像，则 MVEPredL0 等于 MVEPred；否则，如果当前待预测运动矢量预测值 MVEPred 对应参考图像队列 1 中的图像，则 MVEPredL1 等于 MVEPred。

8.6.7.7 运动矢量解码

8.6.7.7.1 概述

如果 AffineFlag 的值为 0，运动矢量解码过程见 8.6.7.7.2；否则，运动矢量解码过程见 8.6.7.7.3。

8.6.7.7.2 普通运动矢量解码

按以下方法解码得到运动矢量：

a) 如果当前预测单元的预测参考模式是‘Pred_List0’：

```
MvDiffXL0 = MvDiffXL0 << AmvrIndex
MvDiffYL0 = MvDiffYL0 << AmvrIndex
mvE->x = Clip3(-32768, 32767, MvDiffXL0 + MVEPredL0->x)
mvE->y = Clip3(-32768, 32767, MvDiffYL0 + MVEPredL0->y)
```

mvE 是当前预测单元的 L0 运动矢量，MvE 等于 mvE。

b) 如果当前预测单元的预测参考模式是‘Pred_List1’：

```
MvDiffXL1 = MvDiffXL1 << AmvrIndex
MvDiffYL1 = MvDiffYL1 << AmvrIndex
mvE->x = Clip3(-32768, 32767, MvDiffXL1 + MVEPredL1->x)
mvE->y = Clip3(-32768, 32767, MvDiffYL1 + MVEPredL1->y)
```

mvE 是当前预测单元的 L1 运动矢量，MvE 等于 mvE。

c) 如果当前预测单元的预测参考模式是‘Pred_List01’：

T/AI 109.8-2025

```
MvDiffXL0 = MvDiffXL0 << AmvrIndex
MvDiffYL0 = MvDiffYL0 << AmvrIndex
MvDiffXL1 = MvDiffXL1 << AmvrIndex
MvDiffYL1 = MvDiffYL1 << AmvrIndex
mvE0->x = Clip3(-32768, 32767, MvDiffXL0 + MVEPredL0->x)
mvE0->y = Clip3(-32768, 32767, MvDiffYL0 + MVEPredL0->y)
mvE1->x = Clip3(-32768, 32767, MvDiffXL1 + MVEPredL1->x)
mvE1->y = Clip3(-32768, 32767, MvDiffYL1 + MVEPredL1->y)
```

mvE0、mvE1 分别为当前预测单元的 L0 运动矢量和 L1 运动矢量，MvE0 等于 mvE0，MvE1 等于 mvE1。

8.6.7.7.3 仿射运动矢量解码

按以下方法解码得到仿射运动矢量：

a) 如果当前预测单元的预测参考模式是‘Pred_List0’：

```
mvrIndex = (AffineAmvrIndex == 0) ? 2 : ((AffineAmvrIndex == 1) ? 4 : 0)
MvDiffXL0 = MvDiffXL0 << mvrIndex
MvDiffYL0 = MvDiffYL0 << mvrIndex
MvDiffXL0Affine = MvDiffXL0Affine << mvrIndex
MvDiffYL0Affine = MvDiffYL0Affine << mvrIndex
mv1->x = Clip3(-131072, 131071, MvDiffXL0 + MVEPredL0->x)
mv1->y = Clip3(-131072, 131071, MvDiffYL0 + MVEPredL0->y)
mv2->x = Clip3(-131072, 131071, MvDiffXL0Affine + MVEPredL0Affine->x)
mv2->y = Clip3(-131072, 131071, MvDiffYL0Affine + MVEPredL0Affine->y)
```

mvsAffine(mv1,mv2)是当前预测单元的仿射控制点运动矢量组，当前预测单元是四参数仿射模式，根据 8.6.7.4.2 规定的过程由 mvsAffine 导出当前预测单元的 L0 运动矢量集合（记作 MvArrayL0）。MvArrayL0 由所有亮度预测子块的 L0 运动矢量组成。

b) 否则，如果当前预测单元的预测参考模式是‘Pred_List1’：

```
mvrIndex = (AffineAmvrIndex == 0) ? 2 : ((AffineAmvrIndex == 1) ? 4 : 0)
MvDiffXL1 = MvDiffXL1 << mvrIndex
MvDiffYL1 = MvDiffYL1 << mvrIndex
MvDiffXL1Affine = MvDiffXL1Affine << mvrIndex
MvDiffYL1Affine = MvDiffYL1Affine << mvrIndex
mv1->x = Clip3(-131072, 131071, MvDiffXL1 + MVEPredL1->x)
mv1->y = Clip3(-131072, 131071, MvDiffYL1 + MVEPredL1->y)
mv2->x = Clip3(-131072, 131071, MvDiffXL1Affine + MVEPredL1Affine->x)
mv2->y = Clip3(-131072, 131071, MvDiffYL1Affine + MVEPredL1Affine->y)
```

mvsAffine(mv1,mv2)是当前预测单元的仿射控制点运动矢量组，当前预测单元是四参数仿射模式，根据 8.6.7.4.2 规定的过程由 mvsAffine 导出当前预测单元中的 L1 运动矢量集合（记作 MvArrayL1）。MvArrayL1 由所有亮度预测子块的 L1 运动矢量组成。

c) 否则，如果当前预测单元的预测参考模式是‘Pred_List01’：

```
mvrIndex = (AffineAmvrIndex == 0) ? 2 : ((AffineAmvrIndex == 1) ? 4 : 0)
MvDiffXL0 = MvDiffXL0 << mvrIndex
```

T/AI 109.8-2025

```
MvDiffYL0 = MvDiffYL0 << mvrIndex
MvDiffXL1 = MvDiffXL1 << mvrIndex
MvDiffYL1 = MvDiffYL1 << mvrIndex
MvDiffXL0Affine = MvDiffXL0Affine << mvrIndex
MvDiffYL0Affine = MvDiffYL0Affine << mvrIndex
MvDiffXL1Affine = MvDiffXL1Affine << mvrIndex
MvDiffYL1Affine = MvDiffYL1Affine << mvrIndex
mv01->x = Clip3(-131072, 131071, MvDiffXL0 + MVEPredL0->x)
mv01->y = Clip3(-131072, 131071, MvDiffYL0 + MVEPredL0->y)
mv02->x = Clip3(-131072, 131071, MvDiffXL0Affine + MVEPredL0Affine->x)
mv02->y = Clip3(-131072, 131071, MvDiffYL0Affine + MVEPredL0Affine->y)
mv11->x = Clip3(-131072, 131071, MvDiffXL1 + MVEPredL1->x)
mv11->y = Clip3(-131072, 131071, MvDiffYL1 + MVEPredL1->y)
mv12->x = Clip3(-131072, 131071, MvDiffXL1Affine + MVEPredL1Affine->x)
mv12->y = Clip3(-131072, 131071, MvDiffYL1Affine + MVEPredL1Affine->y)
```

mvsAffine0(mv01,mv02)是当前预测单元的 L0 仿射控制点运动矢量组，mvsAffine1(mv11,mv12)是当前预测单元的 L1 仿射控制点运动矢量组，当前预测单元是四参数仿射模式，根据 8.6.7.4.2 规定的过程由 mvsAffine0 和 mvsAffine1 分别导出当前预测单元中的 L0 运动矢量集合（记作 MvArrayL0）和 L1 运动矢量集合（记作 MvArrayL1）。MvArrayL0 和 MvArrayL1 分别由所有亮度预测子块的 L0 运动矢量和 L1 运动矢量组成。

8.6.7.8 导出跳过模式和直接模式的运动信息

8.6.7.8.1 概述

如果当前预测单元所在的编码单元的编码单元子类型是‘P_Skip_Temporal’或‘P_Direct_Temporal’，按 8.6.7.8.4 导出运动信息 motionInfo、BgcFlag 和 BgcIndex。

如果当前预测单元所在的编码单元的编码单元子类型是‘P_Skip_SbTemporal’或‘P_Direct_SbTemporal’，按 8.6.7.8.4 导出运动信息集合 MotionArray、BgcFlag 和 BgcIndex。

如果当前预测单元所在的编码单元的编码单元子类型是‘B_Skip_Temporal’或‘B_Direct_Temporal’，按 8.6.7.8.4 导出运动信息 motionInfo、BgcFlag 和 BgcIndex。

如果当前预测单元所在的编码单元的编码单元子类型是‘B_Skip_SbTemporal’或‘B_Direct_SbTemporal’，按 8.6.7.8.4 导出运动信息集合 MotionArray、BgcFlag 和 BgcIndex。

如果当前预测单元所在的编码单元的编码单元子类型是‘P_Skip_Spatial_List0’或‘P_Direct_Spatial_List0’，按 8.6.7.8.3 导出运动信息 motionInfo、BgcFlag 和 BgcIndex。

如果当前预测单元所在的编码单元的编码单元子类型是‘B_Skip_Spatial_list0’、‘B_Skip_Spatial_list1’、‘B_Skip_Spatial_list01’、‘B_Direct_Spatial_list0’、‘B_Direct_Spatial_list1’或‘B_Direct_Spatial_list01’，按 8.6.7.8.3 导出运动信息 motionInfo、BgcFlag 和 BgcIndex。

如果当前预测单元所在的编码单元的编码单元子类型是‘P_Skip_Mvap’、‘P_Direct_Mvap’、B_Skip_Mvap’或‘B_Direct_Mvap’，按 8.6.7.8.8 导出当前预测单元的运动信息阵列 MotionArray、BgcFlag 和 BgcIndex。

如果当前预测单元所在的编码单元的编码单元子类型是‘P_Skip_Hmvp’或‘P_Direct_Hmvp’，先

T/AI 109.8-2025

分别按 8.6.7.8.4 和 8.6.7.8.3 导出 motionInfo0 和 motionInfo1，再按 8.6.7.8.6 导出运动信息 motionInfo、BgcFlag 和 BgcIndex。

如果当前预测单元所在的编码单元的编码单元子类型是‘B_Skip_Hmvp’或‘B_Direct_Hmvp’，先分别按 8.6.7.8.4 和 8.6.7.8.3 导出 motionInfo0、motionInfo1、motionInfo2 和 motionInfo3，再按 8.6.7.8.6 导出运动信息 motionInfo、BgcFlag 和 BgcIndex。

如果当前预测单元所在的编码单元的编码单元子类型是‘P_Skip_Umve’、‘P_Direct_Umve’、‘B_Skip_Umve’或‘B_Direct_Umve’，按 8.6.7.8.5 导出运动信息 motionInfo、BgcFlag 和 BgcIndex。

如果当前预测单元所在的编码单元的编码单元子类型是‘P_Skip_Etmvp’、‘P_Direct_Etmvp’、‘B_Skip_Etmvp’或‘B_Direct_Etmvp’，按 8.6.7.8.10 导出运动矢量阵列 MotionArray、BgcFlag 和 BgcIndex。

如果当前预测单元所在的编码单元的编码单元子类型是‘P_Skip_Affine’、‘P_Direct_Affine’、‘B_Skip_Affine’或‘B_Direct_Affine’，按 8.6.7.8.7 导出仿射运动信息 motionInfoAffine、BgcFlag 和 BgcIndex，以及仿射样本改善信息 asrInfo0 和 asrInfo1。

如果当前预测单元所在的编码单元的编码单元子类型是‘B_Skip_Awp’、‘B_Direct_Awp’、‘P_Skip_Awp’或‘P_Direct_Awp’，按 8.6.7.8.9 导出运动信息 motionInfoAwp0、motionInfoAwp1、BgcFlag 和 BgcIndex。

8.6.7.8.3、8.6.7.8.4 和 8.6.7.8.7 中使用的运动信息存储单元的规定见 8.6.7.10。8.6.7.8.5 中判断运动信息是否相同的方法见 8.6.7.8.3。

8.6.7.8.2 导出有效运动矢量角度预测模式数

令W和H是当前解码单元的宽度和高度，m和n的分别是W/4和H/4，(x,y)是当前解码单元左上角的像素坐标。$A_0, A_1, \ldots, A_{m-1}$是当前块左下角位置的4×4块，$A_m, A_{m+1}, \ldots, A_{m+n-1}$是当前块左侧位置所在的4×4块，$A_{m+n}$是当前块左上角位置所在的4×4块，$A_{m+n+1}, A_{m+n+2}, \ldots, A_{2m+n}$是当前块上边位置所在的4×4块，$A_{2m+n+1}, A_{2m+n+1}, \ldots, A_{2m+2n}$是当前块右上角位置所在的4×4块，见图25。如果i的取值范围是0～m+n，则A_i的坐标分别是(x-1,y+H+W-4×i-1)；如果i的取值范围是m+n+1～2m+2n，则A_i的坐标分别为(x+4×i-W-H-1,y-1)。

A_i位置空域存储单元的运动信息“可用”指A_i满足以下所有条件，否则A_i位置空域存储单元的运动信息“不可用”：

——Ai 位置在图像内；

——A_i位置对应的编码单元与当前编码单元属于同一片；

——A_i位置对应的编码单元已解码且采用帧间预测模式；

——如果 PicturePrivacyEnableFlag 的值等于 1 且 A_i 位置权限存储单元的权限值小于等于 IsPrivacyFlag。

T/AI 109.8-2025

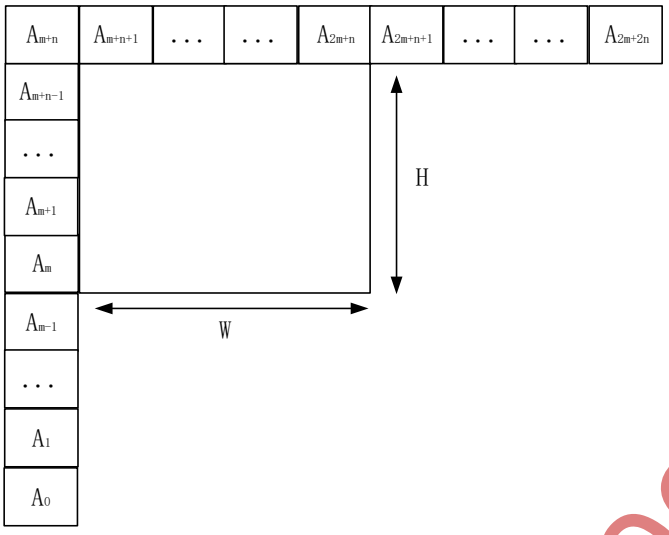


图25 运动矢量角度预测样本的位置

按以下步骤导出ValidMvapModeNum和ValidMvapModeFlag[s]（s=0～4）：

a) 将ValidMvapModeNum初始化为0，ValidMvapModeFlag[s]（s=0～4）初始化为0；

b) 如果满足以下条件之一，则 ValidMvapModeNum 等于 0，ValidMvapModeFlag[s]（s=0～4）等于 0；

1） EtmvpMvapEnableFlag 的值为 0；

2） W 小于 8，或 H 小于 8，或 W 和 H 均等于 8。

c) 否则，按下列步骤更新 ValidMvapModeNum 和 ValidMvapModeFlag[s]的值，其中判断运动信息是否相同的方法见 8.6.7.2。

1） 如果 $A_{m-1+H/8}$ 与 A_{m+n-1} 位置空域运动信息存储单元的运动信息均“可用”且运动信息不同，则 ValidMvapModeFlag[0]等于 1，并将 ValidMvapModeNum 的值加 1；

2） 如果 $A_{m+n+1+W/8}$ 与 A_{m+n+1} 位置空域运动信息存储单元的运动信息均“可用”且运动信息不同，则 ValidMvapModeFlag[1]等于 1，并将 ValidMvapModeNum 的值加 1；

3） 如果 A_{m+n-1} 与 A_{m+n} 位置空域运动信息存储单元的运动信息均“可用”且运动信息不同，或 A_{m+n} 和 A_{m+n+1} 位置空域运动信息存储单元的运动信息均“可用”且运动信息不同，则 ValidMvapModeFlag[2]等于 1，并将 ValidMvapModeNum 的值加 1；

4） 如果 $A_{W/8-1}$ 与 A_{m-1} 位置空域运动信息存储单元的运动信息均“可用”且运动信息不同，或 A_{m-1} 与 $A_{m-1+H/8}$ 位置空域运动信息存储单元的运动信息均“可用”且运动信息不同，则 ValidMvapModeFlag[3]等于 1，并将 ValidMvapModeNum 的值加 1；

5） 如果 $A_{m+n+1+W/8}$ 与 A_{2m+n+1} 位置空域运动信息存储单元的运动信息均“可用”且运动信息不同，或 A_{2m+n+1} 与 $A_{2m+n+1+H/8}$ 位置空域运动信息存储单元的运动信息均“可用”且运动信息不同，则 ValidMvapModeFlag[4]等于 1，并将 ValidMvapModeNum 的值加 1。

8.6.7.8.3 空域运动信息导出方法

8.6.7.8.3.1 概述

如果编码单元子类型是‘P_Skip_Spatial_list0’或‘P_Direct_Spatial_list0’，按 8.6.7.8.3.2 导出 motionInfo(interPredRefMode, mvE0, mvE1, refIndexL0, refIndexL1)、BgcFlag

T/AI 109.8-2025

和 BgcIndex。

如果编码单元子类型是‘B_Skip_Spatial_list01’或‘B_Direct_Spatial_list01’，按 8.6.7.8.3.3 导出 motionInfo(interPredRefMode, mvE0, mvE1, refIndexL0, refIndexL1)、BgcFlag 和 BgcIndex。

如果编码单元子类型是‘B_Skip_Spatial_list1’或‘B_Direct_Spatial_list1’，按 8.6.7.8.3.4 导出 motionInfo(interPredRefMode, mvE0, mvE1, refIndexL0, refIndexL1)、BgcFlag 和 BgcIndex。

如果编码单元子类型是‘B_Skip_Spatial_list0’或‘B_Direct_Spatial_list0’，按 8.6.7.8.3.5 导出 motionInfo(interPredRefMode, mvE0, mvE1, refIndexL0, refIndexL1)、BgcFlag 和 BgcIndex。

如果编码单子类型是‘P_Skip_Hmvp’或‘P_Direct_Hmvp’，按 8.6.7.8.3.2 导出 motionInfo1(interPredRefMode, mvE0, mvE1, refIndexL0, refIndexL1)、BgcFlag 和 BgcIndex。

如果编码单子类型是‘B_Skip_Hmvp’或‘B_Direct_Hmvp’，则分别按 8.6.7.8.3.3、8.6.7.8.3.4 和 8.6.7.8.3.5 导出 motionInfo1，motionInfo2，motionInfo3。

8.6.7.8.3.2 P 图像空域运动信息导出方法

运动信息、BgcFlag 和 BgcIndex 导出方法如下：

a) 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 六者中预测参考模式为‘PRED_List0’且权限值小于等于 IsPrivacyFlag 的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预测参考模式为‘PRED_List0’的预测块，将该预测块的空域运动信息存储单元的 L0 运动矢量和 L0 参考索引分别作为当前预测单元的 L0 运动矢量 mvE0 和 L0 参考索引 refIndexL0；

b) 否则，当前预测单元的 L0 运动矢量 mvE0 为零矢量，且当前预测单元的 L0 参考索引 refIndexL0 的值等于 0；

c) interPredRefMode 的值等于‘PRED_List0’，refIndexL1 的值等于-1，mvE1 是零矢量，BgcFlag 和 BgcIndex 的值均置为 0。

8.6.7.8.3.3 B 图像空域运动信息导出方法 1

运动信息、BgcFlag 和 BgcIndex 导出方法 1 如下：

a) 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 六者中预测参考模式为‘PRED_List01’且权限值小于等于 IsPrivacyFlag 的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预测参考模式为‘PRED_List01’且权限值小于等于 IsPrivacyFlag 的预测块，将该预测块的空域运动信息存储单元的 L0 运动矢量和 L1 运动矢量分别作为当前预测单元的 L0 运动矢量 mvE0 和 L1 运动矢量 mvE1，并将该空域运动信息存储单元的 L0 参考索引和 L1 参考索引分别作为当前预测单元的 L0 参考索引 refIndexL0 和 L1 参考索引 refIndexL1，同时将该空域运动信息存储单元的 BgcFlag 和 BgcIndex 记为 BgcFlagX 和 BgcIndexX，当前预测单元的 BgcFlag 和 BgcIndex 的值分别为(BgcFlagX && !InterPcFlag)和(BgcIndexX && !InterPcFlag)；

b) 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 六者中预测参考模式为‘PRED_List0’且权限值小于等于 IsPrivacyFlag 的预测块的个数大于或等于 1，且预测参考模式为‘PRED_List1’且权限值小于等于 IsPrivacyFlag 的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预测参考模式为

T/AI 109.8-2025

‘PRED_List0’且权限值小于等于 IsPrivacyFlag 的预测块和第一个扫描到的预测参考模式为‘PRED_List1’且权限值小于等于 IsPrivacyFlag 的预测块，将预测参考模式为‘PRED_List0’的预测块的空域运动信息存储单元的 L0 运动矢量和 L0 运动索引作为当前预测单元的 L0 运动矢量 mvE0 和 L0 运动索引 refIndexL0；将预测参考模式为‘PRED_List1’的预测块的空域运动信息存储单元的 L1 运动矢量和 L1 参考索引作为当前预测单元的 L1 运动矢量 mvE1 和 L1 参考索引 refIndexL1，并将当前预测单元的 BgcFlag 和 BgcIndex 均置为 0；

c) 否则，当前预测单元的 L0 运动矢量 mvE0 和 L1 运动矢量 mvE1 均为零矢量，且当前预测单元的 L0 参考索引 refIndexL0 和 L1 参考索引 refIndexL1 的值均等于 0，并将当前预测单元的 BgcFlag 和 BgcIndex 均置为 0；

d) interPredRefMode 的值等于‘PRED_List01’。

8.6.7.8.3.4 B 图像空域运动信息导出方法 2

运动信息、BgcFlag 和 BgcIndex 导出方法如下：

a) 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 六者中预测参考模式为‘PRED_List1’且权限值小于等于 IsPrivacyFlag 的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预测参考模式为‘PRED_List1’且权限值小于等于 IsPrivacyFlag 的预测块，将该预测块的空域运动信息存储单元的 L1 运动矢量和 L1 参考索引作为当前预测单元的 L1 运动矢量 mvE1 和 L1 参考索引 refIndexL1；

b) 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 六者中预测参考模式为‘PRED_List01’且权限值小于等于 IsPrivacyFlag 的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预测参考模式为‘PRED_List01’且权限值小于等于 IsPrivacyFlag 的预测块，将该预测块的空域运动信息存储单元的 L1 运动矢量和 L1 参考索引作为当前预测单元的 L1 运动矢量 mvE1 和 L1 参考索引 refIndexL1；

c) 否则，当前预测单元的 L1 运动矢量 mvE1 为零矢量，且当前预测单元的 L1 参考索引 refIndexL1 的值等于 0；

d) interPredRefMode 的值等于‘PRED_List1’，refIndexL0 的值等于-1，mvE0 是零矢量，BgcFlag 和 BgcIndex 的值均置为 0。

8.6.7.8.3.5 B 图像空域运动信息导出方法 3

运动信息、BgcFlag 和 BgcIndex 导出方法如下：

a) 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 六者中预测参考模式为‘PRED_List0’且权限值小于等于 IsPrivacyFlag 的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预测参考模式为‘PRED_List0’且权限值小于等于 IsPrivacyFlag 的预测块，将该预测块的空域运动信息存储单元的 L0 运动矢量和 L0 参考索引作为当前预测单元的 L0 运动矢量 mvE0 和 L0 参考索引 refIndexL0；

b) 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 六者中预测参考模式为‘PRED_List01’且权限值小于等于 IsPrivacyFlag 的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预测参考模式为‘PRED_List01’且权限值小于等于 IsPrivacyFlag 的预测块，将该预测块的空域运动信息存储单元的 L0 运动矢量和 L0 参考索引作为当前预测单元的 L0 运动矢量 mvE0 和 L0 参考索引 refIndexL0；

T/AI 109.8-2025

c) 否则，当前预测单元的 L0 运动矢量 mvE0 为零矢量，且当前预测单元的 L0 参考索引 refIndexL0 的值等于 0；

d) interPredRefMode 的值等于‘PRED_List0’，refIndexL1 的值等于-1，mvE1 是零矢量，BgcFlag 和 BgcIndex 的值均置为 0。

8.6.7.8.4 时域运动信息导出方法

8.6.7.8.4.1 概述

如果当前编码单元子类型是‘P_Skip_Temporal’、‘P_Direct_Temporal’、‘B_Skip_Temporal’或‘B_Direct_Temporal’，按 8.6.7.8.4.2 导出 BgcFlag、BgcIndex 和 motionInfo0，并分别对 mvE、refIndexL0 和 interPredRefMode 赋值；否则，按 8.6.7.8.4.3 导出 BgcFlag、BgcIndex 和 motionInfo0，并分别对 mvE、refIndexL0 和 interPredRefMode 赋值。

如果当前编码单元子类型是‘P_Skip_Hmvp’、‘P_Direct_Hmvp’、‘B_Skip_Hmvp’或‘B_Direct_Hmvp’且 SbTmvpEnableFlag 的值为 0，按 8.6.7.8.4.2 导出 BgcFlag、BgcIndex 和 motionInfo0。

8.6.7.8.4.2 时域运动信息导出方法 1

如果当前图像是 P 图像，导出运动信息、BgcFlag 和 BgcIndex 的方法如下：

a) 如果 PictureTmvpEnableFlag 的值为 0，或者参考图像队列 0 中参考索引为 0 的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时域运动信息存储单元存储的预测参考模式为‘PRED_List1’或‘PRED_I’，或者参考图像队列 0 中参考索引为 0 的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时域运动信息存储单元存储的权限值大于 IsPrivacyFlag，则当前预测单元的 L0 运动矢量 MvE0 为零矢量，并令当前预测单元的 L0 参考索引值 RefIndexL0 等于 0；

b) 否则，

1) 当前预测单元的 L0 参考索引 refIndex0 等于 0；

2) 将当前图像的距离索引记为 DistanceIndexCur，当前预测单元的 L0 参考索引对应的参考图像的距离索引记为 DistanceIndexL0，计算 BlockDistanceL0 = DistanceIndexCur – DistanceIndexL0；

3) 在参考图像队列 0 中参考索引为 0 的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时域运动信息存储单元的 L0 运动矢量记为 mvRef(mvRef_x, mvRef_y)，该运动信息存储单元所在的图像的距离索引记为 DistanceIndexCol，该运动矢量指向的参考单元所在的图像的距离索引记为 DistanceIndexRef，计算 BlockDistanceRef = DistanceIndexCol – DistanceIndexRef；

4) 计算当前预测单元的 L0 运动矢量 mvE0。

```
mvE0->x = Clip3(-32768, 32767, Sign(mvRef->x * BlockDistanceL0 * BlockDistanceRef) * (((Abs(mvRef->x *
        BlockDistanceL0 * (16384 / BlockDistanceRef))) + 8192 ) >> 14))
mvE0->y = Clip3(-32768, 32767, Sign(mvRef->y * BlockDistanceL0 * BlockDistanceRef) * (((Abs(mvRef->y *
        BlockDistanceL0 * (16384 / BlockDistanceRef))) + 8192 ) >> 14))
```

c) interPredRefMode 的值等于‘PRED_List0’，BgcFlag 和 BgcIndex 的值均置为 0。

如果当前图像是 B 图像，导出运动信息、BgcFlag 和 BgcIndex 的方法如下：

T/AI 109.8-2025

a) 如果 PictureTmvpEnableFlag 的值为 0，或者参考图像队列 1 中参考索引值为 0 的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时域运动信息存储单元存储的预测参考模式为‘PRED_List1’或‘PRED_I’，或者参考图像队列 1 中参考索引为 0 的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时域运动信息存储单元存储的权限值大于 IsPrivacyFlag，则当前预测单元的 L0 参考索引和 L1 参考索引均等于 0。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，然后将按 8.6.7.6.2 得到的 L0 运动矢量预测值和 L1 运动矢量预测值分别作为当前预测单元的 L0 运动矢量 MvE0 和 L1 运动矢量 MvE1，并令当前预测单元的 L0 参考索引 RefIndexL0 和 L1 参考索引 RefIndexL1 均等于 0；

b) 否则，

1) 当前预测单元的 L0 参考索引 refIndex0 和 L1 参考索引 refIndex1 均等于 0；

2) 将当前图像的距离索引记为 DistanceIdexCur，当前预测单元的 L0 参考索引和 L1 参考索引对应的图像的距离索引分别记为 DistanceIndexL0 和 DistanceIndexL1，计算 BlockDistanceL0=DistanceIndexCur – DistanceIndexL0，BlockDistanceL1=DistanceIndexCur – DistanceIndexL1；

3) 在参考图像队列 1 中参考索引为 0 的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时域运动信息存储单元的 L0 运动矢量记为 mvRef(mvRef_x,mvRef_y)，该运动信息存储单元所在的图像的距离索引记为 DistanceIndexCol，该运动矢量指向的参考单元所在的图像的距离索引记为 DistanceIndexRef，计算 BlockDistanceRef = DistanceIndexCol – DistanceIndexRef；

4) 当前预测单元的 L0 参考索引 RefIndexL0 等于 0，计算当前预测单元的 L0 运动矢量 mvE0；

```
mvE0->x = Clip3(-32768, 32767, Sign(mvRef->x * BlockDistanceL0 * BlockDistanceRef) * (((Abs(mvRef->x
        * BlockDistanceL0 * (16384 / BlockDistanceRef))) + 8192 ) >> 14))
mvE0->y = Clip3(-32768, 32767, Sign(mvRef->y * BlockDistanceL0 * BlockDistanceRef) * (((Abs(mvRef->y
        * BlockDistanceL0 * (16384 / BlockDistanceRef))) + 8192 ) >> 14))
```

5) 当前预测单元的 L1 参考索引 RefIndexL1 等于 0，计算当前预测单元的 L1 运动矢量 mvE1。

```
mvE1->x = Clip3(-32768, 32767, Sign(mvRef->x * BlockDistanceL1 * BlockDistanceRef) * (((Abs(mvRef->x
        * BlockDistanceL1 * (16384 / BlockDistanceRef))) + 8192 ) >> 14))
mvE1->y = Clip3(-32768, 32767, Sign(mvRef->y * BlockDistanceL1 * BlockDistanceRef) * (((Abs(mvRef->y
        * BlockDistanceL1 * (16384 / BlockDistanceRef))) + 8192 ) >> 14))
```

c) interPredRefMode 等于‘PRED_List01’，BgcFlag 和 BgcIndex 的值均置为 0。

8.6.7.8.4.3 时域运动信息导出方法 2

第 1 种情况，当前图像是 P 图像，导出运动信息、BgcFlag 和 BgcIndex 的方法如下：

a) 以当前编码单元作为预测单元按 8.6.7.8.4.2 导出的预测单元的 L0 参考索引 RefIndexL0，记为 refIndexT0，以及预测单元的 L0 运动矢量 mvE0，记为 mvT0；

b) 将当前编码单元划为四个子块，每个子块的长和宽为当前编码单元的 1/2，位置关系见图 26。图 26 中数字是每个子块的 index。对于 index 为 0 的子块，（col_x,col_y）是其左上角亮度样本的位置坐标；对于 index 为 1 的子块，（col_x,col_y）是其右上角亮度样本的位置坐标；对于 index 为 2 的子块，（col_x,col_y）是其左下角亮度样本的位置坐标；对于 index 为 3 的子块，（col_x,col_y）是其右下角亮度样本的位置坐标。对每个子块执行以下步骤，导出

T/AI 109.8-2025

每个子块的运动信息 MotionInfo[i](interPredRefMode,mvE0,refIndex0,mvE1, refIndex1)，MotionArray 由 MotionInfo0[i]构成：

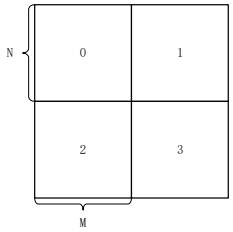


图26 预测单元划分为子块

1) 如果 PictureTmvpEnableFlag 的值为 0，或者参考图像队列 0 中参考索引为 0 的图像中位置坐标为(col_x,col_y)的亮度样本所在的时域运动信息存储单元存储的预测参考模式为‘PRED_I’，或者参考图像队列 0 中参考索引值为 0 的图像中位置坐标为（col_x,col_y）的亮度样本所在的时域运动信息存储单元存储的权限值大于 IsPrivacyFlag，则当前预测单元的 L0 运动矢量 MvE0 为 mvT0，并令当前预测单元的 L0 参考索引值 RefIndexL0 等于 refIndexT0；

2) 否则，

- 当前预测单元的 L0 参考索引 refIndex0 等于 0；
- 将当前图像的距离索引记为 DistanceIndexCur，当前预测单元 L0 参考索引对应的参考图像的距离索引记为 DistanceIndexL0，计算 BlockDistanceL0 = DistanceIndexCur – DistanceIndexL0；
- 如果参考图像队列 0 中参考索引为 0 的图像中位置坐标为(col_x,col_y)的亮度样本所在的时域运动信息存储单元预测参考模式为“PRED_List0”或“PRED_List01”，则将该时域运动信息存储单元 L0 运动矢量记为 mvRef(mvRef_x,mvRef_y)，该运动信息存储单元所在的图像的距离索引记为 DistanceIndexCol，该运动矢量指向的参考单元所在的图像的距离索引记为 DistanceIndexRef，计算 BlockDistanceRef = DistanceIndexCol – DistanceIndexRef；
- 否则，将该时域运动信息存储单元 L1 运动矢量记为 mvRef(mvRef_x,mvRef_y)，该运动信息存储单元所在的图像的距离索引记为 DistanceIndexCol，该运动矢量指向的参考单元所在的图像的距离索引记为 DistanceIndexRef，计算 BlockDistanceRef = DistanceIndexCol – DistanceIndexRef。
- 计算当前预测单元的 L0 运动矢量 mvE0。

```
mvE0->x = Clip3(-32768, 32767, Sign(mvRef->x * BlockDistanceL0 * BlockDistanceRef) *
        (((Abs(mvRef->x * BlockDistanceL0 * (16384 / BlockDistanceRef))) + 8192 ) >> 14))
mvE0->y = Clip3(-32768, 32767, Sign(mvRef->y * BlockDistanceL0 * BlockDistanceRef) *
        (((Abs(mvRef->y * BlockDistanceL0 * (16384 / BlockDistanceRef))) + 8192 ) >> 14))
```

c) interPredRefMode 的值等于‘PRED_List0’，BgcFlag 和 BgcIndex 的值均置为 0。

第 2 种情况，当前图像是 B 图像，导出运动信息、BgcFlag 和 BgcIndex 的方法如下：

a) 以当前编码单元作为预测单元按 8.6.7.8.4.2 导出预测单元的 L0 参考索引 RefIndexL0 和 L1 参考索引 RefIndexL1，分别记为 refIndexT0 和 refIndexT1，以及预测单元的 L0 运动矢量 mvE0 和 L1 运动矢量 mvE1，分别记为 mvT0 和 mvT1；

T/AI 109.8-2025

b) 将当前编码单元划为四个子块，每个子块的长和宽为当前编码单元的 1/2，位置关系见图 26。图 26 中数字是每个子块的 index。对于 index 为 0 的子块，（col_x,col_y）是其左上角亮度样本的位置坐标；对于 index 为 1 的子块，（col_x,col_y）是其右上角亮度样本的位置坐标；对于 index 为 2 的子块，（col_x,col_y）是其左下角亮度样本的位置坐标；对于 index 为 3 的子块，（col_x,col_y）是其右下角亮度样本的位置坐标。对每个子块执行以下步骤，导出每个子块的运动信息 MotionInfo[i](interPredRefMode,mvE0,refIndex0,mvE1, refIndex1)，MotionArray 由 MotionInfo0[i]构成：

1) 如果 PictureTmvpEnableFlag 的值为 0，或者参考图像队列 1 中参考索引值为 0 的图像中位置坐标为（col_x,col_y）的亮度样本所在的时域运动信息存储单元存储的预测参考模式为‘PRED_I’,或者参考图像队列 1 中参考索引为 0 的图像中位置坐标为（col_x,col_y）的所在的时域运动信息存储单元存储的权限值大于 IsPrivacyFlag，则当前预测单元的 L0 运动矢量 MvE0 为 mvT0，L1 运动矢量 MvE1 为 mvT1，并令当前预测单元的 L0 参考索引值 RefIndexL0 等于 refIndexT0，L1 参考索引值 RefIndexL1 等于 refIndexT1；

2) 否则，

- 当前预测单元的 L0 参考索引 refIndex0 和 L1 参考索引 refIndex1 均等于 0；
- 将当前图像的距离索引记为 DistanceIdexCur，当前预测单元的 L0 参考索引和 L1 参考索引对应的图像的距离索引分别记为 DistanceIndexL0 和 DistanceIndexL1，计算 BlockDistanceL0=DistanceIndexCur – DistanceIndexL0，BlockDistanceL1=DistanceIndexCur – DistanceIndexL1；
- 如果参考图像队列 1 中参考索引为 0 的图像中位置坐标为（col_x,col_y）的亮度样本所在的时域运动信息存储单元的预测参考模式为“PRED_List0”或“PRED_List01”，则将该时域运动信息存储单元 L0 运动矢量记为 mvRef(mvRef_x,mvRef_y)，该运动信息存储单元所在的图像的距离索引记为 DistanceIndexCol，该运动矢量指向的参考单元所在的图像的距离索引记为 DistanceIndexRef，计算 BlockDistanceRef = DistanceIndexCol – DistanceIndexRef；
- 否则，将该时域运动信息存储单元 L1 运动矢量记为 mvRef(mvRef_x,mvRef_y)，该运动信息存储单元所在的图像的距离索引记为 DistanceIndexCol，该运动矢量指向的参考单元所在的图像的距离索引记为 DistanceIndexRef，计算 BlockDistanceRef = DistanceIndexCol – DistanceIndexRef；
- 计算当前预测单元的 L0 运动矢量 mvE0；

```
mvE0->x = Clip3(-32768, 32767, Sign(mvRef->x * BlockDistanceL0 * BlockDistanceRef) *
        (((Abs(mvRef->x * BlockDistanceL0 * (16384 / BlockDistanceRef))) + 8192 ) >> 14))
mvE0->y = Clip3(-32768, 32767, Sign(mvRef->y * BlockDistanceL0 * BlockDistanceRef) *
        (((Abs(mvRef->y * BlockDistanceL0 * (16384 / BlockDistanceRef))) + 8192 ) >> 14))
```

- 计算当前预测单元的 L1 运动矢量 mvE1。

```
mvE1->x = Clip3(-32768, 32767, Sign(mvRef->x * BlockDistanceL1 * BlockDistanceRef) *
        (((Abs(mvRef->x * BlockDistanceL1 * (16384 / BlockDistanceRef))) + 8192 ) >> 14))
mvE1->y = Clip3(-32768, 32767, Sign(mvRef->y * BlockDistanceL1 * BlockDistanceRef) *
        (((Abs(mvRef->y * BlockDistanceL1 * (16384 / BlockDistanceRef))) + 8192 ) >> 14))
```

c) interPredRefMode 的值等于‘PRED_List01’，BgcFlag 和 BgcIndex 的值均置为 0。

8.6.7.8.5 高级运动矢量表达运动信息导出方法

T/AI 109.8-2025

本条规定导出运动信息motionInfo、BgcFlag和BgcIndex的方法。

第1步，F、G、C、A和D是当前预测单元E的相邻预测块（见图18），确定mvBaseE0、mvBaseE1、RefIndexL0、RefIndexL1、InterPredRefMode、BgcFlag和BgcIndex。

a) 如果 F 存在且 F 的权限值小于等于 IsPrivacyFlag 且采用帧间预测模式，则 F“可用”；否则，F“不可用”；

b) 如果 G 存在且 G 的权限值小于等于 IsPrivacyFlag 且采用帧间预测模式且 G 和 F 的运动信息不相同，则 G“可用”；否则，G“不可用”；

c) 如果 C 存在且 C 的权限值小于等于 IsPrivacyFlag 且采用帧间预测模式且 C 和 G 的运动信息不相同，则 C“可用”；否则，C“不可用”；

d) 如果 A 存在且 A 的权限值小于等于 IsPrivacyFlag 且采用帧间预测模式且 A 和 F 的运动信息不相同，则 A“可用”；否则，A“不可用”；

e) 如果 D 存在且 D 的权限值小于等于 IsPrivacyFlag 且采用帧间预测模式且 D 和 A 的运动信息不相同且 D 和 G 的运动信息也不相同，则 D“可用”；否则，D“不可用”；

f) 如果 F、G、C、A、D 均“不可用”且 UmveMvIndex 等于 1，则 MvBaseE0 是零矢量，RefIndexL0 等于 0，InterPredRefMode 是‘PRED_List0’，BgcFlag 和 BgcIndex 的值均置为 0；

g) 否则，如果 F、G、C、A、D 中有 UmveMvIndex+1 个“可用”，记第 UmveMvIndex+1 个可用的预测块所在的预测单元为 X，mvUmveBaseInterPredRefMode 为 X 的预测参考模式。

1) 如果 X 的预测参考模式为‘PRED_List0’，则 mvUmveBaseMv0 为 X 的 L0 运动矢量，mvUmveBaseRef0 为 X 的 L0 参考索引，mvBaseE0 等于 mvUmveBaseMv0，mvBaseE1 不存在，RefIndexL0 等于 mvUmveBaseRef0，InterPredRefMode 等于 mvUmveBaseInterPredRefMode，BgcFlag 和 BgcIndex 的值均置为 0；

2) 否则，如果 X 的预测参考模式为‘PRED_List1’，则 mvUmveBaseMv1 为 X 的 L1 运动矢量，mvUmveBaseRef1 为 X 的 L1 参考索引，mvBaseE1 等于 mvUmveBaseMv1，mvBaseE0 不存在，RefIndexL1 等于 mvUmveBaseRef1，InterPredRefMode 等于 mvUmveBaseInterPredRefMode，BgcFlag 和 BgcIndex 的值均置为 0；

3) 否则，如果 X 的预测参考模式为‘PRED_List01’，则 mvUmveBaseMv0 为 X 的 L0 运动矢量，mvUmveBaseRef0 为 X 的 L0 参考索引，mvUmveBaseMv1 为 X 的 L1 运动矢量，mvUmveBaseRef1 为 X 的 L1 参考索引，mvBaseE0 等于 mvUmveBaseMv0，mvBaseE1 等于 mvUmveBaseMv1，RefIndexL0 等于 mvUmveBaseRef0，RefIndexL1 等于 mvUmveBaseRef1，InterPredRefMode 等于 mvUmveBaseInterPredRefMode，BgcFlag 和 BgcIndex 的值分别等于(BgcFlagX && !InterPcFlag)和(BgcIndexX && !InterPcFlag)，其中 BgcFlagX 和 BgcIndexX 分别是 X 的 BgcFlag 和 BgcIndex。

h) 否则，判断当前预测单元所在的编码单元的编码单元类型。

1) 如果编码单元子类型是‘P_Skip_Umve’或‘P_Direct_Umve’，按 8.6.7.8.4.2 规定的方法导出运动矢量 mvUmveBaseMv0、参考帧索引 mvUmveBaseRef0，并令 mvUmveBaseInterPredRefMode 等于‘PRED_List0’，mvBaseE0 等于 mvUmveBaseMv0，mvBaseE1 不存在，RefIndexL0 等于 mvUmveBaseRef0，RefIndexL1 等于 -1，InterPredRefMode 等于 mvUmveBaseInterPredRefMode，BgcFlag 和 BgcIndex 的值置为 0；

2) 如果编码单元子类型是‘B_Skip_Umve’或‘B_Direct_Umve’，按 8.6.7.8.4.2 规定的方法导出 L0 参考索引 mvUmveBaseRef0、L0 运动矢量 UmveMv0、L1 参考索引 mvUmveBaseRef1 和 L1 运动矢量 UmveMv1，并令 mvUmveBaseInterPredRefMode 等于‘PRED_List01’，mvBaseE0 等于 UmveMv0，mvBaseE1 等于 UmveMv1，RefIndexL0 等于 mvUmveBaseRef0，RefIndexL1 等

T/AI 109.8-2025

于 mvUmveBaseRef1，InterPredRefMode 等于 mvUmveBaseInterPredRefMode，BgcFlag 和 BgcIndex 的值置为 0。

第2步，确定运动矢量偏移量mvOffset0和mvOffset1。

a) 根据 UmveStepIndex 和 PictureUmveStepSetIndex 的值查表 113 得到运动矢量偏移量 UmveOffset；

b) 如果 InterPredRefMode 的值为‘PRED_List0’，则 mvOffset0 的值等于 UmveOffset，mvOffset1 不存在；

c) 否则，如果 InterPredRefMode 的值为‘PRED_List1’，则 mvOffset1 的值等于 UmveOffset，mvOffset0 不存在；

d) 否则，如果 InterPredRefMode 的值为‘PRED_List01’，则参考图像队列 0 中参考索引为 RefIndexL0 的图像和参考图像队列 1 中参考索引为 RefIndexL1 的图像的距离索引分别等于 DistanceIndexL0 和 DistanceIndexL1，当前预测单元所在的图像的距离索引记为 DistanceIndexCur，当前预测单元的参考单元所在的参考图像队列 0 和参考图像队列 1 中图像的 BlockDistance 分别记为 BlockDistanceL0 和 BlockDistanceL1。

1) 按 8.6.7.1 得到 BlockDistanceL0 和 BlockDistanceL1；

2) 计算运动矢量偏移量 mvOffset0 和 mvOffset1：

```
if (Abs(BlockDistanceL1) >= Abs(BlockDistanceL0)) {
    mvOffset0 = Clip3(-32768, 32767, Sign(BlockDistanceL1 * BlockDistanceL0) * (Abs((16384 / BlockDistanceL1) * BlockDistanceL0) * UmveOffset + 8192) >> 14)
    mvOffset1 = Clip3(-32768, 32767, (16384 * UmveOffset + 8192) >> 14)
}
else {
    mvOffset0 = Clip3(-32768, 32767, (16384 × UmveOffset + 8192) >> 14)
    mvOffset1 = Clip3(-32768, 32767, Sign(BlockDistanceL1 * BlockDistanceL0) * (Abs((16384 / BlockDistanceL0) * BlockDistanceL1) * UmveOffset + 8192) >> 14)
}
```

表113 UmveOffset 与 UmveStepIndex 的对应关系

UmveStepIndex	UmveOffset	
	PictureUmveStepSetIndex值为0	PictureUmveStepSetIndex值为1
0	1	1
1	2	2
2	4	4
3	8	8
4	16	16
5	-	32
6	-	64
7	-	128

第3步，导出运动矢量：

a) 如果 InterPredRefMode 的值为‘PRED_List0’，导出当前预测单元的 mvE0；

T/AI 109.8-2025

```
if ((UmveDirIndex == 0) || (UmveDirIndex == 1)) {
    mvE0->x = mvBaseE0->x + (-1)^UmveDirIndex * mvOffset0
    mvE0->y = mvBaseE0->y
}
else if ((UmveDirIndex == 2) || (UmveDirIndex == 3)) {
    mvE0->x = mvBaseE0->x
    mvE0->y = mvBaseE0->y + (-1)^UmveDirIndex * mvOffset0
}
mvE0->x = Clip3(-32768, 32767, mvE0->x)
mvE0->y = Clip3(-32768, 32767, mvE0->y)
```

MvE 等于 mvE0。

b) 如果 InterPredRefMode 的值为‘PRED_List1’，导出当前预测单元的 mvE1：

```
if ((UmveDirIndex == 0) || (UmveDirIndex == 1)) {
    mvE1->x = mvBaseE1->x + (-1)^UmveDirIndex * mvOffset1
    mvE1->y = mvBaseE1->y
}
else if ((UmveDirIndex == 2) || (UmveDirIndex == 3)) {
    mvE1->x = mvBaseE1->x
    mvE1->y = mvBaseE1->y + (-1)^UmveDirIndex * mvOffset1
}
mvE1->x = Clip3(-32768, 32767, mvE1->x)
mvE1->y = Clip3(-32768, 32767, mvE1->y)
```

MvE 等于 mvE1。

c) 如果 InterPredRefMode 的值为‘PRED_List01’，导出当前预测单元的 mvE0 和 mvE1。

```
if ((UmveDirIndex == 0) || (UmveDirIndex == 1)) {
    mvE0->x = mvBaseE0->x + (-1)^UmveDirIndex * mvOffset0
    mvE0->y = mvBaseE0->y
    mvE1->x = mvBaseE1->x + (-1)^UmveDirIndex * mvOffset1
    mvE1->y = mvBaseE1->y
}
else if ((UmveDirIndex == 2) || (UmveDirIndex == 3)) {
    mvE0->x = mvBaseE0->x
    mvE0->y = mvBaseE0->y + (-1)^UmveDirIndex * mvOffset0
    mvE1->x = mvBaseE1->x
    mvE1->y = mvBaseE1->y + (-1)^UmveDirIndex * mvOffset1
}
mvE0->x = Clip3(-32768, 32767, mvE0->x)
mvE0->y = Clip3(-32768, 32767, mvE0->y)
mvE1->x = Clip3(-32768, 32767, mvE1->x)
mvE1->y = Clip3(-32768, 32767, mvE1->y)
```

MvE0 等于 mvE0，MvE1 等于 mvE1。

RefIndexL0 等于 mvUmveBaseRef0，RefIndexL1 等于 mvUmveBaseRef1，InterPredRefMode 等于

T/AI 109.8-2025

mvUmveBaseInterPredRefMode。

8.6.7.8.6 历史运动信息候选项导出方法

根据历史运动信息表 HmvpCandList 和 motionInfoX，按以下方法导出运动信息 motionInfo、BgcFlag 和 BgcIndex：

a) 将可选的历史运动信息候选项数量 NumAllowedCand 初始化为 Min(CntHmvp,NumOfHmvpCand)，历史运动信息表索引 hmvpIndex 初始化为 1。如果当前图像是 P 图像，则 candIndex 等于 1 + ValidMvapModeNum；如果当前图像是 B 图像，则 candIndex 等于 3+ValidMvapModeNum；

b) 如果 numAllowedCand 等于 0，结束导出过程。如果当前图像是 P 图像，则 motionInfo 等于 motionInfo1；如果当前图像是 B 图像，则 motionInfo 等于 motionInfo3，BgcFlag 和 BgcIndex 的值均置为 0；

c) 否则，执行以下操作，直到 candIndex 等于 CuSubTypeIndex，或 hmvpIndex 大于 NumAllowedCand：

1) 令 tmpHmvp 等于 HmvpCandList[CntHmvp-hmvpIndex]；

2) 按 8.6.7.2 依次判断 tmpHmvp 中的运动信息与 motionInfoX 是否相同。如果均不相同，则 candIndex 加 1，hmvpIndex 加 1；否则，hmvpIndex 加 1。

d) 如果 candIndex 等于 CuSubTypeIndex，则 motionInfo 等于 tmpHMVP 中的运动信息，BgcFlag 和 BgcIndex 分别等于 tmpHMVP 中的 BgcFlag 和 BgcIndex；如果 candIndex 小于 CuSubTypeIndex，则 motionInfo 等于 HmvpCandList[cntHmvp-1]中的运动信息，BgcFlag 和 BgcIndex 分别等于 HmvpCandList[cntHmvp-1]中的 BgcFlag 和 BgcIndex；

e) 如果 InterPcFlag 的值为 1，则将 BgcFlag 和 BgcIndex 均置为 0。

以上过程中，motionInfoX 中 X 的值如下：

——当 SbTmvpEnableFlag 的值为 0 时，如果当前图像是 P 图像，X 为 0 或 1；如果当前图像是 B 图像，X 为 0、1、2 或 3；

——当 SbTmvpEnableFlag 的值为 1 时，如果当前图像是 P 图像，X 为 1；如果当前图像是 B 图像，X 为 1、2 或 3。

8.6.7.8.7 仿射运动信息导出方法

8.6.7.8.7.1 概述

本条用于导出仿射运动信息、BgcFlag 和 BgcIndex，以及仿射样本改善信息 asrInfo0 和 asrInfo1。

首先，根据 8.6.7.8.7.3 导出仿射控制点运动矢量组、预测参考模式、参考索引、BgcFlag 和 BgcIndex。

如果 AffineUmveFlag 的值等于 1，则根据 8.6.7.8.7.4 对仿射控制点运动矢量组进行调整。

如果当前预测单元的预测参考模式是‘PRED_List0’或‘PRED_List01’，按 8.6.7.4 根据仿射控制点运动矢量组得到 MvArrayL0 和 asrInfo0；

如果当前预测单元的预测参考模式是‘PRED_List1’或‘PRED_List01’，按 8.6.7.4 根据仿射控制点运动矢量组得到 MvArrayL1 和 asrInfo1。

8.6.7.8.7.2 仿射运动矢量继承

令(xN,yN)是相邻块 X 所在预测单元的左上角样本在图像中的坐标，(xC,yC)是当前预测单元的左上角样本在图像中的坐标，widthX 和 heightX 是相邻块 X 所在的预测单元的宽度和高度，width 和

T/AI 109.8-2025

height 是当前预测单元的宽度和高度。

仿射运动继承方法如下：

a) 如果 yC % MaxQtSize 等于 0 且 yN + heightX 等于 yC，则将 isBoundy 初始化为 1，否则，将 isBoundy 初始化为 0；

b) 如果 isBoundy 等于 1，则 mvN0 是 X 所在的预测单元左下角样本所在的 4×4 子块的运动矢量，mvN1 是 X 所在的预测单元右下角样本所在的 4×4 子块的运动矢量，widthOffset 等于 xC-xN，heightOffset 等于 0；否则，mvN0 是 X 所在的预测单元左上角样本所在的 4×4 子块的运动矢量，mvN1 是 X 所在的预测单元右上角样本所在的 4×4 子块的运动矢量，mvN2 是 X 所在的预测单元左下角样本所在的 4×4 子块的运动矢量，widthOffset 等于 xC - xN，heightOffset 等于 yC-yN；

c) 执行以下操作：

1) 计算变量 mvScaleHor、mvScaleVer、dHorX、dVerX、dHorY 和 dVerY；

```
mvScaleHor = mvN0_x << 7
mvScaleVer = mvN0_y<< 7
dHorX = (mvN1->x - mvN0->x) << (7 - Log(widthX))
dHorY = (mvN1->y - mvN0->y) << (7 - Log(widthX))
```

2) 如果 X 所在的预测单元是六参数仿射预测单元且 isBoundy 为 0，则：

```
dVerX = (mvN2->x - mvN0->x) << (7 - Log(heightX))
dVerY = (mvN2->y - mvN0->y) << (7 - Log(heightX))
```

3) 否则：

```
dVerX = -dHorY
dVerY = dHorX
```

d) 如果 X 所在的预测单元是六参数仿射预测单元且 isBoundy 为 0，则当前预测单元是六参数仿射预测单元，仿射控制点运动矢量组是 mvsAffine(mv0,mv1,mv2)，计算运动矢量 mv0、mv1 和 mv2；

```
mv0->x = Clip3(-131072, 131071, Rounding(mvScaleHor + dHorX * widthOffset + dVerX * heightOffset, 7) << 2)
mv0->y = Clip3(-131072, 131071, Rounding(mvScaleVer + dHorY * widthOffset + dVerY * heightOffset, 7) << 2)
mv1->x = Clip3(-131072, 131071, Rounding(mvScaleHor + dHorX * (widthOffset + width) + dVerX * heightOffset,
        7) << 2)
mv1->y = Clip3(-131072, 131071, Rounding(mvScaleVer + dHorY * (widthOffset + width) + dVerY * heightOffset,
        7) << 2)
mv2->x= Clip3(-131072, 131071, Rounding(mvScaleHor + dHorX * widthOffset + dVerX * (heightOffset + height),
    7) << 2)
mv2->y= Clip3(-131072, 131071, Rounding(mvScaleVer + dHorY * widthOffset + dVerY * (heightOffset + height),
    7) << 2)
```

e) 否则，当前预测单元是四参数仿射预测单元，仿射控制点运动矢量组是 mvsAffine(mv0,mv1)，计算运动矢量 mv0 和 mv1。

```
mv0->x = Clip3(-131072, 131071, Rounding(mvScaleHor + dHorX * widthOffset + dVerX * heightOffset, 7) << 2)
mv0->y = Clip3(-131072, 131071, Rounding(mvScaleVer + dHorY * widthOffset + dVerY * heightOffset, 7) << 2)
mv1->x = Clip3(-131072, 131071, Rounding(mvScaleHor + dHorX * (widthOffset + width) + dVerX * heightOffset,
        7) << 2)
```

T/AI 109.8-2025

```
mv1->y = Clip3(-131072, 131071, Rounding(mvScaleVer + dHorY * (widthOffset + width) + dVerY * heightOffset, 7) << 2)
```

8.6.7.8.7.3 导出仿射运动信息

本条规定了导出当前预测单元 E 的仿射运动信息的方法。

第 1 步，确定 E 的相邻亮度样本 H。

a) 如果当前图像是 P 图像，令 H 为参考图像队列 0 中参考索引为 0 的图像中与当前预测单元 E 的右下角亮度样本位置对应的亮度样本（见图 27）。如果 PictureTmvpEnableFlag 的值为 0，或者亮度样本 H 所在的时域运动信息存储单元存储的预测参考模式为‘PRED_I’或者‘PRED_List1’，或者亮度样本 H 所在的时域运动信息存储单元存储的权限值大于 IsPrivacyFlag，则 H“不存在”；否则，H“存在”；

b) 如果当前图像是 B 图像，令 H 为参考图像队列 1 中参考索引为 0 的图像中与当前预测单元 E 的右下角亮度样本位置对应的亮度样本（见图 28）。如果 PictureTmvpEnableFlag 的值为 0，或者亮度样本 H 所在的时域运动信息存储单元存储的预测参考模式为‘PRED_I’或者‘PRED_List1’，或者亮度样本 H 所在的时域运动信息存储单元存储的权限值大于 IsPrivacyFlag，则 H“不存在”；否则，H“存在”。

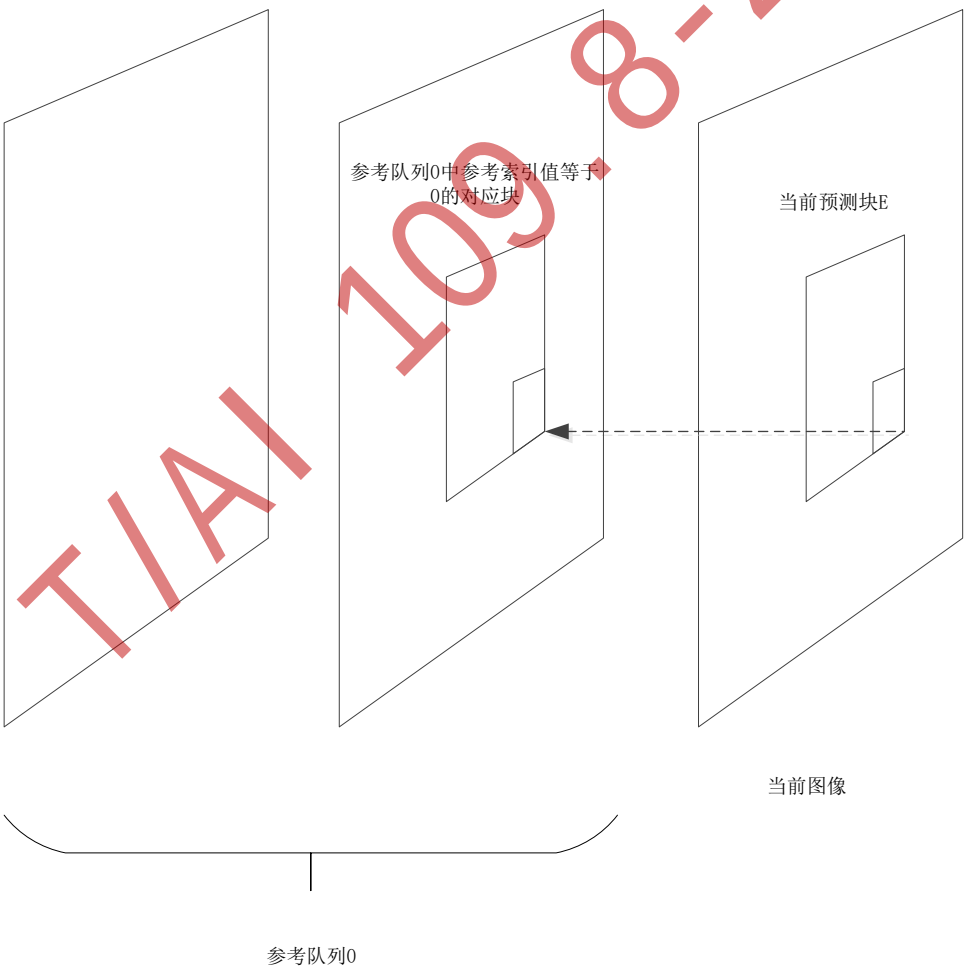


图27 预测单元 E 和相邻亮度样本 H 的空间位置关系（当前图像是 P 图像）

T/AI 109.8-2025

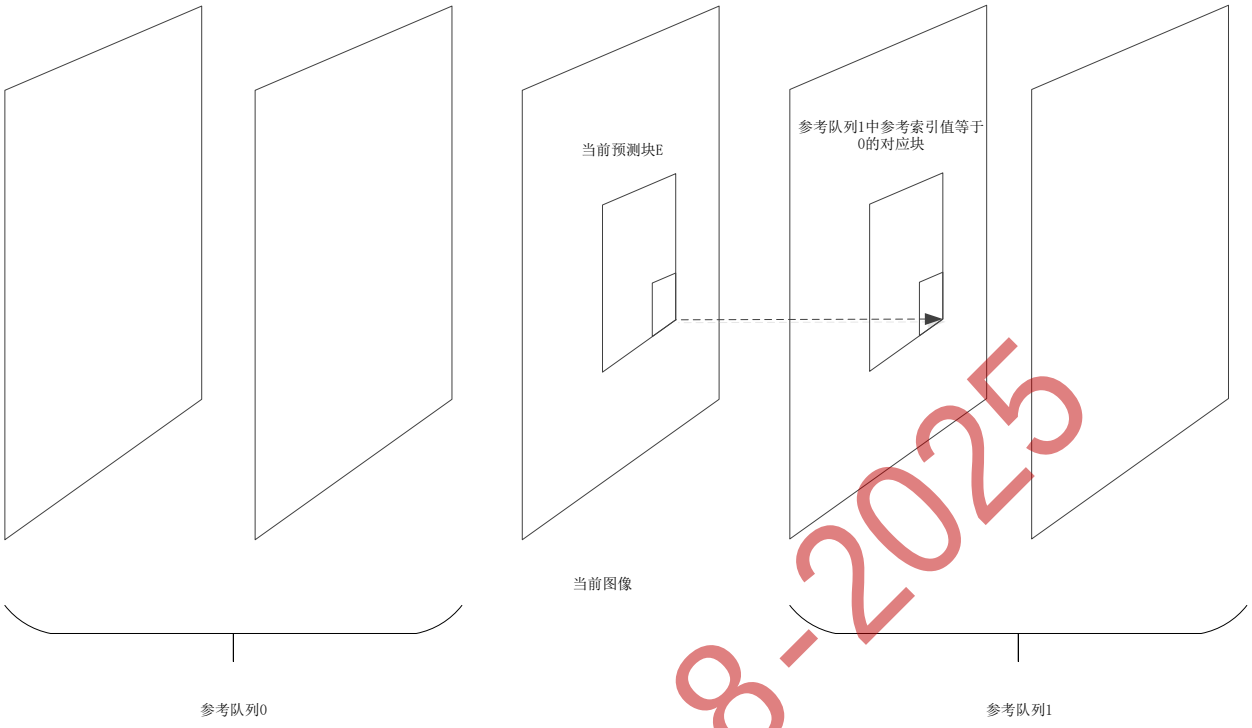


图28 预测单元 E 和相邻亮度样本 H 的空间位置关系（当前图像是 B 图像）

第 2 步，确定 E 的预测参考模式、参考索引和仿射控制点运动矢量组。

a) 当 AffineCandIndex 的值为 0 或 1 时，如果 F、G、C、A 或 D 所在的预测单元中至少有 AffineCandIndex+1 个“可用”且不同的仿射预测单元，记第 AffineCandIndex + 1 个可用的仿射预测单元包括的预测块为 X；否则，执行第 3 步；

b) E 的预测参考模式和参考索引分别等于 X 的空域运动信息存储单元的预测参考模式和参考索引；

c) 如果 X 所在的预测单元的预测参考模式为‘Pred_List0’，由 X 的空域运动信息存储单元的运动信息按 8.6.7.8.7.2 规定的方法得到 E 的 L0 仿射控制点运动矢量组 mvs_L0。E 的 BgcFlag 和 BgcIndex 均为 0；

d) 如果 X 所在的预测单元的预测参考模式为‘Pred_List1’，由 X 的空域运动信息存储单元的运动信息按 8.6.7.8.7.2 规定的方法得到 E 的 L1 仿射控制点运动矢量组 mvs_L1。E 的 BgcFlag 和 BgcIndex 均为 0；

e) 否则，E 的预测参考模式为‘Pred_List01’，由 X 的空域运动信息存储单元的运动信息按 8.6.7.8.7.2 规定的方法得到 E 的 L0 仿射控制点运动矢量组 mvs_L0 和仿射控制点运动矢量组 mvs_L1。E 的 BgcFlag 和 BgcIndex 分别等于 X 的 BgcFlag 和 BgcIndex。

第 3 步，如果 F、G、C、A 和 D 所在的预测单元都不是仿射预测单元，AffineStartOffset 等于 0；如果 F、G、C、A 或 D 所在的预测单元中存在仿射预测单元，且均是同一个仿射预测单元，则 AffineStartOffset 等于 1；否则，AffineStartOffset 等于 2。

第 4 步，确定相邻块。

a) 如果 A、B 和 D 中至少有一个“可用”且为帧间预测模式，则按 A、B、D 的顺序扫描得到第一个“可用”的相邻块 X0；否则，X0“不存在”；

T/AI 109.8-2025

b) 如果 G 和 C 中至少有一个“可用”且为帧间预测模式，则按 G、C 的顺序扫描得到第一个“可用”且为帧间预测模式的相邻块 X1；否则，X1“不存在”；

c) 如果 F“可用”且为帧间预测模式，则记相邻块 F 为 X2；否则，X2“不存在”；

d) 如果 H“存在”，则记 H 为 X3；否则，X3“不存在”。

第 5 步，导出运动矢量。

a) 如果 X3“存在”且当前图像是 B 图像，则按 8.6.7.8.4.2 规定的方法导出 L0 运动矢量 MVX3_L0 和 L1 运动矢量 MVX3_L1，X3 的 L0 和 L1 的参考索引均等于 0；否则，如果 X3“存在”且当前图像是 P 图像，则按 8.6.7.8.4.2 规定的方法导出 L0 运动矢量 MVX3_L0，X3 的 L0 参考索引等于 0，L1 参考索引等于-1；否则 MVX3_L0 和 MVX3_L1“不存在”；

b) 如果 X（X 为 X0、X1 或 X2）“存在”且 X 的 L0 参考索引不等于-1，则 X 的 L0 运动矢量为 MVX_L0；否则 MVX_L0“不存在”；

c) 如果 X（X 为 X0、X1 或 X2）“存在”且 X 的 L1 参考索引不等于-1，则 X 的 L1 运动矢量为 MVX_L1；否则 MVX_L1“不存在”；

d) 如果 MVX0_X 和 MVX2_X 均“存在”（X 为 L0 或 L1）且 X 的参考索引相同，则：

```
MVX2_1_X->x = Rounding(((MVX2_X->y - MVX0_X->y)<<7) * M / N + (MVX0_X->x << 7), 7)
MVX2_1_X->y = Rounding(-((MVX2_X->x - MVX0_X->x) << 7) * M / N + (MVX0_X->y << 7), 7)
```

其中 M 和 N 分别是当前编码单元的宽度和高度。

e) 否则，MVX2_1_X（X 为 L0 或 L1）“不存在”。

第 6 步，导出仿射控制点运动矢量组。

a) 分别判断以下 X（X 为 L0 或 L1）运动矢量组合是否“可用”（如果组合中所有元素包含的运动矢量都“存在”且对应的参考索引相同，则组合“可用”；否则，组合“不可用”）；

```
mvs1_X: { (MVX0_X->x, MVX0_X->y),
          (MVX1_X->x, MVX1_X->y),
          (MVX2_X->x, MVX2_X->y) }
mvs2_X: { (MVX0_X->x, MVX0_X->y),
          (MVX1_X->x, MVX1_X->y),
          (MVX0_X->x + MVX3_X->x - MVX1_X->x, MVX0_X->y + MVX3_X->y - MVX1_X->y) }
mvs3_X: { (MVX0_X->x, MVX0_X->y),
          (MVX0_X->x + MVX3_X->x - MVX2_X->x, MVX0_X->y + MVX3_X->y - MVX2_X->y),
          (MVX2_X->x, MVX2_X->y) }
mvs4_X: { (MVX1_X->x + MVX2_X->x - MVX3_X->x, MVX1_X->y + MVX2_X->y - MVX3_X->y),
          (MVX1_X->x, MVX1_X->y),
          (MVX2_X->x, MVX2_X->y) }
mvs5_X: { (MVX0_X->x, MVX0_X->y),
          (MVX1_X->x, MVX1_X->y) }
mvs6_X: { (MVX0_X->x, MVX0_X->y),
          (MVX2_1_X->x, MVX2_1_X->y) }
```

b) 判断运动矢量候选项 mvsX（X 为 1、2、3、4、5、6）是否“可用”：

1) 如果 mvsX_L0“可用”且 mvsX_L1“不可用”，则 mvsX 可用且 mvsX 只包含 mvsX_L0；

2) 否则，如果 mvsX_L1“可用”且 mvsX_L0“不可用”，则 mvsX“可用”且 mvsX 只包含 mvsX_L1；

3) 否则，如果 mvsX_L0 和 mvsX_L1 均“可用”，则 mvsX“可用”且 mvsX 包含 mvsX_L0 和 mvsX_L1；

T/AI 109.8-2025

4) 否则，mvsX“不可用”。

c) 如果“可用”候选项的数量大于或等于 AffineCandIndex-AffineStartOffset+1，则按 mvs1、mvs2、mvs3、mvs4、mvs5、mvs6 的顺序取第 AffineCandIndex-AffineStartOffset+1 个“可用”候选项并记为 mvs。将 mvs 中所有运动矢量的 x 分量和 y 分量均左移两位，再将 mvs 中所有运动矢量的 x 分量和 y 分量的值均限制在[-131072, 131071]区间内；否则，候选项“不存在”；

d) 如果 mvs 中只包含 mvs_L0，当前预测单元的预测参考模式是‘Pred_List0’，当前预测单元的 L0 参考索引 RefIndex0 等于 mvs_L0 对应的参考索引 refIndex0。如果 mvs_L0 中有三个运动矢量，当前预测单元是六参数仿射预测单元；否则当前预测单元是四参数仿射预测单元。当前预测单元的 L0 仿射控制点运动矢量组是 mvs_L0。当前预测单元 E 的 BgcFlag 和 BgcIndex 均为 0；

e) 如果 mvs 中只包含 mvs_L1，当前预测单元的预测参考模式是‘Pred_List1’，当前预测单元的 L1 参考索引 RefIndex1 等于 mvs_L1 对应的参考索引 refIndex1。如果 mvs_L1 中有三个运动矢量，当前预测单元是六参数仿射预测单元；否则当前预测单元是四参数仿射预测单元。当前预测单元的 L1 仿射控制点运动矢量组是 mvs_L1。当前预测单元 E 的 BgcFlag 和 BgcIndex 均为 0；

f) 如果 mvs 中包含 mvs_L0 和 mvs_L1，当前预测单元的预测参考模式是‘Pred_List01’，当前预测单元的 L0 参考索引 RefIndex0 和 L1 参考索引 RefIndex1 分别等于 mvs_L0 和 mvs_L1 对应的参考索引 refIndex0 和 refIndex1。如果 mvs_L0 或 mvs_L1 中有三个运动矢量，当前预测单元是六参数仿射预测单元；否则当前预测单元是四参数仿射预测单元。当前预测单元的 L0 仿射控制点运动矢量组是 mvs_L0，L1 仿射控制点运动矢量组是 mvs_L1。如果所取候选项为 mvs4，当前预测单元 E 的 BgcFlag 和 BgcIndex 分别等于 X1 的 BgcFlag 和 BgcIndex；否则，当前预测单元 E 的 BgcFlag 和 BgcIndex 分别等于 X0 的 BgcFlag 和 BgcIndex；

g) 如果 mvs“不存在”，当前预测单元的预测参考模式是‘Pred_List0’，当前预测单元的 L0 参考索引 RefIndexL0 为 0，当前预测单元是四参数仿射预测单元，当前预测单元的仿射控制点运动矢量组 mvsAffine 中有且仅有两个零矢量。当前预测单元 E 的 BgcFlag 和 BgcIndex 均为 0。

8.6.7.8.7.4 调整仿射控制点运动矢量组

本条规定了调整仿射控制点运动矢量组的方法。

令 mvsAffineLX 为预测单元的 LX 仿射控制点运动矢量组，mv0 和 mv1 分别是 mvsAffineLX 中的第一个和第二个运动矢量。其中，如果预测参考模式为‘PRED_List0’，则 X 为 0；如果预测参考模式为‘PRED_List1’，则 X 为 1；如果预测参考模式为‘PRED_List01’，则 X 为 0 或 1。

对 mvY（Y 等于 0 或 1）按以下方法调整：

a) 当 Y 等于 0 时，affineUmveStepIndex 的值等于 AffineUmveStepIndex0，affineUmveDirIndex 的值等于 AffineUmveDirIndex0；否则，affineUmveStepIndex 的值等于 AffineUmveStepIndex1，affineUmveDirIndex 的值等于 AffineUmveDirIndex1；

b) 根据 affineUmveStepIndex 查表 114 得到 affineMvOffset；

c) 对 mvY 进行以下调整：

```
if ((affineUmveDirIndex == 0) || (affineUmveDirIndex == 1)) {
    mvY->x = mvY->x + (-1)^affineUmveDirIndex * affineMvOffset
    mvY->y = mvY->y
```

T/AI 109.8-2025

```
}
else if ((affineUmveDirIndex == 2) || (affineUmveDirIndex == 3)) {
    mvY->x = mvY->x
    mvY->y = mvY->y+ (-1)^affineUmveDirIndex * affineMvOffset
}
mvY->x = Clip3(-131072, 131071, mvY->x)
mvY->y = Clip3(-131072, 131071, mvY->y)
```

表114 affineMvOffset 与 affineUmveStepIndex 的对应关系

affineUmveStepIndex	affineMvOffset
0	1
1	2
2	4
3	8
4	16

8.6.7.8.8 运动矢量角度预测运动信息导出方法

本条规定了导出运动矢量角度预测单元子块的运动信息阵列MotionArray、BgcFlag和BgcIndex的方法。

第1步，得到运动矢量角度预测模式号。

a) 令 curMvapMode 是当前编码单元的运动矢量角度预测模式号，将 curMvapMode 初始化为 0；

b) 如果当前图像是 P 图像，candIndex 为 1；如果当前图像是 B 图像，candIndex 为 3。执行以下操作，直到 candIndex 等于 CuSubTypeIndex，结束导出过程。

1) 如果 ValidMvapModeFlag[curMvapMode]等于 1，则 candIndex 加 1，curMvapMode 加 1；

2) 否则，curMvapMode 加 1。

第2步，填充参考运动信息列表neighborMotions[i]（i=0～2m+2n）。

令W和H是当前解码单元的宽度和高度，m和n的分别是W/4和H/4，(x,y)是当前解码单元左上角的像素坐标。$A_0, A_1, \ldots, A_{m-1}$是当前块左下角位置的4×4块，$A_m, A_{m+1}, \ldots, A_{m+n-1}$是当前块左侧位置所在的4×4块，$A_{m+n}$是当前块左上角位置所在的4×4块，$A_{m+n+1}, A_{m+n+2}, \ldots, A_{2m+n}$是当前块上边位置所在的4×4块，$A_{2m+n+1}, A_{2m+n+1}, \ldots, A_{2m+2n}$是当前块右上角位置所在的4×4块，见图25。如果i的取值范围是0～m+n，则A_i的坐标分别是(x-1,y+H+W-4×i-1)；如果i的取值范围是m+n+1～2m+2n，则A_i的坐标分别为(x+4×i-W-H-1,y-1)。

A_i位置空域存储单元的运动信息“可用”指A_i需要满足以下所有条件，否则A_i位置空域存储单元的运动信息“不可用”：

——Ai位置在图像内且A_i位置对应的编码单元与当前编码单元属于同一片；

——A_i位置对应的编码单元已解码且采用帧间预测模式；

——如果PicturePrivacyEnableFlag的值等于1且，Ai权限存储单元的权限值小于等于IsPrivacyFlag。

令 neighborMotions[i] 是参考运动信息列表中的第 i 个运动信息(interPredRefMode,mvE0,mvE1,refIndexL0,refIndexL1)，其中i的取值范围是0～2m+2n。

T/AI 109.8-2025

如果A_i位置空域存储单元的运动信息“可用”，则A_i位置存储单元的运动信息记为motionInfoA_i（i=0～2m+2n）。

a) 填充 neighborMotions[0]：

1) 如果 A_0 位置空域存储单元的运动信息“不可用”，neighborMotions[0]的预测参考模式 interPredRefMode 设为“PRED_List0”，neighborMotions[0]的 mvE0 设为零矢量，neighborMotions[0]的 refIndexL0 设为 0；

2) 否则，neighborMotions[0]设为 motionInfoA_0。

b) 按以下步骤从小到大依次遍历 A_i，填充 neighborMotions[i]。

1) 如果 A_i 位置空域存储单元的运动信息“可用”，neighborMotions[i]为 motionInfoA_i；

2) 否则，neighborMotions[i]等于 neighborMotions[i-1]。

第 3 步，导出运动信息阵列 MotionArray，BgcFlag 和 BgcIndex。

根据 curMvapMode 和 neighborMotions 确定当前编码单元内部子块的运动信息阵列 MotionArray[i][j]（interPredRefMode,MvE0,MvE1,RefIndexL0,RefIndexL1），其中 i=0～（W >> 3）-1，j=0～（H >> 3）-1，(i，j)为当前编码单元内的 8×8 子块的索引，i 为子块的水平索引值，j 为子块的垂直索引值。对每个子块进行遍历，导出运动矢量角度预测的子块运动信息阵列 MotionArray，并将 BgcFlag 和 BgcIndex 置零。

```
m=W >> 2
n= H>> 2
if (curMvapMode == 0)
    MotionArray[i][j] = neighborMotions[m + n - 1 - (j << 1)]
else if (curMvapMode == 1)
    MotionArray[i][j] = neighborMotions[m + n + 1 + (i << 1)]
else if (curMvapMode == 2)
    MotionArray[i][j] = neighborMotions[m + n - (j << 1) + (i << 1)]
else if (curMvapMode == 3)
    MotionArray[i][j] = neighborMotions[m + n - (j << 1) - (i << 1) - 3]
else if (curMvapMode == 4)
    MotionArray[i][j] = neighborMotions[m + n + (j << 1) + (i << 1) + 3]
```

8.6.7.8.9 角度加权预测模式运动信息导出方法

8.6.7.8.9.1 概述

本条规定了导出角度加权预测模式第一运动信息 MotionInfoAwp0(mvAwp0L0,mvAwp0L1,RefIndexAwp0L0, RefIndexAwp0L1,interAwpPredRefMode0)、第二运动信息 MotionInfoAwp1(mvAwp1L0,mvAwp1L1, RefIndexAwp1L0,RefIndexAwp1L1,interAwpPredRefMode1)、BgcFlag和BgcIndex的方法。

8.6.7.8.9.2 角度加权预测模式运动信息查重

如果运动信息A和运动信息B满足下面一个或多个条件，则运动信息A和运动信息B不同；否则，运动信息A和运动信息B相同：

a) 按 8.6.7.3 判定由运动信息 A 中的预测参考模式和参考索引所确定的参考图像和由运动信息 B 中的预测参考模式和参考索引所确定的参考图像不同；

T/AI 109.8-2025

b) 运动信息 A 中的预测参考模式对应的运动矢量和运动信息 B 中的预测参考模式对应的运动矢量不同。

注：预测参考模式‘PRED_List0’对应的运动矢量为L0运动矢量，预测参考模式‘PRED_List1’对应的运动矢量为L1运动矢量。

8.6.7.8.9.3 导出角度加权预测的时域运动信息

令当前预测单元的左上角亮度样本位置是(x,y)，亮度预测块的宽度和高度分别是 W 和 H。

第 1 步，计算(tx,ty)。

a) 初始化 tx = x + W，ty = y + H；

b) 如果 tx 大于当前预测单元所在的 LCU 的最右边的亮度样本位置的横坐标，或 tx 大于当前预测单元所在的图像或 patch 的最右边的亮度样本位置的横坐标，则 tx = x + W − 1；

c) 如果 ty 大于当前预测单元所在的 LCU 的最下边的亮度样本位置的纵坐标，或 ty 大于当前预测单元所在的图像或 patch 的最下边的亮度样本位置的纵坐标，则 ty = y + H − 1。

第2步，导出运动信息。

a) 如果当前图像为 P 图像：

1) 如果 PictureTmvpEnableFlag 的值为 0，或者参考图像队列 0 中参考索引值为 0 的图像中坐标位置是(tx,ty)的亮度样本所在的时域运动信息存储单元存储的预测参考模式为“PRED_I”或“PRED_L1”，或者参考图像队列 0 中参考索引值为 0 的图像中坐标位置是(tx,ty)的亮度样本所在的时域运动信息存储单元存储的权限值大于 IsPrivacyFlag，则当前预测单元的 L0 运动矢量 MvE0 为零矢量，并令当前预测单元的 L0 参考索引值 RefIndexL0 等于 0，结束第 2 步；否则，继续执行以下操作；

2) 当前预测单元的 L0 参考索引等于 0，当前预测单元的 L0 参考索引对应的图像的距离索引记为 DistanceIndexL0，当前预测单元的 L0 参考索引对应的图像的 BlockDistance 记为 BlockDistanceL0；

3) 计算 BlockDistanceRef；

```
BlockDistanceRef = DistanceIndexCol - DistanceIndexRef
```

4) 计算当前预测单元的 L0 运动矢量 mvE0；

```
mvE0_x = Clip3(-32768, 32767, Sign(mvRef_x * BlockDistanceL0 * BlockDistanceRef) * (((Abs(mvRef_x *
        BlockDistanceL0 * (16384 / BlockDistanceRef))) + 8192 ) >> 14))
mvE0_y = Clip3(-32768, 32767, Sign(mvRef_y * BlockDistanceL0 * BlockDistanceRef) * (((Abs(mvRef_y *
        BlockDistanceL0 * (16384 / BlockDistanceRef))) + 8192 ) >> 14))
```

5) interPredRefMode 的值等于‘PRED_List0’。

b) 否则（当前图像为 B 图像）。

1) 如果 PictureTmvpEnableFlag 的值为 0，或者参考图像队列 1 中参考索引值为 0 的图像中坐标位置是(tx,ty)的亮度样本所在的时域运动信息存储单元存储的预测参考模式为‘PRED_I’，或者参考图像队列 1 中参考索引值为 0 的图像中坐标位置是(tx,ty)的亮度样本所在的时域运动信息存储单元存储权限值大于 IsPrivacyFlag，则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，然后将按 8.6.7.6.2 得到的 L0 运动矢量预测值和 L1 运动矢量预测值分别作为当前预测单元的 L0 运动矢量 MvE0 和 L1 运动矢量 MvE1，并令当前预测单元的 L0 参考索引 RefIndexL0 和 L1 参考索引 RefIndexL1 均等于 0，结束第 2 步；否则，继续执行以下操作；

T/AI 109.8-2025

2) 当前预测单元的 L0 参考索引和 L1 参考索引均等于 0，当前预测单元的 L0 参考索引和 L1 参考索引对应的图像的距离索引分别记为 DistanceIndexL0 和 DistanceIndexL1，当前预测单元的 L0 参考索引和 L1 参考索引对应的图像的 BlockDistance 分别记为 BlockDistanceL0 和 BlockDistanceL1；

3) 如果参考图像队列 1 中参考索引为 0 的图像中坐标位置是(tx,ty)的亮度样本所在的时域运动信息存储单元的预测参考模式为‘PRED_L0’或‘PRED_List01’，则将该时域运动信息存储单元 L0 运动矢量记为 mvRef，该时域运动信息存储单元所在的图像的距离索引记为 DistanceIndexCol，该运动矢量指向的参考单元所在的图像的距离索引记为 DistanceIndexRef；否则，如果参考图像队列 1 中参考索引为 0 的图像中坐标位置是(tx,ty)的亮度样本所在的时域运动信息存储单元的预测参考模式为‘PRED_List1’，将该时域运动信息存储单元 L1 运动矢量记为 mvRef，该时域运动信息存储单元所在的图像的距离索引记为 DistanceIndexCol，该运动矢量指向的参考单元所在的图像的距离索引记为 DistanceIndexRef；

4) 计算 BlockDistanceRef；

```
BlockDistanceRef = DistanceIndexCol - DistanceIndexRef
```

5) 令当前预测单元的 L0 参考索引 RefIndexL0 等于 0，计算当前预测单元的 L0 运动矢量 mvE0；

```
mvE0_x = Clip3(-32768, 32767, Sign(mvRef_x * BlockDistanceL0 * BlockDistanceRef) * (((Abs(mvRef_x *
        BlockDistanceL0 * (16384 / BlockDistanceRef))) + 8192 ) >> 14))
mvE0_y = Clip3(-32768, 32767, Sign(mvRef_y * BlockDistanceL0 * BlockDistanceRef) * (((Abs(mvRef_y *
        BlockDistanceL0 * (16384 / BlockDistanceRef))) + 8192 ) >> 14))
```

6) 令当前预测单元的 L1 参考索引 RefIndexL1 等于 0，计算当前预测单元的 L1 运动矢量 mvE1；

```
mvE1_x = Clip3(-32768, 32767, Sign(mvRef_x * BlockDistanceL1 * BlockDistanceRef) * (((Abs(mvRef_x *
        BlockDistanceL1 * (16384 / BlockDistanceRef))) + 8192 ) >> 14))
mvE1_y = Clip3(-32768, 32767, Sign(mvRef_y * BlockDistanceL1 * BlockDistanceRef) * (((Abs(mvRef_y *
        BlockDistanceL1 * (16384 / BlockDistanceRef))) + 8192 ) >> 14))
```

7) interPredRefMode 的值等于‘PRED_List01’。

8.6.7.8.9.4 角度加权预测模式运动信息

本条规定了导出mvAwp0L0、mvAwp0L1，RefIndexAwp0L0、RefIndexAwp0L1、mvAwp1L0、mvAwp1L1、RefIndexAwp1L0、RefIndexAwp1L1、interAwpPredRefMode0和interAwpPredRefMode1的方法。

第1步，F、G、C、A、B和D是当前预测单元E的相邻预测块（见图18），确定F、G、C、A、B和D的“可用”性。

a) 如果 F 存在且 F 的权限值小于等于 IsPrivacyFlag 且采用帧间预测模式，则 F“可用”；否则，F“不可用”；

b) 如果 G 存在且 G 的权限值小于等于 IsPrivacyFlag 且采用帧间预测模式，则 G“可用”；否则，G“不可用”；

c) 如果 C 存在且 C 的权限值小于等于 IsPrivacyFlag 且采用帧间预测模式，则 C“可用”；否则，C“不可用”；

d) 如果 A 存在且 A 的权限值小于等于 IsPrivacyFlag 且采用帧间预测模式，则 A“可用”；否则，A“不可用”；

T/AI 109.8-2025

e) 如果 B 存在且 B 的权限值小于等于 IsPrivacyFlag 且采用帧间预测模式，则 B“可用”；否则，B“不可用”；

f) 如果 D 存在且 D 的权限值小于等于 IsPrivacyFlag 且采用帧间预测模式，则 D“可用”；否则，D“不可用”。

第2步，初始化index等于0，按8.6.7.8.9.3导出运动信息T，然后将T和可用块的运动信息（依照F、G、C、A、B、D的顺序）逐个按照如下步骤，放入运动信息候选列表AwpCandArray。

a) 如果当前运动信息的预测参考模式是‘PRED_List0’或‘PRED_List1’，将其与 AwpCandArray 中已经存在的运动信息按 8.6.7.8.9.2 进行查重操作，若均不重复则放入 AwpCandArray[index]，index 加 1。如果 index 等于 5，则转到第 4 步；

b) 否则，根据当前可用运动信息的遍历顺序编号（从 0 开始）的按照如下规则修改运动信息的预测参考模式，并将修改后的运动信息与 AwpCandArray 中已经存在的运动信息按 8.6.7.8.9.2 进行查重操作，若均不重复则放入 AwpCandArray[index]，index 加 1。如果 index 等于 5，则转到第 4 步。

1) 若遍历顺序编号为偶数，将当前运动信息的预测参考模式修改为‘PRED_List0’；

2) 若遍历顺序编号为奇数，将当前运动信息的预测参考模式修改为‘PRED_List1’。

第3步，如果index小于5，按以下方法生成运动信息motionInfo[i]（i=0～3）。motionInfo[i]的预测参考模式等于AwpCandArray[0]的预测参考模式，AwpCandArray[0]的与预测参考模式对应的运动矢量记为mv，motionInfo[i]与预测参考模式对应的运动矢量记为mvInfo[i]。将motionInfo[i]按i从0到3的顺序依次放入AwpCandArray[index]，index加1。如果index等于5，则转到第4步。

```
amv[0] = mv->x
amv[1] = mv->y
axis = i % 2
sf = i > 1 ? -1 : 1
tmp = Abs(amv[axis])
sign = amv[axis] < 0 ? 1 : 0
amv[axis] = tmp < 8 ? 8 : (tmp <= 64 ? ((tmp * (4 + sf) + 2) >> 2) : (tmp <= 128 ? ((tmp * (8 + sf) + 4) >> 3) : ((tmp * (32
        + sf) + 16) >> 5)))
amv[axis] = amv[axis] * (-1)^sign
if (i > 1 && tmp < 8)
    amv[axis] = -amv[axis]
amv[axis] = Clip3(-32768, 32767, amv[axis])
mvInfo[i]->x = amv[0]
mvInfo[i]->y = amv[1]
```

第4步，将运动信息AwpCandArray[AwpCandIndex1]赋值给motionInfoAwp0。

第5步，将运动信息AwpCandArray[AwpCandIndex0]赋值给motionInfoAwp1。

第6步，将BgcFlag和BgcIndex的值均置为0。如果AwpMvrCandFlag0或者AwpMvrCandFlag1等于1，对mvAwp0L0，mvAwp0L1以及mvAwp1L0，mvAwp1L1进行修正。

a) 根据 AwpMvrCandStep0 和 AwpMvrCandStep1 的值查表 115 分别得到运动矢量偏移量 AwpMvrOffset0 和 AwpMvrOffset1；

b) 如果 AwpMvrCandFlagX 的值等于 1，修正 mvAwpXL0，mvAwpXL1，X 为 0 或 1。

1) 如果 RefIndexAwpXL0 的值大于等于 0，则 mvAwpXL0(mvAwpXL0_x, mvAwpXL0_y)修正为：

T/AI 109.8-2025

```
if ((AwpMvrCandDirX == 0) || (AwpMvrCandDirX == 1)) {
    mvAwpXL0->x = mvAwpXL0->x + (-1) AwpMvrCandDirX * AwpMvrOffsetX
    mvAwpXL0->y = mvAwpXL0->y
}
else if ((AwpMvrCandDirX == 2) || (AwpMvrCandDirX == 3)) {
    mvAwpXL0->x = mvAwpXL0->x
    mvAwpXL0->y = mvAwpXL0->y + (-1) AwpMvrCandDirX * AwpMvrOffsetX
}
mvAwpXL0->x = Clip3(-32768, 32767, mvAwpXL0->x)
mvAwpXL0->y = Clip3(-32768, 32767, mvAwpXL0->y)
```

2) 否则(RefIndexAwpXL1 的值大于等于 0)，mvAwpXL1(mvAwpXL1_x, mvAwpXL1_y)修正为：

```
if ((AwpMvrCandDirX == 0) || (AwpMvrCandDirX == 1)) {
    mvAwpXL1->x = mvAwpXL1->x + (-1) AwpMvrCandDirX * AwpMvrOffsetX
    mvAwpXL1->y = mvAwpXL1->y
}
else if ((AwpMvrCandDirX == 2) || (AwpMvrCandDirX == 3)) {
    mvAwpXL1->x = mvAwpXL1->x
    mvAwpXL1->y = mvAwpXL1->y + (-1) AwpMvrCandDirX * AwpMvrOffsetX
}
mvAwpXL1->x = Clip3(-32768, 32767, mvAwpXL1->x)
mvAwpXL1->y = Clip3(-32768, 32767, mvAwpXL1->y)
```

其中，在 AwpMvrCandDirX 和 AwpMvrOffset 中，X 的值为 0 或 1。

表115 AwpMvrCandStep0、AwpMvrCandStep1 与 AwpMvrOffset0、AwpMvrOffset1 的对应关系

AwpMvrCandStep0, AwpMvrCandStep1	AwpMvrOffset0, AwpMvrOffset1
0	1
1	2
2	4
3	8
4	16

8.6.7.8.10 增强时域运动信息导出方法

8.6.7.8.10.1 概述

按8.6.7.8.10.2导出增强时域运动信息阵列MotionArray、BgcFlag和BgcIndex。

8.6.7.8.10.2 增强时域运动信息导出方法

本条导出增强时域运动矢量预测模式的运动信息阵列MotionArray、BgcFlag和BgcIndex。

第1步,导出缺省运动信息motionInfoDefault。

F是当前预测单元E的相邻预测块（见图18）。将F的运动信息按8.6.7.8.10.3导出当前预测单元的缺省运动信息motionInfoDefault(mvDefaultL0,mvDefaultL1,refDefaultL0,refDefaultL1 ，

T/AI 109.8-2025

predModeDefault)，其中参考预测单元为F所在的空域存储单元，mvDefaultLX等于mvEX，refDefaultLX等于refIndexLX，预测参考模式predModeDefault等于interPredRefMode。如果当前图像是P图像，X为0；如果当前图像是B图像，X为0或1。

第2步，导出坐标(Mx,My)。

令(Xpos,Ypos)是当前预测单元在图像中的左上角坐标，(x_ctb_pos,y_ctb_pos)是当前最大编码单元在图像中的左上角坐标，cu_width和cu_height分别是当前预测单元的宽度和高度，PicWidthInLuma和PicHeightInLuma分别是当前图像的宽度和高度，lcu_size是当前最大编码单元的大小。如果当前图像是P图像，时域参考图像是参考图像队列0中参考帧索引值为0的参考图像，如果当前图像是B图像，时域参考图像是参考图像队列1中参考帧索引值为0的参考图像。按以下方法导出坐标(Mx,My)：

```
Mx = Clip3(x_ctb_pos, min(x_ctb_pos + lcu_size - cu_width, PicWidthInLuma - cu_width), ((Xpos+4)>>3)<<3)
My = Clip3 (y_ctb_pos, min(y_ctb_pos + lcu_size - cu_height, PicHeightInLuma - cu_height), ((Ypos+4)>>3)<<3)
```

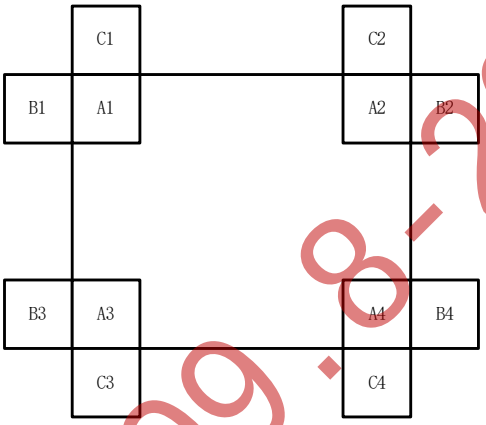


图29 原始匹配块与空间相邻块的关系

第3步，导出(Px,Py)。

令时域参考图像中的左上角坐标是(Mx,My)，所在的时域参考块为当前图像的原始匹配块，原始匹配块的宽度和高度与当前预测单元的宽度和高度一致。A1～A4，B1～B4，C1～C4是时域参考图像中位于原始匹配块周围的相邻4×4块（见图29），按以下方法导出坐标(Px,Py)。

a) 构建增强时域候选列表 EtmvpCandArray[i]（i=0～4），其中判断运动信息是否相同的方法见8.6.7.2；
   1) 将 EtmvpCandArray[i]（i=0～4）初始化为 0；length 初始化为 0；
   2) 令 EtmvpCandArray[0]等于 0，length 加 1；
   3) 如果 My+8 小于或等于 Min(y_ctb_pos + lcu_size - cu_height, PicHeightInLuma - cu_height)，且 A1 与 C3 位置的运动信息不同或 A2 与 C4 位置的运动信息不同，则 EtmvpCandArray[length]等于 1，length 加 1；
   4) 如果 Mx+8 小于或等于 Min(x_ctb_pos + lcu_size - cu_width, PicWidthInLuma - cu_width)，且 A1 与 B2 位置的运动信息不同或 A3 与 B4 位置的运动信息不同，则 EtmvpCandArray[length]等于 2，length 加 1；
   5) 如果 My-8 大于或等于 y_ctb_pos，且 A3 与 C1 位置的运动信息不同或 A4 与 C2 位置的运动信息不同，则 EtmvpCandArray[length]等于 3，length 加 1；
   6) 如果 Mx-8 大于或等于 x_ctb_pos，且 A2 与 B1 位置的运动信息不同或 A4 与 B3 位置的运动信息不同，则 EtmvpCandArray[length]等于 4，length 加 1；

T/AI 109.8-2025

7） 如果 length 小于 5，执行以下操作：

```
while (length < 5) {
    EtmvpCandArray[length] = EtmvpCandArray[length-1]
    length++
}
```

b） 根据 EtmvpCandIndex 和 EtmvpCandArray 查表 116 得到(Px,Py)。

表116 EtmvpCandArray[EtmvpCandIndex]与 Px、Py 的对应关系

EtmvpCandArray[EtmvpCandIndex]值	Px 值	Py 值
0	Mx	My
1	Mx	My + 8
2	Mx + 8	My
3	Mx	My - 8
4	Mx - 8	My

第4步，导出运动信息阵列MotionArray和BgcFlag和BgcIndex。

令 MotionArray[i][j] 是当前预测单元内部 8×8 子块的运动信息(interPredRefMode,MvE0,MvE1,RefIndexL0, RefIndexL1)，(i,j)是当前预测单元内8×8子块的索引，i是子块水平索引值，j是子块垂直索引值，i=0～(cu_width >> 3)-1，j=0～(cu_height >> 3)-1)。

令子块的时域运动信息是时域参考图像中左上角(Px+(i<<3),Py+(j<<3))位置所在的时域存储单元的运动信息。子块的时域运动信息“可用”指时域参考图像中左上角(Px+(i<<3)，Py+(j<<3))位置所在的时域存储单元采用帧间预测，同时，如果PrivacyModeIndex的值为2或3，且时域参考图像左上角(Px+(i<<3)，Py+(j<<3))位置所在的时域运动信息存储单元的权限值小于等于IsPrivacyFlag；否则子块的时域运动信息“不可用”。

a） 根据(Px,Py)确定当前预测单元的每个 8×8 子块的运动信息阵列 MotionArray，并将 BgcFlag 和 BgcIndex 的值均置为 0；

b） 依次遍历时域参考图像左上角(Px+(i<<3)，Py+(j<<3))位置的每个尺寸大小为 8×8 的子块，导出运动信息阵列 MotionArray[i][j]：

1） 如果当前子块的时域运动信息“可用”，将子块的时域运动信息按 8.6.7.8.10.3 导出 MotionArray[i][j],其中参考预测单元是时域参考图像中(Px+(i<<3),Py+(j<<3))位置所在时域存储单元，MotionArray[i][j]中的 MvEX 等于 mvEX，MotionArray[i][j]中的 RefIndexLX 等于 refIndexLX，MotionArray[i][j]中的 interPredRefMode 等于 interPredRefMode。如果当前块所在图像是 P 图像，则 X 为 0；如果当前块所在图像是 B 图像，则 X 为 0 或 1；

2） 否则，MotionArray[i][j]等于 motionInfoDefault。

8.6.7.8.10.3 增强时域运动信息辅助导出方法

如果当前图像是 P 图像，按以下方法导出时域运动信息。

a） 如果参考预测单元的预测参考模式是‘PRED_List0’，将参考预测单元所在图像的距离索引记为 DistanceIndexCol,参考预测单元的参考图像队列 0 中对应的图像的距离索引记为 DistanceIndexRef，参考预测单元所在参考图像队列 0 中图像的 BlockDistance 分别记为

T/AI 109.8-2025

BlockDistanceRef，参考预测单元的 L0 运动矢量记为 mvRef。将当前预测单元所在的图像的距离索引记为 DistanceIndexCur，当前预测单元的 L0 参考索引对应的图像的距离索引分别记为 DistanceIndexL0，当前预测单元的 L0 参考索引对应的图像的 BlockDistance 记为 BlockDistanceL0；

1) 计算 BlockDistanceRef：

```
BlockDistanceRef = DistanceIndexCol - DistanceIndexRef
```

2) 当前预测单元的预测参考模式 interPredRefMode 是‘PRED_List0’，当前预测单元的 L0 参考索引 refIndexL0 等于 0，计算当前预测单元的 L0 运动矢量 mvE0：

```
mvE0->x = Clip3(-32768, 32767, Sign(mvRef->x * BlockDistanceL0 * BlockDistanceRef) * (((Abs(mvRef->x
        * BlockDistanceL0 * (16384 / BlockDistanceRef))) + 8192 ) >> 14))
mvE0->y = Clip3(-32768, 32767, Sign(mvRef->y * BlockDistanceL0 * BlockDistanceRef) * (((Abs(mvRef->y
        * BlockDistanceL0 * (16384 / BlockDistanceRef))) + 8192 ) >> 14))
```

b) 否则，当前预测单元的预测参考模式 interPredRefMode 是‘PRED_List0’；当前预测单元的 L0 参考索引 refIndexL0 等于 0，当前预测单元的 L0 运动矢量 mvE0 是零矢量。

如果当前图像是 B 图像，按以下方法导出时域运动信息。

a) 如果参考预测单元的预测参考模式是‘PRED_List0’，将参考预测单元所在图像的距离索引记为 DistanceIndexCol，参考预测单元的参考图像队列 0 中对应的图像的距离索引记为 DistanceIndexRef，参考预测单元所在参考图像队列 0 中图像的 BlockDistance 分别记为 BlockDistanceRef，参考预测单元的 L0 运动矢量记为 mvRef。将当前预测单元所在的图像的距离索引记为 DistanceIndexCur，当前预测单元的 L0 参考索引对应的图像的距离索引分别记为 DistanceIndexL0，当前预测单元的 L0 参考索引对应的图像的 BlockDistance 记为 BlockDistanceL0；

1) 计算 BlockDistanceRef：

```
BlockDistanceRef = DistanceIndexCol - DistanceIndexRef
```

2) 当前预测单元的预测参考模式 interPredRefMode 是‘PRED_List0’，当前预测单元的 L0 参考索引 refIndexL0 等于 0，计算当前预测单元的 L0 运动矢量 mvE0：

```
mvE->x = Clip3(-32768, 32767, Sign(mvRef->x * BlockDistanceL0 * BlockDistanceRef) * (((Abs(mvRef->x
        * BlockDistanceL0 * (16384 / BlockDistanceRef))) + 8192 ) >> 14))
mvE->y = Clip3(-32768, 32767, Sign(mvRef->y * BlockDistanceL0 * BlockDistanceRef) * (((Abs(mvRef->y
        * BlockDistanceL0 * (16384 / BlockDistanceRef))) + 8192 ) >> 14))
```

b) 否则，如果参考预测单元的预测参考模式是‘PRED_List1’，将参考预测单元所在图像的距离索引记为 DistanceIndexCol，参考预测单元的参考图像队列 1 中对应的图像的距离索引记为 DistanceIndexRef，参考预测单元所在参考图像队列 1 中图像的 BlockDistance 分别记为 BlockDistanceRef，参考预测单元的 L1 运动矢量记为 mvRef。将当前预测单元所在的图像的距离索引记为 DistanceIndexCur，当前预测单元的 L1 参考索引对应的图像的距离索引分别记为 DistanceIndexL1，当前预测单元的 L1 参考索引对应的图像的 BlockDistance 记为 BlockDistanceL1；

1) 计算 BlockDistanceRef：

```
BlockDistanceRef = DistanceIndexCol - DistanceIndexRef
```

2) 当前预测单元的预测参考模式 interPredRefMode 是‘PRED_List1’，当前预测单元的 L1 参考索引 refIndexL1 等于 0，计算当前预测单元的 L1 运动矢量 mvE1：

T/AI 109.8-2025

```
mvE->x = Clip3(-32768, 32767, Sign(mvRef->x * BlockDistanceL1 * BlockDistanceRef) * (((Abs(mvRef->x
        * BlockDistanceL1 * (16384 / BlockDistanceRef))) + 8192 ) >> 14))
mvE->y = Clip3(-32768, 32767, Sign(mvRef->y * BlockDistanceL1 * BlockDistanceRef) * (((Abs(mvRef->y
        * BlockDistanceL1 * (16384 / BlockDistanceRef))) + 8192 ) >> 14))
```

c) 否则，如果参考预测单元的预测参考模式是‘PRED_List01’，将参考预测单元所在图像的距离索引记为DistanceIndexCol,参考预测单元的参考图像队列0和参考图像队列1中对应的图像的距离索引记为DistanceIndexRef0和DistanceIndexRef1，参考预测单元所在参考图像队列0和参考图像队列1中图像的BlockDistance分别记为BlockDistanceRef0和BlockDistanceRef1，参考预测单元的L0和L1运动矢量分别记为mvRef0和mvRef1。将当前预测单元所在的图像的距离索引记为DistanceIndexCur，当前预测单元的L0和L1参考索引对应的图像的距离索引分别记为DistanceIndexL0和DistanceIndexL1；当前预测单元的L0和L1参考索引对应的图像的BlockDistance记为BlockDistanceL0和BlockDistanceL1；

1) 计算BlockDistanceRef0和BlockDistanceRef1：

```
BlockDistanceRef0 = DistanceIndexCol - DistanceIndexRef0
BlockDistanceRef1 = DistanceIndexCol - DistanceIndexRef1
```

2) 当前预测单元的预测参考模式interPredRefMode是‘PRED_List01’，当前预测单元的L0参考索引refIndexL0和L1参考索引refIndexL1均等于0,分别计算当前预测单元的L0运动矢量mvE0和L1运动矢量mvE1：

```
mvE0->x = Clip3(-32768, 32767, Sign(mvRef0->x * BlockDistanceL0 * BlockDistanceRef0) *
        (((Abs(mvRef0->x * BlockDistanceL0 * (16384 / BlockDistanceRef0))) + 8192 ) >> 14))
mvE0->y = Clip3(-32768, 32767, Sign(mvRef0->y * BlockDistanceL0 * BlockDistanceRef0) *
        (((Abs(mvRef0->y * BlockDistanceL0 * (16384 / BlockDistanceRef0))) + 8192 ) >> 14))
mvE1->x = Clip3(-32768, 32767, Sign(mvRef1->x * BlockDistanceL1 * BlockDistanceRef1) *
        (((Abs(mvRef1->x * BlockDistanceL1 * (16384 / BlockDistanceRef1))) + 8192 ) >> 14))
mvE1->y = Clip3(-32768, 32767, Sign(mvRef1->y * BlockDistanceL1 * BlockDistanceRef1) *
        (((Abs(mvRef1->y * BlockDistanceL1 * (16384 / BlockDistanceRef1))) + 8192 ) >> 14))
```

d) 否则，将当前预测单元的预测参考模式interPredRefMode设为‘PRED_List01’，当前预测单元的L0参考索引refIndexL0和L1参考索引均为0,当前预测单元的L0运动矢量mvE0和L1运动矢量mvE1均为零矢量。

8.6.7.9 更新历史运动信息表

完成当前预测单元的解码后，如果满足以下条件之一，则不执行本条规定的操作。

a) 当前预测单元是仿射预测单元或角度加权预测单元；

b) 当前编码单元的编码单元子类型为‘P_Skip_Mvap’或‘P_Direct_Mvap’或‘B_Skip_Mvap’或‘B_Direct_Mvap’；

c) 当前编码单元的编码单元子类型为‘P_Skip_Etmvp’或‘P_Direct_Etmvp’或‘B_Skip_Etmvp’或‘B_Direct_Etmvp’；

d) 当前编码单元的编码单元子类型为‘P_Skip_SbTemporal’或‘P_Direct_SbTemporal’或‘B_Skip_SbTemporal’或‘B_Direct_SbTemporal’；

e) 当前编码单元的PrivacyLevel的值为1。

当NumOfHmvpCand大于0时，根据当前预测块的运动信息、BgcFlag和BgcIndex更新历史运动信息表HmvpCandList；否则，不执行本条规定的操作。

T/AI 109.8-2025

更新历史运动信息表的方法如下：

a) 将 hmvpIndex 初始化为 0；

b) 如果 CntHmvp 等于 0，则 HmvpCandList[CntHmvp] 中的运动信息、BgcFlag 和 BgcIndex 分别等于当前预测单元的运动信息、BgcFlag 和 BgcIndex，CntHmvp 加 1；

c) 否则，按 8.6.7.2 判断当前预测块的运动信息和 HmvpCandList[hmvpIndex]是否相同：

1) 如果运动信息相同，执行步骤 d)，否则，hmvpIndex 加 1；

2) 如果 hmvpIndex 小于 CntHmvp，执行步骤 c)；否则，执行步骤 d)。

d) 如果 hmvpIndex 小于 CntHmvp，则：

1) i 从 hmvpIndex 到 CntHmvp-1，令 HmvpCandList[i]等于 HmvpCandList[i+1]；

2) HmvpCandList[CntHmvp-1] 中的运动信息、BgcFlag 和 BgcIndex 分别等于当前预测单元的运动信息、BgcFlag 和 BgcIndex。

否则，如果 hmvpIndex 等于 CntHmvp 且 CntHmvp 等于 NumOfHmvpCand，则：

1) i 从 0 到 CntHmvp-1，令 HmvpCandList[i]等于 HmvpCandList[i+1]；

2) HmvpCandList[CntHmvp-1] 中的运动信息、BgcFlag 和 BgcIndex 分别等于当前预测单元的运动信息、BgcFlag 和 BgcIndex。

否则，如果 hmvpIndex 等于 CntHmvp 且 CntHmvp 小于 NumOfHmvpCand，则 HmvpCandList[CntHmvp] 中的运动信息、BgcFlag 和 BgcIndex 分别等于当前预测单元的运动信息、BgcFlag 和 BgcIndex，CntHmvp 加 1。

8.6.7.10 运动信息的存储

8.6.7.10.1 导出角度加权预测模式运动信息存储方式

令W和H是当前编码单元的宽度和高度，(x,y)是当前编码单元中的每个4×4块的左上角像素位置，posX等于((x >> 2) << 2) + 2，posY等于((y >> 2) << 2) + 2。按以下方法导出每个4 × 4块的运动信息存储方式AwpMvType：

a) 计算 stepIndex、angleIndex 和 angelAreaIndex 的值；

```
stepIndex = (AwpIndex >> 3) - 3
modAngNum = AwpIndex % 8
if (modAngNum == 2)
    angleIndex = 7
else if (modAngNum == 6)
    angleIndex = 8
else
    angleIndex = modAngNum % 2
angleAreaIndex = modAngNum >> 1
```

b) 根据 angleAreaIndex 查表 117 得到 tP 和 fP，并计算当前编码单元中每个 4 × 4 块的运动信息存储方式 AwpMvType。

```
if (tP >= fP)
    AwpMvType = 0
else
    AwpMvType = 1
```

T/AI 109.8-2025

表117 angleAreaIndex 和 fP、tP 的对应关系

angleAreaIndex	fP	tP
0	H + (W >> angleIndex) - 2 + stepIndex ×(((H + (W >> angleIndex)) >>2) -1)	(posY << 1) + ((posX << 1) >> angleIndex)
1	H + (W >> angleIndex) + stepIndex ×(((H + (W >> angleIndex)) >> 2) -1) - ((W << 1) >> angleIndex)	(posY << 1) - ((posX << 1) >> angleIndex)
2	W + (H >> angleIndex) + stepIndex ×(((W + (H >> angleIndex)) >> 2) -1) - ((H << 1) >> angleIndex)	(posX << 1) - ((posY << 1) >> angleIndex)
3	W + (H >> angleIndex) - 2 + stepIndex ×(((W + (H >> angleIndex)) >> 2) - 1)	(posX << 1) + ((posY << 1) >> angleIndex)

8.6.7.10.2 空域运动信息的存储

首先，确定样本(x,y)对应的空域运动信息存储单元的左上角像素点位置(x_0, y_0)：

$x_0 = (x >> 2) << 2$

$y_0 = (y >> 2) << 2$

其次，设 X 是(x_0, y_0)亮度样本所在的预测单元，RefIndexL0 和 RefIndexL1 分别是 X 的 L0 参考索引和 L1 参考索引，interPredRefMode 是 X 的预测参考模式，按以下方法将运动信息存储到样本(x,y)对应的空域运动信息存储单元：

a) 如果(x_0, y_0)亮度样本所在的预测单元的预测类型是普通帧内预测或块复制帧内预测或串复制帧内预测，则该运动信息存储单元的 L0 运动矢量和 L1 运动矢量均为零矢量，L0 参考索引和 L1 参考索引均为-1，该运动信息存储单元的 BgcFlag 和 BgcIndex 均为 0，该运动信息存储单元的预测参考模式设为‘PRED_I’；

b) 否则，如果 X 的 AffineFlag 值为 0，且 X 的 AwpFlag 值为 0，该运动信息存储单元的预测参考模式是 interPredRefMode；

1) 如果 interPredRefMode 等于 0，则该运动信息存储单元的 L0 运动矢量和 L0 参考索引分别等于 X 的 L0 运动矢量和 L0 参考索引，该运动信息存储单元的 L1 运动矢量是零矢量，L1 参考索引等于-1，该运动信息存储单元的 BgcFlag 和 BgcIndex 均为 0；

2) 否则，如果 interPredRefMode 等于 1，则该运动信息存储单元的 L1 运动矢量和 L1 参考索引分别等于 X 的 L1 运动矢量和 L1 参考索引，该运动信息存储单元的 L0 运动矢量是零矢量，L0 参考索引等于-1，该运动信息存储单元的 BgcFlag 和 BgcIndex 均为 0；

3) 否则，如果 interPredRefMode 等于 2，则该运动信息存储单元的 L0 运动矢量和 L0 参考索引分别等于 X 的 L0 运动矢量和 L0 参考索引，该运动信息存储单元的 L1 运动矢量和 L1 参考索引分别等于 X 的 L1 运动矢量和 L1 参考索引，该运动信息存储单元的 BgcFlag 和 BgcIndex 分别等于 X 的 BgcFlag 和 BgcIndex。

c) 否则，如果 X 的 AffineFlag 值为 0，且 X 的 AwpFlag 值为 1，按 8.6.7.10.1 导出角度加权预测模式的存储方式 AwpMvType。BgcFlag 和 BgcIndex 均为 0；

T/AI 109.8-2025

1) 如果 AwpMvType 等于 0，则该运动信息存储单元的 L0 运动矢量和 L0 参考索引分别等于 mvAwp0L0 和 RefIndexAwp0L0，该运动信息存储单元的 L1 运动矢量和 L1 参考索引分别等于 mvAwp0L1 和 RefIndexAwp0L1；
2) 否则，如果 AwpMvType 等于 1，则该运动信息存储单元的 L0 运动矢量和 L0 参考索引分别等于 mvAwp1L0 和 RefIndexAwp1L0，该运动信息存储单元的 L1 运动矢量和 L1 参考索引分别等于 mvAwp1L1 和 RefIndexAwp1L1。

d) 否则，该运动信息存储单元的预测参考模式是 interPredRefMode，该运动信息存储单元的仿射类型等于该预测单元的仿射类型。
1) 如果 interPredRefMode 等于 0，则该运动信息存储单元的 L0 运动矢量和 L0 参考索引分别等于 mvAffineL0 和 RefIndexL0，该运动信息存储单元的 L1 运动矢量是零矢量，L1 参考索引等于-1，该运动信息存储单元的 BgcFlag 和 BgcIndex 均为 0；
2) 否则，如果 interPredRefMode 等于 1，则该运动信息存储单元的 L1 运动矢量和 L1 参考索引分别等于 mvAffineL1 和 RefIndexL1，该运动信息存储单元的 L0 运动矢量是零矢量，L0 参考索引等于-1，该运动信息存储单元的 BgcFlag 和 BgcIndex 均为 0；
3) 否则，如果 interPredRefMode 等于 2，则该运动信息存储单元的 L0 运动矢量和 L0 参考索引分别等于 mvAffineL0 和 RefIndexL0，该运动信息存储单元的 L1 运动矢量和 L1 参考索引分别等于 mvAffineL1 和 RefIndexL1， 该运动信息存储单元的 BgcFlag 和 BgcIndex 分别等于 X 的 BgcFlag 和 BgcIndex。

其中，mvE0 是 MvArrayL0 在(x_0, y_0)位置的 4×4 单元的 L0 运动矢量，mvE1 是 MvArrayL1 在(x_0, y_0)位置的 4×4 单元的 L1 运动矢量。

mvAffineL0 和 mvAffineL1 为：

```
mvAffineL0->x = Clip3(-32768, 32767, Rounding(mvE0->x,2))
mvAffineL0->y = Clip3(-32768, 32767, Rounding(mvE0->y,2))
mvAffineL1->x = Clip3(-32768, 32767, Rounding(mvE1->x,2)
mvAffineL1->y = Clip3(-32768, 32767, Rounding(mvE1->y,2))
```

8.6.7.10.3 时域运动信息的存储

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：

```
x0 = (x >> 4) << 4
y0 = (y >> 4) << 4
if (x0 + 16 >= PictureWidthInLuma[LayerId])
    nWidth = PictureWidthInLuma[LayerId]  -  x0
else
    nWidth = 16
if (y0 + 16 >= PictureHeightInLuma[LayerId])
    nHeight = PictureHeightInLuma[LayerId]  -  y0
else
    nHeight = 16
```

其次，将(x0+nWidth/2, y0+nHeight/2)样本对应的空域权限存储单元的权限存储到对应的时域运动信息存储单元；

然后，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：

T/AI 109.8-2025

a) 如果 EtmvpMvapEnableFlag、SbTmvpEnableFlag 和 AwpEnableFlag 的值均为 0：
   1) 如果(x0+nWidth/2,y0+nHeight/2)亮度样本所在编码单元的预测类型为普通帧内预测或块复制帧内预测或串复制帧内预测，或(x0+nWidth/2,y0+nHeight/2)亮度样本所在预测单元的预测参考模式是‘PRED_List1’，则将该运动信息存储单元的 L0 参考索引值和 L1 参考索引值均设为-1，将该运动信息存储单元的预测参考模式设为‘PRED_I’；
   2) 否则，将该运动信息存储单元的运动矢量和参考索引设为(x0+nWidth/2,y0+nHeight/2)亮度样本所在 4×4 预测块的 L0 运动矢量和 L0 参考索引，并将该运动信息存储单元的预测参考模式设为‘PRED_List0’；
   3) 该运动信息存储单元的 BgcFlag 和 BgcIndex 均为 0。
b) 如果 EtmvpMvapEnableFlag 的值为 1 或 SbTmvpEnableFlag 的值为 1 或 AwpEnableFlag 的值为 1：
   1) 如果(x0+nWidth/2,y0+nHeight/2)亮度样本所在编码单元的预测类型为普通帧内预测或块复制帧内预测或串复制帧内预测，则将该运动信息存储单元的预测参考模式设为‘PRED_I’；
   2) 否则，如果(x0+nWidth/2,y0+nHeight/2)亮度样本所在预测单元的预测参考模式是‘PRED_List0’将该运动信息存储单元的 L0 的运动矢量和参考索引设为(x0+nWidth/2,y0+nHeight/2)亮度样本所在 4×4 预测块的 L0 的运动矢量和参考索引，将该运动信息存储单元的预测参考模式设为‘PRED_List0’；如果(x0+nWidth/2,y0+nHeight/2)亮度样本所在预测单元的预测参考模式是‘PRED_List1’将该运动信息存储单元的 L1 的运动矢量和参考索引设为(x0+nWidth/2,y0+nHeight/2)亮度样本所在 4×4 预测块的 L1 的运动矢量和参考索引，将该运动信息存储单元的预测参考模式设为‘PRED_List1’；如果(x0+nWidth/2,y0+nHeight/2)亮度样本所在预测单元的预测参考模式是‘PRED_List01’，将该运动信息存储单元的 L0 的运动矢量和参考索引设为(x0+nWidth/2,y0+nHeight/2)亮度样本所在 4×4 预测块的 L0 运动矢量和参考索引，将该运动信息存储单元的 L1 的运动矢量和参考索引设为(x0+nWidth/2,y0+nHeight/2)亮度样本所在 4×4 预测块的 L1 的运动矢量和参考索引，并将该运动信息存储单元的预测参考模式设为‘PRED_List01’；
   3) 该运动信息存储单元的 BgcFlag 和 BgcIndex 均为 0。

8.6.7.11 权限的存储

8.6.7.11.1 空域权限存储方式

首先，确定样本(x,y)对应的空域运动信息存储单元的左上角像素点位置(x0,y0)：

$$x_0 = (x >> 3) << 3$$
$$y_0 = (y >> 3) << 3$$

其次，设 X 是(x0,y0)亮度样本所在的预测单元，PrivacyLevel 是 X 的权限值，将 PrivacyLevel 的值存储到样本(x,y)对应的空域权限存储单元。

8.6.7.11.2 时域权限存储方式

首先，确定样本(x,y)对应的时域权限存储单元的左上角像素点位置(x0,y0)，将该样本对应的空域权限存储单元的权限值存储到样本(x,y)对应的时域权限存储单元。

8.6.7.12 导出相邻块单向运动信息矩阵

T/AI 109.8-2025

当前编码单元的宽度和高度为 W 和 H，当前编码单元左上角样本位置为(xCb,yCb)。

当前编码单元上侧相邻块单向运动信息矩阵为 MotionInfoTop[xSbIndex]，左侧相邻块单向运动信息矩阵为 MotionInfoLeft[ySbIndex],其中 xSbIndex 的取值范围是 0～m-1，ySbIndex 的取值范围是 0～n-1，m 和 n 的分别是 W/4 和 H/4，其中 MotionInfoTop 和 MotionInfoLeft 中每个运动信息单元包含 mvE0，mvE1，refIndexL0,refIndexL1,interPredRefMode。

第 1 步，导出上侧相邻块单向运动信息矩阵 MotionInfoTop。

对于 xSbIndex 从 0～m-1,上侧相邻子块位置(xAbove,yAbove)为((xCb>>2)+xSbIndex,(yCb>>2)-1)。记(xAbove,yAbove)对应的亮度样本所在的空域运动信息存储单元为 aboveMotion。

a) 如果上侧相邻子块不存在，设置上侧相邻块单向运动信息 MotionInfoTop[xSbIndex]为：运动矢量 mvE0 和 mvE1 为零矢量，L0 参考索引 refIndexL0 和 L1 参考索引 refIndexL1 等于-1，interPredRefMode 设置为‘PRED_I’；

b) 否则，如果 aboveMotion 中的 interPredRefMode 不等于‘PRED_List01’，则 MotionInfoTop[xSbIndex]中的运动信息等于 aboveMotion 中的运动信息；

c) 否则（即 aboveMotion 中的 interPredRefMode 等于‘PRED_List01’），将当前图像的距离索引记为 DistanceIdexCur，上侧相邻块 L0 参考索引和 L1 参考索引对应的图像的距离索引分别记为 DistanceIndexL0 和 DistanceIndexL1，计算 BlockDistanceL0=Abs(DistanceIndexCur - DistanceIndexL0)，BlockDistanceL1=Abs(DistanceIndexCur – DistanceIndexL1)，将上侧相邻子块 L0 参考索引和 L1 参考索引对应的图像的图像级量化参数分别记为 QP0 和 QP1。

1) 如果 BlockDistanceL0 小于 BlockDistanceL1，设置上侧相邻块单向运动信息 MotionInfoTop[xSbIndex]为：将运动矢量 mvE0 和参考索引 refIndexL0 分别设置为 aboveMotion 中 L0 运动矢量和 L0 参考索引，运动矢量 mvE1 和参考索引 refIndexL1 分别设置为零矢量和-1，interPredRefMode 设置为‘PRED_List0’；

2) 否则，如果 BlockDistanceL1 小于 BlockDistanceL0，设置上侧相邻块单向运动信息 MotionInfoTop[xSbIndex]为：将运动矢量 mvE1 和参考索引 refIndexL1 分别设置为 aboveMotion 中 L1 运动矢量和 L1 参考索引，运动矢量 mvE0 和参考索引 refIndexL0 分别设置为零矢量和-1，interPredRefMode 设置为‘PRED_List1’；

3) 否则，如果上侧相邻块 L0 参考索引对应的图像的 CuDeltaQpFlag 和 L1 参考索引对应的图像的 CuDeltaQpFlag 均等于 0 且 QP1 小于 QP0，设置上侧相邻块单向运动信息 MotionInfoTop[xSbIndex]为：将运动矢量 mvE1 和参考索引 refIndexL1 分别设置为 aboveMotion 中 L1 运动矢量和 L1 参考索引，运动矢量 mvE0 和参考索引 refIndexL0 分别设置为零矢量和-1，interPredRefMode 设置为‘PRED_List1’；

4) 否则，设置上侧相邻块单向运动信息 MotionInfoTop[xSbIndex]为：将运动矢量 mvE0 和参考索引 refIndexL0 分别设置为 aboveMotion 中 L0 运动矢量和 L0 参考索引，运动矢量 mvE1 和参考索引 refIndexL1 分别设置为零矢量和-1，interPredRefMode 设置为‘PRED_List0’。

第 2 步，导出左侧相邻块单向运动信息矩阵 MotionInfoLeft。

对于 ySbIndex 从 0 到 n-1 每个取值，左侧相邻子块位置(xLeft,yLeft)为((xCb>>2)-1,(yCb>>2)+ySbIndex)。记(xLeft,yLeft)对应的亮度样本所在的空域运动信息存储单元为 leftMotion。

a) 如果左侧相邻子块不存在,设置左侧相邻块单向运动信息 MotionInfoLeft[ySbIndex]为：运动矢量 mvE0 和 mvE1 为零矢量；L0 参考索引 refIndexL0 和 L1 参考索引 refIndexL1 等于-

T/AI 109.8-2025

1；

b) 否则，leftMotion 的 interPredRefMode 不等于‘PRED_List01’，则 MotionInfoLeft[xSbIndex]中的运动信息等于 leftMotion 中的运动信息；

c) 否则(即 leftMotion 的 interPredRefMode 等于‘PRED_List01’)，将当前图像的距离索引记为 DistanceIdexCur，左侧相邻块 L0 参考索引和 L1 参考索引对应的图像的距离索引分别记为 DistanceIndexL0 和 DistanceIndexL1，计算 BlockDistanceL0=Abs(DistanceIndexCur – DistanceIndexL0)，BlockDistanceL1=Abs(DistanceIndexCur – DistanceIndexL1)，将左侧相邻块 L0 参考索引和 L1 参考索引对应的图像的图像级量化参数分别记为 QP0 和 QP1。

1) 如果 BlockDistanceL0 小于 BlockDistanceL1，设置左侧相邻块单向运动信息 MotionInfoLeft[ySbIndex]为：将运动矢量 mvE0 和参考索引 refIndexL0 分别设置为 leftMotion 中 L0 运动矢量和 L0 参考索引，运动矢量 mvE1 和参考索引 refIndexL1 分别设置为零矢量和-1，interPredRefMode 设置为‘PRED_List0’；

2) 否则,如果 BlockDistanceL1 小于 BlockDistanceL0,设置左侧相邻块单向运动信息 MotionInfoLeft[ySbIndex]为：将运动矢量 mvE1 和参考索引 refIndexL1 分别设置为 leftMotion 中 L1 运动矢量和 L1 参考索引，运动矢量 mvE0 和参考索引 refIndexL0 分别设置为零矢量和-1，interPredRefMode 设置为‘PRED_List1’；

3) 否则，如果上侧相邻块 L0 参考索引对应的图像的 CuDeltaQpFlag 和 L1 参考索引对应的图像的 CuDeltaQpFlag 均等于 0 且 QP1 小于 QP0，设置左侧相邻块单向运动信息 MotionInfoLeft[ySbIndex]为：将运动矢量 mvE1 和参考索引 refIndexL1 分别设置为 leftMotion 中 L1 运动矢量和 L1 参考索引，运动矢量 mvE0 和参考索引 refIndexL0 分别设置为零矢量和-1，interPredRefMode 设置为‘PRED_List1’；

4) 否则，设置左侧相邻块单向运动信息 MotionInfoLeft[ySbIndex]为：将运动矢量 mvE0 和参考索引 refIndexL0 分别设置为 leftMotion 中 L0 运动矢量和 L0 参考索引，运动矢量 mvE1 和参考索引 refIndexL1 分别设置为零矢量和-1，interPredRefMode 设置为‘PRED_List0’。

8.7 变换块解码

8.7.1 概述

本条规定 $M_1 \times M_2$ 变换块的解码过程。变换块的量化系数经过反量化(见 8.7.2)和反变换(见 8.7.3)，得到残差样本矩阵。

8.7.2 反量化

本条规定将 $M_1 \times M_2$ 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 的方法。从符合本文件的位流中解码得到的 QuantCoeffMatrix 的元素的取值范围应为-2^{15}～$2^{15}-1$。

如果帧内预测模式既不是‘Intra_Luma_PCM’也不是‘Intra_Chroma_PCM’，二维变换系数矩阵 CoeffMatrix 的计算如下：

```
m = Log(M1 * M2) / 2
shift1 = m + BitDepth - 14
shift2 = 1 << 7
for (i=0; i<M1; i++) {
```

T/AI 109.8-2025

```
        for (j=0; j<M2; j++)
            CoeffMatrix[i][j] = Clip3(-32768, 32767, (((((QuantCoeffMatrix[i][j] * WeightQuantMatrixM1xM2[i][j]) >>
WqmShift) * (DequantTable(QPX)) >> 4) + 2((ShiftTable(QPx)+shift1)-1)) >> (ShiftTable(QPX) + shift1)))
}
if (((M1 == 2 * M2) || (M2 == 2 * M1)) || ((M1 == 8 * M2) || (M2 == 8 * M1))) {
        for (i=0; i<M1; i++) {
            for (j=0; j<M2; j++)
                CoeffMatrix[i][j] = (CoeffMatrix[i][j] * 181 + shift2) >> 8
        }
}
```

如果帧内预测模式是‘Intra_Luma_PCM’或‘Intra_Chroma_PCM’，二维变换系数矩阵 CoeffMatrix 的计算如下：

```
for (i=0; i<M1; i++) {
        for (j=0; j<M2; j++) {
            CoeffMatrix[i][j] = QuantCoeffMatrix[i][j]
        }
}
```

量化参数 QP_X（X 为 Y、Cb 或 Cr）与 DequantTable 和 ShiftTable 的关系见 118。

表118 QP_X与 DequantTable 和 ShiftTable 的关系

QP_X值	DequantTable(QP_X)值	ShiftTable(QP_X)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

T/AI 109.8-2025

表 118 P_x 与 DequantTable 和 ShiftTable 的关系（续）

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

T/AI 109.8-2025

表 118 P_x 与 DequantTable 和 ShiftTable 的关系（续）

QP_x 值	DequantTable(QP_x) 值	ShiftTable(QP_x) 值
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
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

T/AI 109.8-2025

8.7.3 反变换

8.7.3.1 概述

本条规定将 $M_1 \times M_2$ 变换系数矩阵 CoeffMatrix 转换为残差样本矩阵 ResidueMatrix 的方法。

第 1 种情况，当前变换块的帧内预测模式是‘Intra_Luma_PCM’或‘Intra_Chroma_PCM’，按 8.7.3.4 导出残差样本矩阵 ResidueMatrix。

第 2 种情况，当前变换块的帧内预测模式是普通帧内预测模式（不包括‘Intra_Luma_PCM’和‘Intra_Chroma_PCM’模式），或当前块的预测类型是块复制帧内预测，或当前块的预测类型是帧间预测：

——如果 PictureTsEnableFlag 为 0 且 IstTuFlag 为 1 且满足以下条件之一，按 8.7.3.3 导出残差样本矩阵 ResidueMatrix；

- 当前变换块是亮度普通帧内预测残差块，且 M_1 和 M_2 的值均小于 64；
- 当前变换块是亮度帧间预测残差块，且 M_1 和 M_2 的值均小于 32。

——否则，如果 PictureTsEnableFlag 为 1 且 IstTuFlag 不为 0 且 M_1 和 M_2 的值均小于 64 且满足以下条件之一，按 8.7.3.5 导出残差样本矩阵 ResidueMatrix；

- 当前变换块是亮度帧内预测残差块；
- 当前变换块是亮度块复制帧内预测残差块；
- 当前变换块是亮度帧间预测残差块且当前变换块所在编码单元的 DirectFlag 为 0；

——否则，如果 PictureTsEnableFlag 为 1 且 M_1 和 M_2 的值均小于 64 且满足以下条件之一，按 8.7.3.5 导出残差样本矩阵 ResidueMatrix；

- 当前变换块是亮度帧间预测残差块且当前变换块所在编码单元的 SbtCuFlag 的值大于 0。

——否则，按 8.7.3.2 导出残差样本矩阵 ResidueMatrix。

第 3 种情况，如果以上两种情况都不满足，按 8.7.3.2 导出残差样本矩阵 ResidueMatrix。

8.7.3.2 传统反变换方法

本条规定导出残差样值矩阵 ResidueMatrix 的方法。

第 1 步，如果当前变换块是帧内预测残差块，M_1 或 M_2 的值大于 4 且 StEnableFlag 的值等于 1，对 CoeffMatrix 进行以下操作。

a) 由变换系数矩阵得到 4×4 矩阵 C；

```
for (i=0; i<4; i++) {
    for (j=0; j<4; j++)
        C[i][j] = CoeffMatrix[i][j]
}
```

b) 如果当前变换块满足以下 4 个条件之一，先得到矩阵 P，再得到 C，其中 S_4 是 4×4 反变换矩阵；

```
P = C × S4
for (i=0; i<4; i++) {
    for (j=0; j<4; j++)
        C[i][j] = Clip3(-32768, 32767, (P[i][j] + 2^6) >> 7)
}
```

1) 当前变换块是亮度预测残差块，EnhancedStEnableFlag 为 0 且 IntraLumaPredMode 为 0～2 或 13～32 或 44～65，且 8.8.1.2 中坐标为(x0-1, y0)的参考样本“可用”；
2) 当前变换块是亮度预测残差块，DtSplitFlag 为 1，且 IntraLumaPredMode 为 0～2 或 13～32 或 44～65，且 8.8.1.2 中坐标为(x0-1, y0)的参考样本“可用”；
3) 当前变换块是亮度预测残差块，EnhancedStEnableFlag 为 1 且 EstTuFlag 为 1；
4) 当前变换块是色度预测残差块，EnhancedStEnableFlag 为 1 且 ChromaStFlag 为 1。

c) 如果当前变换块满足以下 4 个条件之一，先得到矩阵 Q，再得到 C，其中 S_4^T 是 S_4 的转置；

```
Q = S4^T × C
for (i=0; i<4; i++) {
    for (j=0; j<4; j++)
        C[i][j] = Clip3(-32768, 32767, (Q[i][j] + 2^6) >> 7)
}
```

1) 当前变换块是亮度预测残差块，EnhancedStEnableFlag 为 0 且 IntraLumaPredMode 为 0～23 或 34～57，且 8.8.1.2 中坐标为(x0, y0-1)的参考样本“可用”；
2) 当前变换块是亮度预测残差块，DtSplitFlag 为 1，且 IntraLumaPredMode 为 0～23 或 34～57，且 8.8.1.2 中坐标为(x0, y0-1)的参考样本“可用”；
3) 当前变换块是亮度预测残差块，EnhancedStEnableFlag 为 1 且 EstTuFlag 为 1；
4) 当前变换块是色度预测残差块，EnhancedStEnableFlag 为 1 且 ChromaStFlag 为 1。

T/AI 109.8-2025

d) 根据矩阵 C 修改变换系数矩阵。

```
for (i=0; i<4; i++) {
    for (j=0; j<4; j++)
        CoeffMatrixf[i][j] = C[i][j]
}
```

第 2 步，对变换系数矩阵进行垂直反变换，得到矩阵 K。

a) 由变换系数矩阵和反变换矩阵得到矩阵 V；

1) 如果 M_1 和 M_2 均等于 4 且 StEnableFlag 等于 1 且满足以下条件之一，则矩阵 V 如下，其中 D_4^T 是 4×4 反变换矩阵 D4 的转置：

- 如果当前变换块是亮度帧内预测残差块且 EnhancedStEnableFlag 为 0；
- 如果当前变换块是亮度帧内预测残差块且 DtSplitFlag 为 1；
- 如果当前变换块是亮度帧内预测残差块且 EnhancedStEnableFlag 为 1 且 EstTuFlag 为 1；
- 如果当前变换块是色度帧内预测残差块且 EnhancedStEnableFlag 为 1 且 ChromaStFlag 为 1。

$$V = D_4^T \times CoeffMatrix$$

2) 否则，如果 PbtCuFlag 的值为 1 且变换块的顺序号为 0 或 1，则矩阵 V 如下，其中 $DCT8_{M2}^T$ 是 $M_2 \times M_2$ 反变换矩阵 $DCT8_{M2}$ 的转置；

$$V = DCT8_{M2}^T \times CoeffMatrix$$

3) 否则，如果 PbtCuFlag 的值为 1 且变换块的顺序号为 2 或 3，则矩阵 V 如下，其中 $DST7_{M2}^T$ 是 $M_2 \times M_2$ 反变换矩阵 $DST7_{M2}$ 的转置；

$$V = DST7_{M2}^T \times CoeffMatrix$$

4) 否则，如果 SbtCuFlag 的值为 1 且变换块的顺序号为 0 且变换块的宽度小于 64 且变换块的高度小于 32，且 SbtDirFlag 为 1 且 SbtPosFlag 为 0，则矩阵 V 如下；

$$V = DCT8_{M2}^T \times CoeffMatrix$$

5) 否则，如果 SbtCuFlag 的值为 1 且变换块的顺序号为 0 且变换块的宽度小于 64 且变换块的高度小于 32，且 SbtDirFlag 为 0 或 SbtPosFlag 为 1，则矩阵 V 如下；

$$V = DST7_{M2}^T \times CoeffMatrix$$

6) 否则，矩阵 V 如下，其中 $DCT2_{M2}^T$ 是 $M_2 \times M_2$ 反变换矩阵 $DCT2_{M2}$ 的转置。

$$V = DCT2_{M2}^T \times CoeffMatrix$$

b) 由矩阵 V 得到矩阵 K。

```
for (i=0; i<M1; i++) {
    for (j=0; j<M2; j++)
        K[i][j] = Clip3(-32768, 32767, (V[i][j] + 2^4) >> 5)
}
```

第 3 步，对矩阵 K 进行水平反变换，得到矩阵 H。

a) 由变换系数矩阵和反变换矩阵得到矩阵 W，并得到 MaxValue 和 shift1；

1) 如果 M_1 和 M_2 的值均等于 4 且 StEnableFlag 的值等于 1 且满足以下条件之一，则 MaxValue 和 shift1 以及矩阵 W 如下，其中 D_4 是 4×4 反变换矩阵：

- 如果当前变换块是亮度帧内预测残差块且 EnhancedStEnableFlag 为 0；
- 如果当前变换块是亮度帧内预测残差块且 DtSplitFlag 为 1；

T/AI 109.8-2025

- 如果当前变换块是亮度帧内预测残差块且 EnhancedStEnableFlag 为 1 且 EstTuFlag 为 1；
- 如果当前变换块是色度帧内预测残差块且 EnhancedStEnableFlag 为 1 且 ChromaStFlag 为 1。

$W = K \times D_4$
MaxValue = (1 << BitDepth) − 1
shift1 = 22 - BitDepth

2) 否则，如果 PbtCuFlag 的值为 1 且变换块的顺序号为 0 或 2，则 MaxValue 和 shift1 以及矩阵 W 如下，其中 $DCT8_{M1}$ 是 $M_1 \times M_1$ 反变换矩阵；

$W = K \times DCT8_{M1}$
MaxValue = (1 << BitDepth) − 1
shift1 = 20 - BitDepth

3) 否则，如果 PbtCuFlag 的值为 1 且变换块的顺序号为 1 或 3，则 MaxValue 和 shift1 以及矩阵 W 如下，其中 $DST7_{M1}$ 是 $M_1 \times M_1$ 反变换矩阵；

$W = K \times DST7_{M1}$
MaxValue = (1 << BitDepth) − 1
shift1 = 20 - BitDepth

4) 否则，如果 SbtCuFlag 的值为 1 且变换块的顺序号为 0 且变换块的宽度小于 32 且变换块的高度小于 64，且 SbtDirFlag 为 0 且 SbtPosFlag 为 0，则 MaxValue 和 shift1 以及矩阵 W 如下；

$W = K \times DCT8_{M1}$
MaxValue = (1 << BitDepth) − 1
shift1 = 20 - BitDepth

5) 否则，如果 SbtCuFlag 的值为 1 且变换块的顺序号为 0 且变换块的宽度小于 32 且变换块的高度小于 64，且 SbtDirFlag 为 1 或 SbtPosFlag 为 1，则 MaxValue 和 shift1 以及矩阵 W 如下；

$W = K \times DST7_{M1}$
MaxValue = (1 << BitDepth) − 1
shift1 = 20 - BitDepth

6) 否则，MaxValue 和 shift1 以及矩阵 W 如下，其中 $DCT2_{M1}$ 是 $M_1 \times M_1$ 反变换矩阵。

$W = K \times DCT2_{M1}$
MaxValue = (1 << BitDepth) − 1
shift1 = 20 - BitDepth

c) 由矩阵 W 得到矩阵 H。

```
for (i=0; i<M1; i++) {
    for (j=0; j<M2; j++)
        H[i][j] = Clip3(-MaxValue-1, MaxValue, (W[i][j] + 2^(shift1-1)) >> shift1)
}
```

第 4 步，把矩阵 H 直接作为残差样值矩阵 ResidueMatrix。

二次变换反变换矩阵 S_4 和 D_4 如下：

S_4 = {

T/AI 109.8-2025

{123, -35, -8, -3}
{-32, -120, 30, 10}
{14, 25, 123, -22}
{8, 13, 19, 126}}
D_4 = {
{34, 58, 72, 81}
{77, 69, -7, -75}
{79, -33 -75, 58}
{55, -84, 73, -28}}

DCT2 型变换的变换核尺寸范围是 4×4～64×64，DCT8 型变换的变换核尺寸范围是 4×4～16×16，DST7 型变换的变换核尺寸范围是 4×4～16×16。反变换矩阵见附录 N。

8.7.3.3 隐择反变换方法

本条规定导出残差样值矩阵 ResidueMatrix 的方法。

第 1 步，对变换系数矩阵进行垂直反变换，得到矩阵 K。

a) 由变换系数矩阵和反变换矩阵得到矩阵 V，其中 $DST7_{M2}^{T}$ 是 $M_2 \times M_2$ 反变换矩阵 $DST7_{M2}$ 的转置；

$V = DST7_{M2}^{T} \times CoeffMatrix$

b) 由矩阵 V 得到矩阵 K。

```
for (i=0; i<M1; i++) {
    for (j=0; j<M2; j++)
        K[i][j] = Clip3(-32768, 32767, (V[i][j] + 2^4) >> 5)
}
```

第 2 步，对矩阵 K 进行水平反变换，得到矩阵 H。

a) 由变换系数矩阵和反变换矩阵得到矩阵 W，并得到 MaxValue 和 shift1，其中 $DST7_{M2}$ 是 $M_2 \times M_2$ 反变换矩阵；

```
W = K × DST7M2
MaxValue = (1 << BitDepth) - 1
shift1 = 20 - BitDepth
```

b) 由矩阵 W 得到矩阵 H。

```
for (i=0; i<M1; i++) {
    for (j=0; j<M2; j++)
        H[i][j] = Clip3(-MaxValue-1, MaxValue, (W[i][j] + 2^(shift1-1)) >> shift1)
}
```

第 3 步，把矩阵 H 直接作为残差样值矩阵 ResidueMatrix。

DST7 型变换的变换核尺寸范围是 4×4～32×32。反变换矩阵见附录 N。

8.7.3.4 PCM 模式

本条规定导出残差样值矩阵 ResidueMatrix 的方法。

如果帧内预测模式是‘Intra_Luma_PCM’或‘Intra_Chroma_PCM’，按以下方法由变换系数矩阵 CoeffMatrix 得到残差样值矩阵 ResidueMatrix。

```
for (i=0; i<M1; i++) {
    for (j=0; j<M2; j++)
```

T/AI 109.8-2025

```
            ResidueMatrix[i][j] = CoeffMatrix[i][j] << (BitDepth - SamplePrecision)
}
```

8.7.3.5 隐择反变换跳过方法

本条规定导出残差样值矩阵 ResidueMatrix 的方法。

第 1 步，由变换系数矩阵得到矩阵 W。

a) 计算移位 shift；

```
shift = 15 - BitDepth - ((Log(M1) + Log(M2)) >> 1)
```

b) 计算矩阵 W。

```
rndFactor = 1 << (shift - 1)
for (i=0; i<M1; i++) {
    for (j=0; j<M2; j++)
        W[i][j] = (CoeffMatrix[m][n] + rndFactor) >> shift
}
```

第 2 步，把矩阵 W 直接作为残差样值矩阵 ResidueMatrix。

上述过程中位置索引 m 和 n 的计算如下：

a) 如果当前变换块是亮度帧内预测残差块，且 IstTuFlag 的值等于 2，则：

```
if (((i == (M1 >> 1) - 1) && (j == (M2 >> 1))) || ((i == (M1 >> 1)) && (j == ((M2 >> 1) - 1)))) {
    m = i
    n = j
}
else {
    m = M1 - 1 - i
    n = M2 - 1 - j
}
```

b) 否则：

```
m = i
n = j
```

8.8 帧内预测

8.8.1 普通帧内预测

8.8.1.1 概述

本条规定 M×N 亮度编码块和 K×L 色度编码块的帧内预测过程。

如果当前编码单元的类型是‘I_2M_nU’，将 PredBlockOrder 为 1 的帧内亮度预测块水平拆分成与 PredBlockOrder 为 0 的亮度预测块尺寸相同的 3 个子预测块，3 个子预测块共用相同的帧内预测模式；否则，如果当前编码单元的类型是‘I_2M_nD’，将 PredBlockOrder 为 0 的帧内亮度预测块水平拆分成与 PredBlockOrder 为 1 的亮度预测块尺寸相同的 3 个子预测块，3 个子预测块共用相同的帧内预测模式；否则，如果当前编码单元的类型是‘I_nL_2N’，将 PredBlockOrder 为 1 的帧内亮度预测块垂直拆分成与 PredBlockOrder 为 0 的亮度预测块尺寸相同的 3 个子预测块，3 个子预测块共用相同的帧内预测模式；否则，如果当前编码单元的类型是‘I_nR_2N’，将 PredBlockOrder 为 0 的帧内亮度预测块垂直拆分成与 PredBlockOrder 为 1 的亮度预测块尺寸相同的 3 个子预测块，3 个子预测块共用相同

T/AI 109.8-2025

的帧内预测模式。

分别按 8.8.1.2 和 8.8.1.3 得到当前块的亮度和色度参考样本。

对于亮度编码块，如果 SawpFlag 等于 0，根据 IntraLumaPredMode 按 8.8.1.4 进行亮度帧内预测，得到 M×N 亮度编码块的预测样本矩阵 predMatrix。如果 SawpFlag 等于 1，首先分别将 SawpPredMode0 和 SawpPredMode1 作为 IntraLumaPredMode 按 8.8.1.4 进行亮度帧内预测，得到 M×N 亮度编码块的预测样本矩阵 predLumaMatrix0 和 predLumaMatrix1；然后根据 predLumaMatrix0 和 predLumaMatrix1 按 8.8.1.6 得到 M×N 亮度编码块的预测样本矩阵 predMatrix。

对于色度编码块，如果 SawpFlag 等于 1 且 IntraChromaPredMode 等于 0，首先分别将 SawpPredMode0 和 SawpPredMode1 作为当前编码单元中 PredBlockOrder 的值为 0 的预测块的 IntraLumaPredMode 按 8.8.1.5 进行色度帧内预测，得到 K×L 色度编码块的预测样本矩阵 predChromaMatrix0 和 predChromaMatrix1；然后根据 predChromaMatrix0 和 predChromaMatrix1 按 8.8.1.6 得到 K×L 色度编码块的预测样本矩阵 predMatrix。否则，按 8.8.1.5 进行色度帧内预测，得到 K×L 色度编码块的预测样本矩阵 predMatrix。

对于亮度编码块，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]等于 r[2M]，如果 j 大于 2N，则 c[j]等于 c[2N]。

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

8.8.1.2 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)，其参考样本按以下方法获得：

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(x_0+i-1, y_0-1)（i=1～M）的样本“可用”，则 r[i]等于 $I[x_0+i-1][y_0-1]$，r[i]“可用”；否则 r[i]“不可用”；

c) 如果坐标为(x_0+i-1, y_0-1)（i=M+1～2M）的样本“可用”且坐标为(x_0+i-1, y_0-1)（i=1～M）的样本存在“可用”，则 r[i]等于 $I[x_0+i-1][y_0-1]$；否则 r[i]等于 r[i-1]；

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

e) 如果坐标为(x_0-1, y_0+j-1)（j=N+1～2N）的样本“可用”且坐标为(x_0-1, y_0+j-1)（j=1～N）的样本存在“可用”，则 c[j]等于 $I[x_0-1][y_0+j-1]$；否则 c[j]等于 c[j-1]；

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

1) 如果 r[i]（i=1～M）存在“可用”并且 c[j]（j=1～N）均“不可用”，则 r[0]等于 r[1]，r[0]“可用”；

2) 否则，如果 c[j]（j=1～N）存在“可用”并且 r[i]（i=1～M）均“不可用”，则 r[0]等于 c[1]，r[0]“可用”；

3) 否则，如果 r[i]（i=1～M）存在“可用”并且 c[j]（j=1～N）存在“可用”，则 r[0]等于 r[1]，r[0]“可用”。否则 r[0]“不可用”。

8.8.1.3 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)，其参考样本按以下方法获得：

T/AI 109.8-2025

a) 初始化 row[i]、col[i]为 $2^{BitDepth-1}$（i=0～2K，j=0～2L；BitDepth 是编码样本精度）；
b) 如果坐标为(x_0+i-1, y_0-1)（i=1～K）的样本“可用”，则 row[i]等于 I[x_0+i-1][y_0-1]，row[i]“可用”；否则 row[i]“不可用”；
c) 如果坐标为(x_0+i-1, y_0-1)（i=K+1～2K）的样本“可用”且坐标为(x_0+i-1, y_0-1)（i=1～K）的样本存在“可用”，则 row[i]等于 I[x_0+i-1][y_0-1]；否则 row[i]等于 row[i-1]；
d) 如果坐标为(x_0-1, y_0+j-1)（j=1～L）的样本“可用”，则 col[j]等于 I[x_0-1][y_0+j-1]，col[j]“可用”；否则 col[j]“不可用”；
e) 如果坐标为(x_0-1, y_0+j-1)（j=L+1～2L）的样本“可用”且坐标为(x_0-1, y_0+j-1)（j=1～L）的样本存在“可用”，则 col[j]等于 I[x_0-1][y_0+j-1]；否则 col[j]等于 col[j-1]；
f) 如果坐标为(x_0-1, y_0-1)的样本“可用”，则 row[0]等于 I[x_0-1][y_0-1]，row[0]“可用”。否则：
   1) 如果 row[i]（i=1～K）存在“可用”并且 col[j]（j=1～L）均“不可用”，则 row[0]等于 row[1]，row[0]“可用”；
   2) 否则，如果 col[j]（i=1～L）存在“可用”并且 row[i]（i=1～K）均“不可用”，则 row[0]等于 col[1]，row[0]“可用”；
   3) 否则，如果 row[i]（j=1～K）存在“可用”并且 col[j]（j=1～L）存在“可用”，则 row[0]等于 row[1]，row[0]“可用”；否则 row[0]“不可用”。

8.8.1.4 亮度预测块普通帧内预测

8.8.1.4.1 概述

如果 IntraPfFlag 和 IipFlag 的值均为 0，按 8.8.1.4.2 得到 predMatrix；否则，如果 IntraPfFlag 的值为 1，先按 8.8.1.4.2 得到 orgPredMatrix，再按 8.8.1.4.4 得到 predMatrix；否则，如果 IipFlag 的值为 1，按 8.8.1.4.3 得到 predMatrix。

8.8.1.4.2 预测方法

根据 IntraLumaPredMode 的值进行亮度编码块普通帧内预测方法如下：

a) IntraLumaPredMode 等于 12（Intra_Luma_Vertical 预测）；

```
for (x=0; x<M; x++) {
    for (y=0; y<N; y++) {
        orgPredMatrix[x][y] = r[x+1]
        predMatrix[x][y] = orgPredMatrix[x][y]
    }
}
```

b) IntraLumaPredMode 等于 24（Intra_Luma_Horizontal 预测）；

```
for (x=0; x<M; x++) {
    for (y=0; y<N; y++) {
        orgPredMatrix[x][y] = c[y+1]
        predMatrix[x][y] = orgPredMatrix[x][y]
    }
}
```

c) IntraLumaPredMode 等于 0（Intra_Luma_DC 预测）；

T/AI 109.8-2025

1) 如果 r[i]（i=1～M）中存在一个“可用”且 c[j]（j=1～N）中存在一个“可用”，则：

```
for (x=0; x<M; x++) {
    for (y=0; y<N; y++) {
        ri = 0
        for (i=1; i<=M; i++)
            ri += r[i]
        cj = 0
        for (j=1; j<=N; j++)
            cj += c[j]
        orgPredMatrix[x][y] = ((ri + cj + ((M + N) >> 1)) * (4096 / (M + N))) >> 12
        predMatrix[x][y] = orgPredMatrix[x][y]
    }
}
```

2) 否则，如果 r[i]（i=1～M）中存在一个“可用”，且 c[j]（j=1～N）均“不可用”，则：

```
for (x=0; x<M; x++) {
    for (y=0; y<N; y++) {
        ri = 0
        for (i=1; i<=M; i++)
            ri += r[i]
        orgPredMatrix[x][y] = ((ri + (M >> 1)) >> Log(M)
        predMatrix[x][y] = orgPredMatrix[x][y]
    }
}
```

3) 否则，如果 c[j]（j=1～N）中存在一个“可用”，且 r[i]（i=1～M）均“不可用”，则：

```
for (x=0; x<M; x++) {
    for (y=0; y<N; y++) {
        cj = 0
        for (j=1; j<=N; j++)
            cj += c[j]
        orgPredMatrix[x][y] = ((cj + (N >> 1)) >> Log(N)
        predMatrix[x][y] = orgPredMatrix[x][y]
    }
}
```

4) 否则：

```
for (x=0; x<M; x++) {
    for (y=0; y<N; y++) {
        orgPredMatrix[x][y] = 2^BitDepth-1
        predMatrix[x][y] = orgPredMatrix[x][y]
    }
}
```

其中 BitDepth 是编码样本精度。

d) IntraLumaPredMode 等于 1（Intra_Luma_Plane 预测）

T/AI 109.8-2025

```
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 = 0
for (i=0; i<(M>>1); i++)
    ih += (i + 1) * (r[(M >> 1) + 1 + i] - r[(M >> 1) - 1 - i])
iv = 0
for (i=0; i<(N>>1); i++)
    iv += (i + 1) * (c[(N >> 1) + 1 + i] - c[(N >> 1) - 1 - i])
ia = (r[M] + c[N]) << 4
ib = ((ih << 5) * imh + (1 << (ish - 1))) >> ish
ic = ((iv << 5) * imv + (1 << (isv - 1))) >> isv
for (x=0; x<M; x++) {
    for (y=0; y<N; y++) {
        orgPredMatrix[x][y] = (ia + (x - ((M >> 1) - 1)) * ib + (y - ((N >> 1) - 1)) * ic + 16) >> 5
        predMatrix[x][y] = Clip1(orgPredMatrix[x][y])
    }
}
```

e) IntraLumaPredMode 等于 2（Intra_Luma_Bilinear 预测）

```
bilinearWeight[3] = {21, 13, 7}
weight = bilinearWeight[Log(Max(M, N) / Min(M, N)) - 1]
ishift = Log(Min(M, N))
ia = r[M]
ib = c[N]
if (M == N) {
    ic = (ia + ib + 1) >> 1
}
else {
    ic = (((ia << Log(M)) + (ib << Log(N))) * weight + (1 << (ishift + 5))) >> (ishift + 6)
}
for (x=0; x<M; x++) {
    for (y=0; y<N; y++) {
        orgPredMatrix[x][y] = ((((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)
        predMatrix[x][y] = Clip1(orgPredMatrix[x][y])
    }
}
```

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

T/AI 109.8-2025

1） 首先执行以下操作，然后令 x 从 0 到 M，y 从 0 到 N，依次执行步骤 2）到步骤 5）。

```
r[-1] = c[1]
r[-2] = c[2]
c[-1] = r[1]
c[-2] = r[2]
filterBit[4] = {7, 7, 7, 6}
filterOffset[4] = {64, 64, 64, 32}
xyAxis = dirXYFlag[IntraLumaPredMode]
xySign = dirXYSign[IntraLumaPredMode]
imx = divDxy[IntraLumaPredMode][0]
isx = divDxy[IntraLumaPredMode][1]
imy = divDyx[IntraLumaPredMode][0]
isy = divDyx[IntraLumaPredMode][1]
```

其中 xyAxis、xySign、imx、isx、imy 和 isy 由 IntraLumaPredMode 查表 119 得到。

2） 如果 xySign 小于 0，则

```
if (xyAxis == 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
    if (y == 0)
        filterIndex = MipfEnableFlag ? (M * N <= 64 ? 3 : 2) : 0
    else
        filterIndex = MipfEnableFlag ? (M * N <= 64 ? 2 : 1) : 0
}
else { /* xyAxis == 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)
    if (x == 0)
        filterIndex = MipfEnableFlag ? (M * N <= 64 ? 3 : 2) : 0
    else
        filterIndex = MipfEnableFlag ? (M * N <= 64 ? 2 : 1) : 0
}
```

3） 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)
if (iYy <= -1) { /* 参考上边像素 */
    offset = offsetx
    iX = iXx
    iY = -1
```

T/AI 109.8-2025

```
}
else { /* 参考左边像素 */
    offset = offsety
    iX = -1
    iY = iYy
}
if (xyAxis == 0) {
    if (y == 0)
        filterIndex = MipfEnableFlag ? (M * N <= 64 ? 3 : 2) : 0
    else
        filterIndex = MipfEnableFlag ? (M * N <= 64 ? 2 : 1) : 0
}
else { /* xyAxis == 1 */
    if (x == 0)
        filterIndex = MipfEnableFlag ? (M * N <= 64 ? 3 : 2) : 0
    else
        filterIndex = MipfEnableFlag ? (M * N <= 64 ? 2 : 1) : 0
}
```

4) 如果 iY 等于-1（参考上边像素）：

```
if (xySign < 0) {
    iXn = iX + 1
    iXnP2 = iX + 2
    iXnN1 = iX - 1
}
else {
    iXn = iX - 1
    iXnP2 = iX - 2
    iXnN1 = iX + 1
}
orgPredMatrix[x][y] = (r[iXnN1+1] * filter[0] + r[iX+1] * filter[1] + r[iXn+1] * filter[2] + r[iXnP2+1] * filter[3]
    + filterOffset[filterIndex]) >> filterBit[filterIndex]
if (MipfEnableFlag == 0)
    predMatrix[x][y] = orgPredMatrix[x][y]
else
    predMatrix[x][y] = Clip1( orgPredMatrix[x][y] )
```

其中 filter[0]到 filter[3]由 filterIndex 和 offset 查表 120 得到。

5) 否则，如果 iX 等于-1（参考左边像素）：

```
if (xySign < 0) {
    iYn = iY + 1
    iYnP2 = iY + 2
    iYnN1 = iY - 1
}
```

T/AI 109.8-2025

```
else {
    iYn = iY - 1
    iYnP2 = iY - 2
    iYnN1 = iY + 1
}
orgPredMatrix[x][y] = (c[iYnN1+1] * filter[0] + c[iY+1] * filter[1] + c[iYn+1] * filter[2] + c[iYnP2+1] *
        filter[3] + filterOffset[filterIndex]) >> filterBit[filterIndex]
if (MipfEnableFlag == 0)
    predMatrix[x][y] = orgPredMatrix[x][y]
else
    predMatrix[x][y] = Clip1( orgPredMatrix[x][y] )
```

其中 filter[0]到 filter[3]由 filterIndex 和 offset 查表 120 得到。

g) IntraLumaPredMode 等于‘Intra_Luma_PCM’

```
for (x=0; x<M; x++) {
    for (y=0; y<N; y++)
        predMatrix[x][y] = 0
}
```

表119 角度预测模式的方向

IntraLumaPredMode	dirXYFlag[IntraLumaPredMode]	dirXYSign[IntraLumaPredMode]	divDxy[IntraLumaPredMode]	divDyx[IntraLumaPredMode]
0	-		-	-
1	-	-	-	-
2	-	-	-	-
3	0	-1	{11, 2}	{93, 8}
4	0	-1	{2, 0}	{1, 1}
5	0	-1	{11, 3}	{93, 7}
6	0	-1	{1, 0}	{1, 0}
7	0	-1	{93, 7}	{11, 3}
8	0	-1	{1, 1}	{2, 0}
9	0	-1	{93, 8}	{11, 2}
10	0	-1	{1, 2}	{4, 0}
11	0	-1	{1, 3}	{8, 0}
12	-	-	-	-
13	0	1	{1, 3}	{8, 0}
14	0	1	{1, 2}	{4, 0}
15	0	1	{93, 8}	{11, 2}
16	0	1	{1, 1}	{2, 0}
17	0	1	{93, 7}	{11, 3}
18	1	1	{1, 0}	{1, 0}

T/AI 109.8-2025

表 119 角度预测模式的方向（续）

IntraLumaPredMode	dirXYFlag[IntraLumaPredMode]	dirXYSign[IntraLumaPredMode]	divDxy[IntraLumaPredMode]	divDyx[IntraLumaPredMode]
19	1	1	{11, 3}	{93, 7}
20	1	1	{2, 0}	{1, 1}
21	1	1	{11, 2}	{93, 8}
22	1	1	{4, 0}	{1, 2}
23	1	1	{8, 0}	{1, 3}
24	–	–	–	–
25	1	-1	{8, 0}	{1, 3}
26	1	-1	{4, 0}	{1, 2}
27	1	-1	{11, 2}	{93, 8}
28	1	-1	{2, 0}	{1, 1}
29	1	-1	{11, 3}	{93, 7}
30	1	-1	{1, 0}	{1, 0}
31	1	-1	{93, 7}	{11, 3}
32	1	-1	{1, 1}	{2, 0}
33	–	–	–	–
34	0	-1	{585, 8}	{7, 4}
35	0	-1	{205, 7}	{5, 3}
36	0	-1	{73, 6}	{449, 9}
37	0	-1	{449, 9}	{73, 6}
38	0	-1	{5, 3}	{205, 7}
39	0	-1	{7, 4}	{585, 8}
40	0	-1	{5, 4}	{819, 8}
41	0	-1	{3, 4}	{1365, 8}
42	0	-1	{1, 4}	{16, 0}
43	0	-1	{1, 5}	{32, 0}
44	0	1	{1, 5}	{32, 0}
45	0	1	{1, 4}	{16, 0}
46	0	1	{3, 4}	{1365, 8}
47	0	1	{5, 4}	{819, 8}
48	0	1	{7, 4}	{585, 8}
49	0	1	{5, 3}	{205, 7}
50	0	1	{449, 9}	{73, 6}
51	1	1	{73, 6}	{449, 9}
52	1	1	{205, 7}	{5, 3}
53	1	1	{585, 8}	{7, 4}
54	1	1	{819, 8}	{5, 4}
55	1	1	{1365, 8}	{3, 4}

T/AI 109.8-2025

表 119 角度预测模式的方向（续）

IntraLumaPredMode	dirXYFlag[IntraLumaPredMode]	dirXYSign[IntraLumaPredMode]	divDxy[IntraLumaPredMode]	divDyx[IntraLumaPredMode]
56	1	1	{16,0}	{1,4}
57	1	1	{32,0}	{1,5}
58	1	-1	{32,0}	{1,5}
59	1	-1	{16,0}	{1,4}
60	1	-1	{1365,8}	{3,4}
61	1	-1	{819,8}	{5,4}
62	1	-1	{585,8}	{7,4}
63	1	-1	{205,7}	{5,3}
64	1	-1	{73,6}	{449,9}
65	1	-1	{449,9}	{73,6}

表120 参考像素滤波器系数

filterIndex	0				1			
offset	filter[0]	filter[1]	filter[2]	filter[3]	filter[0]	filter[1]	filter[2]	filter[3]
0	32	64	32	0	23	82	21	2
1	31	63	33	1	21	82	23	2
2	30	62	34	2	19	82	25	2
3	29	61	35	3	18	81	27	2
4	28	60	36	4	16	80	30	2
5	27	59	37	5	15	79	32	2
6	26	58	38	6	13	78	35	2
7	25	57	39	7	12	77	37	2
8	24	56	40	8	11	75	40	2
9	23	55	41	9	10	74	42	2
10	22	54	42	10	9	72	45	2
11	21	53	43	11	8	70	47	3
12	20	52	44	12	7	68	50	3
13	19	51	45	13	6	67	52	3
14	18	50	46	14	6	64	54	4
15	17	49	47	15	5	62	57	4
16	16	48	48	16	4	60	60	4
17	15	47	49	17	4	57	62	5
18	14	46	50	18	4	54	64	6
19	13	45	51	19	3	52	67	6

T/AI 109.8-2025

表 120 参考像素滤波器系数（续）

filterIndex	0				1			
offset	filter[0]	filter[1]	filter[2]	filter[3]	filter[0]	filter[1]	filter[2]	filter[3]
20	12	44	52	20	3	50	68	7
21	11	43	53	21	3	47	70	8
22	10	42	54	22	2	45	72	9
23	9	41	55	23	2	42	74	10
24	8	40	56	24	2	40	75	11
25	7	39	57	25	2	37	77	12
26	6	38	58	26	2	35	78	13
27	5	37	59	27	2	32	79	15
28	4	36	60	28	2	30	80	16
29	3	35	61	29	2	27	81	18
30	2	34	62	30	2	25	82	19
31	1	33	63	31	2	23	82	21
filterIndex	2				3			
offset	filter[0]	filter[1]	filter[2]	filter[3]	filter[0]	filter[1]	filter[2]	filter[3]
0	11	106	12	-1	0	64	0	0
1	9	105	15	-1	-1	63	2	0
2	7	105	18	-2	-2	62	4	0
3	6	103	21	-2	-2	60	7	-1
4	4	102	25	-3	-2	58	10	-2
5	2	100	29	-3	-3	57	12	-2
6	1	99	32	-4	-4	56	14	-2
7	0	96	36	-4	-4	55	15	-2
8	-1	94	39	-4	-4	54	16	-2
9	-2	92	43	-5	-5	53	18	-2
10	-3	89	47	-5	-6	52	20	-2
11	-4	86	51	-5	-6	49	24	-3
12	-4	83	55	-6	-6	46	28	-4
13	-5	80	59	-6	-5	44	29	-4
14	-5	77	62	-6	-4	42	30	-4
15	-6	74	66	-6	-4	39	33	-4
16	-6	70	70	-6	-4	36	36	-4
17	-6	66	74	-6	-4	33	39	-4
18	-6	62	77	-5	-4	30	42	-4

T/AI 109.8-2025

表 120 参考像素滤波器系数（续）

filterIndex	2				3			
offset	filter[0]	filter[1]	offset	filter[0]	filter[1]	offset	filter[0]	filter[1]
19	-6	59	80	-5	-4	29	44	-5
20	-6	55	83	-4	-4	28	46	-6
21	-5	51	86	-4	-3	24	49	-6
22	-5	47	89	-3	-2	20	52	-6
23	-5	43	92	-2	-2	18	53	-5
24	-4	39	94	-1	-2	16	54	-4
25	-4	36	96	0	-2	15	55	-4
26	-4	32	99	1	-2	14	56	-4
27	-3	29	100	2	-2	12	57	-3
28	-3	25	102	4	-2	10	58	-2
29	-2	21	103	6	-1	7	60	-2
30	-2	18	105	7	0	4	62	-2
31	-1	15	105	9	0	2	63	-1

8.8.1.4.3 改进型预测方法

第 1 步，初始化。

```
r[-1] = c[1]
r[-2] = c[2]
r[-3] = c[3]
r[-4] = c[4]
c[-1] = r[1]
c[-2] = r[2]
c[-3] = r[3]
c[-4] = r[4]
```

第 2 步，根据 IntraLumaPredMode 的值进行亮度编码块改进型普通帧内预测：

a) IntraLumaPredMode 等于 12（Intra_Luma_Vertical 预测）

```
for (x=0; x<M; x++) {
    for (y=0; y<N; y++) {
        orgPredMatrix[x][y] = (21 * r[x-1] + 63 * r[x] + 88 * r[x+1] + 63 * r[x+2] + 21 * r[x+3]) >> 8
    }
}
```

b) IntraLumaPredMode 等于 24（Intra_Luma_Horizontal 预测）

```
for (x=0; x<M; x++) {
    for (y=0; y<N; y++) {
        orgPredMatrix[x][y] = (21 * c[y-1] + 63 * c[y] + 88 * c[y+1] + 63 * c[y+2] + 21 * c[y+3]) >> 8
```

T/AI 109.8-2025

```
    }
}
```

c) IntraLumaPredMode 等于 0（Intra_Luma_DC 预测）

1）如果 r[i]（i=1～M）中存在一个“可用”且 c[j]（j=1～N）中存在一个“可用”，则 base 的值如下。

```
ri = 0
for (i=1; i<=M; i++)
    ri += r[i]
cj = 0
for (j=1; j<=N; j++)
    cj += c[j]
base = ((ri + cj + ((M + N) >> 1)) * (4096 / (M + N))) >> 12
```

2）否则，如果 r[i]（i=1～M）中存在一个“可用”，且 c[j]（j=1～N）均“不可用”，则 base 的值如下。

```
ri = 0
for (i=1; i<=M; i++)
    ri += r[i]
base = (ri + (M >> 1)) >> Log(M)
```

3）否则，如果 c[j]（j=1～N）中存在一个“可用”，且 r[i]（i=1～M）均“不可用”，则 base 的值如下。

```
cj = 0
for (j=1; j<=N; i++)
    cj += c[j]
base = (cj + (N >> 1)) >> Log(N)
```

4）否则，base 的值如下。

```
base = 2^BitDepth-1
```

其中 BitDepth 是编码样本精度。

5）导出 orgPredMatrix。

```
orgPredMatrix[0][0] = (base * 126 + (c[0] + c[2] + r[2]) * 18 + (c[1]+r[1]) * 38 + 128) >> 8
orgPredMatrix[1][0] = (base * 169 + (r[1] + r[3]) * 18 + r[2] * 38 + c[1] * 13 + 128) >> 8
orgPredMatrix[0][1] = (base * 169 + (c[1] + c[3]) * 18 + c[2] * 38 + r[1] * 13 + 128) >> 8
orgPredMatrix[1][1] = (base * 230 + r[2] * 13 + c[2] * 13 + 128) >> 8
for (x=2; x<M; x++) {
    for (y=2; y<N; y++) {
        orgPredMatrix[x][0] = (base * 182 + (r[x] + r[x+2]) * 18 + r[x+1] * 38 + 128) >> 8
        orgPredMatrix[0][y] = (base * 182 + (c[y] + c[y+2]) * 18 + c[y+1] * 38 + 128) >> 8
        orgPredMatrix[x][1] = (base * 243 + r[x+1] * 13 + 128) >> 8
        orgPredMatrix[1][y] = (base * 243 + c[y+1] * 13 + 128) >> 8
        orgPredMatrix[x][y] = base
    }
}
```

T/AI 109.8-2025

d) IntraLumaPredMode 等于 1（Intra_Luma_Plane 预测）

1）导出 ia, ib, ic 和 base。

```
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 = 0
for (i=0; i<(M>>1); i++)
    ih += (i + 1) * (r[(M >> 1) + 1 + i] - r[(M >> 1) - 1 - i])
iv = 0
for (i=0; i<(N>>1); i++)
    iv += (i + 1) * (c[(N >> 1) + 1 + i] - c[(N >> 1) - 1 - i])
ia = (r[M] + c[N]) << 4
ib = ((ih << 5) * imh + (1 << (ish - 1))) >> ish
ic = ((iv << 5) * imv + (1 << (isv - 1))) >> isv
base= ia - ((M >> 1) - 1) * ib - ((N >> 1) - 1) * ic + 16
```

2）导出 orgPredMatrix。

```
orgPredMatrix[0][0] = (((base * 126 + 69 * ib + 69 * ic) >> 5) + (c[0] + c[2] + r[2]) * 18 + (c[1]+r[1]) * 38 +
        128) >> 8
orgPredMatrix[1][0] = (((base * 169 + 195 * ib + 87 * ic) >> 5) + (r[1] + r[3]) * 18 + r[2] * 38 + c[1] * 13 +
        128) >> 8
orgPredMatrix[0][1] = (((base * 169 + 195 * ic + 87 * ib) >> 5) + (c[1] + c[3]) * 18 + c[2] * 38 + r[1] * 13 +
        128) >> 8
orgPredMatrix[1][1] = (((base * 230 + 256 * ib + 256 * ic) >> 5) + r[2] * 13 + c[2] * 13 + 128) >> 8
for (x=2; x<M; x++) {
    for (y=2; y<N; y++) {
        orgPredMatrix[x][0] = (((base * 182 + x * ib * 182 + 87 * ic) >> 5) + (r[x] + r[x+2]) * 18 + r[x+1] *
38 + 128) >> 8
        orgPredMatrix[0][y] = (((base * 182 + y * ic * 182 + 87 * ib) >> 5) + (c[x] + c[y+2]) * 18 + c[y+1] *
38 + 128) >> 8
        orgPredMatrix[x][1] = (((base * 243 + x * ib * 243 + 269 * ic) >> 5) + r[x+1] * 13 + 128) >> 8
        orgPredMatrix[1][y] = (((base * 243 + y * ic * 243 + 269 * ib) >> 5) + c[y+1] * 13 + 128) >> 8
            orgPredMatrix[x][y] = (base + x * ib + y * ic) >> 5
    }
}
```

e) IntraLumaPredMode 等于 2（Intra_Luma_Bilinear 预测）

```
bilinearWeight[3] = {21, 13, 7}
weight = bilinearWeight[Log(Max(M, N) / Min(M, N)) - 1]
ishift = Log(Min(M, N))
ia = r[M]
```

T/AI 109.8-2025

```
ib = c[N]
if (M == N)
    ic = (ia + ib + 1) >> 1
else
    ic = (((ia << Log(M)) + (ib << Log(N))) * weight + (1 << (ishift + 5))) >> (ishift + 6)
for (x=0; x<M; x++) {
    for (y=0; y<N; y++) {
        orgPredMatrix[x][y] = (((((ia<<10) - (82 * c[y-1] + 252 * c[y] + 356 * c[y+1] + 252 * c[y+2] + 82 * c[y+3])) * (x+1)) << Log(N)) + ((((ib<<10) - (82 * r[x-1] + 252 * r[x] + 356 * r[x+1] + 252 * r[x+2] + 82 * r[x+3])) * (y+1)) << Log(M)) + ((82 * r[x-1] + 252 * r[x] + 356 * r[x+1] + 252 * r[x+2] + 82 * r[x+3] + 82 * c[y-1] + 252 * c[y] + 356 * c[y+1] + 252 * c[y+2] + 82 * c[y+3]) << (Log(M) + Log(N))) + (((ic<<1) - ia - ib) * x * y<<10) + (1 << (Log(M) + Log(N) + 10))) >> (Log(M) + Log(N) + 11)
    }
}
```

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

1) 首先执行以下操作，然后令 x 从 0 到 M，y 从 0 到 N，依次执行步骤 2）到步骤 5）。

```
filterBitshiftLong[4] = {10, 10, 10, 9}
filterOffsetLong[4] = {512, 512, 512, 256}
xyAxis = dirXYFlag[IntraLumaPredMode]
xySign = dirXYSign[IntraLumaPredMode]
imx = divDxy[IntraLumaPredMode][0]
isx = divDxy[IntraLumaPredMode][1]
imy = divDyx[IntraLumaPredMode][0]
isy = divDyx[IntraLumaPredMode][1]
```

其中 xyAxis、xySign、imx、isx、imy 和 isy 由 IntraLumaPredMode 查表 119 得到。

2) 如果 xySign 小于 0，则

```
if (xyAxis == 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
    if (y == 0)
        filterIndex = MipfEnableFlag ? (M * N <= 64 ? 3 : 2) : 0
    else
        filterIndex = MipfEnableFlag ? (M * N <= 64 ? 2 : 1) : 0
}
else { /* xyAxis == 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)
    if (x == 0)
        filterIndex = MipfEnableFlag ? (M * N <= 64 ? 3 : 2) : 0
    else
```

T/AI 109.8-2025

```
            filterIndex = MipfEnableFlag ? (M * N <= 64 ? 2 : 1) : 0
}
```

3) 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)
if (iYy <= -1) { /* 参考上边像素 */
    offset = offsetx
    iX = iXx
    iY = -1
}
else { /* 参考左边像素 */
    offset = offsety
    iX = -1
    iY = iYy
}
if (xyAxis == 0) {
    if (y == 0)
        filterIndex = MipfEnableFlag ? (M * N <= 64 ? 3 : 2) : 0
    else
        filterIndex = MipfEnableFlag ? (M * N <= 64 ? 2 : 1) : 0
}
else { /* xyAxis == 1 */
    if (x == 0)
        filterIndex = MipfEnableFlag ? (M * N <= 64 ? 3 : 2) : 0
    else
        filterIndex = MipfEnableFlag ? (M * N <= 64 ? 2 : 1) : 0
}
```

4) 如果 iY 等于-1（参考上边像素）：

```
if (xySign < 0) {
    iXn = iX + 1
    iXnP2 = iX + 2
    iXnP3 = iX + 3
    iXnP4 = iX + 4
    iXnN1 = iX − 1
    iXnN2 = iX − 2
    iXnN3 = iX − 3
}
else {
    iXn = iX − 1
    iXnP2 = iX − 2
```

T/AI 109.8-2025

```
        iXnP3 = iX  -  3
        iXnP4 = iX  -  4
        iXnN1 = iX + 1
        iXnN2 = iX + 2
        iXnN3 = iX + 3
}
orgPredMatrix[x][y] = (r[iXnN3+1] * filter[0] + r[iXnN2+1] * filter[1] + r[iXnN1+1] * filter[2] + r[iX+1] *
        filter[3] + r[iXn+1] * filter[4] + r[iXnP2+1] * filter[5] + r[iXnP3+1] * filter[6] + r[iXnP4+1] * filter[7]
        + filterOffsetLong[filterIndex]) >> filterBitshiftLong[filterIndex]
```

其中 filter[0]到 filter[7]由 filterIndex 和 offset 查表 121 得到。

5) 否则，如果 iX 等于-1（参考左边像素）：

```
if (xySign < 0) {
        iYn = iY + 1
        iYnP2 = iY + 2
        iYnP3 = iY + 3
        iYnP4 = iY + 4
        iYnN1 = iY  -  1
        iYnN2 = iY  -  2
        iYnN3 = iY  -  3
}
else {
        iYn = iY  -  1
        iYnP2 = iY  -  2
        iYnP3 = iY  -  3
        iYnP4 = iY  -  4
        iYnN1 = iY + 1
        iYnN2 = iY + 2
        iYnN3 = iY + 3
}
orgPredMatrix[x][y] = (c[iYnN3+1] * filter[0] + c[iYnN2+1] * filter[1] + c[iYnN1+1] * filter[2] + c[iY+1] *
        filter[3] + c[iYn+1] * filter[4] + c[iYnP2+1] * filter[5] + c[iYnP3+1] * filter[6] + c[iYnP4+1] * filter[7]
        + filterOffsetLong[filterIndex]) >> filterBitshiftLong[filterIndex]
```

其中 filter[0]到 filter[7]由 filterIndex 和 offset 查表 121 得到。

g) IntraLumaPredMode 等于‘Intra_Luma_PCM’。

```
for (x=0; x<M; x++) {
    for (y=0; y<N; y++)
        predMatrix[x][y] = 0
}
```

上述改进型预测操作中，若 IntraLumaPredMode 不等于‘Intra_Luma_PCM’，则执行如下操作得到 predMatrix。如果 x 或 y 等于 0，则 d 等于 1；否则 d 等于 2。如果 x 或 y 小于 2，则 w0 等于 10，w1 等于 216；否则 w0 等于 13，w1 等于 204。其中 x 的取值范围是 0～M-1，y 的取值范围是 0～N-1。

```
r[M+1] = r[M]
```

T/AI 109.8-2025

```
r[M+2] = r[M]
c[N+1] = c[N]
c[N+2] = c[N]
for (x=0; x<M; x++) {
    for (y=0; y<N; y++) {
        predMatrix[x][y] = Clip1((orgPredMatrix[x][y] * w1 + (r[x+d+1] + r[x-d+1] + c[y+d+1] + c[y-d+1]) * w0 + 128) >> 8)
    }
}
```

表121 参考像素滤波器系数

filter Index	0							
offset	filter[0]	filter[1]	filter[2]	filter[3]	filter[4]	filter[5]	filter[6]	filter[7]
0	46	145	210	222	210	145	46	0
1	44	142	208	221	211	147	49	2
2	43	139	206	221	211	149	52	3
3	41	136	204	221	212	151	55	4
4	40	133	202	220	212	153	58	6
5	39	130	200	220	212	155	61	7
6	37	126	198	220	213	157	64	9
7	36	123	196	219	213	160	67	10
8	34	120	194	219	213	162	71	11
9	33	117	192	218	214	163	74	13
10	31	114	190	218	214	166	77	14
11	30	111	188	218	214	167	80	16
12	28	108	186	217	215	170	83	17
13	27	105	184	217	215	172	86	18
14	26	101	182	217	215	174	89	20
15	24	99	180	216	216	176	92	21
16	23	95	178	216	216	178	95	23
17	21	92	176	216	216	180	99	24
18	20	89	174	215	217	182	101	26
19	18	86	172	215	217	184	105	27
20	17	83	170	215	217	186	108	28
21	16	80	167	214	218	188	111	30
22	14	77	166	214	218	190	114	31
23	13	74	163	214	218	192	117	33
24	11	71	162	213	219	194	120	34
25	10	67	160	213	219	196	123	36

T/AI 109.8-2025

表 121 参考像素滤波器系数（续）

filter Index	0							
offset	filter[0]	filter[1]	filter[2]	filter[3]	filter[4]	filter[5]	filter[6]	filter[7]
26	9	64	157	213	220	198	126	37
27	7	61	155	212	220	200	130	39
28	6	58	153	212	220	202	133	40
29	4	55	151	212	221	204	136	41
30	3	52	149	211	221	206	139	43
31	2	49	147	211	221	208	142	44
filter Index	1							
offset	filter[0]	filter[1]	filter[2]	filter[3]	filter[4]	filter[5]	filter[6]	filter[7]
0	33	156	209	223	211	156	33	3
1	30	153	208	223	212	159	36	3
2	27	149	208	223	213	162	39	3
3	26	146	207	223	213	164	42	3
4	23	141	206	223	214	168	46	3
5	21	138	205	223	215	170	49	3
6	18	133	204	223	216	174	53	3
7	17	130	204	222	216	176	56	3
8	16	125	203	222	217	178	60	3
9	14	123	202	222	217	180	63	3
10	13	118	201	222	218	182	67	3
11	12	113	199	222	218	184	72	4
12	10	109	198	221	219	187	76	4
13	9	106	198	221	219	188	79	4
14	9	101	195	221	219	189	84	6
15	7	97	195	220	220	191	88	6
16	6	92	194	220	220	194	92	6
17	6	88	191	220	220	195	97	7
18	6	84	189	219	221	195	101	9
19	4	79	188	219	221	198	106	9
20	4	76	187	219	221	198	109	10
21	4	72	184	218	222	199	113	12
22	3	67	182	218	222	201	118	13
23	3	63	180	217	222	202	123	14
24	3	60	178	217	222	203	125	16
25	3	56	176	216	222	204	130	17
26	3	53	174	216	223	204	133	18

T/AI 109.8-2025

表 121 参考像素滤波器系数（续）

filter Index	1							
offset	filter[0]	offset	filter[0]	offset	filter[0]	offset	filter[0]	offset
27	3	49	170	215	223	205	138	21
28	3	46	168	214	223	206	141	23
29	3	42	164	213	223	207	146	26
30	3	39	162	213	223	208	149	27
31	3	36	159	212	223	208	153	30
filter Index	2							
offset	filter[0]	filter[1]	filter[2]	filter[3]	filter[4]	filter[5]	filter[6]	filter[7]
0	16	170	215	226	214	169	15	-1
1	13	165	214	226	215	173	20	-2
2	10	162	215	226	216	176	22	-3
3	8	157	214	226	216	179	27	-3
4	6	152	214	226	217	182	31	-4
5	3	146	213	226	218	186	36	-4
6	2	143	214	226	218	188	39	-6
7	0	137	213	226	219	190	45	-6
8	-1	132	212	226	220	192	49	-6
9	-3	128	213	226	220	194	53	-7
10	-4	122	211	225	221	197	59	-7
11	-6	116	210	225	222	200	64	-7
12	-6	112	211	225	222	200	68	-8
13	-7	106	210	225	222	203	74	-9
14	-7	101	209	225	223	204	78	-9
15	-9	96	208	224	224	206	84	-9
16	-9	90	207	224	224	207	90	-9
17	-9	84	206	224	224	208	96	-9
18	-9	78	204	223	225	209	101	-7
19	-9	74	203	222	225	210	106	-7
20	-8	68	200	222	225	211	112	-6
21	-7	64	200	222	225	210	116	-6
22	-7	59	197	221	225	211	122	-4
23	-7	53	194	220	226	213	128	-3
24	-6	49	192	220	226	212	132	-1
25	-6	45	190	219	226	213	137	0
26	-6	39	188	218	226	214	143	2
27	-4	36	186	218	226	213	146	3

T/AI 109.8-2025

表 121 参考像素滤波器系数（续）

filter Index	2							
offset	filter[0]	offset	filter[0]	offset	filter[0]	offset	filter[0]	offset
28	-4	31	182	217	226	214	152	6
29	-3	27	179	216	226	214	157	8
30	-3	22	176	216	226	215	162	10
31	-2	20	173	215	226	214	165	13
filter Index	3							
offset	filter[0]	filter[1]	filter[2]	filter[3]	filter[4]	filter[5]	filter[6]	filter[7]
0	0	91	108	114	108	91	0	0
1	-1	88	107	114	108	93	3	0
2	-3	85	106	114	109	95	6	0
3	-3	82	107	114	109	96	8	-1
4	-3	79	109	114	109	96	11	-3
5	-4	76	108	114	109	98	14	-3
6	-6	73	107	114	110	100	17	-3
7	-6	72	107	114	110	100	18	-3
8	-6	70	107	114	110	100	20	-3
9	-7	67	106	114	111	102	22	-3
10	-8	64	106	113	111	104	25	-3
11	-9	60	106	113	112	105	29	-4
12	-9	56	107	113	112	106	33	-6
13	-7	54	107	113	112	104	35	-6
14	-6	53	107	113	112	103	36	-6
15	-6	49	106	113	112	104	40	-6
16	-6	45	105	112	112	105	45	-6
17	-6	40	104	112	113	106	49	-6
18	-6	36	103	112	113	107	53	-6
19	-6	35	104	112	113	107	54	-7
20	-6	33	106	112	113	107	56	-9
21	-4	29	105	112	113	106	60	-9
22	-3	25	104	111	113	106	64	-8
23	-3	22	102	111	114	106	67	-7
24	-3	20	100	110	114	107	70	-6
25	-3	18	100	110	114	107	72	-6
26	-3	17	100	110	114	107	73	-6
27	-3	14	98	109	114	108	76	-4
28	-3	11	96	109	114	109	79	-3

T/AI 109.8-2025

表 121 参考像素滤波器系数（续）

filter Index	3							
offset	filter[0]	offset	filter[0]	offset	filter[0]	offset	filter[0]	offset
29	-1	8	96	109	114	107	82	-3
30	0	6	95	109	114	106	85	-3
31	0	3	93	108	114	107	88	-1

8.8.1.4.4 亮度帧内预测滤波

亮度帧内预测滤波过程如下：

```
predMatrixIPF[-1][-1] = 0
for (x=0; x<M; x++)
    predMatrixIPF[x][-1] = r[x+1]
for (y=0; y<N; y++)
    predMatrixIPF[-1][y] = c[y+1]
for (x=0; x<M; x++) {
    for (y=0; y<N; y++)
        predMatrixIPF[x][y] = orgPredMatrix[x][y]
}
if ((IntraLumaPredMode == 0) || (IntraLumaPredMode == 1) || (IntraLumaPredMode == 2)) {
    for (x=0; x<M; x++) {
        for (y=0; y<N; y++)
            predMatrix[x][y] = Clip1((f[x] * predMatrixIPF[-1][y] + f[y] * predMatrixIPF[x][-1] + (64 - f[x] - f[y]) * predMatrixIPF[x][y] + 32 ) >> 6)
    }
}
else if (((IntraLumaPredMode >= 3) && (IntraLumaPredMode <= 18)) || ((IntraLumaPredMode >= 34) && (IntraLumaPredMode <= 50))) {
    for (x=0; x<M; x++) {
        for (y=0; y<N; y++)
            predMatrix[x][y] = Clip1((f[x] * predMatrixIPF[-1][y] + (64- f[x]) * predMatrixIPF[x][y] + 32) >> 6)
    }
}
else if (((IntraLumaPredMode >= 19) && (IntraLumaPredMode <= 32)) || (IntraLumaPredMode >= 51) && (IntraLumaPredMode <= 65)) {
    for (x=0; x<M; x++) {
        for (y=0; y<N; y++)
            predMatrix[x][y] = Clip1((f[y] * predMatrixIPF[x][-1] + (64 - f[y]) * predMatrixIPF[x][y] + 32) >> 6)
    }
}
```

T/AI 109.8-2025

其中，predMatrixIPF 是(M+1)×(N+1)大小的样本矩阵，M×N 块内坐标为(x,y)的像素对应的滤波系数 f[x]和 f[y]分别由 M、x 和 N、y 查表 122 得到。

表122　帧内预测边界滤波系数表

x 或 y	预测块宽度或高度				
	4	8	16	32	64
0	24	44	40	36	52
1	6	25	27	27	44
2	2	14	19	21	37
3	0	8	13	16	31
4	0	4	9	12	26
5	0	2	6	9	22
6	0	1	4	7	18
7	0	1	3	5	15
8	0	0	2	4	13
9	0	0	1	3	11
10～63	0	0	0	0	0

b)

8.8.1.5　色度预测块普通帧内预测

8.8.1.5.1　概述

如果 IntraPfChromaEnableFlag 的值为 1 且 CodingTreeComponent 为‘COMPONENT_LUMACHROMA’且 IntraPfFlag 的值为 1，先根据 8.8.1.5.2 得到 orgPredMatrix，再根据 8.8.1.5.3 得到 predMatrix；否则，根据 8.8.1.5.2 得到 predMatrix。

8.8.1.5.2　预测方法

根据 IntraChromaPredMode 的值进行色度编码块普通帧内预测：

a)　IntraChromaPredMode 等于 0（Intra_Chroma_DM 和 Intra_Chroma_PCM 预测）：

1)　如果 IntraLumaPredMode 等于 12，则 IntraChromaPredMode 的值为 3，转到步骤 d)；

2)　如果 IntraLumaPredMode 等于 24，则 IntraChromaPredMode 的值为 2，转到步骤 c)；

3)　如果 IntraLumaPredMode 等于 0，则 IntraChromaPredMode 的值为 1，转到步骤 b)；

4)　如果 IntraLumaPredMode 等于 2，则 IntraChromaPredMode 的值为 4，转到步骤 e)；

5)　如果 IntraLumaPredMode 等于 1，则：

```
ibMult[5] = {13, 17, 5, 11, 23}
ibShift[5] = {7, 10, 11, 15, 19}
imh = ibMult[Log(K) – 2]
ish = ibShift[Log(K) – 2]
imv = ibMult[Log(L) – 2]
```

T/AI 109.8-2025

```
isv = ibShift[Log(L) - 2]
ih = 0
for (i=0; i<(K>>1); i++)
    ih += (i + 1) * (row[(K >> 1) + 1 + i] - row[(K >> 1) - 1 - i])
iv = 0
for (i=0; i<(L>>1); i++)
    iv += (i + 1) * (col[(L >> 1) + 1 + i] - col[(L >> 1) - 1 - i])
ia = (row[K] + col[L]) << 4
ib = ((ih << 5) * imh + (1 << (ish - 1))) >> ish
ic = ((iv << 5) * imv + (1 << (isv - 1))) >> isv
for (x=0; x<K; x++) {
    for (y=0; y<L; y++)
        predMatrix[x][y] = Clip1((ia + (x-((K>>1)-1)) * ib + (y - ((L>>1) - 1)) * ic + 16) >> 5)
}
```

6) 如果 IntraLumaPredMode 等于 33，则：

```
for (x=0; x<K; x++) {
    for (y=0; y<L; y++)
        predMatrix[x][y] = 0
}
```

7) 如果 IntraLumaPredMode 不等于 0、1、2、12、24、33。

- 初始化：

```
row[-1] = col[1]
row[-2] = col[2]
col[-1] = row[1]
col[-2] = row[2]
filterBit[4] = {7, 7, 7, 6}
filterOffset[4] = {64, 64, 64, 32}
xyAxis = dirXYFlag[IntraLumaPredMode]
xySign = dirXYSign[IntraLumaPredMode]
imx = divDxy[IntraLumaPredMode][0]
isx = divDxy[IntraLumaPredMode][1]
imy = divDyx[IntraLumaPredMode][0]
isy = divDyx[IntraLumaPredMode][1]
```

- 如果 xySign 小于 0，则：

```
if (xyAxis == 0) {
    offset = (((y + 1) * imx * 32) >> isx) − (((y + 1) * imx) >> isx) * 32
    iX = x + (((y + 1) * imx) >> isx)
    iY = -1
    if (y<=1)
        filterIndex = MipfEnableFlag ? (K * L <= 32 ? 3 : 2) : 0
    else
        filterIndex = MipfEnableFlag ? (K * L <= 32 ? 2 : 1) : 0
```

T/AI 109.8-2025

```
}
else {
    offset = (((x + 1) * imy * 32) >> isy) − (((x + 1) * imy) >> isy) * 32
    iX = -1
    iY = y + (((x + 1) * imy) >> isy)
    if (x<=1)
        filterIndex = MipfEnableFlag ? (K * L <= 32 ? 3 : 2) : 0
    else
        filterIndex = MipfEnableFlag ? (K * L <= 32 ? 2 : 1) : 0
}
```

- 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)
if (iYy <= -1) { /* 参考上边像素 */
    offset = offsetx
    iX = iXx
    iY = -1
}
else { /* 参考左边像素 */
    offset = offsety
    iX = -1
    iY = iYy
}
if (xyAxis == 0) {
    if (y <= 1)
        filterIndex = MipfEnableFlag ? (K * L <= 32 ? 3 : 2) : 0
    else
        filterIndex = MipfEnableFlag ? (K * L <= 32 ? 2 : 1) : 0
}
else { /* xyAxis == 1 */
    if (x <= 1)
        filterIndex = MipfEnableFlag ? (K * L <= 32 ? 3 : 2) : 0
    else
        filterIndex = MipfEnableFlag ? (K * L <= 32 ? 2 : 1) : 0
}
```

- 如果 iY 等于-1（参考上边像素）：

```
if (dx * dy < 0) {
    iXn = iX + 1
    iXnP2 = iX + 2
    iXnN1 = iX − 1
```

T/AI 109.8-2025

```
}
else {
    iXn = iX - 1
    iXnP2 = iX - 2
    iXnN1 = iX + 1
}
for (x=0; x<K; x++) {
    for (y=0; y<L; y++) {
        orgPredMatrix[x][y] = (row[iXnN1+1] * filter[0] + row[iX+1] * filter[1] + row[iXn+1] * filter[2] + row[iXnP2+1] * filter[3] + filterOffset[filterIndex]) >> filterBit[filterIndex]
        if (MipfEnableFlag == 0)
            predMatrix[x][y] = orgPredMatrix[x][y]
        else
            predMatrix[x][y] = Clip1( orgPredMatrix[x][y] )
    }
}
```

其中 filter[0]到 filter[3]由 filterIndex 和 offset 查表 120 得到。

- 否则，如果 iX 等于-1（参考左边像素）：

```
if (dx * dy < 0) {
    iYn = iY + 1
    iYnP2 = iY + 2
    iYnN1 = iY - 1
}
else {
    iYn = iY - 1
    iYnP2 = iY - 2
    iYnN1 = iY + 1
}
for (x=0; x<K; x++) {
    for (y=0; y<L; y++) {
        orgPredMatrix[x][y] = (col[iYnN1+1] * filter[0] + col[iY+1] * filter[1] + col[iYn+1] * filter[2] + col[iYnP2+1] * filter[3] + filterOffset[filterIndex]) >> filterBit[filterIndex]
        if (MipfEnableFlag == 0)
            predMatrix[x][y] = orgPredMatrix[x][y]
        else
            predMatrix[x][y] = Clip1( orgPredMatrix[x][y] )
    }
}
```

其中 filter[0]到 filter[3]由 filterIndex 和 offset 查 120 得到。

b) IntraChromaPredMode 等于 1（Intra_Chroma_DC 预测）；

1) 如果 row[i](i=1～K)中存在一个“可用”且 col[j](j=1～L)中存在一个“可用”，则：

```
rowi = 0
```

T/AI 109.8-2025

```
for (i=1; i<=K; i++)
    rowi += row[i]
colj = 0
for (j=1; j<=L; j++)
    colj += col[j]
for (x=0; x<K; x++) {
    for (y=0; y<L; y++)
        predMatrix[x][y] = ((rowi + colj + ((K + L) >> 1)) * (4096 / (K + L))) >> 12
}
```

2) 否则，如果 row[i]（i=1～K）中存在一个“可用”，且 col[j]（j=1～L）均“不可用”，则：

```
rowi = 0
for (i=1; i<=K; i++)
    rowi += row[i]
for (x=0; x<K; x++) {
    for (y=0; y<L; y++)
        predMatrix[x][y] = (rowi + (K >> 1)) >> Log(K)
}
```

3) 否则，如果 col[j]（i=1～L）中存在一个“可用”，且 row[i]（i=1～K）均“不可用”，则：

```
colj = 0
for (j=1; j<=L; j++)
    colj += col[j]
for (x=0; x<K; x++) {
    for (y=0; y<L; y++)
        predMatrix[x][y] = (colj + (L >> 1)) >> Log(L)
}
```

4) 否则：

```
for (x=0; x<K; x++) {
    for (y=0; y<L; y++)
        predMatrix[x][y] = 2^BitDepth-1
}
```

其中 BitDepth 是编码样本精度。

c) IntraChromaPredMode 等于 2（Intra_Chroma_Horizontal 预测）；

```
for (x=0; x<K; x++) {
    for (y=0; y<L; y++) {
        orgPredMatrix[x][y] = col[y+1]
        predMatrix[x][y] = orgPredMatrix[x][y]
    }
}
```

d) IntraChromaPredMode 等于 3（Intra_Chroma_Vertical 预测）；

T/AI 109.8-2025

```
for (x=0; x<K; x++) {
    for (y=0; y<L; y++) {
        orgPredMatrix[x][y] = row[x+1]
        predMatrix[x][y] = orgPredMatrix[x][y]
    }
}
```

e) IntraChromaPredMode 等于 4（Intra_Chroma_Bilinear 预测）；

```
bilinearWeight[3] = {21, 13, 7}
weight = bilinearWeight[Log(Max(K, L) / Min(K, L)) - 1]
ishift = Log(Min(K, L))
ia = row[K]
ib = col[L]
if (K == L)
    ic = (ia + ib + 1) >> 1
else
    ic = (((ia << Log(K)) + (ib << Log(L))) * weight + (1 << (ishift + 5))) >> (ishift + 6)
for (x=0; x<M; x++) {
    for (y=0; y<N; y++)
        predMatrix[x][y] = Clip1(((((ia - col[y+1]) * (x + 1)) << Log(L)) + (((ib - row[x+1]) * (y + 1)) << Log(K)) + ((row[x+1] + col[y+1]) << (Log(K) + Log(L))) + ((ic << 1) - ia - ib) * x * y + (1 << (Log(K) + Log(L)))) >> (Log(K) + Log(L) + 1))
}
```

f) IntraChromaPredMode 等于 5～20，令当前亮度预测块的尺寸为 M×N，色度预测块的尺寸为 K×L，I 表示当前块所在的图像的补偿后样本的亮度样值矩阵，Cb 表示当前块所在的图像的补偿后样本的色度 Cb 样值矩阵，r[i]、c[j]（i=0～2M，j=0～2N）是亮度参考样本，row[i]、col[i]（i=0～2K，j=0～2L）是色度参考样本。

1) 如果 IntraChromaPredMode 等于 5、6、7 或 8，或如果 IntraChromaPredMode 等于 9、10、11、12、13、14、15、16、17、18、19 或 20 且当前为 Cb 分量，则：

```
for (x=0; x<2K; x++) {
    for (y=0; y<2L; y++)
        predChroma[x][y] = Clip1(((α * I[x][y]) >> iShift) + β)
}
for (x=0; x<K; x++) {
    for (y=0; y<L; y++) {
        if (x == 0)
            predMatrix[0][y] = (predChroma[0][2*y] + predChroma[0][2*y+1] + 1) >> 1
        else
            predMatrix[x][y] = (predChroma[2*x-1][2*y] + 2 * predChroma[2*x][2*y] + predChroma[2*x+1][2*y] + predChroma[2*x-1][2*y+1] + 2 * predChroma[2*x][2*y+1] + predChroma[2*x+1][2*y+1] + 4) >> 3
    }
}
```

T/AI 109.8-2025

2） 否则：

```
if (IntraChromaPredMode == 9 || IntraChromaPredMode == 10 || IntraChromaPredMode == 11 ||
        IntraChromaPredMode == 12) {
    k1 = 1
    k2 = 1
    p = 0
}
else if (IntraChromaPredMode == 13 || IntraChromaPredMode == 14 || IntraChromaPredMode == 15 ||
        IntraChromaPredMode == 16) {
    k1 = 1
    k2 = 2
    p = 1
}
else if (IntraChromaPredMode == 17 || IntraChromaPredMode == 18 || IntraChromaPredMode == 19 ||
        IntraChromaPredMode == 20) {
    k1 = 2
    k2 = 1
    p = 0
}
for (x=0; x<2K; x++) {
    for (y=0; y<2L; y++)
        predChroma[x][y] = Clip3(0, (1 << (BitDepth + 2)) - 1, ((((k1*αCb + p)/k2 + αCr) * I[x][y]) >>
iShift) + (k1*βCb + p)/k2 + βCr)
}
for (x=0; x<K; x++) {
    for (y=0; y<L; y++) {
        if (x == 0)
            predMatrixTemp[0][y] = (predChroma[0][2*y] + predChroma[0][2*y+1] + 1) >> 1
        else
            predMatrixTemp[x][y] = (predChroma[2*x-1][2*y] + 2 * predChroma[2*x][2*y] +
predChroma[2*x+1][2*y] + predChroma[2*x-1][2*y+1] + 2 * predChroma[2*x][2*y+1] +
predChroma[2*x+1][2*y+1] + 4) >> 3
        predMatrix[x][y] = Clip1(predMatrixTemp[x][y] - (k1 * Cb[x][y] + p) / k2)
    }
}
```

上述色度帧内预测操作中的 iShift 以及线性模型系数α_x和β_x（X 是 Cb 或 Cr）按以下方法导出：

a） 如果 row[i]（i=1～K）和 col[i]（i=1～L）均“不可用”，或 IntraChromaPredMode 等于 8、12、16 或 20 且 row[i]（i=1～K）均“不可用”，或 IntraChromaPredMode 等于 7、11、15 或 19 且 col[i]（i=1～L）均“不可用”，则：

```
αX = 0
βX = 1 << (BitDepth - 1)
iShift = 0
```

T/AI 109.8-2025

b) 否则，$iShift_X$等于 16，并按以下方法导出α_X和β_X（X 是 Cb 或 Cr）：

1) 如果 IntraChromaPredMode 等于 5、9、13 或 17：

- 如果 row[i]（i=1～K）中存在一个“可用”，且 col[j]（j=1～L）中存在一个“可用”：

```
posA0 = 0
posL0 = 0
if (K >= L) {
    posA1 = K - (K / L)
    posL1 = L - 1
}
else {
    posA1 = K - 1
    posL1 = L - (L / K)
}
x[0] = (r[2*posA0] + 2 * r[2*posA0+1] + r[2*posA0+2] + 2) >> 2
y[0] = row[posA0+1]
x[1] = (r[2*posA1] + 2 * r[2*posA1+1] + r[2*posA1+2] + 2) >> 2
y[1] = row[posA1+1]
x[2] = (c[2*posL0+1] + c[2*posL0+2] + 1) >> 1
y[2] = col[posL0+1]
x[3] = (c[2*posL1+1] + c[2*posL1+2] + 1) >> 1
y[3] = col[posL1+1]
```

- 如果 row[i]（i=1～K）中存在一个“可用”且 col[j]均“不可用”（j=1～L）：

```
posA0 = 0
posA1 = K / 4
posA2 = 2 * K / 4
posA3 = 3 * K / 4
x[0] = (3 * r[2*posA0+1] + r[2*posA0+2] + 2) >> 2
y[0] = row[posA0+1]
x[1] = (r[2*posA1] + 2 * r[2*posA1+1] + r[2*posA1+2] + 2) >> 2
y[1] = row[posA1+1]
x[2] = (r[2*posA2] + 2 * r[2*posA2+1] + r[2*posA2+2] + 2) >> 2
y[2] = row[posA2+1]
x[3] = (r[2*posA3] + 2 * r[2*posA3+1] + r[2*posA3+2] + 2) >> 2
y[3] = row[posA3+1]
```

- 如果 row[i]（i=1～K）均“不可用”且 col[j]中存在一个“可用”（j=1～L）：

```
posL0 = 0
posL1 = L / 4
posL2 = 2 * L / 4
posL3 = 3 * L / 4
x[0] = (c[2*posL0+1] + c[2*posL0+2] + 1) >> 1
y[0] = col[posL0+1]
x[1] = (c[2*posL1+1] + c[2*posL1+2] + 1) >> 1
```

T/AI 109.8-2025

```
y[1] = col[posL1+1]
x[2] = (c[2*posL2+1] + c[2*posL2+2] + 1) >> 1
y[2] = col[posL2+1]
x[3] = (c[2*posL3+1] + c[2*posL3+2] + 1) >> 1
y[3] = col[posL3+1]
```

2) 如果 IntraChromaPredMode 等于 6、10、14 或 18：

- 如果 row[i]（i=1～K）中存在一个“可用”且 col[j]（j=1～L）中存在一个“可用”：

```
posA0 = 3*K / 8
posL0 = 3*L / 8
posA1 = 5*K / 8
posL1 = 5*L / 8
x[0] = (r[2*posA0] + 2 * r[2*posA0+1] + r[2*posA0+2] + 2) >> 2
y[0] = row[posA0+1]
x[1] = (r[2*posA1] + 2 * r[2*posA1+1] + r[2*posA1+2] + 2) >> 2
y[1] = row[posA1+1]
x[2] = (c[2*posL0+1] + c[2*posL0+2] + 1) >> 1
y[2] = col[posL0+1]
x[3] = (c[2*posL1+1] + c[2*posL1+2] + 1) >> 1
y[3] = col[posL1+1]
```

- 如果 row[i]（i=1～K）中存在一个“可用”且 col[j]均“不可用”（j=1～L）：

```
posA0 = K / 8
posA1 = 3 * K / 8
posA2 = 5 * K / 8
posA3 = 7 * K / 8
x[0] = (posA0 == 0) ? ( (3 * r[2*posA0+1] + r[2*posA0+2] + 2) >> 2) : ((r[2*posA0] + 2 *
       r[2*posA0+1] + r[2*posA0+2] + 2) >> 2)
y[0] = row[posA0+1]
x[1] = (r[2*posA1] + 2 * r[2*posA1+1] + r[2*posA1+2] + 2) >> 2
y[1] = row[posA1+1]
x[2] = (r[2*posA2] + 2 * r[2*posA2+1] + r[2*posA2+2] + 2) >> 2
y[2] = row[posA2+1]
x[3] = (r[2*posA3] + 2 * r[2*posA3+1] + r[2*posA3+2] + 2) >> 2
y[3] = row[posA3+1]
```

- 如果 row[i]（i=1～K）均“不可用”且 col[j]中存在一个“可用”（j=1～L）：

```
posL0 = L / 8
posL1 = 3 * L / 8
posL2 = 5 * L / 8
posL3 = 7 * L / 8
x[0] = (c[2*posL0+1] + c[2*posL0+2] + 1) >> 1
y[0] = col[posL0+1]
x[1] = (c[2*posL1+1] + c[2*posL1+2] + 1) >> 1
y[1] = col[posL1+1]
```

T/AI 109.8-2025

```
x[2] = (c[2*posL2+1] + c[2*posL2+2] + 1) >> 1
y[2] = col[posL2+1]
x[3] = (c[2*posL3+1] + c[2*posL3+2] + 1) >> 1
y[3] = col[posL3+1]
```

3) 如果 IntraChromaPredMode 等于 8、12、16 或 20，如果 row[i]中存在一个“可用”，令 len 等于 K：

```
posA0 = 0
posA1 = len / 4
posA2 = 2 * len / 4
posA3 = 3 * len / 4
x[0] = (3 * r[2*posA0+1] + r[2*posA0+2] + 2) >> 2
y[0] = row[posA0+1]
x[1] = (r[2*posA1] + 2 * r[2*posA1+1] + r[2*posA1+2] + 2) >> 2
y[1] = row[posA1+1]
x[2] = (r[2*posA2] + 2 * r[2*posA2+1] + r[2*posA2+2] + 2) >> 2
y[2] = row[posA2+1]
x[3] = (r[2*posA3] + 2 * r[2*posA3+1] + r[2*posA3+2] + 2) >> 2
y[3] = row[posA3+1]
```

4) 如果 IntraChromaPredMode 等于 7、11、15 或 19，如果 col[j]中存在一个“可用”，令 len 等于 L：

```
posL0 = 0
posL1 = len / 4
posL2 = 2 * len / 4
posL3 = 3 * len / 4
x[0] = (c[2*posL0+1] + c[2*posL0+2] + 1) >> 1
y[0] = col[posL0+1]
x[1] = (c[2*posL1+1] + c[2*posL1+2] + 1) >> 1
y[1] = col[posL1+1]
x[2] = (c[2*posL2+1] + c[2*posL2+2] + 1) >> 1
y[2] = col[posL2+1]
x[3] = (c[2*posL3+1] + c[2*posL3+2] + 1) >> 1
y[3] = col[posL3+1]
```

5) 线性模型系数α_x和β_x：

```
minIndex[2] = {0, 2}
maxIndex[2] = {1, 3}
if (x[minIndex[0]] > x[minIndex[1]])
    Exchange(minIndex[0], minIndex[1])
if (x[maxIndex[0]] > x[maxIndex[1]])
    Exchange(maxIndex[0], maxIndex[1])
if (x[minIndex[0]] > x[maxIndex[1]]) {
    Exchange(minIndex[0], maxIndex[0])
    Exchange(minIndex[1], maxIndex[1])
```

T/AI 109.8-2025

```
}
if (x[minIndex[1]] > x[maxIndex[0]])
    Exchange(minIndex[1], maxIndex[0])
xMin = (x[minIndex[0]] + x[minIndex[1]] + 1) >> 1
yMin = (y[minIndex[0]] + y[minIndex[1]] + 1) >> 1
xMax = (x[maxIndex[0]] + x[maxIndex[1]] + 1) >> 1
yMax = (y[maxIndex[0]] + y[maxIndex[1]] + 1) >> 1
diffX = xMax - xMin
diffY = yMax - yMin
if (diffX > 64) {
    shift = (BitDepth > 8) ? BitDepth - 6 : 2
    add = 1 << (shift - 1)
    αx = (diffY * TscpmTable[((diffX+add)>>shift)-1] + add) >> shift
    βx = yMin - ((αx * xMin) >> iShiftx)
}
else if (diffX > 0) {
    αx = diffY * TscpmTable[diffX-1]
    βx = yMin - ((αx * xMin) >> iShiftx)
}
else {
    αx = 0
    βx = yMin
}
```

其中 TscpmTable 见表 123。

表123 TscpmTable 查找表

索引	TscpmTable	索引	TscpmTable	索引	TscpmTable	索引	TscpmTable
0	65536	16	3855	32	1985	48	1337
1	32768	17	3640	33	1927	49	1310
2	21845	18	3449	34	1872	50	1285
3	16384	19	3276	35	1820	51	1260
4	13107	20	3120	36	1771	52	1236
5	10922	21	2978	37	1724	53	1213
6	9362	22	2849	38	1680	54	1191
7	8192	23	2730	39	1638	55	1170
8	7281	24	2621	40	1598	56	1149
9	6553	25	2520	41	1560	57	1129
10	5957	26	2427	42	1524	58	1110
11	5461	27	2340	43	1489	59	1092
12	5041	28	2259	44	1456	60	1074
13	4681	29	2184	45	1424	61	1057

T/AI 109.8-2025

表 123 TscpmTable 查找表（续）

索引	TscpmTable	索引	TscpmTable	索引	TscpmTable	索引	TscpmTable
14	4369	30	2114	46	1394	62	1040
15	4096	31	2048	47	1365	63	1024

8.8.1.5.3 色度帧内预测滤波

色度帧内预测滤波方法如下：

```
predMatrixIPF[-1][-1] = 0
for (x=0; x<K; x++)
    predMatrixIPF[x][-1] = row[x+1]
for (y=0; y<L; y++)
    predMatrixIPF[-1][y] = col[y+1]
for (x=0; x<K; x++) {
    for (y=0; y<L; y++)
        predMatrixIPF[x][y] = orgPredMatrix[x][y]
}
if (IntraChromaPredMode == 3 || IntraChromaPredMode == 8) {
    for (x=0; x<K; x++) {
        for (y=0; y<L; y++)
            predMatrix[x][y] = Clip1((f[x] * predMatrixIPF[-1][y] + (64- f[x]) * predMatrixIPF[x][y] + 32) >> 6)
    }
}
else if (IntraChromaPredMode == 2 || IntraChromaPredMode == 7) {
    for (x=0; x<K; x++) {
        for (y=0; y<L; y++)
            predMatrix[x][y] = Clip1((f[y] * predMatrixIPF[x][-1] + (64 - f[y]) * predMatrixIPF[x][y] + 32) >> 6)
    }
}
else {
    for (x=0; x<K; x++) {
        for (y=0; y<L; y++) {
            predMatrix[x][y] = orgPredMatrix[x][y]
        }
    }
}
```

其中，predMatrixIPF 是(K+1)×(L+1)大小的样本矩阵，KxL 块内坐标为(x,y)的像素对应的滤波系数 f[x]和 f[y]分别由 K、x 和 L、y 查表 122 得到。

8.8.1.6 导出空域角度加权预测样本

8.8.1.6.1 概述

T/AI 109.8-2025

本条的输入是空域角度加权预测模式索引值 SawpIndex、predMatrixSawp0、predMatrixSawp1 以及当前编码单元的宽度 W 和高度 H，本条的输出是空域角度加权预测样本矩阵 predMatrixSawp。

本条的输出是空域角度加权预测样本矩阵 predMatrixSawp。

首先按 8.8.1.6.2 得到空域角度加权预测权重矩阵，再按 8.8.1.6.3 得到空域角度加权预测样本矩阵。

8.8.1.6.2 导出空域角度加权预测权重矩阵

本条的输入是当前预测单元的宽度W和高度H。

本条的输出是空域角度加权预测的亮度权重矩阵SawpWeightMatrixY和色度权重矩阵SawpWeightMatrixUV。

第1步，计算变量stepIndex、angleIndex和angleAreaIndex。

```
stepIndex = (SawpIndex >> 3) - 3
modAngNum = SawpIndex % 8
if (modAngNum == 2)
    angleIndex = 7
else if (modAngNum == 6)
    angleIndex = 8
else
    angleIndex = modAngNum % 2
angleAreaIndex = modAngNum >> 1
```

第2步，先根据angleAreaIndex查表124得到vL，再根据angleAreaIndex和PictureAwpRefineIndex查表125得到shift和fP，最后得到参考权重列表ReferenceWeights。

```
for (x=0; x<vL; x++)
    ReferenceWeights[x] = Clip3(0, 8, (x - fP) << shift)
```

表124 angleAreaIndex 和 vL 的对应关系

angleAreaIndex	vL
0	(H + (W >> angleIndex)) << 1
1	
2	(W + (H >> angleIndex)) << 1
3	

表125 angleAreaIndex，PictureAwpRefineIndex 和 shift，fP 的对应关系

angleAreaIndex	PictureAwpRefineIndex	shift	fP
0	0	0	(vL >> 1) - 6 + stepIndex * ((vL >> 3) - 1)
	1	2	(vL >> 1) - 3 + stepIndex * ((vL >> 3) - 1)
1	0	0	(vL >> 1) - 4 + stepIndex * ((vL >> 3) -1) - ((W << 1) >> angleIndex)

T/AI 109.8-2025

表 125 angleAreaIndex，PictureAwpRefineIndex 和 shift，fP 的对应关系（续）

angleAreaIndex	PictureAwpRefineIndex	shift	fP
	1	2	(vL >> 1) - 1 + stepIndex * ((vL >> 3) -1) - ((W << 1) >> angleIndex)
2	0	0	(vL >> 1) - 4 + stepIndex * ((vL >> 3) -1) - ((H << 1) >> angleIndex)
	1	2	(vL >> 1) - 1 + stepIndex * ((vL >> 3) -1) - ((H << 1) >> angleIndex)
3	0	0	(vL >> 1) - 6 + stepIndex *((vL >> 3) -1)
	1	2	(vL >> 1) - 3 + stepIndex * ((vL >> 3) -1)

第3步，先根据angleAreaIndex查表126得到tP，再得到亮度权重矩阵SawpWeightMatrixY。

```
for (y=0; y<H; x++) {
    for (x=0; x<W; x++)
        SawpWeightMatrixY[x][y] = ReferenceWeights[tP]
}
```

表126 angleAreaIndex 和 tP 的对应关系

angleAreaIndex	tP
0	(y << 1) + ((x << 1) >> angleIndex)
1	(y << 1) - ((x << 1) >> angleIndex)
2	(x << 1) - ((y << 1) >> angleIndex)
3	(x << 1) + ((y << 1) >> angleIndex)

第4步，得到色度权重矩阵SawpWeightMatrixUV。

```
for (y=0; y<H/2; y++) {
    for (x=0; x<W/2; x++)
        SawpWeightMatrixUV[x][y] = SawpWeightMatrixY[x<<1][y<<1]
}
```

8.8.1.6.3 导出空域角度加权预测样本

本条的输入是空域角度加权预测亮度权重矩阵 SawpWeightMatrixY 和色度权重矩阵 SawpWeightMatrixUV，当前预测块左上角样本位置(xE,yE)，当前预测块的宽度 W 和高度 H。

本条的输出是加权预测样本矩阵 predMatrix。

如果当前预测块是亮度预测块：

```
for (x=0; x<W; x++) {
    for (y=0; y<H; y++)
```

T/AI 109.8-2025

```
        predMatrix[x][y] = (predLumaMatrix0[x][y] * SawpWeightMatrixY[x][y] + predLumaMatrix1 [x][y] * (8 - SawpWeightMatrixY[x][y] ) + 4) >> 3
}
```

如果当前预测块是色度预测块：

```
for (x=0; x<W/2; x++) {
    for (y=0; y<H/2; y++)
        predMatrix[x][y] = (predChromaMatrix0[x][y] * SawpWeightMatrixUV[x][y] + predChromaMatrix1 [x][y] * (8 - SawpWeightMatrixUV[x][y] ) + 4) >> 3
}
```

8.8.2 块复制帧内预测

8.8.2.1 概述

本条规定了导出块复制帧内预测样本矩阵的方法。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。

8.8.2.2 导出预测样本

如果当前预测块是亮度预测块，令当前预测块左上角样本在当前图像亮度样本矩阵中的位置是(xE,yE)，亮度预测样本矩阵 predMatrixIbc 的宽度和高度分别是 W 和 H，当前预测单元块矢量是 BvE。predMatrixIbc 的元素 predMatrixIbc[x][y]的值是当前图像未滤波重建的整像素精度亮度样本矩阵中(xPos,yPos)位置的样本值。其中(xPos,yPos)的值如下。

```
xPos = xE + x + (BvE->x >> 2)
yPos = yE + y + (BvE->y >> 2)
```

如果当前预测块是色度预测块，令包含当前预测块左上角样本的亮度预测块的左上角样本在当前图像的亮度样本矩阵中的位置是(xE,yE)，色度预测样本矩阵 predMatrixIbc 的宽度和高度分别是 W 和 H，包含当前预测块所在预测单元的亮度编码块的块矢量是 BvE。predMatrixIbc 的元素 predMatrixIbc[x][y]的值是当前图像未滤波重建的 1/2 精度色度样本矩阵中(xPos,yPos)位置的样本值。当前图像未滤波重建的色度 1/2 精度样本矩阵中各个位置的元素值按 8.8.2.3 插值得到。其中(xPos,yPos)的值如下。

```
xPos = (xE + 2 * x) + (BvE->x >> 2)
yPos = (yE + 2 * y) + (BvE->y >> 2)
```

8.8.2.3 块复制帧内预测色度样本的插值

参考图像色度样本矩阵中的样本位置见图 30，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 fx&1，dy 等于 fy& 1，其中(fx,fy)是该分像素样本在 1/2 精度的色度样本矩阵中的坐标。整像素 Ax,y 和周边的 3 个分像素样本 ax,y(dx,dy)的具体位置见图 31。

T/AI 109.8-2025

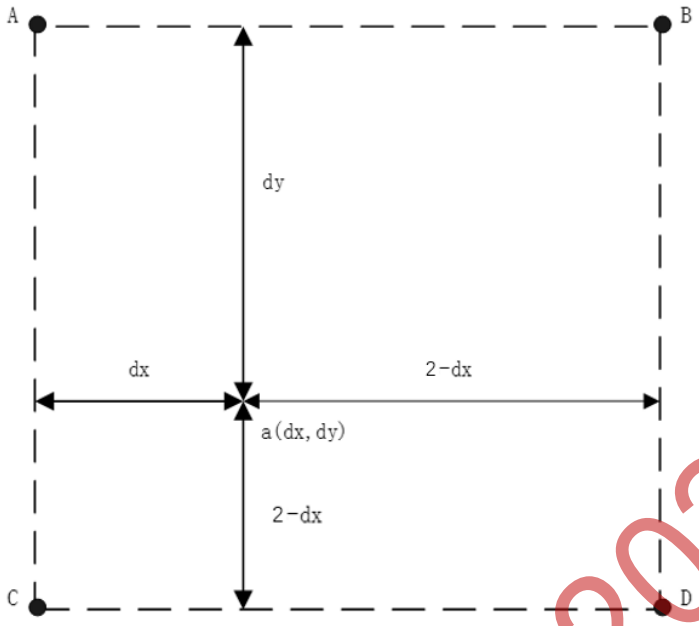


图30 块复制帧内预测色度样本位置

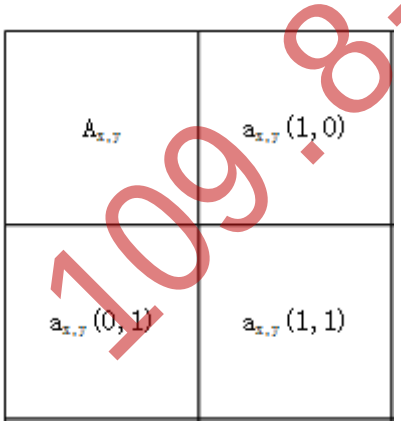


图31 块复制帧内预测整像素样本和分像素样本的位置

色度滤波系数见表 127。

表127 块复制帧内预测色度滤波系数

数组标识	滤波器系数
C[0]	{64, 0}
C[1]	{32, 32}

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：

```
if (dx == 0)
    a_{x,y}(0,dy) = Clip3(0, (1 << BitDepth) - 1, (C[dy][0] * A_{x,y} + C[dy][1] * A_{x,y+1} + 32) >> 6)
else if (dy == 0)
    a_{x,y}(dx,0) = Clip3(0, (1 << BitDepth) - 1, (C[dx][0] * A_{x,y} + C[dx][1] * A_{x+1,y} + 32) >> 6)
```

T/AI 109.8-2025

```
else
    a_{x,y}(dx,dy) = Clip3(0, (1 << BitDepth) - 1, (C[dy][0] * a'_{x,y}(dx,0) + C[dy][1] * a'_{x,y+1}(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，规定为：

```
a'_{x,y}(dx,0) = (C[dx][0] * A_{x,y} + C[dx][1] * A_{x+1,y} + ((1 << (BitDepth - 8)) >> 1)) >> (BitDepth - 8)
```

8.8.3 串复制帧内预测

8.8.3.1 概述

本条规定用于串复制帧内预测的预测样本矩阵的方法。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。

如果IscSubtypeFlag的值为0，则按8.8.3.2导出预测样本矩阵predMatrixIsc；否则，按8.8.3.3导出预测样本矩阵predMatrixIsc。

8.8.3.2 导出普通串子模式的预测样本

当前编码单元按照往返扫描顺序划分为 IscPartNum 个部分。

第 1 步，令 cuWidth 和 cuHeight 分别是当前编码单元的宽度和高度，将 strPos 和 IscPartNumSplit 初始化为 0。

第 2 步，令 i 从 0 到 IscPartNum-1，如果 IscMatchType[i]等于 1，第 i 部分的长度是 StrLen[i]，执行操作 a)～c)；否则，如果 IscMatchType[i]等于 0，仅执行操作 c)：

a) 初始化：

```
NoOverlapStrNum = 1
RemainingStrLen = StrLen[i]
```

b) 计算 NoOverlapStrLen；

```
NoOverlapStrLen[1] = StrLen[i]
if (Sv[i]->y < 0 && Abs(Sv[i]->x) < cuWidth) {
    xStartPos = TravScan[Log(cuWidth)-2][Log(cuHeight)-2][strPos][0]
    yStartPos = TravScan[Log(cuWidth)-2][Log(cuHeight)-2][strPos][1]
    NoOverlapStrLen[1] = Min((Abs(Sv[i]->y) - 1) * cuWidth + ((yStartPos & 0x1) ? xStartPos + 1 : cuWidth - xStartPos), RemainingStrLen)
}
while (RemainingStrLen > NoOverlapStrLen[NoOverlapStrNum]) {
    RemainingStrlen -= NoOverlapStrLen[NoOverlapStrNum]
    IscPartNumSplit++
    NoOverlapStrNum++
    NoOverlapStrLen[NoOverlapStrNum] = Min(Abs(Sv[i]->y) * cuWidth, RemainingStrLen)
}
```

c) 如果 IscMatchType[i]等于 1，令 k 从 1～NoOverlapStrNum。对每个 k，令 j 从 0～NoOverlapStrLen[k]-1，执行以下操作；否则，如果 IscMatchType[i]等于 0，令 j 从 0～3，执行以下操作：

1) 计算当前亮度样本的坐标(xj,yj)：

```
xj = TravScan[Log(CuWidth)-2][Log(CuHeight)-2][strPos][0]
```

T/AI 109.8-2025

```
yj = TravScan[Log(CuWidth)-2][Log(CuHeight)-2][strPos][1]
```

2) 如果当前预测块是亮度预测块，令当前预测块左上角样本在当前图像的亮度样本矩阵中的位置是(xE,yE)，当前预测单元第 i 部分串矢量是 Sv[i]。执行以下操作：

- 如果 IscMatchType[i] 等于 1，或 IscMatchType[i] 等于 0 且 PixelMatchTypeFlag[i][j]等于 1，如果(xPos,yPos)不在当前编码单元内，则亮度预测样本矩阵 predMatrixIsc 元素 predMatrixIsc[xj][yj]的值是当前图像未滤波重建的整像素精度亮度样本矩阵中(xPos,yPos)位置的样本值；否则，亮度预测样本矩阵 predMatrixIsc 元素 predMatrixIsc[xj][yj]的值是 predMatrixIsc[xj + Sv[i]->x][yj + Sv[i]->y]的值；

```
xPos = xE + xj + Sv[i]->x
yPos = yE + yj + Sv[i]->y
```

- 否则，如果 IscMatchType[i]等于 0 且 PixelMatchTypeFlag[i][j]等于 0，predMatrixIsc[xj][yj]等于当前预测单元的 IscUnmatchedPixelY[i][j]的值。

3) 如果当前预测块是色度预测块，且 xj & 0x1 的值和 yj & 0x1 的值均等于 0，令包含当前预测块左上角样本的亮度预测块的左上角样本在当前图像的亮度样本矩阵中的位置是(xE,yE)，当前预测单元第 i 部分串矢量是 Sv[i]。执行以下操作：

- 如果 IscPixelMatchType[i][j]等于 1 且 IscMatchType[i]等于 0，或 IscMatchType[i]等于 1，色度预测样本矩阵 predMatrixIsc 的元素 predMatrixIsc[x][y]的值是当前图像未滤波重建的整像素精度色度样本矩阵中(xPos,yPos)位置的样本值；

```
xPos = (xE + xj + Sv[i]->x) >> 1
yPos = (yE + yj + Sv[i]->y) >> 1
```

- 如果 IscMatchType[i]等于 0 且 IscPixelMatchType[i][j]等于 0，Cb 分量色度预测样本矩阵 predMatrixIsc 的元素 predMatrixIsc[x][y]的值等于当前预测单元的 IscUnmatchedPixelCb[i][j]的值； Cr 分量色度预测样本矩阵 predMatrixIsc 的元素 predMatrixIsc[x][y]的值等于当前预测单元的 IscUnmatchedPixelCr[i][j]的值。

4) strPos 加 1。

8.8.3.3 导出非普通串子模式的预测样本

令 cuWidth 和 cuHeight 分别是当前编码单元的宽度和高度，当前编码单元按照往返扫描顺序划分为 IscPartNum 个部分，令 strPos 和 IscPartNumSplit 为 0，令 i 从 0～IscPartNum-1，按以下方法依次导出各部分的预测样本。

第 1 情况，当前预测块第 i 部分是单位基矢量串或等值串。

a) 该部分的长度等于 StrLen[i]；

b) 如果当前串是单位基矢量串，执行以下操作；

1) 初始化；

```
NoOverlapStrNum = 1
RemainingStrLen = StrLen[i]
```

2) 计算 xStartPos、yStartPos 和 yEndPos；

```
xStartPos = TravScan[Log(cuWidth)-2][Log(cuHeight)-2][strPos][0]
yStartPos = TravScan[Log(cuWidth)-2][Log(cuHeight)-2][strPos][1]
yEndPos = TravScan[Log(cuWidth)-2][Log(cuHeight)-2][strPos+StrLen[i]][1]
```

3) 计算 NoOverlapStrLen。

T/AI 109.8-2025

```
if (yEndPos - yStartPos < 1)
    NoOverlapStrLen[1] = StrLen[i]
else
    NoOverlapStrLen[1] = ((yStartPos & 0x1) ? xStartPos + 1 : cuWidth - xStartPos)
while (RemainingStrLen > NoOverlapStrLen[NoOverlapStrNum]) {
    NoOverlapStrNum++
    IscPartNumSplit++
    RemainingStrlen -= NoOverlapStrLen[NoOverlapStrNum-1]
    NoOverlapStrLen[NoOverlapStrNum] = Min(cuWidth, RemainingStrLen)
}
```

c) 如果当前串是等值串，令 j 从 0～StrLen[i]-1，执行以下操作；否则，如果当前串是单位基矢量串，令 k 从 1～NoOverlapStrNum，对每个 k，令 j 从 0～NoOverlapStrLen[k]-1，执行以下操作。

1) 计算当前亮度样本的坐标，记为(xj,yj)：

```
xj = TravScan[Log(cuWidth)-2][Log(cuHeight)-2][strPos][0]
yj = TravScan[Log(cuWidth)-2][Log(cuHeight)-2][strPos][1]
```

2) 如果当前预测块是亮度预测块，记当前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)，亮度预测样本矩阵 predMatrixIsc 的元素 predMatrixIsc[xj][yj]的值如下：

```
if (StringType[i] == 1)
    predMatrixIsc[xj][yj] = LcuRowBufY[PpInfoList[PvAddr[strPos]][0]][PpInfoList[PvAddr[strPos]][1]]
else
    predMatrixIsc[xj][yj] = predMatrixIsc[xj][yj-1]
```

3) 如果当前预测块是色度预测块，且 xj & 0x1 的值和 yj & 0x1 的值均等于 0，导出 C00、C10、C01、C11 和 uvCnt 的值，色度预测样本矩阵 predMatrixIsc 中的元素 predMatrixIsc[xj>>1][yj>>1]的值如下：

```
predMatrixIsc[xj>>1][yj>>1] = PvFlag[xj][yj]? C00: (C00 + C10 + C01 + C11 + (uvCnt >> 1)) / Max(uvCnt,
    1)
```

4) strPos 加 1。

第 2 种情况，当前预测块第 i 部分的类型是未匹配像素。

a) 计算当前亮度样本的坐标，记为(xj,yj)：

```
xj = TravScan[Log(cuWidth)-2][Log(cuHeight)-2][strPos][0]
yj = TravScan[Log(cuWidth)-2][Log(cuHeight)-2][strPos][1]
```

b) 如果当前预测块是亮度预测块，则亮度预测样本矩阵 predMatrixIsc 的元素 predMatrixIsc[xj][yj]等于 IscNosUpY[i]。其中 IscNosUpY[i]是当前预测单元第 i 部分的未匹配像素的 Y 分量值；

c) 如果当前预测块是色度预测块，且 xj & 0x1 的值和 yj & 0x1 的值均等于 0，导出 C00，C10，C01，C11 和 uvCnt 的值，色度预测样本矩阵 predMatrixIsc 的元素 predMatrixIsc[xj>>1][yj>>1]的值等于：

```
predMatrixIsc[xj>>1][yj>>1] = (C00 + C10 + C01 + C11 + (uvCnt>>1)) / Max(uvCnt,1)
```

d) strPos 加 1。

T/AI 109.8-2025

IscNosUpC[i]是当前预测单元第 i 部分的未匹配像素的色度分量值，C00、C10、C01、C11、LcuRowBufC 和 IscNosUpC 中的 C 为 Cb 或 Cr。导出 C00、C10、C01、C11 和 uvCnt 的方法如下：

```
offsetAbove = 2 * (cuWidth – TravScan[Log(cuWidth)-2][Log(cuHeight)-2][strPos][0]) – 1
x00InLcuR = PPInfoList[PvAddr[strPos]][0]
y00InLcuR = PPInfoList[PvAddr[strPos]][1]
x10InLcuR = PPInfoList[PvAddr[strPos+1]][0]
y10InLcuR = PPInfoList[PvAddr[strPos+1]][1]
x01InLcuR = PPInfoList[PvAddr[strPos+offsetAbove]][0]
y01InLcuR = PPInfoList[PvAddr[strPos+offsetAbove]][1]
x11InLcuR = PPInfoList[PvAddr[strPos+offsetAbove-1]][0]
y11InLcuR = PPInfoList[PvAddr[strPos+offsetAbove-1]][1]
uvCnt = 4 – (PixelType[strPos] != 2 ? ((x00InLcuR | y00InLcuR) & 1) : upYonly[UnmatchedPixelIndex[strPos]]) –
        (PixelType[strPos + 1] != 2 ? ((x10InLcuR|y10InLcuR)&1) : upYonly[UnmatchedPixelIndex[strPos + 1]]) –
        (PixelType[strPos + offsetAbove] != 2 ? ((x01InLcuR|y01InLcuR)&1) : upYonly[UnmatchedPixelIndex[strPos +
        offsetAbove]]) – (PixelType[strPos + offsetAbove - 1] != 2 ? ((x11InLcuR|y11InLcuR)&1) :
        upYonly[UnmatchedPixelIndex[strPos + offsetAbove - 1]])
eP00 = ((x00InLcuR | y00InLcuR) & 1) == 0 ? LcuRowBufC[x00InLcuR>>1][y00InLcuR>>1] : 0
uP00 = upYonly[UnmatchedPixelIndex[strPos]] == 0 ? IscNosUpC[UnmatchedPixelIndex[strPos]] : 0
eP10 = ((x10InLcuR | y10InLcuR) & 1) == 0 ? LcuRowBufC[x10InLcuR>>1][y10InLcuR>>1] : 0
uP10 = upYonly[UnmatchedPixelIndex[strPos + 1]] == 0 ? IscNosUpC[UnmatchedPixelIndex[strPos + 1]] : 0
eP01 = ((x01InLcuR | y01InLcuR) & 1) == 0 ? LcuRowBufC[x01InLcuR>>1][y01InLcuR>>1] : 0
uP01 = upYonly[UnmatchedPixelIndex[strPos + offsetAbove]] == 0 ? IscNosUpC[UnmatchedPixelIndex[strPos +
        offsetAbove]] : 0
eP11 = ((x11InLcuR | y11InLcuR) & 1) == 0 ? LcuRowBufC[x11InLcuR>>1][y11InLcuR>>1] : 0
uP11 = upYonly[UnmatchedPixelIndex[strPos + offsetAbove] - 1] == 0 ? IscNosUpC[UnmatchedPixelIndex[strPos +
        offsetAbove - 1]] : 0
C00 = PixelType[strPos] != 2 ? eP00 : uP00
C10 = PixelType[strPos+1] != 2 ? eP10 : uP10
C01 = PixelType[strPos+offsetAbove] != 2 ? eP01 : uP01
C11 = PixelType[strPos+offsetAbove-1] != 2 ? eP11 : uP11
```

8.9 帧间预测

8.9.1 概述

本条规定帧间预测过程。

先按 8.9.2 得到 PredMatrixL0、PredMatrixL1、GradXL0、GradYL0、GradXL1 和 GradYL1。

再由 PredMatrixL0、PredMatrixL1、GradXL0、GradYL0、GradXL1 和 GradYL1 按 8.9.3 得到 PredMatrixComb。

最后由 PredMatrixL0、PredMatrixL1、PredMatrixComb 按 8.9.4 得到 PredMatrixInter。

8.9.2 导出预测样本

8.9.2.1 概述

T/AI 109.8-2025

本条的输入是当前编码单元的宽度 W 和高度 H，当前编码单元左上角样本在当前图像的亮度样本矩阵中的位置(xE,yE)。

本条的输出是预测样本矩阵 PredMatrixL0 和 PredMatrixL1，梯度矩阵 GradXL0、GradYL0、GradXL1 和 GradYL1。

第 1 种情况，当前编码单元只包含色度编码块，由运动信息 MotionInfo、(xE,yE)、W 和 H 按 8.9.2.2 得到 predMatrixL0 和 predMatrixL1。

第 2 种情况，当前编码单元的编码单元子类型是‘P_Skip_Affine’、‘P_Direct_Affine’、‘P_Inter_Affine’、‘B_Skip_Affine’、‘B_Direct_Affine’或‘B_Inter_Affine’，则 DmvrFlag 的值等于 0，按以下方法导出当前预测单元的预测样本。

a) 如果满足以下条件之一，则子块的宽度 sWa 和高度 sHa 均等于 8，iShift 等于 3；否则子块的宽度 sWa 和高度 sHa 均等于 4，iShift 等于 2；
   1) AffineSubblockSizeFlag 的值为 1；
   2) 当前预测单元的预测参考模式为‘PRED_List01’。
b) 令 i 和 j 是当前 sWa×sHa 子块的水平和垂直索引值，i=0～((W>>iShift)-1)，j=0～((H>>iShift)-1)，当前子块 X 左上角的坐标是(x,y)。如果 i 等于 0 且 j 等于 0，则 X 是 A 子块；否则，如果 i 等于 0 且 j 等于(W/sWa)-1，则 X 是 B 子块；否则，如果 i 等于(H/sHa)-1 且 j 等于 0，则 X 是 C 子块；否则，X 是 N 子块，见图 24。按以下方法导出 mvC0 和 mvC1；
   1) 如果 sWa 等于 8，则 mvC0 等于 MvArrayL0[i][j]，mvC1 等于 MvArrayL1[i][j]；
   2) 否则：

```
x = xE + sWa * i
y = yE + sHa * j
ti = (i >> 1) << 1
tj = (j >> 1) << 1
mvC0->x = (MvArrayL0[ti][tj]->x + MvArrayL0[ti+1][tj]->x + MvArrayL0[ti][tj+1]->x +
      MvArrayL0[ti+1][tj+1]->x + 2) >> 2
mvC0->y = (MvArrayL0[ti][tj]->y + MvArrayL0[ti+1][tj]->y + MvArrayL0[ti][tj+1]->y +
      MvArrayL0[ti+1][tj+1]->y + 2) >> 2
mvC1->x = (MvArrayL1[ti][tj]->x + MvArrayL1[ti+1][tj]->x + MvArrayL1[ti][tj+1]->x +
      MvArrayL1[ti+1][tj+1]->x + 2) >> 2
mvC1->y = (MvArrayL1[ti][tj]->y + MvArrayL1[ti+1][tj]->y + MvArrayL1[ti][tj+1]->y +
      MvArrayL1[ti+1][tj+1]->y + 2) >> 2
```

c) 当前子块的运动信息 MotionInfo 由当前预测单元的仿射运动信息 motionInfoAffine 中的预测参考模式、L0 运动矢量集合中的运动矢量 MvArrayL0[i][j]、L1 运动矢量集合中的运动矢量 MvArrayL1[i][j]、当前预测单元的 L0 参考索引值 refIndexL0 和当前预测单元的 L0 参考索引值 refIndexL1 组成。由运动信息 MotionInfo、(x,y)、mvC0、mvC1、sWa、sHa、asrInfo0、asrInfo1 和 X（X 是 A、B、C 或 N）按 8.9.2.3 得到 predMatrixL0 和 predMatrixL1；
d) PredMatrixL0 由各个子块的 predMatrixL0 组成，PredMatrixL1 由各个子块的 predMatrixL1 组成，GradXL0 由各个子块的 gradXL0 组成，GradYL0 由各个子块的 gradYL0 组成，GradXL1 由各个子块的 gradXL1 组成，GradYL1 由各个子块的 gradYL1 组成。

第 3 种情况，当前编码单元的编码单元子类型是‘P_Skip_Mvap’、‘P_Direct_Mvap’、‘B_Skip_Mvap’或‘B_Direct_Mvap’，令 i 和 j 是当前 8×8 子块的水平和垂直索引值，i=0～((W>>3)-1),j=0～((H>>3)-1)，按以下方法导出当前预测单元的预测样本。

T/AI 109.8-2025

a) 当前子块的运动信息 MotionInfo 是 MotionArray[i][j]，当前子块左上角的坐标(x,y)等于(xE+8*i,yE+8*j)，子块的宽度 sWa 和高度 sHa 均等于 8。如果满足以下条件之一，则 DmvrFlag 的值为 0；否则，DmvrFlag 的值为 1；
   1) DmvrEnableFlag 的值为 0；
   2) 当前子块的预测参考模式不是‘PRED_List01’；
   3) 当前预测单元的 InterPcFlag 的值为 1；
   4) 当前子块的参考图像队列 0 中参考索引是 RefIndexL0 的图像和参考图像队列 1 中参考索引是 RefIndexL1 的图像均位于当前图像显示顺序之前或之后；
   5) 当前子块的参考图像队列 0 中参考索引是 RefIndexL0 的图像的图像距离索引是 DistanceIndexRef0，当前子块的参考图像队列 1 中参考索引是 RefIndexL1 的图像的图像距离索引是 DistanceIndexRef1，当前图像的距离索引是 DistanceIndexCur，Abs(DistanceIndexRef0-DistanceIndexCur)和 Abs(DistanceIndexRef1-DistanceIndexCur)不相等；
   6) 当前子块的参考图像队列 0 中参考索引是 refIndexL0 的图像和参考图像队列 1 中参考索引是 refIndexL1 的图像是同一参考图像，且 L0 运动矢量和 L1 运动矢量相同。
b) 如果 DmvrFlag 的值为 1，由运动信息 MotionInfo、(x,y)、sWa 和 sHa 按 8.9.2.4 得到 predMatrixL0、predMatrixL1、gradXL0、gradYL0、gradXL1 和 gradYL1；
c) 否则，由运动信息 MotionInfo、(x,y)、sWa 和 sHa 按 8.9.2.2 得到 predMatrixL0、predMatrixL1、gradXL0、gradYL0、gradXL1 和 gradYL1；
d) PredMatrixL0 由各个子块的 predMatrixL0 组成，PredMatrixL1 由各个子块的 predMatrixL1 组成，GradXL0 由各个子块的 gradXL0 组成，GradYL0 由各个子块的 gradYL0 组成，GradXL1 由各个子块的 gradXL1 组成，GradYL1 由各个子块的 gradYL1 组成。

第 4 种情况，当前编码单元的编码单元子类型是‘P_Skip_Etmvp’、‘P_Direct_Etmvp’、‘B_Skip_Etmvp’或‘B_Direct_Etmvp’，则 DmvrFlag 等于 0，令 i 和 j 是当前 8×8 子块的水平和垂直索引值，i=0～((W>>3)-1),j=0～((H>>3)-1)，按以下方法导出当前预测单元的预测样本。

a) 当前子块的运动信息 MotionInfo 是 MotionArray[i][j]，当前子块左上角的坐标(x,y)等于(xE+8*i,yE+8*j)，子块的宽度 sWa 和高度 sHa 均等于 8；
b) 由运动信息 MotionInfo、(x,y)、sWa 和 sHa 按 8.9.2.2 得到 predMatrixL0、predMatrixL1、gradXL0、gradYL0、gradXL1 和 gradYL1；
c) PredMatrixL0 由各个子块的 predMatrixL0 组成，PredMatrixL1 由各个子块的 predMatrixL1 组成，GradXL0 由各个子块的 gradXL0 组成，GradYL0 由各个子块的 gradYL0 组成，GradXL1 由各个子块的 gradXL1 组成，GradYL1 由各个子块的 gradYL1 组成。

第 5 种情况，当前编码单元子类型是‘P_Skip_SbTemporal’、‘P_Direct_SbTemporal’、‘B_Skip_SbTemporal’或‘B_Direct_SbTemporal’，则将当前预测单元划分为 4 个子块（见图 26，图中的数字是子块的索引值），按以下方法导出当前预测单元的预测样本。

a) 对于索引值是 i 的子块，运动信息 MotionInfo 是 MotionArray[i]，左上角的坐标(x,y)等于(xE+(W/2*(i%2)),yE+(H/2*(i/2)))，宽度 sWa 和高度 sHa 分别等于 W/2 和 H/2。如果满足以下条件之一，则 DmvrFlag 的值为 0；否则，DmvrFlag 的值为 1；
   1) DmvrEnableFlag 的值为 0；
   2) 当前子块的预测参考模式不是‘PRED_List01’；
   3) 当前预测单元的 InterPcFlag 的值为 1；
   4) 当前子块的参考图像队列 0 中参考索引是 RefIndexL0 的图像和参考图像队列 1 中参考索

T/AI 109.8-2025

引是 RefIndexL1 的图像均位于当前图像显示顺序之前或之后；

5) 当前子块的参考图像队列 0 中参考索引是 RefIndexL0 的图像的图像距离索引是 DistanceIndexRef0，当前子块的参考图像队列 1 中参考索引是 RefIndexL1 的图像的图像距离索引是 DistanceIndexRef1，当前图像的距离索引是 DistanceIndexCur，Abs(DistanceIndexRef0-DistanceIndexCur)和 Abs(DistanceIndexRef1-DistanceIndexCur)不相等；

6) 当前子块的参考图像队列 0 中参考索引是 refIndexL0 的图像和参考图像队列 1 中参考索引是 refIndexL1 的图像是同一参考图像，且 L0 运动矢量和 L1 运动矢量相同。

b) 如果 DmvrFlag 的值为 1，由运动信息 MotionInfo、(x,y)、sWa 和 sHa 按 8.9.2.4 得到 predMatrixL0、predMatrixL1、gradXL0、gradYL0、gradXL1 和 gradYL1；

c) 否则，由运动信息 MotionInfo、(x,y)、sWa 和 sHa 按 8.9.2.2 得到 predMatrixL0、predMatrixL1、gradXL0、gradYL0、gradXL1 和 gradYL1；

d) PredMatrixL0 由各个子块的 predMatrixL0 组成，PredMatrixL1 由各个子块的 predMatrixL1 组成，GradXL0 由各个子块的 gradXL0 组成，GradYL0 由各个子块的 gradYL0 组成，GradXL1 由各个子块的 gradXL1 组成，GradYL1 由各个子块的 gradYL1 组成。

第 6 种情况，当前编码单元子类型是‘B_Skip_Awp’、‘B_Direct_Awp’、‘P_Skip_Awp’或‘P_Direct_Awp’，按以下方法导出 PredMatrixAwp0 以及 PredMatrixAwp1。

a) 由运动信息 MotionInfoAwp0、(xE,yE)、W 和 H 按 8.9.2.2 得到 PredMatrixAwp0。如果预测参考模式 interAwpPredRefMode0 是‘PRED_List0’，令 predMatrixAwp0 等于 predMatrixL0；否则，令 PredMatrixAwp0 等于 predMatrixL1；

b) 由运动信息 MotionInfoAwp1、(xE,yE)、W 和 H 按 8.9.2.2 得到 PredMatrixAwp1。如果预测参考模式 interAwpPredRefMode1 是‘PRED_List0’，令 predMatrixAwp1 等于 predMatrixL0；否则，令 PredMatrixAwp1 等于 predMatrixL1。

如果以上 6 种情况都不满足，按以下步骤导出 predMatrixL0 和 predMatrixL1。

a) 如果满足以下条件之一，则 DmvrFlag 的值为 0；否则，DmvrFlag 的值为 1；

1) DmvrEnableFlag 的值为 0；

2) 当前预测单元的 DirectFlag 的值为 0 且 SkipFlag 的值为 0；

3) 当前预测单元的 InterPcFlag、UmveFlag、AffineFlag、AwpFlag 或 EtmvpFlag 中有一个的值为 1；

4) 当前预测单元的预测参考模式不是“PRED_List01”；

5) 当前预测单元的高度或宽度小于 8；

6) 当前预测单元的高度等于 4 且宽度等于 8；

7) 当前预测单元的高度等于 8 且宽度等于 4；

8) 当前预测单元的参考图像队列 0 中参考索引是 RefIndexL0 的图像和参考图像队列 1 中参考索引是 RefIndexL1 的图像均位于当前图像显示顺序之前或之后；

9) 当前预测单元的参考图像队列 0 中参考索引是 RefIndexL0 的图像的图像距离索引是 DistanceIndexRef0，当前预测单元的参考图像队列 1 中参考索引是 RefIndexL1 的图像的图像距离索引是 DistanceIndexRef1，当前图像的距离索引是 DistanceIndexCur，Abs(DistanceIndexRef0-DistanceIndexCur)和 Abs(DistanceIndexRef1-DistanceIndexCur)不相等；

10) 当前预测单元的参考图像队列 0 中参考索引是 refIndexL0 的图像和参考图像队列 1 中参考索引是 refIndexL1 的图像是同一参考图像，且 L0 运动矢量和 L1 运动矢量相同。

T/AI 109.8-2025

b) 如果 DmvrFlag 的值为 1，由当前编码单元的运动信息 MotionInfo、(xE,yE)、W 和 H 按 8.9.2.4 得到 predMatrixL0、predMatrixL1、gradXL0、gradYL0、gradXL1 和 gradYL1；

c) 否则，由当前编码单元的运动信息 MotionInfo、(xE,yE)、W 和 H 按 8.9.2.2 得到 predMatrixL0、predMatrixL1、gradXL0、gradYL0、gradXL1 和 gradYL1；

d) PredMatrixL0 由各个子块的 predMatrixL0 组成，PredMatrixL1 由各个子块的 predMatrixL1 组成，GradXL0 由各个子块的 gradXL0 组成，GradYL0 由各个子块的 gradYL0 组成，GradXL1 由各个子块的 gradXL1 组成，GradYL1 由各个子块的 gradYL1 组成。

8.9.2.2 导出普通预测样本

本条的输入是运动信息 motionInfo(mvE0,mvE1,refIndexL0,refIndexL1,interPredRefMode)，左上角样本位置(xE,yE)，以及当前编码单元的宽度 W 和高度 H。

本条的输出是预测样本矩阵 predMatrixL0 和 predMatrixL1。

由运动信息 motionInfo 按 8.9.2.5 得到 scaleFlagL0、scaleFlagL1、scaleShiftL0 和 scaleShiftL1。

对于亮度块，执行以下操作：

——如果当前块的预测参考模式 interPredRefMode 等于‘PRED_List0’：

如果 scaleFlagL0 等于 0，则亮度预测样本矩阵 predMatrixL0 的元素 predMatrixL0[x][y]的值是参考图像队列 0 中参考索引为 RefIndexL0 的参考图像的 1/4 精度亮度样本矩阵中(xPos,yPos)位置的样本值。其中 MvE 是当前预测单元的 L0 运动矢量。

```
xPos = ((xE + x) << 2) + MvE->x
yPos = ((yE + y) << 2) + MvE->y
```

否则（scaleFlagL0 等于 1），则亮度预测样本矩阵 predMatrixL0 的元素 predMatrixL0[x][y]的值是参考图像队列 0 中参考索引为 RefIndexL0 的参考图像的 1/16 精度亮度样本矩阵中(xPos,yPos)位置的样本值。其中 MvE 是当前预测单元的 L0 运动矢量。

```
xPos = ((((xE + x) << 2) + MvE->x) << (scaleShiftL0 + 2)) >> 1
yPos = ((((yE + y) << 2) + MvE->y) << (scaleShiftL0 + 2)) >> 1
```

——如果 interPredRefMode 等于‘PRED_List1’：

如果 scaleFlagL1 等于 0，则亮度预测样本矩阵 predMatrixL1 的元素 predMatrixL1[x][y]的值是参考图像队列 1 中参考索引为 RefIndexL1 的参考图像的 1/4 精度亮度样本矩阵中(xPos,yPos)位置的样本值。其中 MvE 是当前预测单元的 L1 运动矢量。

```
xPos = ((xE + x) << 2) + MvE->x
yPos = ((yE + y) << 2) + MvE->y
```

否则（scaleFlagL1 等于 1），则亮度预测样本矩阵 predMatrixL1 的元素 predMatrixL1[x][y]的值是参考图像队列 1 中参考索引为 RefIndexL1 的参考图像的 1/16 精度亮度样本矩阵中(xPos,yPos)位置的样本值。其中 MvE 是当前预测单元的 L1 运动矢量。

```
xPos = ((((xE + x) << 2) + MvE->x) << (scaleShiftL1 + 2)) >> 1
yPos = ((((yE + y) << 2) + MvE->y) << (scaleShiftL1 + 2)) >> 1
```

——如果 interPredRefMode 等于‘PRED_List01’：

如果 scaleFlagL0 等于 0，则亮度预测样本矩阵 predMatrixL0 的元素 predMatrixL0[x][y]的值是参考图像队列 0 中参考帧索引为 RefIndexL0 的参考图像的 1/4 精度亮度样本矩阵中(x0Pos,y0Pos)位置的样本值。其中 MvE0 是当前预测单元的 L0 运动矢量。

```
x0Pos = ((xE + x) << 2) + MvE0->x
```

T/AI 109.8-2025

```
y0Pos = ((yE + y) << 2) + MvE0->y
```

否则（scaleFlagL0 等于 1），亮度预测样本矩阵 predMatrixL0 的元素 predMatrixL0[x][y]的值是参考图像队列 0 中参考帧索引为 RefIndexL0 的参考图像的 1/16 精度亮度样本矩阵中(x0Pos,y0Pos)位置的样本值。其中 MvE0 是当前预测单元的 L0 运动矢量。

```
x0Pos = ((((xE + x) << 2) + MvE0->x) << (scaleShiftL0 + 2)) >> 1
y0Pos = ((((yE + y) << 2) + MvE0->y) << (scaleShiftL0 + 2)) >> 1
```

如果 scaleFlagL1 等于 0，亮度预测样本矩阵 predMatrixL1 的元素 predMatrixL1[x][y]的值是参考图像队列 1 中参考索引为 RefIndexL1 的参考图像的 1/4 精度亮度样本矩阵中(x1Pos,y1Pos)位置的样本值。其中 MvE1 是当前预测单元的 L1 运动矢量。

```
x1Pos = ((xE + x) << 2) + MvE1->x
y1Pos = ((yE + y) << 2) + MvE1->y
```

否则（scaleFlagL1 等于 1），亮度预测样本矩阵 predMatrixL1 的元素 predMatrixL1[x][y]的值是参考图像队列 1 中参考索引为 RefIndexL1 的参考图像的 1/16 精度亮度样本矩阵中(x1Pos,y1Pos)位置的样本值。其中 MvE1 是当前预测单元的 L1 运动矢量。

```
x1Pos = ((((xE + x) << 2) + MvE1->x) << (scaleShiftL1 + 2)) >> 1
y1Pos = ((((yE + y) << 2) + MvE1->y) << (scaleShiftL1 + 2)) >> 1
```

对于色度块，执行以下操作：

——如果 interPredRefMode 等于‘PRED_List0’：

如果 scaleFlagL0 等于 0，则色度预测样本矩阵 predMatrixL0 的元素 predMatrixL0[x][y]的值是参考图像队列 0 中参考索引为 RefIndexL0 的参考图像的 1/8 精度色度样本矩阵中(xPos,yPos)位置的样本值。其中 MvE 是当前预测单元的 L0 运动矢量。

```
xPos = ((xE + 2 * x) << 2) + MvE->x
yPos = ((yE + 2 * y) << 2) + MvE->y
```

否则（scaleFlagL0 等于 1），则色度预测样本矩阵 predMatrixL0 的元素 predMatrixL0[x][y]的值是参考图像队列 0 中参考索引为 RefIndexL0 的参考图像的 1/32 精度色度样本矩阵中(xPos,yPos)位置的样本值。其中 MvE 是当前预测单元的 L0 运动矢量。

```
xPos = ((((xE + 2 * x) << 2) + MvE->x) << (scaleShiftL0 + 2)) >> 1
yPos = ((((yE + 2 * y) << 2) + MvE->y) << (scaleShiftL0 + 2)) >> 1
```

——如果 interPredRefMode 等于‘PRED_List1’：

如果 scaleFlagL1 等于 0，则色度预测样本矩阵 predMatrixL1 的元素 predMatrixL1[x][y]的值是参考图像队列 1 中参考索引为 RefIndexL1 的参考图像的 1/8 精度色度样本矩阵中(xPos,yPos)位置的样本值。其中 MvE 是当前预测单元的 L1 运动矢量。

```
xPos = ((xE + 2 * x) << 2) + MvE->x
yPos = ((yE + 2 * y) << 2) + MvE->y
```

否则（scaleFlagL1 等于 1），则色度预测样本矩阵 predMatrixL1 的元素 predMatrixL1[x][y]的值是参考图像队列 1 中参考索引为 RefIndexL1 的参考图像的 1/32 精度色度样本矩阵中(xPos,yPos)位置的样本值。其中 MvE 是当前预测单元的 L1 运动矢量。

```
xPos = ((((xE + 2 * x) << 2) + MvE->x) << (scaleShiftL1 + 2)) >> 1
yPos = ((((yE + 2 * y) << 2) + MvE->y) << (scaleShiftL1 + 2)) >> 1
```

——如果 interPredRefMode 等于‘PRED_List01’：

如果 scaleFlagL0 等于 0，则色度预测样本矩阵 predMatrixL0 的元素 predMatrixL0[x][y]的

T/AI 109.8-2025

值是参考图像队列 0 中参考索引为 RefIndexL0 的参考图像的 1/8 精度色度样本矩阵中(x0Pos,y0Pos)位置的样本值。其中 MvE0 是当前预测单元的 L0 运动矢量。

```
x0Pos = ((xE + 2 * x) << 2) + MvE0->x
y0Pos = ((yE + 2 * y) << 2) + MvE0->y
```

否则（scaleFlagL0 等于 1），则色度预测样本矩阵 predMatrixL0 的元素 predMatrixL0[x][y]的值是参考图像队列 0 中参考索引为 RefIndexL0 的参考图像的 1/32 精度色度样本矩阵中(x0Pos,y0Pos)位置的样本值。其中 MvE0 是当前预测单元的 L0 运动矢量。

```
x0Pos = ((((xE + 2 * x) << 2) + MvE0->x) << (scaleShiftL0 + 2)) >> 1
y0Pos = ((((yE + 2 * y) << 2) + MvE0->y) << (scaleShiftL0 + 2)) >> 1
```

如果 scaleFlagL1 等于 0，则色度预测样本矩阵 predMatrixL1 的元素 predMatrixL1[x][y]的值是参考图像队列 1 中参考索引为 RefIndexL1 的参考图像的 1/8 精度色度样本矩阵中(x1Pos,y1Pos)位置的样本值。其中 MvE1 是当前预测单元的 L1 运动矢量。

```
x1Pos = ((xE + 2 * x) << 2) + MvE1->x
y1Pos = ((yE + 2 * y) << 2) + MvE1->y
```

否则（scaleFlagL1 等于 1），则色度预测样本矩阵 predMatrixL1 的元素 predMatrixL1[x][y]的值是参考图像队列 1 中参考索引为 RefIndexL1 的参考图像的 1/32 精度色度样本矩阵中(x1Pos,y1Pos)位置的样本值；其中 MvE1 是当前预测单元的 L1 运动矢量。

```
x1Pos = ((((xE + 2 * x) << 2) + MvE1->x) << (scaleShiftL1 + 2)) >> 1
y1Pos = ((((yE + 2 * y) << 2) + MvE1->y) << (scaleShiftL1 + 2)) >> 1
```

上述过程中，参考图像的亮度 1/4 精度样本矩阵和色度 1/8 精度样本矩阵中各个位置的值分别按 8.9.2.6.2 和 8.9.2.7.2 插值得到；参考图像的亮度 1/16 精度样本矩阵和色度 1/32 精度样本矩阵中各个位置的值分别按 8.9.2.6.4 和 8.9.2.7.3 插值得到。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。x 的取值范围是 0～W-1，y 的取值范围是 0～H-1。

8.9.2.3 导出仿射预测样本

8.9.2.3.1 概述

本条的输入是运动信息 motionInfo(mvE0,mvE1,refIndexL0,refIndexL1,interPredRefMode)，当前编码单元的宽度 W 和高度 H，左上角样本位置(xE,yE)，色度 L0 运动矢量 mvC0，色度 L1 运动矢量 mvC1，仿射改善信息 asrInfo0(ASRFlag0,ASRHorFlag0,ASRVerFlag0,dMvA0,dMvB0,dMvC0,dMvN0)和 asrInfo1(ASRFlag1,ASRHorFlag1,ASRVerFlag1,dMvA1,dMvB1,dMvC1,dMvN1)，以及参数 X（X 是 A、B、C 或 N，见图 24）。

本条的输出是预测样本矩阵 predMatrixL0 和 predMatrixL1。

由 motionInfo、(xE,yE)、mvC0、mvC1、W 和 H 按 8.9.2.3.2 得到预测样本矩阵 predMatrixL0 和 predMatrixL1。

如果 ASRFlag0 的值为 1，由参考队列 0、mvE0、refIndexL0、predMatrixL0、ASRHorFlag0、ASRVerFlag0、dMvX0、(xE,yE)、W 和 H 按 8.9.2.3.3 得到改善后的预测样本矩阵 predMatrixL0。

如果 ASRFlag1 的值为 1，由参考队列 1、mvE1、refIndexL1、predMatrixL1、ASRHorFlag1、ASRVerFlag1、dMvX1、(xE,yE)、W 和 H 按 8.9.2.3.3 得到改善后的预测样本矩阵 predMatrixL1。

8.9.2.3.2 导出仿射预测样本的方法

T/AI 109.8-2025

本条的输入是运动信息 motionInfo(mvE0,mvE1,refIndexL0,refIndexL1,interPredRefMode)，左上角样本位置(xE,yE)，色度 L0 运动矢量 mvC0，色度 L1 运动矢量 mvC1，宽度 W 和高度 H。

由运动信息 motionInfo 按 8.9.2.5 得到 scaleFlagL0、scaleFlagL1、scaleShiftL0 和 scaleShiftL1。

对于亮度编码块，执行以下操作：

——如果当前块的预测参考模式 interPredRefMode 等于‘PRED_List0’，则亮度预测样本矩阵 predMatrixL0 的元素 predMatrixL0[x][y]的值是参考图像队列 0 中参考索引为 refIndexL0 的参考图像的 1/16 精度亮度样本矩阵中(xPos,yPos)位置的样本值。其中 MvE 是当前预测单元的 L0 运动矢量；

```
xPos = ((((xE + x) << 4) + MvE->x) << scaleShiftL0) >> 1
yPos = ((((yE + y) << 4) + MvE->y) << scaleShiftL0) >> 1
```

——如果 interPredRefMode 等于‘PRED_List1’，则亮度预测样本矩阵 predMatrixL1 的元素 predMatrixL1[x][y]的值是参考图像队列 1 中参考索引为 RefIndexL1 的参考图像的 1/16 精度亮度样本矩阵中(xPos,yPos)位置的样本值。其中 MvE 是当前预测单元的 L1 运动矢量；

```
xPos = ((((xE + x) << 4) + MvE->x) << scaleShiftL1) >> 1
yPos = ((((yE + y) << 4) + MvE->y) << scaleShiftL1) >> 1
```

——如果 interPredRefMode 等于‘PRED_List01’，则亮度预测样本矩阵 predMatrixL0 的元素 predMatrixL0[x][y]的值是参考图像队列 0 中参考帧索引为 RefIndexL0 的参考图像的 1/16 精度亮度样本矩阵中(x0Pos,y0Pos)位置的样本值；predMatrixL1 的元素 predMatrixL1[x][y]的值是参考图像队列 1 中参考索引为 RefIndexL1 的参考图像的 1/16 精度亮度样本矩阵中(x1Pos,y1Pos)位置的样本值。其中 MvE0 和 MvE1 分别是当前预测单元的 L0 运动矢量和 L1 运动矢量。

```
x0Pos = ((((xE + x) << 4) + MvE0->x) << scaleShiftL0) >> 1
y0Pos = ((((xE + x) << 4) + MvE0->y) << scaleShiftL0) >> 1
x1Pos = ((((xE + x) << 4) + MvE1->x) << scaleShiftL1) >> 1
y1Pos = ((((xE + x) << 4) + MvE1->y) << scaleShiftL1) >> 1
```

对于色度编码块，执行以下操作：

——如果 interPredRefMode 等于‘PRED_List0’，则色度预测样本矩阵 predMatrixL0 的元素 predMatrixL0[x][y]的值是参考图像队列 0 中参考索引为 RefIndexL0 的参考图像的 1/32 精度色度样本矩阵中(xPos,yPos)位置的样本值；

```
xPos = ((((xE + 2 * x) << 4) + mvC0->x) << scaleShiftL0) >> 1
yPos = ((((yE + 2 * y) << 4) + mvC0->y) << scaleShiftL0) >> 1
```

——如果 interPredRefMode 等于‘PRED_List1’，则色度预测样本矩阵 predMatrixL1 的元素 predMatrixL1[x][y]的值是参考图像队列 1 中参考索引为 RefIndexL1 的参考图像的 1/32 精度色度样本矩阵中(xPos,yPos)位置的样本值；

```
xPos = ((((xE + 2 * x) << 4) + mvC1->x) << scaleShiftL1) >> 1
yPos = ((((yE + 2 * y) << 4) + mvC1->y) << scaleShiftL1) >> 1
```

——如果 interPredRefMode 等于‘PRED_List01’，则色度预测样本矩阵 predMatrixL0 的元素 predMatrixL0[x][y]的值是参考图像队列 0 中参考索引为 RefIndexL0 的参考图像的 1/32 精度色度样本矩阵中 x0Pos,y0Pos)位置的样本值；predMatrixL1 的元素 predMatrixL1[x][y]的值是参考图像队列 1 中参考索引为 RefIndexL1 的参考图像的 1/32 精度色度样本矩阵中 x1Pos,y1Pos)位置的样本值。

T/AI 109.8-2025

```
x0Pos = ((((xE + 2 * x) << 4) + mvC0->x) << scaleShiftL0) >> 1
y0Pos = ((((yE + 2 * y) << 4) + mvC0->y) << scaleShiftL0) >> 1
x1Pos = ((((xE + 2 * x) << 4) + mvC1->x) << scaleShiftL1) >> 1
y1Pos = ((((yE + 2 * y) << 4) + mvC1->y) << scaleShiftL1) >> 1
```

上述过程中，参考图像的亮度 1/16 精度样本矩阵和色度 1/32 精度样本矩阵中各个位置的值分别按 8.9.2.6.4 和 8.9.2.7.3 插值得到。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，x 的取值范围是 0～W-1，y 的取值范围是 0～H-1。

8.9.2.3.3 改善仿射预测样本

本条的输入是参考队列 K（K 为 0 或 1），运动矢量 mv，参考索引 refIndex，预测样本矩阵 predMatrix，ASRHorFlag，ASRVerFlag，子块偏差矩阵 dMv，左上角样本位置(xE,yE)，宽度 W 和高度 H。

本条的输出是改善后的预测样本矩阵 predMatrixLK（K 为 0 或 1）。

第 1 步，设 predMatrixTemp 是(W+2)×(H+2)的临时预测样本矩阵，refLumaMatrix 是参考图像队列 K 中参考帧索引为 refIndex 的参考图像亮度样本矩阵。按以下方法给 predMatrixTmp 赋值。

```
for (i=0; i<=W+1; i++) {
    for (j=0; j<=H+1; j++) {
        xPos = xE  －  1 + i + ((mv->x + 7) >> 4)
        yPos = yE  －  1 + j + ((mv->y + 7) >> 4)
        predMatrixTemp[i][j] = refLumaMatrix[xPos][yPos]
    }
}
for (i=0; i<W; i++) {
    for (j=0; j<H; j++)
        predMatrixTemp[i+1][j+1] = predMatrix[i][j]
}
```

第 2 步，计算 predMatrix[i][j]。

a) 如果 ASRHorFlag 等于 1 且 ASRVerFlag 等于 1：

```
for (i=0; i<W; i++) {
    for (j=0; j<H; j++)
        predMatrix[i][j] = (PredMatrixTemp[i][j] * (-dMv[i][j][0]  －  dMv[i][j][1]) +
            PredMatrixTemp[i+1][j] * ((-dMv[i][j][1]) << 3) +
            PredMatrixTemp[i+2][j] * (dMv[i][j][0]  －  dMv[i][j][1]) +
            PredMatrixTemp[i][j+1] * ((-dMv[i][j][0]) << 3) +
            PredMatrixTemp[i+1][j+1] * (1 << 9) +
            PredMatrixTemp[i+2][j+1] * (dMv[i][j][0] << 3) +
            PredMatrixTemp[i][j+2] * (-dMv[i][j][0] + dMv[i][j][1]) +
            PredMatrixTemp[i+1][j+2] * (dMv[i][j][1] << 3) +
            PredMatrixTemp[i+2][j+2] * (dMv[i][j][0] + dMv[i][j][1]) + (1 << 8)) >> 9
}
```

b) 否则，如果 ASRHorFlag 等于 1：

T/AI 109.8-2025

```
for (i=0; i<W; i++) {
    for (j=0; j<H; j++)
        predMatrix[i][j] = (PredMatrixTemp[i][j+1] * (-dMv[i][j][0]) +
            PredMatrixTemp[i+1][j+1] * (1 << 6) +
            PredMatrixTemp[i+2][j+1] * dMv[i][j][0] + (1 << 5)) >> 6
}
```

c) 否则：

```
for (i=0; i<W; i++) {
    for (j=0; j<H; j++)
        predMatrix[i][j] = (PredMatrixTemp[i+1][j] * (-dMv[i][j][1]) +
            PredMatrixTemp[i+1][j+1] * (1 << 6) +
            PredMatrixTemp[i+1][j+2] * dMv[i][j][1] + (1 << 5)) >> 6
}
```

第 3 步，限制 predMatrix[i][j]的取值范围。

```
for (i=0; i<W; i++) {
    for (j=0; j<H; j++)
        predMatrix[i][j] = Clip3(0, (1 << BitDepth) - 1, predMatrix [i][j])
}
```

8.9.2.4 导出解码端运动改良的样本

8.9.2.4.1 概述

本条的输入是运动信息 motionInfo(interPredRefMode, mvE0, mvE1, RefIndexL0, RefIndexL1)，左上角样本位置(xCb, yCb)，宽度 nNbW 和高度 nNbH。

本条的输出是改良后的预测样本矩阵 predMatrixL0 和 predMatrixL1。

按 8.9.2.4.2 得到改良后的预测样本矩阵 predMatrixL0 和 predMatrixL1。

8.9.2.4.2 解码端运动矢量的改良方法

本条的输入是运动信息 motionInfo(interPredRefMode, mvE0, mvE1, RefIndexL0, RefIndexL1)，左上角样本位置(xCb, yCb)，宽度 nNbW 和高度 nNbH。

本条的输出是改良后的预测样本矩阵 predMatrixL0 和 predMatrixL1。

第 1 步，首先，将运动矢量 mvE0 和 mvE1 水平分量范围限制在((-LcuSize-4 - xCb)<<2)到((PicWidthInLuma+LcuSize+4 - xCb - nNbW)<<2)之间，垂直分量范围限制在((-LcuSize- 4 - yCb)<<2)和((PicHeightInLuma+LcuSize+4 - yCb - nNbH)<<2)之间；然后，将运动矢量 mvE0 和 mvE1 水平分量和垂直分量限制在-32768～32767；再由 mvE0 和 mvE1 得到起始运动矢量 sMV0 和 sMV1。

```
signMV0->x = mvE0->x >= 0 ? 1 : (-1)
signMV0->y = mvE0->y >= 0 ? 1 : (-1)
signMV1->x = mvE1->x >= 0 ? 1 : (-1)
signMV1->y = mvE1->y >= 0 ? 1 : (-1)
absMV0->x = Clip3(0, 32763, Abs(mvE0->x))
absMV0->y = Clip3(0, 32763, Abs(mvE0->y))
```

T/AI 109.8-2025

```
absMV1->x = Clip3(0, 32763, Abs(mvE1->x))
absMV1->y = Clip3(0, 32763, Abs(mvE1->y))
xFracMV0->x = absMV0->x & 3
yFracMV0->y = absMV0->y & 3
xFracMV1->x = absMV1->x & 3
yFracMV1->y = absMV1->y & 3
sMV0->x = (((absMV0->x >> 2) << 2) + (((xFracMV0->x > 2) ? 1 : 0) << 2)) * signMV0->x
sMV0->y = (((absMV0->y >> 2) << 2) + (((yFracMV0->y > 2) ? 1 : 0) << 2)) * signMV0->y
sMV1->x = (((absMV1->x >> 2) << 2) + (((xFracMV1->x > 2) ? 1 : 0) << 2)) * signMV1->x
sMV1->y = (((absMV1->y >> 2) << 2) + (((yFracMV1->y > 2) ? 1 : 0) << 2)) * signMV1->y
sMV0->x = Clip3(-32760, 32756, sMV0->x)
sMV0->y = Clip3(-32760, 32756, sMV0->y)
sMV1->x = Clip3(-32760, 32756, sMV1->x)
sMV1->y = Clip3(-32760, 32756, sMV1->y)
```

第 2 步，设 subW 和 subH 分别是子块的宽度和高度，将当前块的亮度图像块划分为 subWNum×subWNum 个不重叠且位置相邻的子块。计算第 i 个子块的左上角样本位置(xCbSub, yCbSub)，子块的运动矢量 ssubMV0 和 ssubMV1。其中 i 的取值范围是 0～subWNum×subWNum-1。

```
subW = nNbW < 16 ? nNbW : 16
subH = nNbH < 16 ? nNbH : 16
subWNum= nNbW / subW
subHNum= nNbH / subH
xCbSub= subW * (i % subWNum)
yCbSub = subH * (i / subWNum)
ssubMV0->x= sMV0->x
ssubMV0->y= sMV0->y
ssubMV1->x= sMV1->x
ssubMV1->y= sMV1->y
```

第 3 步，由(xCb, yCb) 和(nNbW, nNbH)，对 ssubMV0 和 ssubMV1 进行限制。

```
if ((ssubMV0->x < ((-128 - xCb) << 2)) || (ssubMV0->x > ((PicWidthInLuma + 128 - nNbW - xCb) << 2)) {
    ssubMV0->x = Clip3(((-128 - xCb) << 2), ((PicWidthInLuma + 128 - nNbW - xCb) << 2), ssubMV0->x)
}
if ((ssubMV0->y < ((-128 - yCb) << 2)) || (ssubMV0->y > ((PicHeightInLuma + 128 - nNbH - yCb) << 2)) {
    ssubMV0->y = Clip3(((-128 - yCb) << 2), ((PicHeightInLuma + 128 - nNbH - yCb) <<2 ), ssubMV0->y)
if ((ssubMV1->x < ((-128 - xCb) << 2)) || (ssubMV1->x > ((PicWidthInLuma + 128 - nNbW - xCb) << 2)) {
    ssubMV1->x = Clip3(((-128 - xCb) << 2), ((PicWidthInLuma + 128 - nNbW - xCb) << 2), ssubMV1->x)
}
if ((ssubMV1->y < ((-128 - yCb) << 2)) || (ssubMV1->y > ((PicHeightInLuma + 128 - nNbH - yCb) << 2)) {
    ssubMV1->y = Clip3(((-128 - yCb) << 2), ((PicHeightInLuma + 128 - nNbH - yCb) <<2 ), ssubMV1->y)
}
```

第 4 步，由(xCb, yCb)、(xCbSub, yCbSub)、ssubMV0 和 ssubMV1 按 8.9.2.4.3 得到子块的亮度搜索窗口样本矩阵 predSearchMatrixL0 和 predSearchMatrixL1，亮度搜索窗口样本矩阵的宽度和高度分别是 SearchW 和 SearchH；按 8.9.2.4.4 得到子块的色度搜索窗口样本矩阵 predSearchMatrixC0 和

T/AI 109.8-2025

predSearchMatrixC1，色度搜索窗口样本矩阵的宽度和高度分别是 SearchWC 和 SearchHC。

```
SearchW = subW+FilterSize+2*IterCount
SearchH = subH+FilterSize+2*IterCount
SearchWC = (subW>>1)+FilterSizeC+2*IterCount
SearchHC = (subH>>1)+FilterSizeC+2*IterCount
```

其中 IterCount 等于 2，FilterSize 等于 8，FilterSizeC 等于 4，见图 32。

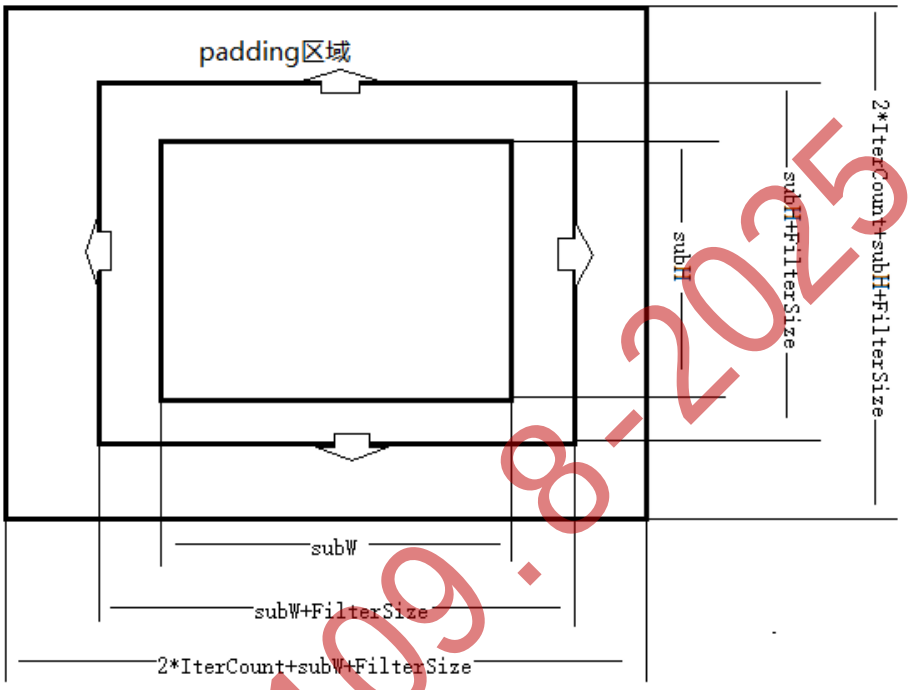


图32　亮度/色度搜索窗口样本矩阵生成方式示意图

第 5 步，遍历 index=0～20，计算 5×5 的子块绝对值误差和矩阵 sadMatrix。

```
LeftFilter = (FilterSize >> 1)  – 1
searchStart = IterCount + LeftFilter
xPos = arrayPosX[index]
yPos = arrayPosY[index]
sadTmp = 0
for (x=0; x<subW; x++) {
      for (y=0; y<subH; y++)
            sadTmp          +=          Abs(predSearchMatrixL0[x+SearchStart+xPos][y+searchStart+yPos]          –
predSearchMatrixL1[x+searchStart-xPos][y+SearchStart-yPos])
}
sadMatrix[xPos+2][yPos+2] = sadTmp
```

其中 arrayPosX[index]和 arrayPosY[index]的值见表 128。

T/AI 109.8-2025

表128 arrayPosX 和 arrayPosY 的值

index 值	arrayPosX 值	arrayPosY 值
0	0	0
1	0	1
2	0	-1
3	1	0
4	-1	0
5	0	2
6	1	1
7	2	0
8	1	-1
9	0	-2
10	-1	-1
11	-2	0
12	-1	1
13	1	2
14	2	1
15	-1	2
16	-2	1
17	1	-2
18	2	-1
19	-1	-2
20	-2	-1

第 6 步，在 sadMatrix 中查找目标位置 arrayPosBestX 和 arrayPosBestY。

```
startX = 2
startY = 2
arrayPosBestX = startX
arrayPosBestY = startY
for (j=0; j<3; j++){
    sadEMin = sadMatrix[startX][startY]
    best0 = 0
    for (index=1; index<5; index++) {
        if (startX+arrayPosX[idx]>= 0 && startX+arrayPosX[idx] < 5 && startY+arrayPosY[idx]>= 0 && startY+arrayPosY[idx] < 5) {
            if (sadEMin > sadMatrix[startX+arrayPosX[index]][startY+arrayPosY[index]]) {
                sadEMin = sadMatrix[startX+arrayPosX[index]][startY+arrayPosY[index]]
                best0 = index
            }
        }
```

T/AI 109.8-2025

```
    }
    arrayPosBestX = startX + arrayPosX[best0]
    arrayPosBestY = startY + arrayPosY[best0]
    if (sadEMin == 0 || best0 == 0 || j == 2)
        break
    startX += arrayPosX[best0]
    startY += arrayPosY[best0]
}
```

第 7 步，从 arrayPosBestX 和 arrayPosBestY 得到子块整数像素位置偏移 subnMV0 和 subnMV1。

```
subnMV0->x-= arrayPosBestX  -  2
subnMV0->y = arrayPosBestY  -  2
subnMV1->x-= 2  -  arrayPosBestX
subnMV1->y = 2  -  arrayPosBestY
```

第 8 步，计算子块亚像素位置偏移 subqpelX 和 subqpelY。将 subqpelX 和 subqpelY 初始化为 0，如果以下条件均满足，由 sadH0、sadH1 和 sadH2 按 8.9.2.4.5 得到 subqpelX；由 sadV0、sadV1 和 sadV2 按 8.9.2.4.5 得到 subqpelY。其中，sadH0、sadH1、sadH2、sadV0、sadV1 和 sadV2 的计算如下：

a) sadH1 不等于 0；

b) array_pos_best_x 大于或等于-1 且小于或等于 1，同时 array_pos_best_y 大于或等于-1 且小于或等于 1。

```
sadH0 = sadMatrix[arrayPosBestX-1][ arrayPosBestY]
sadH1 = sadMatrix[arrayPosBestX][arrayPosBestY]
sadH2 = sadMatrix[arrayPosBestX+1][arrayPosBestY]
sadV0 = sadMatrix[arrayPosBestX][arrayPosBestY-1]
sadV1 = sadMatrix[arrayPosBestX][arrayPosBestY]
sadV2 = sadMatrix[arrayPosBestX][arrayPosBestY+1]
```

第 9 步，根据子块整数像素位置偏移 subnMV0 和 subnMV1，子块亚像素位置偏移 subqpelX 和 subqpelY，分别从亮度和色度搜索窗口样本矩阵 predSearchMatrixL0、predSearchMatrixL1、predSearchMatrixC0、和 predSearchMatrixC1 得到亮度预测样本矩阵和色度预测样本矩阵。

对 predMatrixL0 的操作如下：

令subqpelXL0等于subqpelX，subqpelYL0等于subqpelY。

a) 限制 subnMV0、subqpelXL0 和 subqpelYL0 的取值范围；

```
subMVTempL0->x = ((subnMV0->x) << 2) + subqpelXL0 + sMV0->x
if ((subMVTempL0->x < ((-128  -  xCb  -  xCbSub) << 2)) || (subMVTempL0->x > ((PicWidthInLuma + 128
        -  subW  -  xCb  -  xCbSub) << 2)) {
    subMVTempL0->x = Clip3(((-128  -  xCb  -  xCbSub) << 2), ((PicWidthInLuma + 128  -  subW  -
xCb  -  xCbSub) << 2), subMVTempL0->x)
    subqpelXL0 = subMVTempL0->x & 0x3
    subnMV0->x = (subMVTempL0->x >> 2)  -  (sMV0->x >> 2)
    subnMV0->x = Clip3(-IterCount, IterCount, subnMV0->x)
}
subMVTemp->L0->y = ((subnMV0->y) << 2) + subqpelYL0 + sMV0->y
```

T/AI 109.8-2025

```
if ((subMVTempL0->y < ((-128 - yCb - yCbSub) << 2)) || (subMVTempL0->y > ((PicHeightInLuma + 128
        - subH - yCb - yCbSub) << 2)) {
    subMVTempL0->y = Clip3(((-128 - yCb - yCbSub) << 2), ((PicHeightInLuma + 128 - subH - yCb
- yCbSub) <<2 ), subMVTempL0->y)
    subqpelYL0 = subMVTempL0->y & 0x3
    subnMV0->y = (subMVTempL0->y >> 2) - (sMV0->y >> 2)
    subnMV0->y = Clip3(-IterCount, IterCount, subnMV0->y)
}
```

b) 亮度预测样本矩阵 predMatrixL0 的元素 predMatrixL0[x+xCbSub][y+yCbSub]的值等于 predSearchMatrixL0 的 1/4 精度亮度样本矩阵中位置为(x0, y0)的样本值；

```
x0 = ((x + IterCount + LeftFilter + subnMV0->x) <<2) + subqpelXL0
y0 = ((y + IterCount + LeftFilter + subnMV0->y) <<2) + subqpelYL0
```

c) 色度预测样本矩阵 predMatrixL0 的元素 predMatrixL0[x+(xCbSub>>1)][y+(yCbSub>>1)]的值等于 predSearchMatrixC0 的 1/8 精度色度样本矩阵中位置为(x0, y0)的样本值。

```
x0 = ((2 * x + 2 * IterCount + 2 * LeftFilterC + subnMV0->x) << 2) + (subqpelXL0)
y0 = ((2 * y + 2 * IterCount + 2 * LeftFilterC + subnMV0->y) << 2) + (subqpelYL0)
```

对 predMatrixL1 的操作如下：

令subqpelXL1等于-subqpelX，subqpelYL1等于-subqpelY。

a) 限制 subnMV1、subqpelXL1 和 subqpelYL1 的取值范围；

```
subMVTempL1->x = ((subnMV1->x) << 2) + subqpelXL1 + sMV1->x
if ((subMVTempL1->x < ((-128 - xCb - xCbSub) << 2)) || (subMVTempL1->x > ((PicWidthInLuma + 128 -
        subW - xCb - xCbSub) << 2)) {
    subMVTempL1->x = Clip3(((-128 - xCb - xCbSub) << 2), ((PicWidthInLuma + 128 - subW - xCb
- xCbSub) << 2), subMVTempL1->x)
    subqpelXL1 = subMVTempL1->x & 0x3
    subnMV1->x = (subMVTempL1->x >> 2) - (sMV1->x >> 2)
    subnMV1->x = Clip3(-IterCount, IterCount, subnMV1->x)
}
subMVTemp->L1->y = ((subnMV1->y) << 2) + subqpelYL1 + sMV1->y
if ((subMVTempL1->y < ((-128 - yCb - yCbSub) << 2)) || (subMVTempL1->y > ((PicHeightInLuma + 128
    - subH - yCb - yCbSub) << 2)) {
    subMVTempL1->y = Clip3(((-128 - yCb - yCbSub) << 2), ((PicHeightInLuma + 128 - subH - yCb
- yCbSub) <<2 ), subMVTempL1->y)
    subqpelYL1 = subMVTempL1->y & 0x3
    subnMV1->y = (subMVTempL1->y >> 2) - (sMV1->y >> 2)
    subnMV1->y = Clip3(-IterCount, IterCount, subnMV1->y)
}
```

b) 亮度预测样本矩阵 predMatrixL1 的元素 predMatrixL1[x+xCbSub][y+yCbSub]的值等于 predSearchMatrixL1 的 1/4 精度亮度样本矩阵中位置为(x1, y1)的样本值；

```
x1 = ((x + IterCount + LeftFilter + subnMV1->x) <<2) + subqpelXL1)
y1 = ((y + IterCount + LeftFilter + subnMV1->y) <<2) + subqpelYL1)
```

T/AI 109.8-2025

c) 色度预测样本矩阵 predMatrixL1 的元素 predMatrixL1[x+(xCbSub>>1)][y+(yCbSub>>1)]的值等于 predSearchMatrixC1 的 1/8 精度色度样本矩阵中位置为(x1, y1)的样本值。

```
x1 = ((2 * x + 2 * IterCount + 2 * LeftFilterC + subnMV1->x) << 2) + (subqpelXL1)
y1 = ((2 * y + 2 * IterCount + 2 * LeftFilterC + subnMV1->y) << 2) + (subqpelYL1)
```

上述过程中，predSearchMatrixL0、predSearchMatrixL1、predSearchMatrixC0、predSearchMatrixC1 的亮度 1/4 精度样本矩阵和色度 1/8 精度样本矩阵中各个位置的值分别按 8.9.2.6.2 和 8.9.2.7.2 插值得到，paddingBuf 外的整数样本应使用该 paddingBuf 内距离该样本最近的整数样本（边缘或角样本）代替。

8.9.2.4.3 生成亮度搜索窗口样本矩阵

本条的输入是当前块左上角样本的位置(xCb, yCb)，当前子块左上角样本的位置(xCbSub, yCbSub)，当前子块的运动矢量 ssubMV0 和 ssubMV1。

本条的输出是亮度搜索窗口样本矩阵 predSearchMatrixL0 和 predSearchMatrixL1。

第 1 步，设 ref0LumaMatrix 是参考图像队列 0 中参考帧索引为 RefIndexL0 的整数精度亮度样本矩阵，ref1LumaMatrix 是参考图像队列 1 中参考帧索引为 RefIndexL1 的整数精度亮度样本矩阵。计算 predSearchMatrixL0[x+IterCount][y+IterCount]和 predSearchMatrixL1[x+IterCount][y+IterCount]。

```
LeftFilter = (FilterSize >> 1) - 1
for (x=0; x<subW+FilterSize; x++) {
    for (y=0; y<subH+FilterSize; y++) {
        x0Pos = (xCb + xCbSub + x - LeftFilter) + (ssubMV0->x >> 2)
        y0Pos = (yCb + yCbSub + y - LeftFilter) + (ssubMV0->y >> 2)
        predSearchMatrixL0[x+IterCount][y+IterCount] = ref0LumaMatrix[x0Pos][y0Pos]
        x1Pos = (xCb + xCbSub + x - LeftFilter) + (ssubMV1->x >> 2)
        y1Pos = (yCb + yCbSub + y - LeftFilter) + (ssubMV1->y >> 2)
        predSearchMatrixL1[x+IterCount][y+IterCount] = ref1LumaMatrix[x1Pos][y1Pos]
    }
}
```

第 2 步，计算 predSearchMatrixL0[x1][y1]和 predSearchMatrixL1[x1][y1]。

```
for (x1=0; x1<IterCount; x1++) {
    for (y1=IterCount; y1<subH+FilterSize+IterCount; y1++) {
        predSearchMatrixL0[x1][y1] = predSearchMatrixL0[IterCount][y1]
        predSearchMatrixL1[x1][y1] = predSearchMatrixL1[IterCount][y1]
    }
}
```

第 3 步，计算 predSearchMatrixL0[x2][y2]和 predSearchMatrixL1[x2][y2]。

```
for (x2= IterCount+subW+FilterSize; x2<2*IterCount+subW+FilterSize; x2++) {
    for (y2=InterCount; y2<subH+FilterSize+IterCount; y2++) {
        predSearchMatrixL0[x2][y2] = predSearchMatrixL0[IterCount+subW+FilterSize-1][y2]
        predSearchMatrixL1[x2][y2] = predSearchMatrixL1[IterCount+subW+FilterSize-1][y2]
    }
}
```

T/AI 109.8-2025

第 4 步，计算 predSearchMatrixL0[x3][y3]和 predSearchMatrixL1[x3][y3]。

```
for (x3= 0; x3<2*IterCount+subW+FilterSize; x3++) {
    for (y3=0; y3<IterCount; y3++) {
        predSearchMatrixL0[x3][y3] = predSearchMatrixL0[x3][IterCount]
        predSearchMatrixL1[x3][y3] = predSearchMatrixL1[x3][IterCount]
    }
}
```

第 5 步，计算 predSearchMatrixL0[x4][y4]和 predSearchMatrixL1[x4][y4]。

```
for (x4= 0; x4<2*IterCount+subW+FilterSize; x4++) {
    for (y4= IterCount+subH+FilterSize; y4<2*IterCount+subH+FilterSize; y4++) {
        predSearchMatrixL0[x4][y4] = predSearchMatrixL0[x4][IterCount+subH+FilterSize-1]
        predSearchMatrixL1[x4][y4] = predSearchMatrixL1[x4][IterCount+subH+FilterSize-1]
    }
}
```

8.9.2.4.4 生成色度搜索窗口样本矩阵

本条的输入是当前块左上角样本的位置(xCb, yCb)，当前子块左上角样本的位置(xCbSub, yCbSub)，当前子块的运动矢量 ssubMV0 和 ssubMV1。

本条的输出是色度搜索窗口样本矩阵 predSearchMatrixC0 和 predSearchMatrixC1。

第 1 步，设 ref0ChromaMatrix 是参考图像队列 0 中参考帧索引为 RefIndexL0 的整数精度色度样本矩阵，ref1ChromaMatrix 是参考图像队列 1 中参考帧索引为 RefIndexL1 的整数精度色度样本矩阵。计算 predSearchMatrixL0[x+IterCount][y+IterCount]和 predSearchMatrixL1[x+IterCount][y+IterCount]。

```
LeftFilterC = (FilterSizeC >> 1)  -  1
for (x=0; x<(subW>>1)+FilterSizeC; x++) {
    for (y=0; y<(subH>>1)+FilterSizeC; y++) {
        x0Pos = ((xCb + xCbSub) >> 1) + x  -  LeftFilterC + (ssubMV0->x >> 3)
        y0Pos = ((yCb + yCbSub) >> 1) + y  -  LeftFilterC + (ssubMV0->y >> 3)
        predSearchMatrixC0[x+IterCount][y+IterCount] = ref0ChromaMatrix[x0Pos][y0Pos]
        x1Pos = ((xCb + xCbSub) >> 1) + x  -  LeftFilterC + (ssubMV1->x >> 3)
        y1Pos = ((yCb + yCbSub) >> 1) + y  -  LeftFilterC + (ssubMV1->y >> 3)
        predSearchMatrixC1[x+IterCount][y+IterCount] = ref1ChromaMatrix[x1Pos][y1Pos]
    }
}
```

第 2 步，计算 predSearchMatrixC0[x1][y1]和 predSearchMatrixC1[x1][y1]。

```
for (x1=0; x1<IterCount; x1++) {
    for (y1=IterCount; y1<(subH>>1)+FilterSizeC+IterCount; y1++) {
        predSearchMatrixC0[x1][y1] = predSearchMatrixC0[IterCount][y1]
        predSearchMatrixC1[x1][y1] = predSearchMatrixC1[IterCount][y1]
    }
}
```

第 3 步，计算 predSearchMatrixC0[x2][y2]和 predSearchMatrixC1[x2[y2]。

T/AI 109.8-2025

```
for (x2=IterCount+(subW>>1)+FilterSizeC; x2<2*IterCount+(subW>>1)+FilterSizeC; x2++) {
    for (y2=IterCount; y2<(subH>>1)+FilterSizeC+IterCount; y2++) {
        predSearchMatrixC0[x2][y2] = predSearchMatrixC0[IterCount+(subW>>1)+FilterSizeC-1][y2]
        predSearchMatrixC1[x2][y2] = predSearchMatrixC1[IterCount+(subW>>1)+FilterSizeC-1][y2]
    }
}
```

第 4 步，计算 predSearchMatrixC0[x3][y3]和 predSearchMatrixC1[x3[y3]。

```
for (x3=0; x3<2*IterCount+(subW>>1)+FilterSizeC; x3++) {
    for (y3=0; y3<IterCount; y3++) {
        predSearchMatrixC0[x3][y3] = predSearchMatrixC0[x3][IterCount]
        predSearchMatrixC1[x3][y3] = predSearchMatrixC1[x3][IterCount]
    }
}
```

第 5 步，计算 predSearchMatrixC0[x4][y4]和 predSearchMatrixC1[x4[y4]。

```
for (x4=0; x4<2*IterCount+(subW>>1)+FilterSizeC; x4++) {
    for (y4=IterCount+(subH>>1)+FilterSizeC; y4<2*IterCount+(subH>>1)+FilterSizeC; y4++) {
        predSearchMatrixC0[x4][y4] = predSearchMatrixC0[x4][IterCount+(subH>>1)+FilterSizeC-1]
        predSearchMatrixC1[x4][y4] = predSearchMatrixC0[x4][IterCount+(subH>>1)+FilterSizeC-1]
    }
}
```

8.9.2.4.5 计算亚像素位置偏移

本条的输入是平均绝对误差和 sad0、sad1 和 sad2。

本条的输出是计算亚像素位置偏移 deltaMv。

第 1 步，计算 iNum 和 iDenom。

```
iNum = (sad0 – sad2) << 4)
iDenom = ((sad0 + sad2 – (sad1 << 1)))
```

第 2 步，如果 iDenom 不等于 0，计算 deltaMv。

```
if ((sad0 != sad1) && (sad2 != sad1)) {
    sign = 0
    if (iNum < 0) {
        sign = 1
        iNum = -iNum
    }
    q = 0
    iDenom = (iDenom << 3)
    if (iNum >= iDenom) {
        iNum -= iDenom
        q++
    }
    q = (q << 1)
```

T/AI 109.8-2025

```
        iDenom = (iDenom >> 1)
        if (iNum >= iDenom) {
            iNum -= iDenom
            q++
        }
        q = (q << 1)
        if (iNum >= (iDenom >> 1))
            q++
        if (sign != 0)
            deltaMv = -q
        else
            deltaMv = q
    }
    else if (sad0 == sad1)
        deltaMv = -8
    else
        deltaMv = 8
    deltaMv = deltaMv >> 2
```

8.9.2.5 导出参考帧缩放参数

本条的输入是运动信息 motionInfo(mvE0，mvE1，refIndexL0，refIndexL1，interPredRefMode)。

本条的输出是参考帧缩放标志位 scaleFlagL0 和 scaleFlagL1，以及参考帧缩放偏移 scaleShiftL0 和 scaleShiftL1。

按照以下方式导出 scaleFlagL0、scaleFlagL1、scaleShiftL0 和 scaleShiftL1。

如果当前块的预测参考模式 interPredRefMode 等于‘PRED_List0’或‘PRED_List01’：

——如果参考图像队列 0 中参考索引为 RefIndexL0 的参考图像的水平尺寸等于当前图像的水平尺寸，将 scaleFlagL0 设置为 0，将 scaleShiftL0 设置为 1；

——否则，如果参考图像队列 0 中参考索引为 RefIndexL0 的参考图像的水平尺寸大于当前图像的水平尺寸，将 scaleFlagL0 设置为 1，将 scaleShiftL0 设置为 2；

——否则（参考图像队列 0 中参考索引为 RefIndexL0 的参考图像的水平尺寸小于当前图像的水平尺寸），将 scaleFlagL0 设置为 1，scaleShiftL0 设置为 0。

如果当前块的预测参考模式 interPredRefMode 等于‘PRED_List1’或‘PRED_List01’，

——如果参考图像队列 1 中参考索引为 RefIndexL1 的参考图像的水平尺寸等于当前图像的水平尺寸，将 scaleFlagL1 设置为 0，将 scaleShiftL1 设置为 1；

——否则，如果参考图像队列 1 中参考索引为 RefIndexL1 的参考图像的水平尺寸大于当前图像的水平尺寸，将 scaleFlagL1 设置为 1，scaleShiftL1 设置为 2；

——否则（参考图像队列 1 中参考索引为 RefIndexL1 的参考图像的水平尺寸小于当前图像的水平尺寸），将 scaleFlagL1 设置为 1，scaleShiftL1 设置为 0。

否则，将 scaleFlagL1 设置为 0，将 scaleShiftL1 设置为 1。

8.9.2.6 亮度样本的插值过程

T/AI 109.8-2025

8.9.2.6.1 概述

如果当前预测单元的 AffineFlag 的值为 0，先按 8.9.2.6.2 进行亮度样本插值，再按 8.9.2.6.3 进行梯度计算；否则，按 8.9.2.6.4 进行亮度样本插值。

8.9.2.6.2 普通亮度样本的插值过程

图 33 给出了亮度样本矩阵中初始整数样本、1/2 样本和 1/4 样本的位置，其中用大写字母标记的是整数样本位置，用小写字母标记的是 1/2 和 1/4 样本位置。这些样本位置与其在 1/4 精度的亮度样本矩阵中的坐标(f_x, f_y)的对应关系见表 129，其中 xFracL 等于 f_x & 3, yFracl 等于 f_y & 3。

如果 IsPrivacyFlag 的值等于 1，整数样本值等于初始整数样本值。

否则，IsPrivacyFlag 的值等于 0，获取初始整数样本对应参考帧权限储存单元的权限值，若初始整数样本对应权限储存单元的权限值大于 0，则整数样本等于 1<<(BitDepth-1)。



图33 整数样本、1/2 样本和 1/4 样本的位置

T/AI 109.8-2025

表129 预测样本矩阵元素

xFracL值	yFracL值	图33中样本位置
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：

$$a0,0 = Clip1((-A_{-3,0} + 4*A_{-2,0} - 10*A_{-1,0} + 57*A_{0,0} + 19*A_{1,0} - 7*A_{2,0} + 3*A_{3,0} - A_{4,0} + 32) >> 6)$$

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

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

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：

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

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

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

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：

$$e_{0,0} = Clip1((-a'_{0,-3} + 4*a'_{0,-2} - 10*a'_{0,-1} + 57*a'_{0,0} + 19*a'_{0,1} - 7*a'_{0,2} + 3*a'_{0,3} - a'_{0,4} + (1 << 19 - BitDepth)) >> (20 - BitDepth))$$

T/AI 109.8-2025

$i_{0,0} = Clip1((-a'_{0,-3} + 4*a'_{0,-2} - 11*a'_{0,-1} + 40*a'_{0,0} + 40*a'_{0,1} - 11*a'_{0,2} + 4*a'_{0,3} - a'_{0,4} + (1 << 19 - BitDepth)) >> (20 - BitDepth))$

$p_{0,0} = Clip1((-a'_{0,-3} + 3*a'_{0,-2} - 7*a'_{0,-1} + 19*a'_{0,0} + 57*a'_{0,1} - 10*a'_{0,2} + 4*a'_{0,3} - a'_{0,4} + (1 << 19 - BitDepth)) >> (20 - BitDepth))$

$f_{0,0} = Clip1((-b'_{0,-3} + 4*b'_{0,-2} - 10*b'_{0,-1} + 57*b'_{0,0} + 19*b'_{0,1} - 7*b'_{0,2} + 3*b'_{0,3} - b'_{0,4} + (1 << 19 - BitDepth)) >> (20 - BitDepth))$

$j_{0,0} = Clip1((-b'_{0,-3} + 4*b'_{0,-2} - 11*b'_{0,-1} + 40*b'_{0,0} + 40*b'_{0,1} - 11*b'_{0,2} + 4*b'_{0,3} - b'_{0,4} + (1 << 19 - BitDepth)) >> (20 - BitDepth))$

$q_{0,0} = Clip1((-b'_{0,-3} b'_{0,-2} - 7*b'_{0,-1} + 19*b'_{0,0} + 57*b'_{0,1} - 10*b'_{0,2} + 4*b'_{0,3} - b'_{0,4} + (1 << 19 - BitDepth)) >> (20 - BitDepth))$

$g_{0,0} = Clip1((-c'_{0,-3} + 4*c'_{0,-2} - 10*c'_{0,-1} + 57*c'_{0,0} + 19*c'_{0,1} - 7*c'_{0,2} + 3*c'_{0,3} c'_{0,4} + (1 << 19 - BitDepth)) >> (20 - BitDepth))$

$k_{0,0} = Clip1((-c'_{0,-3} + 4*c'_{0,-2} - 11*c'_{0,-1} + 40*c'_{0,0} + 40*c'_{0,1} - 11*c'_{0,2} + 4*c'_{0,3} - c'_{0,4} + (1 << 19 - BitDepth)) >> (20 - BitDepth))$

$r_{0,0} = Clip1((-c'_{0,-3} c'_{0,-2} - 7*c'_{0,-1} + 19*c'_{0,0} + 57*c'_{0,1} - 10*c'_{0,2} + 4*c'_{0,3} - c'_{0,4} + (1 << 19 - BitDepth)) >> (20 - BitDepth))$

其中：

$a'_{0,i} = (-A_{-3,I} + 4*A_{-2,i} - 10*A_{-1,i} + 57*A_{0,i} + 19*A_{1,i} - 7*A_{2,i} + 3*A_{3,i} - A_{4,i}) >> (BitDepth - 8)$

$b'_{0,i} = (-A_{-3,i} + 4*A_{-2,i} - 11*A_{-1,i} + 40*A_{0,i} + 40*A_{1,i} - 11*A_{2,i} + 4*A_{3,i} - A_{4,i}) >> (BitDepth - 8)$

$c'_{0,i} = (-A_{-3,i} + 3*A_{-2,i} - 7*A_{-1,i} + 19*A_{0,i} + 57*A_{1,i} - 10*A_{2,i} + 4*A_{3,i} - A_{4,i}) >> (BitDepth - 8)$

T/AI 109.8-2025

8.9.2.6.3 双向光流亮度样本梯度的计算过程

图33给出了亮度样本矩阵中初始整数样本、1/2样本和1/4样本的位置，其中用大写字母标记的是整数样本位置，用小写字母标记的是1/2和1/4样本位置。这些样本位置与其在1/4精度的亮度样本矩阵中的坐标(f_x, f_y)的对应关系见表129，其中xFracL等于f_x & 3,yFracl等于f_y & 3。

如果 PrivacyLevel 等于 1，整数样本值等于初始整数样本值。

否则，PrivacyLevel等于0，获取初始整数样本对应参考帧权限储存单元的权限值，若初始整数样本对应权限储存单元的权限值大于0，则整数样本值等于1<<(BitDepth-1)。

梯度矩阵 gradXL0 的元素 gradXL0[x][y]的值是参考图像队列 0 中参考索引为 RefIndexL0 的参考图像的 1/4 精度亮度样本矩阵中(f_x, f_y)位置的水平梯度值，梯度矩阵 gradYL0 的元素 gradYL0[x][y]的值是其竖直梯度值。梯度矩阵 gradXL1 的元素 gradXL1[x][y]的值是参考图像队列 1 中参考索引为

RefIndexL1 的参考图像的 1/4 精度亮度样本矩阵中(f_x, f_y)位置的水平梯度值，梯度矩阵 gradYL1 的元素 gradYL1[x][y]的值是其竖直梯度值。使用 8 抽头滤波器按以下方法得到。

xFracL 的值为 0，1，2，3 的 gradXL[x][y]（包括 gradXL0[x][y]和 gradXL1[x][y]）中间值为

```
if (xFracL == 0)
    gradXL’[x][y]= (−4*A−3,y + 11*A−2,y − 39*A−1,y -1*A0,y + 41*A1,y − 14*A2,y + 8*A3,y − 2*A4,y + 2) >> 2
else if (xFracL == 1)
    gradXL’[x][y]= (−2*A−3,y + 6*A−2,y − 19*A−1,y -31*A0,y + 53*A1,y − 12*A2,y + 7*A3,y − 2*A4,y + 2) >> 2
else if (xFracL == 2)
    gradXL’[x][y]= ( -A−2,y − 50*A0,y + 50*A1,y + A3,y + 2) >> 2
else
    gradXL’[x][y]= ( 2*A−3,y − 7*A−2,y + 12*A−1,y − 53*A0,y + 31*A1,y +19*A2,y − 6*A3,y + 2* A4,y + 2) >>
 2
```

yFracL 的值为 0，1，2，3 的 gradXL[x][y]（包括 gradXL0[x][y]和 gradXL1[x][y]）值为

```
if (yFracL == 0)
    gradXL[x][y]= (64* gradXL’[x][0] + 8192) >> 14
else if (yFracL == 1)
    gradXL[x][y]= (−1* gradXL’[x][-3] + 4* gradXL’[x][-2] − 10* gradXL’[x][-1] + 57* gradXL’[x][0] + 19*
 gradXL’[x][1] - 7* gradXL’[x][2] + 3* gradXL’[x][3] -1* gradXL’[x][4] + 8192) >> 14
else if (yFracL == 2)
    gradXL[x][y]= (−1* gradXL’[x][-3] + 4* gradXL’[x][-2] − 11* gradXL’[x][-1] + 40* gradXL’[x][0] + 40*
 gradXL’[x][1] - 11* gradXL’[x][2] + 4* gradXL’[x][3] -1* gradXL’[x][4] + 8192) >> 14
else
    gradXL[x][y]= (−1* gradXL’[x][-3] + 3* gradXL’[x][-2] − 7* gradXL’[x][-1] + 19* gradXL’[x][0] + 57*
 gradXL’[x][1] - 10* gradXL’[x][2] + 4* gradXL’[x][3] -1* gradXL’[x][4] + 8192) >> 14
```

xFracL 的值为 0，1，2，3 的 gradYL[x][y]（包括 gradYL0[x][y]和 gradYL1[x][y]）中间值为

```
if (xFracL == 0)
    gradYL’[x][y]= (64*A0,y + 2) >> 2
else if (xFracL == 1)
    gradYL’[x][y]= (−1*A−3,y + 4*A−2,y − 10*A−1,y +57*A0,y + 19*A1,y - 7*A2,y + 3*A3,y -1*A4,y + 2) >> 2
else if (xFracL == 2)
    gradYL’[x][y] = ( −1*A−3,y + 4*A−2,y − 11*A−1,y +40*A0,y + 40*A1,y - 11*A2,y + 4*A3,y -1*A4,y + 2) >> 2
else
    gradYL’[x][y] = ( -1*A−3,y +3*A−2,y − 7*A−1,y +19*A0,y + 57*A1,y -10*A2,y +4*A3,y -1* A4,y + 2) >> 2
```

yFracL 的值为 0，1，2，3 的 gradYL[x][y]（包括 gradYL0[x][y]和 gradYL1[x][y]）值为

```
if (yFracL == 0)
    gradYL[x][y]= (−4* gradYL’[x][-3] + 11* gradYL’[x][-2] − 39* gradYL’[x][-1] -1* gradYL’[x][0] + 41*
 gradYL’[x][1] - 14* gradYL’[x][2] + 8* gradYL’[x][3] -2* gradYL’[x][4] + 8192) >> 14
else if (yFracL == 1)
    gradYL[x][y]= (−2* gradYL’[x][-3] + 6* gradYL’[x][-2] − 19* gradYL’[x][-1] -31* gradYL’[x][0] + 53*
 gradYL’[x][1] - 12* gradYL’[x][2] + 7* gradYL’[x][3] -2* gradYL’[x][4] + 8192) >> 14
else if (yFracL == 2)
    gradYL[x][y]= (-1* gradYL’[x][-2] -50* gradYL’[x][0] + 50* gradYL’[x][1] + gradYL’[x][3] + 8192) >> 14
```

T/AI 109.8-2025

```
else
    gradYL[x][y]= (2* gradYL’[x][-3] -7* gradYL’[x][-2] +12* gradYL’[x][-1] -53* gradYL’[x][0] + 31* gradYL’[x][1] +19* gradYL’[x][2] -6* gradYL’[x][3] +2* gradYL’[x][4] + 8192) >> 14
```

8.9.2.6.4 仿射亮度样本的插值过程

参考图像亮度样本矩阵中的样本位置见图 34，A，B，C，D 是相邻初始整像素样本，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 fx & 15，dy 等于 fy & 15，其中(fx,fy)是该分像素样本在 1/16 精度的亮度样本矩阵中的坐标。整像素 $A_{x,y}$ 和周边的 255 个分像素样本 $a_{x,y}$(dx,dy)的具体位置见图 35。

如果 IsPrivacyFlag 的值等于 1，整数样本值等于初始整数样本值。

否则，IsPrivacyFlag 的值等于 0，获取初始整数样本对应参考帧权限储存单元的权限值，若初始整数样本对应权限储存单元的权限值大于 0，则整数样本等于 1<<(BitDepth-1)。

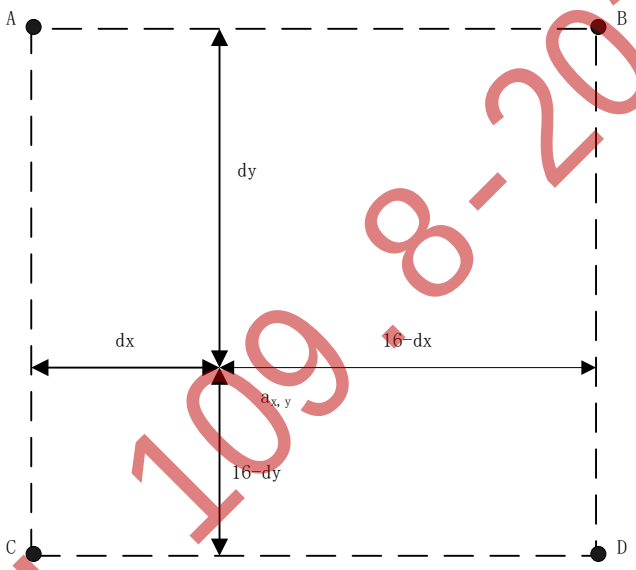


图34 仿射预测亮度样本的位置

T/AI 109.8-2025

$A_{x,y}$	$a_{x,y}(1,0)$	$a_{x,y}(2,0)$	$a_{x,y}(3,0)$	$a_{x,y}(4,0)$	$a_{x,y}(5,0)$	…	$a_{x,y}(15,0)$
$a_{x,y}(0,1)$	$a_{x,y}(1,1)$	$a_{x,y}(2,1)$	$a_{x,y}(3,1)$	$a_{x,y}(4,1)$	$a_{x,y}(5,1)$	…	$a_{x,y}(15,1)$
$a_{x,y}(0,2)$	$a_{x,y}(1,2)$	$a_{x,y}(2,2)$	$a_{x,y}(3,2)$	$a_{x,y}(4,2)$	$a_{x,y}(5,2)$	…	$a_{x,y}(15,2)$
$a_{x,y}(0,3)$	$a_{x,y}(1,3)$	$a_{x,y}(2,3)$	$a_{x,y}(3,3)$	$a_{x,y}(4,3)$	$a_{x,y}(5,3)$	…	$a_{x,y}(15,3)$
$a_{x,y}(0,4)$	$a_{x,y}(1,4)$	$a_{x,y}(2,4)$	$a_{x,y}(3,4)$	$a_{x,y}(4,4)$	$a_{x,y}(5,4)$	…	$a_{x,y}(15,4)$
$a_{x,y}(0,5)$	$a_{x,y}(1,5)$	$a_{x,y}(2,5)$	$a_{x,y}(3,5)$	$a_{x,y}(4,5)$	$a_{x,y}(5,5)$	…	$a_{x,y}(15,5)$
…	…	…	…	…	…	…	…
$a_{x,y}(0,15)$	$a_{x,y}(1,15)$	$a_{x,y}(2,15)$	$a_{x,y}(3,15)$	$a_{x,y}(4,15)$	$a_{x,y}(5,15)$	…	$a_{x,y}(15,15)$

图35　仿射预测整像素样本和分像素样本的位置

样本位置 $a_{x,0}$（x=1~15）由水平方向上距离插值点最近得 8 个整数值滤波得到，预测值得获取方式如下：

$$a_{x,0} = Clip1((f_L[x][0]*A_{-3,0} + f_L[x][1]*A_{-2,0} + f_L[x][2]*A_{-1,0} + f_L[x][3]*A_{0,0} + f_L[x][4]*A_{1,0} + f_L[x][5]*A_{2,0} + f_L[x][6]*A_{3,0} + f_L[x][7]*A_{4,0} + 32) >> 6)$$

样本位置 $a_{0,y}$（y=1~15）由垂直方向上距离插值点最近得 8 个整数值滤波得到，预测值得获取方式如下：

$$a_{0,y} = Clip1((f_L[y][0]*A_{0,-3} + f_L[y][1]*A_{0,-2} + f_L[y][2]*A_{0,-1} + f_L[y][3]*A_{0,0} + f_L[y][4]*A_{0,1} + f_L[y][5]*A_{0,2} + f_L[y][6]*A_{0,3} + f_L[y][7]*A_{0,4} + 32) >> 6)$$

样本位置 $a_{x,y}$（x=1～15，y=1～15）的预测值获取方式如下：

$$a_{x,y} = Clip1((f_L[y][0]*a'_{x,y-3} + f_L[y][1]*a'_{x,y-2} + f_L[y][2]*a'_{x,y-1} + f_L[y][3]*a'_{x,y} + f_L[y][4]*a'_{x,y+1} + f_L[y][5]*a'_{x,y+2} + f_L[y][6]*a'_{x,y+3} + f_L[y][7]*a'_{x,y+4} + (1 << (19 - BitDepth))) >> (20 - BitDepth))$$

其中：

$$a'_{x,y} = (f_L[x][0]*A_{-3,y} + f_L[x][1]*A_{-2,y} + f_L[x][2]*A_{-1,y} + f_L[x][3]*A_{0,y} + f_L[x][4]*A_{1,y} + f_L[x][5]*A_{2,y} + f_L[x][6]*A_{3,y} + f_L[x][7]*A_{4,y}) >> (BitDepth - 8)$$

亮度插值滤波器系数见表 130。

T/AI 109.8-2025

表130 仿射预测亮度插值滤波器系数

分像素点位置	亮度插值滤波器系数							
	$f_L[p][0]$	$f_L[p][1]$	$f_L[p][2]$	$f_L[p][3]$	$f_L[p][4]$	$f_L[p][5]$	$f_L[p][6]$	$f_L[p][7]$
1	0	1	−3	63	4	−2	1	0
2	−1	2	−5	62	8	−3	1	0
3	−1	3	−8	60	13	−4	1	0
4	−1	4	−10	58	17	−5	1	0
5	−1	4	−11	52	26	−8	3	−1
6	−1	3	−9	47	31	−10	4	−1
7	−1	4	−11	45	34	−10	4	−1
8	−1	4	−11	40	40	−11	4	−1
9	−1	4	−10	34	45	−11	4	−1
10	−1	4	−10	31	47	−9	3	−1
11	−1	3	−8	26	52	−11	4	−1
12	0	1	−5	17	58	−10	4	−1
13	0	1	−4	13	60	−8	3	−1
14	0	1	−3	8	62	−5	2	−1
15	0	1	−2	4	63	−3	1	0

8.9.2.7 色度样本的插值过程

8.9.2.7.1 概述

如果当前预测单元的 AffineFlag 的值为 0，色度样本的插值过程见 8.9.2.7.2；否则，色度样本的插值过程见 8.9.2.7.3。

8.9.2.7.2 普通色度样本的插值过程

参考图像色度样本矩阵中的样本位置见图 36，A，B，C，D 是相邻初始整像素样本，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 fx & 7，dy 等于 fy & 7，其中(fx,fy)是该分像素样本在 1/8 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 37。

如果 IsPrivacyFlag 的值等于 1，整数样本值等于初始整数样本值。

否则，IsPrivacyFlag 的值等于 0，获取初始整数样本对应参考帧权限储存单元的权限值，若初始整数样本对应权限储存单元的权限值大于 0，则整数样本值等于 1<<(BitDepth-1)。

T/AI 109.8-2025

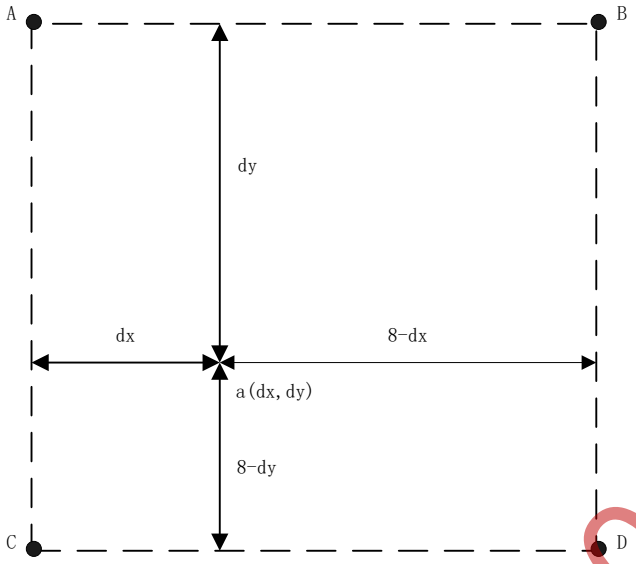


图36 普通预测色度样本位置

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

图37 普通预测整像素样本和分像素样本的位置

色度滤波系数见表 131。

T/AI 109.8-2025

表131 普通预测色度滤波系数

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：

```
if (dx == 0)
    a_{x,y}(0,dy) = Clip3(0, (1<<BitDepth)-1, (C[dy][0]*A_{x,y-1}+C[dy][1]*A_{x,y}+C[dy][2]*A_{x,y+1}+C[dy][3]*A_{x,y+2}+32) >> 6)
else if (dy == 0)
    a_{x,y}(dx,0) = Clip3(0, (1<<BitDepth)-1, (C[dx][0]*A_{x-1,y}+C[dx][1]*A_{x,y}+C[dx][2]*A_{x+1,y}+C[dx][3]*A_{x+2,y}+32) >> 6)
else
    a_{x,y}(dx,dy) = Clip3(0, (1<<BitDepth)-1, (C[dy][0]*a'_{x,y-1}(dx,0) + C[dy][1]*a'_{x,y}(dx,0) + C[dy][2]*a'_{x,y+1}(dx,0) + C[dy][3]*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，规定为：

```
a'_{x,y}(dx,0) = (C[dx][0]*A_{x-1,y} + C[dx][1]*A_{x,y} + C[dx][2]*A_{x+1,y} + C[dx][3]*A_{x+2,y}) >> (BitDepth - 8)
```

8.9.2.7.3 仿射色度样本的插值过程

参考图像色度样本矩阵中的样本位置见图 38，A，B，C，D 是相邻初始整像素样本，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 fx & 31，dy 等于 fy & 31，其中(fx, fy)是该分像素样本在 1/32 精度的色度样本矩阵中的坐标。整像素 $A_{x,y}$和周边的 1023 个分像素样本 $a_{x,y}(dx,dy)$的具体位置见图 39。

如果 IsPrivacyFlag 的值等于 1，整数样本值等于初始整数样本值。

否则，IsPrivacyFlag 的值等于 0，获取初始整数样本对应参考帧权限储存单元的权限值，若初始整数样本对应权限储存单元的权限值大于 0，则整数样本值等于 1<<(BitDepth-1)。

T/AI 109.8-2025

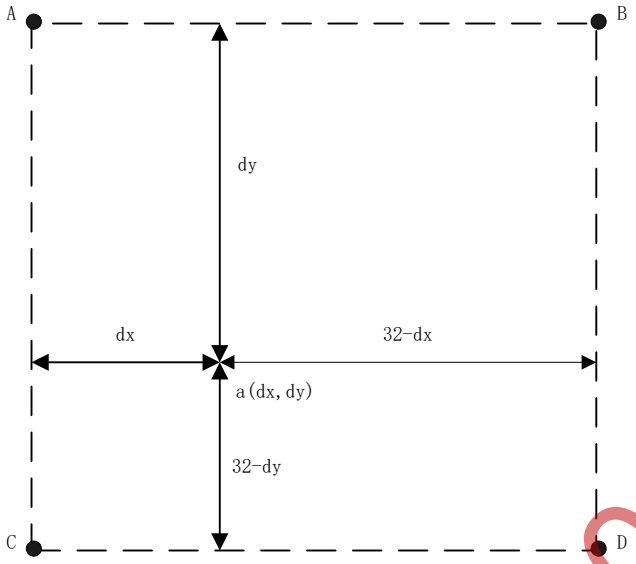


图38 仿射预测色度样本的位置

$A_{x,y}$	$a_{x,y}(1,0)$	$a_{x,y}(2,0)$	$a_{x,y}(3,0)$	$a_{x,y}(4,0)$	$a_{x,y}(5,0)$	…	$a_{x,y}(31,0)$
$a_{x,y}(0,1)$	$a_{x,y}(1,1)$	$a_{x,y}(2,1)$	$a_{x,y}(3,1)$	$a_{x,y}(4,1)$	$a_{x,y}(5,1)$	…	$a_{x,y}(31,1)$
$a_{x,y}(0,2)$	$a_{x,y}(1,2)$	$a_{x,y}(2,2)$	$a_{x,y}(3,2)$	$a_{x,y}(4,2)$	$a_{x,y}(5,2)$	…	$a_{x,y}(31,2)$
$a_{x,y}(0,3)$	$a_{x,y}(1,3)$	$a_{x,y}(2,3)$	$a_{x,y}(3,3)$	$a_{x,y}(4,3)$	$a_{x,y}(5,3)$	…	$a_{x,y}(31,3)$
$a_{x,y}(0,4)$	$a_{x,y}(1,4)$	$a_{x,y}(2,4)$	$a_{x,y}(3,4)$	$a_{x,y}(4,4)$	$a_{x,y}(5,4)$	…	$a_{x,y}(31,4)$
$a_{x,y}(0,5)$	$a_{x,y}(1,5)$	$a_{x,y}(2,5)$	$a_{x,y}(3,5)$	$a_{x,y}(4,5)$	$a_{x,y}(5,5)$	…	$a_{x,y}(31,5)$
…	…	…	…	…	…	…	…
$a_{x,y}(0,31)$	$a_{x,y}(1,31)$	$a_{x,y}(2,31)$	$a_{x,y}(3,31)$	$a_{x,y}(4,31)$	$a_{x,y}(5,31)$	…	$a_{x,y}(31,31)$

图39 仿射预测整像素样本和分像素样本的位置

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：

T/AI 109.8-2025

if (dx == 0)

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

else if (dy == 0)

$a_{x,y}(dx,0) = Clip3(0, (1<<BitDepth)-1, (f_C[dx][0]*A_{x-1,y}+f_C[dx][1]*A_{x,y}+f_C[dx][2]*A_{x+1,y}+f_C[dx][3]*A_{x+2,y}+32) >> 6)$

else

$a_{x,y}(dx,dy) = Clip3(0, (1<<BitDepth)-1, (C[dy][0]*a'_{x,y-1}(dx,0) + C[dy][1]*a'_{x,y}(dx,0) + C[dy][2]*a'_{x,y+1}(dx,0) + C[dy][3]*a'_{x,y+2}(dx,0) + (1 << (19-BitDepth))) >> (20-BitDepth))$

其中，$a'_{x,y}(dx,0)$是整像素行上的分像素的临时值，规定为：

$a'_{x,y}(dx,0) = (f_C[dx][0]*A_{x-1,y} + f_C[dx][1]*A_{x,y} + f_C[dx][2]*A_{x+1,y} + f_C[dx][3]*A_{x+2,y}) >> (BitDepth-8)$

色度插值滤波器系数见表 132。

T/AI 109.8-2025

表132 仿射预测色度插值滤波器系数

分像素位置	插值滤波器系数			
	$f_C[p][0]$	$f_C[p][1]$	$f_C[p][2]$	$f_C[p][3]$
1	−1	63	2	0
2	−2	62	4	0
3	−2	60	7	−1
4	−2	58	10	−2
5	−3	57	12	−2
6	−4	56	14	−2
7	−4	55	15	−2
8	−4	54	16	−2
9	−5	53	18	−2
10	−6	52	20	−2
11	−6	49	24	−3
12	−6	46	28	−4
13	−5	44	29	−4
14	−4	42	30	−4
15	−4	39	33	−4
16	−4	36	36	−4
17	−4	33	39	−4
18	−4	30	42	−4
19	−4	29	44	−5
20	−4	28	46	−6
21	−3	24	49	−6
22	−2	20	52	−6
23	−2	18	53	−5
24	−2	16	54	−4
25	−2	15	55	−4
26	−2	14	56	−4
27	−2	12	57	−3
28	−2	10	58	−2
29	−1	7	60	−2
30	0	4	62	−2
31	0	2	63	−1

T/AI 109.8-2025

8.9.3 预测样本合成

8.9.3.1 概述

本条的输入是运动信息

motionInfo(mvE0,mvE1,refIndexL0,refIndexL1,interPredRefMode,interPredRefMode)，当前编码单元的宽度 W 和高度 H，预测样本矩阵 predMatrixL0 和 predMatrixL1。

本条的输出是合成后的预测样本矩阵 PredMatrixComb。

第 1 种情况，当前编码单元只包含色度编码块，由运动信息中的预测参考模式 interPredRefMode、W 和 H 按 8.9.3.3 得到合成后的预测样本矩阵 PredMatrixComb。

第 2 种情况，当前编码单元的编码单元子类型是‘P_Skip_Affine’、‘P_Direct_Affine’、‘P_Inter_Affine’、‘B_Skip_Affine’、‘B_Direct_Affine’或‘B_Inter_Affine’，由 interPredRefMode、predMatrixL0、predMatrixL1、W 和 H 按 8.9.3.3 得到 PredMatrixComb。

第 3 种情况，当前编码单元的编码单元子类型是‘P_Skip_Mvap’、‘P_Direct_Mvap’、‘B_Skip_Mvap’、‘B_Direct_Mvap’、‘P_Skip_Etmvp’、‘P_Direct_Etmvp’、‘B_Skip_Etmvp’或‘B_Direct_Etmvp’，按以下方法得到 PredMatrixComb。

a) 将当前预测单元划分为宽度 subW 和高度 subH 均等于 8 的子块。令 i 和 j 分别是子块在当前预测单元内的水平索引值和垂直索引值，i 的取值范围是 0～(W>>3)-1，j 的取值范围是 0～(H>>3)-1，MotionArray[i][j]是当前子块的运动信息，子块左上角坐标(x,y)等于(xE+8*i,yE+8*j)；

b) 如果当前子块的 interPredRefMode 等于‘PRED_List01’：

1) 如果 BioEnableFlag 的值等于 1，且当前子块的参考图像队列 0 中参考帧索引值为 refIndexL0 的参考帧和参考图像队列 1 中参考索引值为 refIndexL1 的参考帧显示顺序分别位于当前图像的两侧，且当前预测块的 AmvrIndex 的值等于 0，且当前预测块的 InterPcFlag 的值为 0，则 BioFlag 的值为 1，根据 predMatrixL0、predMatrixL1、gradXL0、gradXL1、gradYL0、gradYL1、subW 和 subH 按 8.9.3.2 得到 predMatrixCombSub；

2) 否则，BioFlag 的值为 0，并由 interPredRefMode、predMatrixL0、predMatrixL1、subW 和 subH 按 8.9.3.3 得到 predMatrixCombSub。

c) 否则，如果当前子块的 interPredRefMode 等于‘PRED_List0’或‘PRED_List1’，由 interPredRefMode、predMatrixL0、predMatrixL1、subW 和 subH 按 8.9.3.3 得到 predMatrixSingleSub。

1) 如果 ObmcEnableFlag 等于 1，且当前图像类型不为 P 图像，且当前编码单元包含不小于 64 个亮度样本，且 i 或 j 等于 0，则令 wFlag 等于(W>=16?1:0)，hFlag 等于(H>=16?1:0)，由当前子块左上角位置(xE+8*i,yE+8*j)，当前子块宽度 subW 和高度 subH，当前子块水平索引 i 和竖直索引 j，标志位 wFlag 和 hFlag，以及当前子块 predMatrixSingleSub 按 8.9.3.5 得到 predMatrixCombSub；

2) 否则，令 predMatrxiCombSub 等于 predMatrixSignleSub。

d) PredMatrixComb 由各个子块的 predMatrixCombSub 组成。

第 4 种情况，当前编码单元的编码单元子类型是‘P_Skip_SbTemporal’、‘P_Direct_SbTemporal’、‘B_Skip_SbTemporal’或‘B_Direct_SbTemporal’，则将当前预测单元划分为 4 个子块（见图 26，图中的数字是子块的索引值），按以下方法得到 PredMatrixComb。

a) 令 MotionArray[i]是索引值为 i 的子块的运动信息，子块的宽度 subW 和高度 subH 分别等于 W/2 和 H/2，子块左上角坐标(x,y)等于(xE+(W/2*(i%2)),yE+((H/2)*(i/2)))；

b) 如果当前子块的 interPredRefMode 等于‘PRED_List01’，且 BioEnableFlag 的值等于 1，且当前子块的参考图像队列 0 中参考帧索引值为 refIndexL0 的参考帧和参考图像队列 1 中参考索引值为 refIndexL1 的参考帧显示顺序分别位于当前图像的两侧，且当前预测块的 AmvrIndex

T/AI 109.8-2025

的值等于 0，且当前预测块的 InterPcFlag 的值为 0，则 BioFlag 的值为 1；否则，BioFlag 的值为 0；

c) 如果 BioFlag 的值等于 1，由 predMatrixL0、predMatrixL1、gradXL0、gradXL1、gradYL0、gradYL1、subW 和 subH 按 8.9.3.2 得到 predMatrixComb；

d) 否则，由 interPredRefMode、predMatrixL0、predMatrixL1、subW 和 subH 按 8.9.3.3 得到 predMatrixCombSub；

e) PredMatrixComb 由各个子块的 predMatrixCombSub 组成。

第 5 种情况，当前编码单元的编码单元子类型是‘B_Skip_Awp’、‘B_Direct_Awp’、‘P_Skip_Awp’或‘P_Direct_Awp’，按 8.9.3.4 得到 predMatrixAwp，作为 PredMatrixComb。

如果以上 5 种情况都不满足，按以下方法得到 PredMatrixComb。

a) 如果当前预测单元的 interPredRefMode 等于‘PRED_List01’：

1) 如果 BioEnableFlag 的值等于 1，且当前预测单元的参考图像队列 0 中参考帧索引值为 refIndexL0 的参考帧和参考图像队列 1 中参考索引值为 refIndexL1 的参考帧显示顺序分别位于当前图像的两侧，且当前预测块的 AmvrIndex 的值等于 0，且当前预测块的 InterPcFlag 的值为 0，则 BioFlag 的值为 1，由 predMatrixL0、predMatrixL1、gradXL0、gradXL1、gradYL0、gradYL1、W 和 H 按 8.9.3.2 得到 predMatrixCombSub；

2) 否则，BioFlag 的值为 0，由 interPredRefMode、predMatrixL0、predMatrixL1、W 和 H 按 8.9.3.3 得到 predMatrixCombSub。

b) 否则，如果当前预测单元的 InterPredRefMode 等于‘PRED_List0’或‘PRED_List1’，由 interPredRefMode、predMatrixL0、predMatrixL1、W 和 H 按 8.9.3.3 得到 predMatrixSingleSub；

1) 如果 ObmcEnableFlag 等于 1，且当前图像类型不为 P 图像，且当前编码单元包含不小于 64 个亮度样本，则令 wFlag 等于(W>=16?1:0)，hFlag 等于(H>=16?1:0)，由当前编码单元左上角位置(xE,yE)，编码单元宽度 W 和高度 H，水平方向索引 0 和垂直方向索引 0，标志位 wFlag 和 hFlag，以及当前预测单元 predMatrixSingleSub 按 8.9.3.5 得到 predMatrixCombSub；

2) 否则，令 predMatrxiCombSub 等于 predMatrixSignleSub。

c) PredMatrixComb 等于 predMatrixCombSub。

8.9.3.2 导出双向光流预测样本改善值

本条的输入是predMatrixL0、predMatrixL1、gradXL0、gradXL1、gradYL0和gradYL1，以及高度W和宽度H。

本条的输出是合成后的预测样本矩阵predMatrixComb。

第1步，初始化W×H矩阵s1、s2、s3、s4和s5。

```
for (x=0; x<W; x++) {
    for (y=0; y<H; y++) {
        t = predMatrixL0[x][y] - predMatrixL1[x][y]
        tx = gradXL0[x][y] + gradXL1[x][y]
        ty = gradYL0[x][y] + gradYL1[x][y]
        s1[x][y] = tx * tx
        s2[x][y] = tx * ty
        s3[x][y] = -tx * t
        s4[x][y] = ty * ty
```

T/AI 109.8-2025

```
            s5[x][y] = -ty * t
        }
    }
```

第2步，令(tx,ty)是4×4子块的左上角坐标，(i,j)是4×4子块中样本点位置坐标，tx的取值范围是0～(W>>2)，ty的取值范围是0～(H>>2)。按以下方法依次处理当前块的各个4×4子块，得到子块的predMatrixComb。

a) 计算 t1，t2，t3，t4 和 t5；

```
t1 = 0
t2 = 0
t3 = 0
t4 = 0
t5 = 0
for (i=tx*4; i<tx*4+4; i++) {
    for (j=ty*4; j<ty*4+4; j++) {
        t1 += s1[i][j]
        t2 += s2[i][j]
        t3 += s3[i][j]
        t4 += s4[i][j]
        t5 += s5[i][j]
    }
}
```

b) 令 LIMITBIO 等于 768，DENOMBIO 等于 2，计算 vx 和 vy；

```
if (t1 > DENOMBIO)
    vx = Clip3(-LIMITBIO, LIMITBIO, (t3 << 5) >> Floor(Log(t1 + DENOMBIO))
else
    vx = 0
if (t4 > DENOMBIO)
    vy = Clip3(-LIMITBIO, LIMITBIO, ((t5 << 6) - vx * t2) >> Floor(Log((t4 + DENOMBIO ) << 1))
else
    vy = 0
```

c) 计算子块样本位置的预测样本改善值b；

```
for (i=tx*4; i<tx*4+4; i++) {
    for (j=ty*4; j<ty*4+4; j++) {
        b[i][j] = vx * (gradXL0[i][j] - gradXL1[i][j]) + vy * (gradYL0[i][j] - gradYL1[i][j])
        b[i][j] = (b[i][j] > 0) ? ((b[i][j] + 32) >> 6) : (-((-b[i][j] + 32) >> 6))
    }
}
```

d) 得到predMatrixComb。

```
for (i=tx*4; i<tx*4+4; i++) {
    for (j=ty*4; j<ty*4+4; j++)
        predMatrixComb[i][j] = Clip1((predMatrixL0[i][j] + predMatrixL1[i][j] + b[i][j]+1) >> 1)
```

T/AI 109.8-2025

```
}
```

第3步，当前块的predMatrixComb由4×4子块的predMatrixComb组成。

8.9.3.3 合成普通预测样本

本条的输入是预测参考模式 interPredRefMode，predMatrixL0 和 predMatrixL1，以及宽度 W 和高度 H。

本条的输出是合成后的预测样本矩阵 predMatrixComb。

如果 interPredRefMode 等于‘PRED_List0’：

```
for (x=0; x<W; x++) {
    for (y=0; y<H; y++)
        predMatrixComb[x][y] = predMatrixL0[x][y]
}
```

如果 interPredRefMode 等于‘PRED_List1’：

```
for (x=0; x<W; x++) {
    for (y=0; y<H; y++)
        predMatrixComb[x][y] = predMatrixL1[x][y]
}
```

如果 interPredRefMode 等于‘PRED_List01’：

```
for (x=0; x<W; x++) {
    for (y=0; y<H; y++)
        predMatrixComb[x][y] = (predMatrixL0[x][y] + predMatrixL1[x][y] + 1) >> 1
}
```

8.9.3.4 导出角度加权预测样本

8.9.3.4.1 概述

本条的输入是角度加权预测模式索引值 AwpIndex、PredMatrixAwp0、PredMatrixAwp1，以及宽度 W 和高度 H。

本条的输出是角度加权预测样本矩阵 predMatrixAwp。

先按 8.9.3.4.2 得到角度加权预测权重矩阵，再按 8.9.3.4.3 得到角度加权预测样本矩阵。

8.9.3.4.2 导出角度加权预测权重矩阵

本条的输入是角度加权预测模式索引值AwpIndex，以及宽度W和高度H。

本条的输出是角度加权预测的亮度权重矩阵AwpWeightMatrixY和色度权重矩阵AwpWeightMatrixUV。

第1步，计算变量stepIndex、angleIndex和angleAreaIndex。

```
stepIndex = (AwpIndex >> 3) - 3
modAngNum = AwpIndex % 8
if (modAngNum == 2)
    angleIndex = 7
else if (modAngNum == 6)
    angleIndex = 8
else
```

T/AI 109.8-2025

```
    angleIndex = modAngNum % 2
angleAreaIndex = modAngNum >> 1
```

第2步，如果当前图像是P图像，令PictureAwpRefineIndex等于1。先根据angleAreaIndex查表133得到vL，再根据angleAreaIndex和PictureAwpRefineIndex查表134得到shift和fP，如果当前图像是P图像，令shift等于3。最后得到参考权重列表ReferenceWeights。

```
for (x=0; x<vL; x++)
    ReferenceWeights[x] = Clip3(0, 8, (x  -  fP) << shift)
```

表133 angleAreaIndex 和 vL 的对应关系

angleAreaIndex	vL
0	(H + (W >> angleIndex)) << 1
1	
2	(W + (H >> angleIndex)) << 1
3	

表134 angleAreaIndex，PictureAwpRefineIndex 和 shift，fP 的对应关系

angleAreaIndex	PictureAwpRefineIndex	shift	fP
0	0	0	(vL >> 1) - 6 + stepIndex * ((vL >> 3) - 1)
	1	2	(vL >> 1) - 3 + stepIndex * ((vL >> 3) - 1)
1	0	0	(vL >> 1) - 4 + stepIndex * ((vL >> 3) -1) - ((W << 1) >> angleIndex)
	1	2	(vL >> 1) - 1 + stepIndex * ((vL >> 3) -1) - ((W << 1) >> angleIndex)
2	0	0	(vL >> 1) - 4 + stepIndex * ((vL >> 3) -1) - ((H << 1) >> angleIndex)
	1	2	(vL >> 1) - 1 + stepIndex * ((vL >> 3) -1) - ((H << 1) >> angleIndex)
3	0	0	(vL >> 1) - 6 + stepIndex * ((vL >> 3) - 1)
	1	2	(vL >> 1) - 3 + stepIndex * ((vL >> 3) - 1)

第3步，先根据当前图像类型和angleAreaIndex查表135得到tP，再得到亮度权重矩阵AwpWeightMatrixY。

```
for (x=0; x<W; x++) {
    for (y=0; y<H; y++)
        AwpWeightMatrixY[x][y] = ReferenceWeights[tP]
}
```

T/AI 109.8-2025

表135 angleAreaIndex 和 tP 的对应关系

图像类型	angleAreaIndex	tP
B 图像	0	(y << 1) + ((x << 1) >> angleIndex)
	1	(y << 1) - ((x << 1) >> angleIndex)
	2	(x << 1) - ((y << 1) >> angleIndex)
	3	(x << 1) + ((y << 1) >> angleIndex)
P 图像	0	((y >> 2) << 3) + 4 + ((((x >> 2) << 3) + 4) >> angleIndex)
	1	((y >> 2) << 3) + 4 - ((((x >> 2) << 3) + 4) >> angleIndex)
	2	((x >> 2) << 3) + 4 - ((((y >> 2) << 3) + 4) >> angleIndex)
	3	((x >> 2) << 3) + 4 + ((((y >> 2) << 3) + 4) >> angleIndex)

第4步，得到色度权重矩阵AwpWeightMatrixUV。

如果当前图像是 P 图像，则：

```
for (x=0; x<W/2; x++) {
    for (y=0; y<H/2; y++)
        AwpWeightMatrixUV[x][y] = AwpWeightMatrixY[(x>>2)<<3][(y>>2)<<3]
}
```

否则：

```
for (x=0; x<W/2; x++) {
    for (y=0; y<H/2; y++)
        AwpWeightMatrixUV[x][y] = AwpWeightMatrixY[x<<1][y<<1]
}
```

8.9.3.4.3 导出角度加权预测样本

本条的输入是角度加权预测亮度权重矩阵 AwpWeightMatrixY 和色度权重矩阵 AwpWeightMatrixUV，当前预测块的宽度 W 和高度 H。

本条的输出是加权预测样本矩阵 predMatrixAwp。

如果当前预测块是亮度预测块：

```
for (x=0; x<W; x++)
    for (y=0; y<H; y++)
        predMatrixAwp[x][y] = (PredMatrixAwp0[x][y] * AwpWeightMatrixY[x][y] + PredMatrixAwp1[x][y] * (8 - AwpWeightMatrixY[x][y] ) + 4) >> 3
}
```

如果当前预测块是色度预测块：

```
for (x=0; x<W/2; x++)
```

T/AI 109.8-2025

```
    for (y=0; y<H/2; y++)
        predMatrixAwp[x][y] = (PredMatrixAwp0[x][y] * AwpWeightMatrixUV[x][y] + PredMatrixAwp1[x][y] * (8 -
AwpWeightMatrixUV[x][y] ) + 4) >> 3
}
```

8.9.3.5 合成重叠块预测样本

8.9.3.5.1 概述

本条的输入是当前编码单元或子块左上角样本的位置(xSb,ySb)，宽度 subW 和高度 subH，水平索引 i 和竖直索引 j，标志位 wFlag 和 hFlag，预测样本矩阵 PredMatrixSingleSub，当前编码单元上侧相邻块单向运动信息矩阵 MotionInfoTop 和左侧相邻块单向运动信息矩阵 MotionInfoLeft。

本条的输出是调整后的当前子块的预测样本矩阵 PredMatrixCombSub。

8.9.3.5.2 重叠块运动补偿过程

本条的输入是当前编码单元左上角样本的位置(xCb,yCb)，当前子块左上角样本位置（xSb，ySb)，当前编码单元的宽度 W 和高度 H，当前子块宽度 subW 和高度 subH，当前子块水平索引 i 和竖直索引 j，当前子块的预测样本矩阵 PredMatrixSingleSub，当前编码单元上侧相邻块单向运动信息矩阵 MotionInfoTop 和左侧相邻块单向运动信息矩阵 MotionInfoLeft。

本条的输出是调整后的当前子块的预测样本矩阵 PredMatrixCombSub。

第 1 步，更新当前编码单元上边缘像素。

a) 步骤 1.1，如果 hFlag 等于 1，令色度加权标识 chromaBlending 为 1；否则，令 chromaBlending 为 0。将当前子块划分为宽度 subWidth 为 4，高度 subHeight 为(subH>>1)的加权块。令水平索引 xSbIndex 从 0～(subW>>2)-1，竖直索引 ySbIndex 从 0～1，设当前加权块位置(xCurr,yCurr)为(xSb+(xSbIndex<<2),ySb+ySbIndex*subHeight)，其相邻运动信息 topMotionInfo 为 MotionInfoTop[(subW>>2)*i+xSbIndex]中的运动信息，加权方向标识 blendingDir 为 0，令 xSbIndex 从 0～(subW>>2)-1，循环执行步骤 1.2；

b) 步骤 1.2，如果 j 等于 0，对于竖直索引 ySbIndex 等于 0 的加权块依次执行以下操作，导出加权块预测样本矩阵 PredMatrixobmcSub；否则（j 不等于 0，或竖直索引 ySbIndex 等于 1 的加权块），令预测样本矩阵 PredMatrixobmcSub 等于 PredMatrixSingleSub 中与当前加权块对应的部分；

   1) 如果 topMotionInfo 中的预测参考模式为‘Pred_List0’或‘Pred_List1’，由运动信息 topMotionInfo、(xCurr,yCurr)、subWidth 和 subHeight 按 8.9.2.2 得到预测样本 predAboveOBMCL0 和 predAboveOBMCL1；

   2) 如果 topMotionInfo 中的预测参考模式为‘Pred_List0’，由 subWidth、subHeight、predAboveOBMCL0、blendingDir、chromaBlending 和预测样本矩阵 PredMatrixSingleSub 中与当前加权块对应的部分按 8.9.3.5.3 导出预测样本矩阵 PredMatrixCombSub；

   3) 如果 topMotionInfo 中的预测参考模式为‘Pred_List1’，由 subWidth、subHeight、predAboveOBMCL1、blendingDir、chromaBlending 和预测样本矩阵 PredMatrixSingleSub 中与当前加权块对应的部分按 8.9.3.5.3 导出预测样本矩阵 PredMatrixCombSub。

c) 步骤 1.3，当前块的预测样本矩阵 PredMatrixObmcSub 由其中各个加权块的预测样本矩阵 PredMatrixobmcSub 组成。

第 2 步，更新当前编码单元左边缘像素。

T/AI 109.8-2025

a) 步骤 2.1，如果 wFlag 等于 1，令色度加权标识 chromaBlending 为 1；否则，令 chromaBlending 置为 0。将当前子块划分为宽度 subSWidth 为(subW>>1)，高度 subHeight 为 4 的加权块。令加权块水平索引 xSbIndex 从 0～1，ySbIndex 从 0～(subH>>2)-1，设当前子块位置(xCurr,yCurr)为(xCb+subH/2,yCb+(ySbIndex<<2))，运动信息 leftMotoinInfo 为 MotionInfoLeft[(subH>>2]*j+ySbIndex]中的运动信息，加权方向标识 blendingDir 为 1，令 ySbIndex 从 0～(subH>>2)-1，循环执行步骤 2.2；

b) 步骤 2.2，如果 i 等于 0，对于水平索引 xSbIndex 等于 0 的加权块依次执行以下操作，导出加权块预测样本矩阵 PredMatrixobmcSub；否则（i 不等于 0，或水平索引 xSbIndex 等于 1 的加权块），令预测样本矩阵 PredMatrixCombSub 等于 PredMatrixObmcSub 中与当加权块对应的部分；

1) 如果 leftMotoinInfo 中的预测参考模式为‘Pred_List0’或‘Pred_List1’，由运动信息 leftMotionInfo、(xCurr,yCurr)、subWidth 和 subHeight 按 8.9.2.2 得到预测样本 predLeftOBMCL0 和 predLeftOBMCL1；

2) 如果 leftMotionInfo 中的预测参考模式为‘Pred_List0’，由 subWidth、subHeight、predLeftOBMCL0、blendingDir、chromaBlending 和预测样本矩阵 PredMatrixObmcSub 中与当前加权块对应的部分按 8.9.3.5.3 导出预测样本矩阵 PredMatrixCombSub；

3) 如果 leftMotionInfo 中的预测参考模式为‘Pred_List1’，由 subWidth、subHeight、predLeftOBMCL1、blendingDir、chromaBlending 和预测样本矩阵 PredMatrixObmcSub 中与当前加权块对应的部分按 8.9.3.5.3 导出预测样本矩阵 PredMatrixCombSub。

c) 步骤 2.3，当前块的预测样本矩阵 PredMatrixComSub 由其中各个加权块的预测样本矩阵 PredMatrixcomSub 组成。

8.9.3.5.3 重叠块运动补偿值加权过程

本条的输入是加权块宽度 subWidth 和高度 subHeight，当前加权块加权预测样本矩阵 predObmc，加权方向标识 blendingDir 和色度加权标识 chromaBlending，以及当前加权块合成前预测样本矩阵 predMatrixIn。

本条的输出是合成后预测样本矩阵 PredMatrixOut。

如果 CodingTreeComponent 不等于‘COMPONENT_CHROMA’，predMatrixInLuma 是 predMatrixIn 中的亮度预测样本矩阵，predObmcLuma 是 predObmc 中的亮度预测样本矩阵，PredMatrixOut 是 PredMatrixOut 中的亮度预测样本矩阵。

如果 CodingTreeComponent 不等于‘COMPONENT_LUMA’，predMatrixInCb 和 predMatrixInCr 是 predMatrixIn 中的两个色度样本矩阵，predObmcCb 和 predObmcCr 是 predObmc 中的两个色度样本矩阵，PredMatrixOutCb 和 PredMatrixOutCr 是 PredMatrixOut 中的两个色度样本矩阵。

令 wL 和 wC 分别是亮度和色度样本使用的加权系数向量。

如果 blendingDir 等于 0，令 curSize 等于 subHeight；否则（blendingDir 等于 1），令 curSize 等于 subWidth。按以下步骤更新当前子块预测样本。

a) 如果 PictureObmcBlendingSetIndex 等于 0，设 wL 和 wC 分别为表 136 中 W[curSize]和 W[curSize>>1]向量中加权系数；否则，设 wL 和 wC 分别为表 137 中 W[curSize]和 W[curSize>>1]向量中加权系数；

b) 如果 CodingTreeComponent 不等于‘COMPONENT_CHROMA’，按以下方法对当前子块亮度预测样本进行加权调整；

```
for (x=0; x<subWidth; x++) {
```

T/AI 109.8-2025

```
    for (y=0; y<subHeight; y++) {
        if (blendingDir == 0)
            PredMatrixOutLuma[x][ y] = Clip1((wL[y] * PredMatrixInLuma[x][y] + (64 - wL[y]) *
predObmcLuma[x][y] + 32 ) >> 6)
        else
            PredMatrixOutLuma[x][ y] = Clip1((wL[x] * PredMatrixInLuma[x][y] + (64 - wL[x]) *
predObmcLuma[x][y] + 32 ) >> 6)
    }
}
```

c) 如果 CodingTreeComponent 不等于‘COMPONENT_LUMA’且 chromaBlending 等于 1，按以下方法对当前子块色度预测样本进行加权调整。

```
for (x=0; x<subWidth>>1; x++) {
    for (y=0; y<subHeight>>1; y++) {
        if (blendingDir == 0) {
            PredMatrixOutCb[(xSb>>1)+x][(ySb>>1)+y] = Clip1((wC[y] * PredMatrixInCb[x][y] + (64 -
wC[y]) * predObmcCb[x][y] + 32 ) >> 6)
            PredMatrixOutCr[(xSb>>1)+x][ (ySb>>1)+y] = Clip1((wC[y] * PredMatrixInCr[x][y] + (64 -
wC[y]) * predObmcCr[x][y] + 32 ) >> 6)
        } else {
            PredMatrixOutCb[(xSb>>1)+x][(ySb>>1)+y] = Clip1((wC[x] * PredMatrixInCb[x][y] + (64 -
wC[x]) * predObmcCb[x][y] + 32 ) >> 6)
            PredMatrixOutCr[(xSb>>1)+x][ (ySb>>1)+y] = Clip1((wC[x] * PredMatrixInCr[x][y] + (64 -
wC[x]) * predObmcCr[x][y] + 32 ) >> 6)
        }
    }
}
```

表136 重叠块运动补偿加权系数表一

Index	W[64]	W[32]	W[16]	W[8]	W[4]	W[2]
0	56	56	56	57	58	59
1	56	57	57	58	60	63
2	56	57	58	60	63	-
3	57	57	59	61	64	-
4	57	58	59	62	-	-
5	57	58	60	63	-	-
6	57	59	61	64	-	-
7	57	59	61	64	-	-
8	58	59	62	-	-	-
9	58	60	62	-	-	-
10	58	60	63	-	-	-
11	58	60	63	-	-	-

T/AI 109.8-2025

表 136 重叠块运动补偿加权系数表一（续）

Index	W[64]	W[32]	W[16]	W[8]	W[4]	W[2]
12	58	61	64	–	–	–
13	59	61	64	–	–	–
14	59	61	64	–	–	–
15	59	62	64	–	–	–
16	59	62	–	–	–	–
17	59	62	–	–	–	–
18	60	62	–	–	–	–
19	60	63	–	–	–	–
20	60	63	–	–	–	–
21	60	63	–	–	–	–
22	60	63	–	–	–	–
23	60	63	–	–	–	–
24	61	63	–	–	–	–
25	61	64	–	–	–	–
26	61	64	–	–	–	–
27	61	64	–	–	–	–
28	61	64	–	–	–	–
29	61	64	–	–	–	–
30	61	64	–	–	–	–
31	62	64	–	–	–	–
32	62	–	–	–	–	–
33	62	–	–	–	–	–
34	62	–	–	–	–	–
35	62	–	–	–	–	–
36	62	–	–	–	–	–
37	62	–	–	–	–	–
38	62	–	–	–	–	–
39	63	–	–	–	–	–
40	63	–	–	–	–	–
41	63	–	–	–	–	–
42	63	–	–	–	–	–
43	63	–	–	–	–	–
44	63	–	–	–	–	–
45	63	–	–	–	–	–
46	63	–	–	–	–	–
47	63	–	–	–	–	–
48	63	–	–	–	–	–
49	63	–	–	–	–	–

T/AI 109.8-2025

表 136 重叠块运动补偿加权系数表一（续）

Index	W[64]	W[32]	W[16]	W[8]	W[4]	W[2]
50	64	–	–	–	–	–
51	64	–	–	–	–	–
52	64	–	–	–	–	–
53	64	–	–	–	–	–
54	64	–	–	–	–	–
55	64	–	–	–	–	–
56	64	–	–	–	–	–
57	64	–	–	–	–	–
58	64	–	–	–	–	–
59	64	–	–	–	–	–
60	64	–	–	–	–	–
61	64	–	–	–	–	–
62	64	–	–	–	–	–
63	64	–	–	–	–	–

表137 重叠块运动补偿加权系数表二

Index	W[64]	W[32]	W[16]	W[8]	W[4]	W[2]
0	40	41	41	42	45	49
1	41	42	44	47	53	62
2	41	43	46	51	60	–
3	42	44	48	55	64	–
4	43	45	50	59	–	–
5	43	46	52	61	–	–
6	44	48	54	63	–	–
7	44	49	56	64	–	–
8	45	50	58	–	–	–
9	46	51	59	–	–	–
10	46	52	61	–	–	–
11	47	53	62	–	–	–
12	47	54	63	–	–	–
13	48	55	63	–	–	–
14	48	56	64	–	–	–
15	49	57	64	–	–	–
16	49	57	–	–	–	–
17	50	58	–	–	–	–
18	51	59	–	–	–	–
19	51	60	–	–	–	–

T/AI 109.8-2025

表 137 重叠块运动补偿加权系数表二（续）

Index	W[64]	W[32]	W[16]	W[8]	W[4]	W[2]
20	52	60	–	–	–	–
21	52	61	–	–	–	–
22	53	61	–	–	–	–
23	53	62	–	–	–	–
24	54	62	–	–	–	–
25	54	63	–	–	–	–
26	55	63	–	–	–	–
27	55	63	–	–	–	–
28	55	64	–	–	–	–
29	56	64	–	–	–	–
30	56	64	–	–	–	–
31	57	64	–	–	–	–
32	57	–	–	–	–	–
33	58	–	–	–	–	–
34	58	–	–	–	–	–
35	58	–	–	–	–	–
36	59	–	–	–	–	–
37	59	–	–	–	–	–
38	59	–	–	–	–	–
39	60	–	–	–	–	–
40	60	–	–	–	–	–
41	60	–	–	–	–	–
42	61	–	–	–	–	–
43	61	–	–	–	–	–
44	61	–	–	–	–	–
45	62	–	–	–	–	–
46	62	–	–	–	–	–
47	62	–	–	–	–	–
48	62	–	–	–	–	–
49	62	–	–	–	–	–
50	63	–	–	–	–	–
51	63	–	–	–	–	–
52	63	–	–	–	–	–
53	63	–	–	–	–	–
54	63	–	–	–	–	–
55	63	–	–	–	–	–
56	64	–	–	–	–	–
57	64	–	–	–	–	–

T/AI 109.8-2025

表 137 重叠块运动补偿加权系数表二（续）

Index	W[64]	W[32]	W[16]	W[8]	W[4]	W[2]
58	64	-	-	-	-	-
59	64	-	-	-	-	-
60	64	-	-	-	-	-
61	64	-	-	-	-	-
62	64	-	-	-	-	-
63	64	-	-	-	-	-

8.9.4 预测样本修正

8.9.4.1 概述

本条的输入是当前编码单元的宽度 W 和高度 H，当前编码单元左上角样本的坐标(xE,yE)，预测样本矩阵 predMatrixL0、predMatrixL1 和合成后的预测样本矩阵 predMatrixComb。

本条的输出是帧间预测的预测样本矩阵 predMatrixInter。

如果 BgcFlag 的值等于 1，由 predMatrixL0、predMatrixL1、predMatrixComb 和 BgcIndex 按 8.9.4.2 得到 predMatrixBgc；否则，predMatrixBgc 等于 predMatrixComb。

如果 InterPfFlag 的值等于 1，由 predMatrixBgc 和 InterPfIndex 按 8.9.4.3 得到 predMatrixInterPf；否则，predMatrixInterPf 等于 predMatrixBgc。

如果 InterPcFlag 的值等于 1，由 predMatrixInterPf 和 InterPcIndex 按 8.9.4.4 得到 predMatrixInter；否则，predMatrixInter 等于 predMatrixInterPf。

8.9.4.2 导出双向梯度修正后的预测样本

本条的输入是预测样本矩阵predMatrixL0、predMatrixL1和predMatrixComb，以及宽度M和高度N。

本条的输出是修正后的预测样本矩阵predMatrixBgc。

第1步，导出修正样本改善值矩阵g。

a) 如果 BgcIndex 值为 1：

```
for (x=0; x<M; x++) {
    for (y=0; y<N; y++)
        g[x][y] = (predMatrixL0[x][y] - predMatrixL1[x][y]) >> 3
}
```

b) 如果 BgcIndex 值为 0：

```
for (x=0; x<M; x++) {
    for (y=0; y<N; y++)
        g[x][y] = (predMatrixL1[x][y] - predMatrixL0[x][y]) >> 3
}
```

第2步，导出修正后的预测样本矩阵predMatrixBgc。

```
for (x=0; x<M; x++) {
    for (y=0; y<N; y++)
        predMatrixBgc[x][y] = Clip1(PredMatrixComb[x][y] + g[x][y])
```

T/AI 109.8-2025

```
}
```

8.9.4.3 导出帧间预测滤波后的预测样本

本条的输入是修正后的预测样本矩阵predMatrixBgc，以及亮度块的宽度M和高度N，色度块的宽度K和高度L。

本条的输出是帧间预测的预测样本矩阵predMatrixInterPf。

第 1 步，分别按 8.8.1.2 和 8.8.1.3 得到当前块的亮度和色度参考样本。

a) 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]。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[j]，其中 row[0]等于 col[0]；

b) 如果当前块是亮度块：

```
for (i=0; i<2M; i++)
    topPel[i] = r[i]
for (j=0; j<2N; j++)
    leftPel[j] = c[j]
```

c) 如果当前块是色度块：

```
for (i=0; i<2K; i++)
    topPel[i] = row[i]
for (j=0; j<2L; j++)
    leftPel[j] = col[j]
```

第 2 步，如果 InterPfIndex 为 0，按以下方法得到预测样本矩阵 predMatrixTmp。其中，如果当前预测块是亮度块，x 的取值范围是 0～M-1，y 的取值范围是 0～N-1；如果当前预测块是色度块，x 的取值范围是 0～K-1，y 的取值范围是 0～L-1。

a) 得到预测样本矩阵 predPlane；

```
predV = ((N - 1 - y) * topPel[x+1] + (y + 1) * leftPel[N+1] + (N >> 1)) >> Log(N)
predH = ((M - 1 - x) * leftPel[y+1] + (x + 1) * topPel[M+1] + (M >> 1)) >> Log(M)
predPlane[x][y] = (predV + predH + 1) >> 1
```

b) 得到预测样本矩阵 predMatrixTmp。

```
predMatrixTmp[x][y] = ((predMatrixBgc[x][y] * 5 + predPlane[x][y] * 3 + 4) >> 3)
```

第 3 步，否则（InterPfIndex 为 1），按以下方法得到预测样本矩阵 predMatrixTmp。其中，如果当前预测块是亮度块，x 的取值范围是 0～M-1，y 的取值范围是 0～N-1；如果当前预测块是色度块，x 的取值范围是 0～K-1，y 的取值范围是 0～L-1。

```
predMatrixTmp[x][y] = Clip1((f[x] * leftPel[y+1] + f[y] * topPel[x+1] + (64 - f[x] - f[y]) * predMatrixBgc[x][y] +
        32 ) >> 6)
```

第 4 步，得到帧间预测的预测样本矩阵 predMatrixInterPf。其中，如果当前预测块是亮度块，x 的取值范围是 0～M-1，y 的取值范围是 0～N-1；如果当前预测块是色度块，x 的取值范围是 0～K-1，y 的取值范围是 0～L-1。

```
predMatrixInterPf[x][y] = predMatrixTmp[x][y]
```

上述过程中，M×N 亮度块内坐标为(x,y)的像素对应的滤波系数 f[x]和 f[y]分别由 M、x 和 N、y 查表 122 得到；K×L 色度块内坐标为(x,y)的像素对应的滤波系数 f[x]和 f[y]分别由 K、x 和 L、y 查表 122 得到。

T/AI 109.8-2025

8.9.4.4 导出帧间预测修正后的预测样本

本条的输入是预测样本矩阵predMatrixInterPf，以及亮度块的宽度M和高度N，色度块的宽度K和高度L。

本条的输出是帧间预测的预测样本矩阵predMatrixInter。

分别按 8.8.1.2 和 8.8.1.3 得到当前块的亮度和色度参考样本。

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 i=0～M，j=0～N。令 rp[i+1]等于亮度 predMatrixInterPf[i][0]（i=0～M-1），cp[j+1]等于亮度 predMatrixInterPf[0][j](j=0～N-1)。

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[j]，其中 i=0～K，j=0～L。令 rowp[i+1]等于色度 predMatrixInterPf[i][0]（i=0～K-1），colp[j+1]等于色度 predMatrixInterPf[0][j](j=0～L-1)。

亮度块的处理如下：

a) 如果满足以下条件之一，则：

```
for (x=0; x<M; x++) {
    for (y=0; y<N; y++)
        predMatrixInter[x][y] = 1 << (Bitdepth - 1)
}
```

1) r[i]（i=1～M）、c[j]（j=1～N）均“不可用”。
2) InterPcIndex 等于 1 且 r[i]（i=1～M）均“不可用”。
3) InterPcIndex 等于 2 且 c[j]（j=1～N）均“不可用”。

b) 否则，执行以下操作：

1) 如果当前编码单元子类型为‘P_Skip_SbTemporal’、‘P_Direct_SbTemporal’、‘P_Skip_Mvap’、‘P_Direct_Mvap’、‘B_Skip_SbTemporal’、‘B_Direct_SbTemporal’、‘B_Skip_Mvap’或‘B_Direct_Mvap’，则 P 等于 8；否则，P 等于 16；
2) 如果 M 大于 P，则 ML 等于 P；否则 ML 等于 M。如果 N 大于 P，则 NL 等于 P；否则 NL 等于 N；
3) 如果 InterPcIndex 等于 0 且 r[i]（i=1～M）中存在一个“可用”且 c[j]（j=1～N）中存在一个“可用”，则：

```
posA0 = 0
posL0 = 0
if (ML >= NL) {
    posA1 = ML - ML / NL
    posL1 = NL - 1
}
else {
    posA1 = ML - 1
    posL1 = NL - NL / ML
}
x[0] = rp[posA0+2]
y[0] = r[posA0+2]
x[1] = rp[Min(posA1+2, ML)]
```

T/AI 109.8-2025

```
y[1] = r[Min(posA1+2, ML)]
x[2] = cp[posL0+2]
y[2] = c[posL0+2]
x[3] = cp[Min(posL1+2, NL)]
y[3] = c[Min(posL1+2, NL)]
```

4) 否则，如果 InterPcIndex 等于 0 且 r[i]中存在一个“可用”且 c[j]均“不可用”（i=1～M，j=1～N），或 InterPcIndex 等于 1 且 r[i]中存在一个“可用”（i=1～M）：

```
posA0 = 0
posA1 = ML / 4
posA2 = 2 * (ML / 4)
posA3 = 3 * (ML / 4)
x[0] = rp[posA0+1]
y[0] = r[posA0+1]
x[1] = rp[posA1+1]
y[1] = r[posA1+1]
x[2] = rp[posA2+1]
y[2] = r[posA2+1]
x[3] = rp[posA3+1]
y[3] = r[posA3+1]
```

5) 否则，如果 InterPcIndex 等于 0 且 r[i]均“不可用”且 c[j]中存在一个“可用”（i=1～M，j=1～N），或 InterPcIndex 等于 2 且 c[j]中存在一个“可用”j=（1～N）：

```
posL0 = 0
posL1 = NL / 4
posL2 = 2 * (NL / 4)
posL3 = 3 * (NL / 4)
x[0] = cp[posL0+1]
y[0] = c[posL0+1]
x[1] = cp[posL1+1]
y[1] = c[posL1+1]
x[2] = cp[posL2+1]
y[2] = c[posL2+1]
x[3] = cp[posL3+1]
y[3] = c[posL3+1]
```

6) 令 minIndex[2]={0,2}，maxIndex[2]={1,3}，执行以下操作：

```
if (x[minIndex[0]] > x[minIndex[1]])
    Exchange(minIndex[0], minIndex[1])
if (x[maxIndex[0]] > x[maxIndex[1]])
    Exchange(maxIndex[0], maxIndex[1])
if (x[minIndex[0]] > x[maxIndex[1]]) {
    Exchange(minIndex[0], maxIndex[0])
    Exchange(minIndex[1], maxIndex[1])
}
```

T/AI 109.8-2025

```
if (x[minIndex[1]] > x[maxIndex[0]])
    Exchange(minIndex[1], maxIndex[0])
xMin = (x[minIndex[0]]+x[minIndex[1]]+1)>>1
yMin = (y[minIndex[0]]+y[minIndex[1]]+1)>>1
xMax = (x[maxIndex[0]]+x[maxIndex[1]]+1)>>1
yMax = (y[maxIndex[0]]+y[maxIndex[1]]+1)>>1
diffX = xMax - xMin
diffY = yMax - yMin
iShift = 16
```

7） 执行以下操作：

```
if (diffX > 64) {
    shift = (BitDepth > 8) ? BitDepth - 6 : 2
    add = 1 << (shift - 1)
    α = (diffY * TscpmTable[((diffX + add) >> shift) - 1] + add) >> shift
    β = yMin - ((α * xMin) >> iShift)
}
else if (diffX > 0) {
    α = (diffY * TscpmTable[diffX-1])
    β = yMin - ((α * xMin) >> iShift)
}
else {
    α = 0
    β = yMin
}
```

8） 导出修正后的预测值：

```
for (x=0; x<M; x++) {
    for (y=0; y<N; y++)
        predMatrixInter[x][y] = Clip1(((α * predMatrixPf[x][y]) >> iShift) + β)
}
```

色度块的处理如下：

a） 如果满足以下条件之一，则：

```
for (x=0; x<K; x++) {
    for (y=0; y<L; y++)
        predMatrixInter[x][y] = 1 << (Bitdepth - 1)
}
```

1） row[i]（i=1～K）、col[j]（j=1～L）均“不可用”；

2） InterPcIndex 等于 1 且 row[i]（i=1～K）均“不可用”；

3） InterPcIndex 等于 2 且 col[j]（j=1～L）均“不可用”。

b） 否则，执行以下操作：

1） 如果当前编码单元子类型为‘P_Skip_SbTemporal’、‘P_Direct_SbTemporal’、‘P_Skip_Mvap’、‘P_Direct_Mvap’、‘B_Skip_SbTemporal’、‘B_Direct_SbTemporal’、‘B_Skip_Mvap’或‘B_Direct_Mvap’，则 P 等于 4；否则，P 等于 8；

T/AI 109.8-2025

2) 如果 K 大于 P，则 KL 等于 P；否则 KL 等于 K。如果 L 大于 P，则 LL 等于 P；否则 LL 等于 L；

3) 如果 InterPcIndex 等于 0 且 row[i]（i=1～K）中存在一个“可用”且 col[j]（j=1～L）中存在一个“可用”：

```
posA0 = 0
posL0 = 0
if (LL >= LL) {
    posA1 = KL - KL / LL
    posL1 = LL - 1
}
else {
    posA1 = KL - 1
    posL1 = LL - LL / KL
}
x[0] = rowp[posA0+2]
y[0] = row[posA0+2]
x[1] = rowp[Min(posA1+2, KL)]
y[1] = row[Min(posA1+2, KL)]
x[2] = colp[posL0+2]
y[2] = col[posL0+2]
x[3] = colp[Min(posL1+2, LL)]
y[3] = col[Min(posL1+2, LL)]
```

4) 否则，如果 InterPcIndex 等于 0 且 row[i]中存在一个“可用”且 col[j]均“不可用”（i=1～K，j=1～L），或 InterPcIndex 等于 1 且 row[i]中存在一个“可用”（i=1～K）：

```
posA0 = 0
posA1 = KL / 4
posA2 = 2 * (KL / 4)
posA3 = 3 * (KL / 4)
x[0] = rowp[posA0+1]
y[0] = row[posA0+1]
x[1] = rowp[posA1+1]
y[1] = row[posA1+1]
x[2] = rowp[posA2+1]
y[2] = row[posA2+1]
x[3] = rowp[posA3+1]
y[3] = row[posA3+1]
```

5) 否则，如果 InterPcIndex 等于 0 且 row[i]均“不可用”且 col[j]中存在一个“可用”（i=1～K，j=1～L），或 InterPcIndex 等于 2 且 col[j]中存在一个“可用”（j=1～L）：

```
posL0 = 0
posL1 = LL / 4
posL2 = 2 * (LL / 4)
posL3 = 3 * (LL / 4)
```

T/AI 109.8-2025

```
x[0] = colp[posL0+1]
y[0] = col[posL0+1]
x[1] = colp[posL1+1]
y[1] = col[posL1+1]
x[2] = colp[posL2+1]
y[2] = col[posL2+1]
x[3] = colp[posL3+1]
y[3] = col[posL3+1]
```

6) 令 minIndex[2]={0,2}，maxIndex[2]={1,3}，执行以下操作：

```
if (x[minIndex[0]] > x[minIndex[1]])
    Exchange(minIndex[0], minIndex[1])
if (x[maxIndex[0]] > x[maxIndex[1]])
    Exchange(maxIndex[0], maxIndex[1])
if (x[minIndex[0]] > x[maxIndex[1]]) {
    Exchange(minIndex[0], maxIndex[0])
    Exchange(minIndex[1], maxIndex[1])
}
if (x[minIndex[1]] > x[maxIndex[0]])
    Exchange(minIndex[1], maxIndex[0])
xMin = (x[minIndex[0]]+x[minIndex[1]]+1)>>1
yMin = (y[minIndex[0]]+y[minIndex[1]]+1)>>1
xMax = (x[maxIndex[0]]+x[maxIndex[1]]+1)>>1
yMax = (y[maxIndex[0]]+y[maxIndex[1]]+1)>>1
diffX = Abs(xMax - xMin)
diffY = Abs(yMax - yMin)
if (diffX > 64) {
    shift = (BitDepth > 8) ? BitDepth – 6 : 2
    add = 1 << (shift – 1)
    diffX = TscpmTable[((diffX + add) >> shift) - 1]
}
```

7) 如果 diffX 小于等于 1<<(Bitdepth-8)或 diffY 小于等于 1<<(Bitdepth-8)，predMatrixInter[x][y]等于 Clip1((yMin+yMax)/2)，(x=0～K-1,y=0～L-1)；否则，predMatrixInter 等于 predMatrixInterPf。

8.10 预测补偿

8.10.1 概述

本条规定预测补偿的过程。

先按 8.10.2 导出预测样本矩阵 predMatrixTmp，再按 8.10.3 由 predMatrixTmp 导出补偿后样本矩阵 CompMatrix。

如果当前编码单元的分量模式是‘COMPONENT_CHROMA’且 PictureIscEvsUbvsEnableFlag 的值等于

T/AI 109.8-2025

1，按 8.10.4 导出常现位置色度样本值。

8.10.2 导出预测样本矩阵

本条导出预测样本矩阵 predMatrixTmp。

令当前预测单元的宽度和高度分别是 W 和 H，x 的取值范围是 0～W-1，y 的取值范围是 0～H-1。

如果当前预测单元的预测类型是普通帧内预测：

```
for (x=0; x<W; x++) {
    for (y=0; y<H; y++)
        predMatrixTmp[x][y] = predMatrix[x][y]
}
```

如果当前预测单元的预测类型是块复制帧内预测：

```
for (x=0; x<W; x++) {
    for (y=0; y<H; y++)
        predMatrixTmp[x][y] = predMatrixIbc[x][y]
}
```

如果当前预测单元的预测类型是串复制帧内预测：

```
for (x=0; x<W; x++) {
    for (y=0; y<H; y++)
        predMatrixTmp[x][y] = predMatrixIsc[x][y]
}
```

如果当前预测单元的预测类型是帧间预测：

```
for (x=0; x<W; x++) {
    for (y=0; y<H; y++)
        predMatrixTmp[x][y] = predMatrixInter[x][y]
}
```

上述过程中，predMatrix 是帧内预测的预测样本矩阵，predMatrixIbc 是当前编码帧未滤波重建图像的预测样本矩阵，predMatrixInter 是帧间预测的预测样本矩阵。

8.10.3 导出补偿后样本矩阵

本条导出补偿后样本矩阵 CompMatrix。

令当前预测单元的宽度和高度分别是 W 和 H，x 的取值范围是 0～W-1，y 的取值范围是 0～H-1，补偿后样本矩阵如下：

```
for (x=0; x<W; x++) {
    for (y=0; y<H; y++)
        CompMatrix[x][y] = Clip1(predMatrixTmp[x][y] + ResidueMatrix[x][y])
}
```

8.10.4 常现位置色度样本坐标

令 k 从 0～PrevPvBufSize-1，如果 PrevCompLumaFreqOccurPos[k]等于 1，执行以下操作：

a) 记当前编码单元的左上角样本在当前图像的色度样本矩阵中的位置为(xC,yC)，当前最大编码单元行左上角样本在当前图像中的坐标为(LcuRx0,LcuRy0)，历史点预测信息表 PrevPpInfoList 中第 k 个点矢量(PrevPpInfoList[k][0],PrevPpInfoList[k][1])为

T/AI 109.8-2025

(P_vX_k, P_vY_k)；

b) 计算(x_k, y_k)；

$$x_k = (P_vX_k >> 1) + (LcuRx0 >> 1) - xC$$
$$y_k = (P_vY_k >> 1) + (LcuRy0 >> 1) - yC$$

c) 将 CompMatrix[x_k][y_k]作为 LcuRowBufC[P_vX_k>>1][P_vY_k>>1]用于后续在分量模式为‘COMPONENT_LUMACHROMA’的非普通串子模式中从 LcuRowBufC 中取出常现位置的色度分量完成色度预测样本导出，见 8.8.3.3，其中 C 为 Cb 或 Cr；

d) 令 PrevCompLumaFreqOccurPos[k]等于 0，即该常现位置以及点矢量从此等同于来自分量模式为‘COMPONENT_LUMACHROMA’的编码单元。

8.11 去块效应滤波

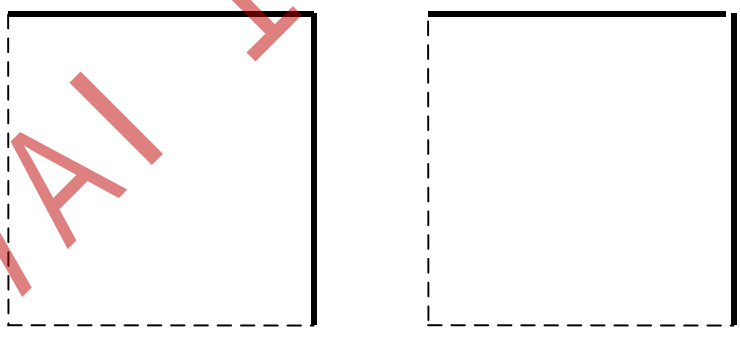
8.11.1 概述

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 40 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，首先按 8.11.2 判断该边界是否需要滤波。如果需要滤波，则按 8.11.3 确定去块滤波调整参数，并按 8.11.4 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 8.11.5 到 8.11.10）；否则，直接将补偿后样本的值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，虚线为其他滤波块待滤波的边界。

图40 滤波单元中需要处理的边界示意

8.11.2 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：

——如果待滤波边界是图像边界，则该边界不需要滤波；

——如果待滤波边界是片边界且 CplfEnableFlag 为 0，则该边界不需要滤波；

——如果待滤波边界是亮度滤波边界，且该亮度滤波边界不是亮度编码块或亮度变换块的边界，则

T/AI 109.8-2025

该边界不需要滤波；

——如果待滤波边界是色度滤波边界，且该色度滤波边界不是色度编码块或色度变换块的边界，且该色度滤波边界对应的亮度滤波边界不是亮度编码块的边界，则该边界不需要滤波；

——如果待滤波边界是亮度滤波边界，且该亮度滤波边界所在的编码单元的 SbtCuFlag 为 1，且该亮度滤波边界不是亮度编码块的边界，则该边界不需要滤波；

——如果待滤波边界所在的编码单元的 IscCuFlag 为 1，则该边界不需要滤波；

——如果待滤波边界两边样本所在的权限储存单元的权限值不相等，则该边界不需要滤波。

8.11.3 确定去块滤波调整参数

本条确定去块滤波调整参数 DbrThresold、DbrOffset0、DbrOffset1、DbrAltOffset0 和 DbrAltOffset1。

如果当前为垂直边界且PictureDbrVEnableFlag为1,或当前为水平边界且PictureDbrHEnableFlag为 1，则 PictureDbrEnableFlag 的值为 1；否则，PictureDbrEnableFlag 的值为 0。

如果当前为垂直边界且 PictureAltDbrVEnableFlag 为 1，或当前为水平边界且 PictureAltDbrHEnableFlag 为 1，则 PictureAltDbrEnableFlag 的值为 1；否则，PictureAltDbrEnableFlag 的值为 0。

对于垂直边界：

```
DbrThreshold = DbrVThreshold
DbrOffset0 = DbrVOffset0
DbrOffset1 = DbrVOffset1
DbrAltOffset0 = DbrVAltOffset0
DbrAltOffset1 = DbrVAltOffset1
```

对于水平边界：

```
DbrThreshold = DbrHThreshold
DbrOffset0 = DbrHOffset0
DbrOffset1 = DbrHOffset1
DbrAltOffset0 = DbrHAltOffset0
DbrAltOffset1 = DbrHAltOffset1
```

8.11.4 边界滤波强度的推导过程

图 41 表示某一段滤波块边界（用黑色粗线表示），其两侧的 8 个样本分别记为 p_0、p_1、p_2、p_3和 q_0、q_1、q_2、q_3。

T/AI 109.8-2025

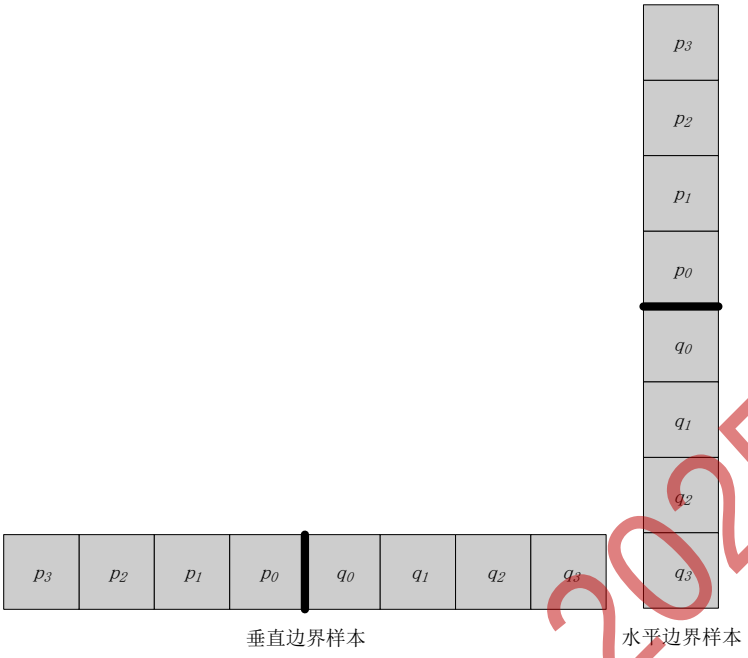


图41 滤波块边界样本

如果满足以下所有条件，则边界滤波强度 Bs 等于 0。

a) p_0和 q_0所在的编码单元所有变换块的量化系数均为 0。其中，如果 p_0（或 q_0）是亮度样本且 p_0（或 q_0）所在的编码单元只包含亮度样本，则 p_0（或 q_0）所在的编码单元指包含 p_0（或 q_0）的亮度编码单元；如果 p_0（或 q_0）是色度样本且 p_0（或 q_0）所在的编码单元只包含色度样本，则 p_0（或 q_0）所在的编码单元指包含 p_0（或 q_0）对应亮度样本的编码单元；否则（即 p_0或 q_0所在的编码单元同时包含亮度样本和色度样本），则 p_0（或 q_0）所在的编码单元指包含 p_0（或 q_0）的编码单元；

b) p_0和 q_0所在的编码单元的预测类型不是帧内；

c) 记 B_P和 B_Q分别是 p_0和 q_0对应的 4×4 亮度编码块，B_P和 B_Q的运动信息同时满足下列条件 1 和 2 或同时满足条件 3、4 和 5。

1) B_P和 B_Q分别对应的空域运动信息存储单元的 L0 参考索引均等于-1，或 B_P和 B_Q分别对应的空域运动信息存储单元的 L0 参考索引对应的参考帧为同一帧且空域运动信息存储单元的 L0 运动矢量的所有分量的差均小于一个整像素点；

2) B_P和 B_Q分别对应的空域运动信息存储单元的 L1 参考索引均等于-1，或 B_P和 B_Q分别对应的空域运动信息存储单元的 L1 参考索引对应的参考帧为同一帧且空域运动信息存储单元的 L1 运动矢量的所有分量的差均小于一个整像素点；

3) 满足以下条件之一：

- B_P对应的空域运动信息存储单元的 L0 参考索引等于-1 且 B_Q对应的空域运动信息存储单元的 L1 参考索引等于-1；
- B_P对应的空域运动信息存储单元的 L0 参考索引对应的参考帧与 B_Q对应的空域运动信息存储单元的 L1 参考索引对应的参考帧为同一帧且 B_P对应的空域运动信息存储单元的 L0 运动矢量和 B_Q对应的空域运动信息存储单元的 L1 运动矢量的所有分量的差均小于一个整像素点。

4) 满足以下条件之一：

T/AI 109.8-2025

- B_Q对应的空域运动信息存储单元的L0参考索引等于-1且B_P对应的空域运动信息存储单元的L1参考索引等于-1；
- B_Q对应的空域运动信息存储单元的L0参考索引对应的参考帧与B_P对应的空域运动信息存储单元的L1参考索引对应的参考帧为同一帧且B_Q对应的空域运动信息存储单元的L0运动矢量和B_P对应的空域运动信息存储单元的L1运动矢量的所有分量的差均小于一个整像素点。

5) B_P对应的空域运动信息存储单元的L0参考索引对应的参考帧和B_Q对应的空域运动信息存储单元的L0参考索引对应的参考帧不为同一帧；B_P对应的空域运动信息存储单元的L1参考索引对应的参考帧和B_Q对应的空域运动信息存储单元的L1参考索引对应的参考帧不为同一帧。

否则，按以下方法计算边界滤波强度Bs。

a) 计算p_0和q_0所在的编码单元的平均量化参数QP_{av}；

计算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。

——如果是亮度样本，则使用亮度编码块的量化参数；

——如果是色度样本，按以下方法根据对应位置的亮度量化参数QP_Y计算色度编码块的量化参数。

第一步，计算p0和q0所在的编码单元的xCb和xCr，其取值范围应是-16～63。

```
xCb = Clip3(-16, 63, QPY – 8 * (BitDepth – 8) + chroma_quant_param_delta_cb)
xCr = Clip3(-16, 63, QPY – 8 * (BitDepth – 8) + chroma_quant_param_delta_cr)
```

第二步，分别以xCb和xCr为索引查表104得到p0和q0所在的编码单元的QP'cb和QP'cr，并计算Cb、Cr色度编码块的量化参数QPcb和QPcr。

```
QPCb = Clip3(0, 63 + 8 * (BitDepth – 8), QP'Cb + 8 * (BitDepth – 8))
QPCr = Clip3(0, 63 + 8 * (BitDepth – 8), QP'Cr + 8 * (BitDepth – 8))
```

计算平均量化参数为：

```
QPav = ( QPp + QPq + 1 ) >> 1
```

b) 计算索引IndexA和IndexB；

```
IndexA = Clip3(0, 63, QPav + 8 * (BitDepth – 8) + AlphaCOffset)
IndexB = Clip3(0, 63, QPav + 8 * (BitDepth – 8) + BetaOffset)
```

c) 分别根据IndexA和IndexB查表138得到α'和β'的值，再根据BitDepth得到α、β的值；

```
α = α' << (BitDepth – 8)
β = β' << (BitDepth – 8)
```

d) 如果DeblockingFilterType为1，且Abs(p0-q0)大于或等于4×α，则Bs等于0；否则，按以下方法计算Bs。

1) 将fL和fR的值均置为0，计算fS；

```
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++
```

T/AI 109.8-2025

```
fS = fL + fR
```

2) 根据 fS 确定 Bs；

- 当 fS 等于 6 时，如果 Abs(p0−p1)小于或等于β/4 且 Abs(q0−q1)小于或等于β/4 且 Abs(p0−q0)小于α，则 Bs 等于 4；否则 Bs 等于 3；
- 当 fS 等于 6 且 DeblockingFilterType 等于 1 时，如果 Abs(p0−p1)小于或等于β/4 且 Abs(q0−q1)小于或等于β/4 且 Abs(p0−p3)小于或等于β/2 且 Abs(q0−q3)小于或等于β/2 且 Abs(p0−q0)小于α，则 Bs 等于 4；否则 Bs 等于 3；
- 当 fS 等于 5 时，如果 p_0 等于 p_1 且 q_0 等于 q_1，则 Bs 等于 3；否则 Bs 等于 2；
- 当 fS 等于 5 且 DeblockingFilterType 等于 1 时，如果 p0 等于 p1 且 q0 等于 q1 且 Abs(p2−q2)小于α，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，如果 Abs(p_1−q_1)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。

3) 如果步骤 2)得到的 Bs 不等于 0 且滤波的边界是色度编码块边界，则 Bs 减 1。

表138 IndexA 和 IndexB 与块边界阈值α'和β'与的关系

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

8.11.5 亮度分量 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)：

$$P_0 = (p_2 * 3 + p_1 * 8 + p_0 * 10 + q_0 * 8 + q_1 * 3 + 16) >> 5$$

$$P_1 = (p_2 * 4 + p_1 * 5 + p_0 * 4 + q_0 * 3 + 8) >> 4$$

T/AI 109.8-2025

$P_2 = (p_3 * 2 + p_2 * 2 + p_1 * 2 + p_0 * 1 + q_0 * 1 + 4) >> 3$

$Q_0 = (p_1 * 3 + p_0 * 8 + q_0 * 10 + q_1 * 8 + q_2 \times 3 + 16) >> 5$

$Q_1 = (p_0 * 3 + q_0 * 4 + q_1 * 5 + q_2 * 4 + 8) >> 4$

$Q_2 = (p_0 * 1 + q_0 * 1 + q_1 * 2 + q_2 * 2 + q_3 * 2 + 4) >> 3$

如果 PictureDbrEnableFlag 的值为 1，按 8.11.11 调整 P_0、P_1、P_2、Q_0、Q_1 和 Q_2 的值。

8.11.6 亮度分量 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)：

$P_0 = (p_2 + (p_1 << 2) + (p_0 << 2) + (p_0 << 1) + (q_0 << 2) + q_1 + 8) >> 4$

$P_1 = ((p_2 << 1) + p_2 + (p_1 << 3) + (p_0 << 2) + q_0 + 8) >> 4$

$Q_0 = (p_1 + (p_0 << 2) + (q_0 << 2) + (q_0 << 1) + (q_1 << 2) + q_2 + 8) >> 4$

$Q_1 = ((q_2 << 1) + q_2 + (q_1 << 3) + (q_0 << 2) + p_0 + 8) >> 4$

如果 PictureDbrEnableFlag 的值为 1，按 8.11.11 调整 P_0、P_1、Q_0 和 Q_1 的值。

8.11.7 亮度分量 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)：

$P_0 = ((p_1 << 1) + p_1 + (p_0 << 3) + (p_0 << 1) + (q_0 << 1) + q_0 + 8) >> 4$

$Q_0 = ((p_0 << 1) + p_0 + (q_0 << 3) + (q_0 << 1) + (q_1 << 1) + q_1 + 8) >> 4$

如果 PictureDbrEnableFlag 的值为 1，按 8.11.11 调整 P_0 和 Q_0 的值。

8.11.8 亮度分量 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)：

$P_0 = ((p_0 << 1) + p_0 + q_0 + 2) >> 2$

$Q_0 = ((q_0 << 1) + q_0 + p_0 + 2) >> 2$

如果 PictureDbrEnableFlag 的值为 1，按 8.11.11 调整 P_0 和 Q_0 的值。

8.11.9 亮度分量 Bs 等于 0 时的边界滤波过程

边界滤波强度 Bs 的值为 0 且 PictureAltDbrEnableFlag 的值为 1 时，对 p_0 和 q_0 滤波调整的计算过程如下(P_0 和 Q_0 是滤波调整后的值)：

$DP_0 = (q_0 - p_0+2) >> 2$

if (DP_0< -DbrThreshold)

$P_0 = Clip1(p_0 + DbrAltOffset0)$

else if (DP_0 >DbrThreshold)

$P_0 = Clip1(p_0 + DbrAltOffset1)$

$DQ_0 = (p_0 - q_0+2) >> 2$

if (DQ_0 < -DbrThreshold)

$Q_0 = Clip1(q_0 + DbrAltOffset0)$

else if (DQ_0> DbrThreshold)

$Q_0 = Clip1(q_0 + DbrAltOffset1)$

8.11.10 色度分量 Bs 大于 0 时的边界滤波过程

T/AI 109.8-2025

边界滤波强度 Bs 的值大于 0 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)：

$$P_0 = ((p_1 << 1) + p_1 + (p_0 << 3) + (p_0 << 1) + (q_0 << 1) + q_0 + 8) >> 4$$
$$Q_0 = ((p_0 << 1) + p_0 + (q_0 << 3) + (q_0 << 1) + (q_1 << 1) + q_1 + 8) >> 4$$

边界滤波强度 Bs 的值等于 3 时，对 p_1 和 q_1 滤波的计算过程如下(P_1 和 Q_1 是滤波后的值)：

$$P_1 = ((p_2 << 1) + p_2 + (p_1 << 3) + (p_0 << 1) + p_0 + (q_0 << 1) + 8) >> 4$$
$$Q_1 = ((q_2 << 1) + q_2 + (q_1 << 3) + (q_0 << 1) + q_0 + (p_0 << 1) + 8) >> 4$$

8.11.11 去块效应滤波调整过程

本条调整 P_i 和 Q_i（i 可为 0、1 或 2）的值：

```
if (pi > Pi + DbrThreshold)
    Pi = Clip1((Pi + pi + 1) >> 1 + DbrOffset0)
else if (pi < Pi - DbrThreshold)
    Pi = Clip1((Pi + pi + 1) >> 1 + DbrOffset1)
if (qi > Qi + DbrThreshold)
    Qi = Clip1((Qi + qi + 1) >> 1 + DbrOffset0)
else if (qi < Qi - DbrThreshold)
    Qi = Clip1((Qi + qi + 1) >> 1 + DbrOffset1)
```

8.12 样值偏移补偿

8.12.1 基本样值偏移补偿

8.12.1.1 概述

如果 PatchSaoEnableFlag[compIndex]的值为 1，先按 8.12.1.2 导出基本样值偏移补偿单元，再按 8.12.1.3 导出与当前基本样值偏移补偿单元对应的基本样值偏移补偿信息，最后按 8.12.1.4 对当前基本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。

否则，直接将滤波后样本对应分量的值作为偏移后该样本分量的值。

8.12.1.2 导出基本样值偏移补偿单元

先确定当前最大编码单元，再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。

T/AI 109.8-2025

8.12.1.3 导出基本样值偏移补偿信息

本条规定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[compIndex]、样值偏移补偿的偏移值 SaoOffset[compIndex][j]、样值偏移补偿边缘模式类型 SaoEdgeType[compIndex]、样值偏移补偿区间模式的起始偏移子区间位置 SaoIntervalStartPos[compIndex]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[compIndex]。

第 1 步，如果 SaoMergeLeftAvai 为 1 或 SaoMergeUpAvai 为 1，则 MergeFlagExist 的值为 1；否则 MergeFlagExist 的值为 0。根据 SaoMergeLeftAvai、SaoMergeUpAvai 和 SaoMergeTypeIndex 查表 139 得到样值偏移补偿合并模式 SaoMergeMode。

表139 样值偏移补偿模式

SaoMergeLeftAvai 值	SaoMergeUpAvai 值	SaoMergeTypeIndex 值	样值偏移补偿合并模式
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

第 2 步，导出 SaoMode[compIndex]。

a) 如果 SaoMergeMode 的值是‘SAO_NON_MERGE’，从位流中解析得到 SaoMode[compIndex]的值；

b) 如果 SaoMergeMode 的值是‘SAO_MERGE_LEFT’，SaoMode[compIndex]的值等于左边样值偏移补偿单元的 SaoMode[compIndex]的值；

c) 如果 SaoMergeMode 值是‘SAO_MERGE_UP’,SaoMode[compIndex]的值等于上边样值偏移补偿单元的 SaoMode[compIndex]的值。

第 3 步，导出 SaoOffset[compIndex]、SaoIntervalStartPos[compIndex]、SaoIntervalDeltaPosMinus2[compIndex]和 SaoEdegType[compIndex]。

a) 如果 SaoMode[compIndex]的值是‘SAO_Interval’，先按以下方法导出 SaoOffset[compIndex][j]（j=0～3）、SaoIntervalStartPos[compIndex]和 SaoIntervalDeltaPosMinus2[compIndex]，然后按 8.12.1.4.1 进行样值偏移补偿操作：

1) 如果 SaoMergeMode 的值为‘SAO_NON_MERGE’，从位流中解析得到 SaoIntervalOffsetAbs[compIndex][j](j=0～3)、SaoIntervalOffsetSign[compIndex][j]（j=0～3）、SaoIntervalStartPos[compIndex]和 SaoIntervalDeltaPosMinus2[compIndex]，并计算 SaoOffset[compIndex][j]；

```
for (j=0; j<4; j++)
    SaoOffset[compIndex][j] = SaoIntervalOffsetAbs[compIndex][j] * SaoIntervalOffsetSign[compIndex][j]
```

2) 否则，如果 SaoMergeMode 的值是‘SAO_MERGE_LEFT’，SaoOffset[compIndex][j]（j=0～3）的值等于左边样值偏移补偿单元的 SaoOffset[compIndex][j]（j=0～3）的值，SaoIntervalStartPos[compIndex]和 SaoIntervalDeltaPosMinus2[compIndex]的值等于

T/AI 109.8-2025

左边样值偏移补偿单元的 SaoIntervalStartPos[compIndex] 和 SaoIntervalDeltaPosMinus2[compIndex]的值；

3) 否则，SaoOffset[compIndex][j]（j=0～3）的值等于上边样值偏移补偿单元的 SaoOffset[compIndex][j]（j=0～3）的值，SaoIntervalStartPos[compIndex]和 SaoIntervalDeltaPosMinus2[compIndex]的值等于上边样值偏移补偿单元的 SaoIntervalStartPos[compIndex]和 SaoIntervalDeltaPosMinus2[compIndex]的值。

b) 否则，如果 SaoMode[compIndex]的值是‘SAO_Edge’，先按以下方法导出 SaoOffset[compIndex][j]（j=0～3）和 SaoEdgeType[compIndex]，再按 0 进行样值偏移补偿操作：

1) 如果 SaoMergeMode 的值是‘SAO_NON_MERGE’，从位流中解析得到 SaoEdgeOffset[compIndex][j]（j=0～3）和 SaoEdgeType[compIndex]的值，并计算 SaoOffset[compIndex][j]；

```
for (j=0; j<4; j++)
    SaoOffset[compIndex][j] = SaoEdgeOffset[compIndex][j]
```

2) 否则，如果 SaoMergeMode 的值是‘SAO_MERGE_LEFT’，SaoOffset[compIndex][j]（j=0～3）和 SaoEdgeType[compIndex]的值等于左边样值偏移补偿单元的 SaoOffset[compIndex][j]（j=0～3）和 SaoEdgeType[compIndex]的值；

3) 否则，SaoOffset[compIndex][j]（j=0～3）和 SaoEdgeType[compIndex]的值等于上边样值偏移补偿单元的 SaoOffset[compIndex][j]（j=0～3）和 SaoEdgeType[compIndex]的值。

8.12.1.4 基本样值偏移补偿操作

8.12.1.4.1 SAO_Interval 模式的操作

如果样值偏移补偿单元的 SaoMode[compIndex]是‘SAO_Interval’，进行以下操作：

第 1 步，根据滤波后样本的 compIndex 分量值查表 140 得到该分量对应的 saoOffset[compIndex]。

表140 区间模式的偏移值

样值	偏移值（saoOffset[compIndex]）
SaoIntervalOffsetPos[compIndex][0] << shift1 ～ (SaoIntervalOffsetPos[compIndex][0] << shift1) + ((1 << shift1) - 1)	SaoOffset[compIndex][0]
SaoIntervalOffsetPos[compIndex][1] << shift1 ～ (SaoIntervalOffsetPos[compIndex][1] << shift1) + ((1 << shift1) - 1)	SaoOffset[compIndex][1]
SaoIntervalOffsetPos[compIndex][2] << shift1 ～ (SaoIntervalOffsetPos[compIndex][2] << shift1) + ((1 << shift1) - 1)	SaoOffset[compIndex][2]
SaoIntervalOffsetPos[compIndex][3] << shift1 ～ (SaoIntervalOffsetPos[compIndex][3] << shift1) + ((1 << shift1) - 1)	SaoOffset[compIndex][3]
其他	0

表 140 中：

```
shift1 = BitDepth - 5
```

T/AI 109.8-2025

```
SaoIntervalOffsetPos[compIndex][0] = SaoIntervalStartPos[compIndex]
SaoIntervalOffsetPos[compIndex][1] = (SaoIntervalStartPos[compIndex] + 1) % 32
SaoIntervalOffsetPos[compIndex][2]=(SaoIntervalStartPos[compIndex]+SaoIntervalDeltaPosMinus2[compIndex] + 2) %
        32
SaoIntervalOffsetPos[compIndex][3]=(SaoIntervalStartPos[compIndex]+SaoIntervalDeltaPosMinus2[compIndex] + 3) %
        32
```

第 2 步，根据当前样本的滤波后样本的 compIndex 分量 x[compIndex]计算偏移后样本的 compIndex 分量 y[compIndex]。

```
y[compIndex] = Clip1(x[compIndex] + saoOffset[compIndex])
```

8.12.1.4.2 SAO_Edge 模式的操作

如果样值偏移补偿单元的 SaoMode[compIndex]是‘SAO_Edge’，进行以下操作：

第 1 步，根据 SaoEdgeType[compIndex]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 42）。如果满足以下条件之一，则根据当前样本的滤波后样本的 compIndex 分量 x[compIndex]计算偏移后样本的 compIndex 分量 y[compIndex]，并结束补偿过程；否则，继续执行下面的步骤。

```
y[compIndex] = x[compIndex]
```

——a 或 b 不与 c 处在同一片内，且 CplfEnableFlag 的值为 0；

——a 或 b 不在当前图像内；

——从当前帧的权限储存单元中获取 a, b, c 的权限值且 a 或 b 的权限值大于 c 的权限值。

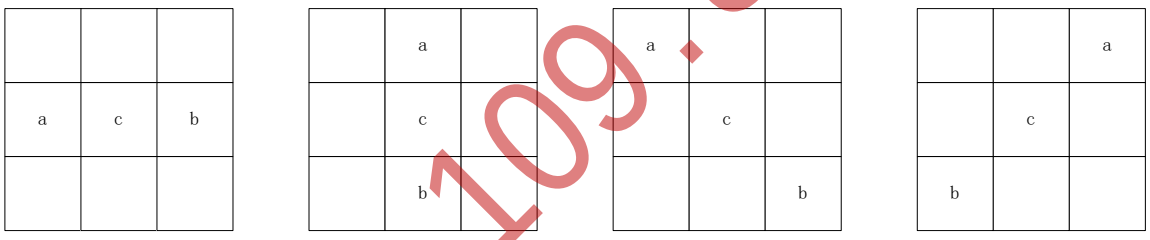


图42 当前滤波后样本和相邻滤波后样本的关系

第 2 步，根据表 141 利用当前样本 c 的滤波后 compIndex 分量值 x_c 与相邻样本 a 和 b 的滤波后样本 compIndex 分量值 x_a 和 x_b 的关系确定当前样本 compIndex 分量的 saoOffset[compIndex]。

表141 边缘模式的偏移值

样本分量值关系	偏移值（saoOffset[compIndex]）
$x_c < x_a \&\& x_c < x_b$	SaoOffset[compIndex][0]
$(x_c < x_a \&\& x_c == x_b) \mid\mid (x_c == x_a \&\& x_c < x_b)$	SaoOffset[compIndex][1]
$(x_c > x_a \&\& x_c == x_b) \mid\mid (x_c == x_a \&\& x_c > x_b)$	SaoOffset[compIndex][2]
$x_c > x_a \&\& x_c > x_b$	SaoOffset[compIndex][3]
其他	0

第 3 步，根据当前样本的滤波后样本的 compIndex 分量 x[compIndex]计算偏移后样本的 compIndex 分量 y[compIndex]。

T/AI 109.8-2025

```
y[compIndex] = Clip1(x[compIndex] + saoOffset[compIndex])
```

8.12.1.4.3 SAO_Off 模式的操作

如果样值偏移补偿单元的 SaoMode[compIndex]是‘SAO_Off’，直接将当前样本滤波后样本 compIndex 分量的值作为该样本偏移后样本 compIndex 分量值。

8.12.2 增强样值偏移补偿

8.12.2.1 概述

如果 PictureEsaoEnableFlag[compIndex]为 1，先按 8.12.2.2 导出增强样值偏移补偿单元，再按 8.12.2.3 对增强样值偏移补偿单元内的各个样本的各个分量进行操作，得到偏移后样本值；否则，直接将滤波后样本对应分量的值作为偏移后该样本分量的值。

8.12.2.2 导出增强样值偏移补偿单元

先确定当前最大编码单元，如果当前分量为色度分量，则直接将当前最大编码单元所对应的色度样本区域作为色度增强样值偏移补偿单元；否则，将当前最大编码单元按下列步骤得到与其对应的增强样值偏移补偿单元。

第 1 种情况，CplfEnableFlag 的值等于 0：

a) 当前最大编码单元所在亮度样本区域为 E1；
b) 如果区域 E1 超出当前片边界，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包含当前片最左列的样本或最右列的样本或最上行的样本或最下行的样本，则：
   1) 如果当前最大编码单元包含当前片最左列的样本，则将区域 E2 的左边界向右缩进一列，得到区域 E3；
   2) 如果当前最大编码单元包含当前片最右列的样本，则将区域 E2 的右边界向左缩进一列，得到区域 E3；
   3) 如果当前最大编码单元包含当前片最上行的样本，则将区域 E2 的上边界向下缩进一行，得到区域 E3；
   4) 如果当前最大编码单元包含当前片最下行的样本，则将区域 E2 的下边界向上缩进一行，得到区域 E3。
d) 否则，令 E3 等于 E2；
e) 将区域 E3 作为当前增强样值偏移补偿单元。

第 2 种情况，CplfEnableFlag 的值等于 1：

a) 当前最大编码单元所在亮度样本区域为 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包含图像最左列的样本或最右列的样本或最上行的样本或最下行的样本，则：
   1) 如果当前最大编码单元包含图像最左列的样本，则将区域 E2 的左边界向右缩进一列，得到区域 E3；
   2) 如果当前最大编码单元包含图像最右列的样本，则将区域 E2 的右边界向左缩进一列，得到区域 E3；

T/AI 109.8-2025

3) 如果当前最大编码单元包含图像最上行的样本，则将区域 E2 的上边界向下缩进一行，得到区域 E3；
4) 如果当前最大编码单元包含图像最下行的样本，则将区域 E2 的下边界向上缩进一行，得到区域 E3。

d) 否则，令 E3 等于 E2；
e) 将区域 E3 作为当前增强样值偏移补偿单元。

8.12.2.3 增强样值偏移补偿操作

如果 EsaoLcuEnableFlag[compIndex][LcuIndex]等于 1，且当前分量为亮度分量，且样本(x,y)对应权限储存单元的权限大于等于(x-1,y-1)，(x-1,y)，(x-1,y+1)，(x,y+1)，(x,y-1)，(x+1,y-1)，(x+1,y)，(x+1,y+1)中任何一个样本对应的权限储存单元的权限，则对 compIndex 亮度分量使用第 EsaoLcuSetIndex[compIndex][LcuIndex]组偏移值进行增强样值偏移补偿；否则不进行增强样值偏移补偿。其中，(x,y)为当前滤波样本位置。亮度分量的增强样值偏移补偿操作如下：

```
setIndex = EsaoLcuSetIndex[compIndex][LcuIndex]
C = 0
C1 = 0
for (a=-1; a<=1; a++) {
    for (b=-1; b<=1; b++) {
        if (a != 0 || b != 0) {
            if (EsaoLumaParamType[setIndex] == 0)
                C1+=( p(x, y)< p(x+a, y+b) ? 1 : ( p(x, y)== p(x+a, y+b)?0:-1))
            else
                C1 += (p(x, y) < p(x+a, y+b) ? 1 : 0)
        }
    }
}
C1 = (EsaoLumaParamType[setIndex] == 0) ? C1+8 : C1
C = ((p(x,y) * PictureEsaoAdaptiveParam[setIndex]) >> BitDepth) * (EsaoLumaParamType[setIndex] ? 9 : 17) + C1
p’(x, y) = Clip1(p(x, y) + EsaoFilterCoeff[0][setIndex][C])
```

其中，p(x,y)是滤波后样本，p’(x,y)是增强样值偏移补偿后样本。

如果 EsaoLcuEnableFlag[compIndex][LcuIndex]等于 1，且当前分量为色度分量，则对色度分量使用第 EsaoLcuSetIndex[compIndex][LcuIndex]组偏移值进行增强样值偏移补偿。色度分量的增强样值偏移补偿操作如下：

```
setIndex = EsaoLcuSetIndex[compIndex][LcuIndex]
C = ((p(x,y) * EsaoTableUV[PictureEsaoAdaptiveParamIndex[compIndex][setIndex]])) >> BitDepth
p’(x, y) = Clip1(p(x, y) + EsaoFilterCoeff[compIndex][setIndex][C])
```

其中，p(x,y)是滤波后样本，p’(x,y)是增强样值偏移补偿后样本。EsaoTableUV 的值见表 142。

表142 EsaoTableUV 的值

k	EsaoTableUV 值
0 ~ 59	(k + 1) * 2
60	122

T/AI 109.8-2025

表 142 EsaoTableUV 的值（续）

k	EsaoTableUV 值
61	124
62	128
63	132
64	136
65	140
66	144
67	148
68	152
69	156
70	160
71	164
72	168
73	172
74	176
75	180
76	184
77	188
78	192
79	196
80	200
81	204
82	208
83	212
84	216
85	220
86	224
87	228
88	232
89	236
90	240
91	244
92	248
93	256
94	264
95	272

T/AI 109.8-2025

8.12.3 跨分量样值偏移补偿

8.12.3.1 概述

如果 PictureCcsaoEnableFlag[compIndex]的值为 1，先按 8.12.3.2 导出跨分量样值偏移补偿单元，然后按 8.12.3.3 对当前跨分量样值偏移补偿单元内的各分量样本进行操作，得到跨分量偏移后样本值。

否则，直接将偏移后样本对应分量的值作为跨分量偏移后该样本分量的值。

8.12.3.2 导出跨分量样值偏移补偿单元

先确定当前最大编码单元，再根据当前最大编码单元按下列步骤得到与其对应的当前跨分量样值偏移补偿单元。

第 1 种情况，CplfEnableFlag 的值等于 0：

a) 当前最大编码单元所在色度样本区域为 E1；
b) 如果区域 E1 超出当前片边界，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包含当前片最左列的样本且不包含当前片最上行的样本，则将区域 E2 的左边界向右缩进一列，得到区域 E3；
d) 否则，如果当前最大编码单元包含当前片最上行的样本且不包含当前片最左列的样本，则将区域 E2 的上边界向下缩进一行，得到区域 E3；
e) 否则，如果当前最大编码单元同时包含当前片最左列的样本和最上行的样本，则将区域 E2 的左边界向右缩进一列后再将新区域的上边界向下缩进一行，得到区域 E3；
f) 否则，令 E3 等于 E2；
g) 将区域 E3 作为当前跨分量样值偏移补偿单元。

第 2 种情况，CplfEnableFlag 的值等于 1：

a) 当前最大编码单元所在色度样本区域为 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包含图像最左列的样本且不包含图像最上行的样本，则将区域 E2 的左边界向右缩进一列，得到区域 E3；
d) 否则，如果当前最大编码单元包含图像最上行的样本且不包含图像最左列的样本，则将区域 E2 的上边界向下缩进一行，得到区域 E3；
e) 否则，如果当前最大编码单元同时包含图像最左列的样本和最上行的样本，则将区域 E2 的左边界向右缩进一列后再将新区域的上边界向下缩进一行，得到区域 E3；
f) 否则，令 E3 等于 E2；
g) 将区域 E3 作为当前跨分量样值偏移补偿单元。

8.12.3.3 跨分量样值偏移补偿操作

8.12.3.3.1 概述

如果 CcSaoLCUEnableFlag[compIndex][LcuIndex]等于 1，则对 compIndex 分量进行跨分量样值偏移补偿；否则不进行跨分量样值偏移补偿。

如果 PictureCcsaoMode[compIndex][CcsaoLcuSetIndex[compIndex][LcuIndex]]的值等于 0，则按 8.12.3.3.2 对 compIndex 分量根据第 CcsaoLcuSetIndex[compIndex][LcuIndex]组偏移值进行区间模式跨分量样值偏移补偿。否则，如果

T/AI 109.8-2025

PictureCcsaoMode[compIndex][CcsaoLcuSetIndex[compIndex][LcuIndex]]的值等于 1，则按 8.12.3.3.3 对 compIndex 分量根据第 CcsaoLcuSetIndex[compIndex][LcuIndex]组偏移值进行边缘模式跨分量样值偏移补偿。

8.12.3.3.2 区间模式跨分量样值偏移补偿操作

令当前色度分量样本位置为(xC, yC)，当前色度分量滤波后样本值为 p(xC, yC)，当前色度分量偏移后样本值为 p'(xC, yC)，对应的亮度分量样本位置为(xL, yL)，若存在亮度分量样本对应的权限储存单元内的权限值大于当前色度分量样本对应的权限储存单元内的权限值，当前色度分量样本不进行区间模式跨分量样值偏移补偿操作，否则对应的亮度分量滤波后样本值为 p(xL, yL)，跨分量偏移后样本值为 p''(xC, yC)，区间模式的跨分量样值偏移补偿操作如下。

```
setIndex = CcsaoLcuSetIndex[compIndex][LcuIndex]
xL = (xC << 1) + ccsaoCandPosX
yL = (yC << 1) + ccsaoCandPosY[lineIndex]
bandL = (p(xL, yL) * PictureCcsaoLumaBandNum[compIndex][setIndex]) >> BitDepth
bandC = (p(xC, yC) * PictureCcsaoChromaBandNum[compIndex][setIndex]) >> BitDepth
classIndex = bandL * PictureCcsaoChromaBandNum[compIndex][setIndex] + bandC
p''(xC, yC) = Clip1(p'(xC, yC) + CcsaoOffset[compIndex][setIndex][classIndex])
```

其中 ccsaoCandPosX 和 ccsaoCandPosY[lineIndex]的值由 PictureCcsaoCandIndex[compIndex][setIndex]查表 143 得到。如果 CplfEnableFlag 的值为 0 且当前最大编码单元包含当前片最下行的样本，或 CplfEnableFlag 的值为 1 且当前最大编码单元包含图像最下行的样本，则 lineIndex 的值为 0，否则 lineIndex 的值如下：

```
lcuHeightC = 1<< (LcuSizeInBit-1)
lineIndex = (yC % lcuHeightC) >= lcuHeightC - 4 ? (lcuHeightC - (yC % lcuHeightC) - 1) : 0
```

表143 跨分量样值偏移补偿坐标偏移值

PictureCcsaoCandIndex[compIndex][setIndex]	ccsaoCandPosX	ccsaoCandPosY[0]	ccsaoCandPosY[1]	ccsaoCandPosY[2]	ccsaoCandPosY[3]
0	-1	-1	0	2	4
1	0	-1	0	2	4
2	1	-1	0	2	4
3	-1	0	0	2	4
4	0	0	0	2	4
5	1	0	0	2	4
6	-1	1	1	2	4
7	0	1	1	2	4
8	1	1	1	2	4

8.12.3.3.3 边缘模式跨分量样值偏移补偿操作

令当前色度分量样本位置为(xC, yC)，当前色度分量滤波后样本值为 p(xC, yC)，当前色度分量偏移后样本值为 p'(xC, yC)，对应的亮度分量样本位置为(xL, yL)，对应的亮度分量滤波后样本值为 p(xL, yL)，

T/AI 109.8-2025

相邻的亮度分量滤波后样本值分别为 p(xL+a1, yL+a2)和 p(xL+b1, yL+b2)。跨分量偏移后样本值为 p”(xC, yC)，若存在亮度分量样本对应的权限储存单元内的权限值大于当前色度分量样本对应的权限储存单元内的权限值，当前色度分量样本不进行边缘模式跨分量样值偏移补偿操作，否则边缘模式的跨分量样值偏移补偿操作如下。

```
setIndex = CcsaoLcuSetIndex[compIndex][LcuIndex]
xL = (xC << 1)
yL = (yC << 1) + (lineIndex == 3 ? 4 : (lineIndex == 2 ? 2 : 0))
a1 = ccsaoEdgeTypeX[0]
a2 = ccsaoEdgeTypeY[0][lineIndex]
b1 = ccsaoEdgeTypeX[1]
b2 = ccsaoEdgeTypeY[1][lineIndex]
diff1 = p(xL,yL) – p(xL+a1, (yC<<1)+a2)
diff2 = p(xL,yL) – p(xL+b1, (yC<<1)+b2)
if (PictureCcsaoEdgeBandNumIndex[compIndex][setIndex] < 2)
    band = (p(xL,yL) * PictureCcsaoEdgeBandNum[compIndex][setIndex]) >> BitDepth
else
    band = (p(xC,yC) * PictureCcsaoEdgeBandNum[compIndex][setIndex]) >> BitDepth
classIndex = band *16 + C1 * 4 + C2
p’’(xC, yC) = Clip1(p’(xC, yC) + CcsaoOffset[compIndex][setIndex][classIndex])
```

其中 CX（X 为 1 或 2）的计算如下：

```
thr = PictureCcsaoEdgeThr[compIndex][setIndex]
CX = diffX < 0 ? ( diffX < -thr ? 0 : 1 ) : ( diffX < thr ? 2 : 3 )
```

ccsaoEdgeTypeX[0]、ccsaoEdgeTypeX[1]、ccsaoEdgeTypeY[0][lineIndex]和 ccsaoEdgeTypeY[1][lineIndex]的值由 PictureCcsaoEdgeType[compIndex][setIndex]查表 144 得到。如果 CplfEnableFlag 的值为 0 且当前最大编码单元包含当前片最下行的样本，或 CplfEnableFlag 的值为 1 且当前最大编码单元包含图像最下行的样本，则 lineIndex 的值为 0，否则 lineIndex 的值如下：

```
lcuHeightC = 1<< (LcuSizeInBit-1)
lineIndex = (yC % lcuHeightC) >= lcuHeightC – 4 ? (lcuHeightC - yC - 1) : 0
```

表144 跨分量样值偏移补偿边缘模式坐标偏移值

PictureCcsaoEdgeType[compIndex][setIndex]	ccsaoEdgeTypeX[0]	ccsaoEdgeTypeX[1]	ccsaoEdgeTypeY[0][0]	ccsaoEdgeTypeY[0][1]	ccsaoEdgeTypeY[0][2]	ccsaoEdgeTypeY[0][3]	ccsaoEdgeTypeY[1][0]	ccsaoEdgeTypeY[1][1]	ccsaoEdgeTypeY[1][2]	ccsaoEdgeTypeY[1][3]
0	-1	1	0	0	2	4	0	0	2	4
1	0	0	-1	0	2	4	1	1	2	4
2	-1	1	-1	0	2	4	1	1	2	4
3	1	-1	-1	0	2	4	1	1	2	4

T/AI 109.8-2025

8.13 自适应修正滤波

8.13.1 概述

如果 PictureAlfEnableFlag[compIndex]为 0，直接将偏移后样本分量的值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。

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。首先按 8.13.2 解码各分量的自适应修正滤波系数，再按 8.13.3 导出自适应修正滤波单元，并按 8.13.4 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，最后按 8.13.5 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。

8.13.2 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：

第 1 步，对系数 AlfCoeffLuma 进行处理。

a) 如果 EalfEnableFlag 为 0，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 43 的系数 C8）进行以下处理：

```
for (i=0; i<=alf_filter_num_minus1; i++) {
    coeffLuma = 0
    for(j=0; j<8; j++)
        coeffLuma += 2 * AlfCoeffLuma[i][j]
    AlfCoeffLuma[i][8] += 64 - coeffLuma
}
```

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) 如果 EalfEnableFlag 为 1，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=0～13）。对系数 AlfCoeffLuma[i][14]（即图 44 的系数 C14）进行以下处理。

```
for (i=0; i<=alf_filter_num_minus1; i++) {
    coeffLuma = 0
    sum = 1 << AlfLumaShift[i]
    for(j=0; j<14; j++)
        coeffLuma += 2 * AlfCoeffLuma[i][j]
    AlfCoeffLuma[i][14] += sum - coeffLuma
}
```

其中 AlfCoeffLuma[i][j]（j=0～13）的位宽是 7 位，取值范围是-64～63。经上述处理后 AlfCoeffLuma[i][14]的取值范围是 0～127。

第 2 步，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 alfCoeffIndexTab[FilterNum]）。

```
count = 0
alfCoeffIndexTab[0] = 0
```

T/AI 109.8-2025

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

第 3 步，对系数 AlfCoeffChroma 进行处理。

a) 如果 EalfEnableFlag 为 0，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）。分别对系数 AlfCoeffChroma[0][8]和 AlfCoeffChroma[1][8]（即图 43 的系数 C8）进行以下处理：

```
coeffChroma0 = 0
coeffChroma1 = 0
for(j=0; j<8; j++) {
    coeffChroma0 += 2 * AlfCoeffChroma[0][j]
    coeffChroma1 += 2 * AlfCoeffChroma[1][j]
}
AlfCoeffChroma[0][8] += 64 - coeffChroma0
AlfCoeffChroma[1][8] += 64 - coeffChroma1
```

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。

b) 如果 EalfEnableFlag 为 1，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=0～13）。分别对系数 AlfCoeffChroma[0][14]和 AlfCoeffChroma[1][14]（即图 44 的系数 C14）进行以下处理。

```
coeffChroma0 = 0
coeffChroma1 = 0
sumChroma0 = 1 << AlfChromaShift[0]
sumChroma1 = 1 << AlfChromaShift[1]
for(j=0; j<14; j++) {
    coeffChroma0 += 2 * AlfCoeffChroma[0][j]
    coeffChroma1 += 2 * AlfCoeffChroma[1][j]
}
AlfCoeffChroma[0][14] += sumChroma0 - coeffChroma0
AlfCoeffChroma[1][14] += sumChroma1 - coeffChroma1
```

其中 AlfCoeffChroma[i][j]（j=0～13）的位宽是 7 位，取值范围是-64～63。经上述处理后 AlfCoeffChroma[i][14]的取值范围是 0～127。

T/AI 109.8-2025

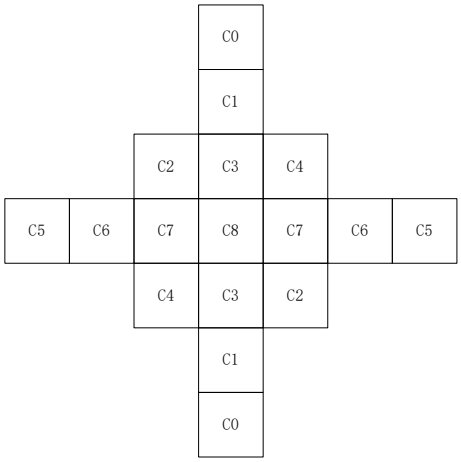


图43　自适应修正滤波系数（7×7 十字形加 3×3 方形）

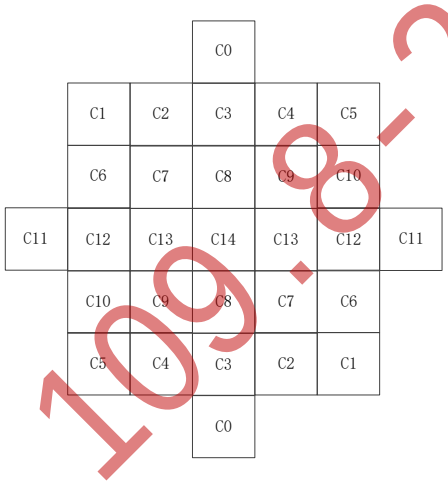


图44　自适应修正滤波系数（7×7 十字形加 5×5 方形）

8.13.3　导出自适应修正滤波单元

先确定当前最大编码单元，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45）：

a) 将当前最大编码单元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， 或者属于片边界且 CplfEnableFlag 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。

T/AI 109.8-2025

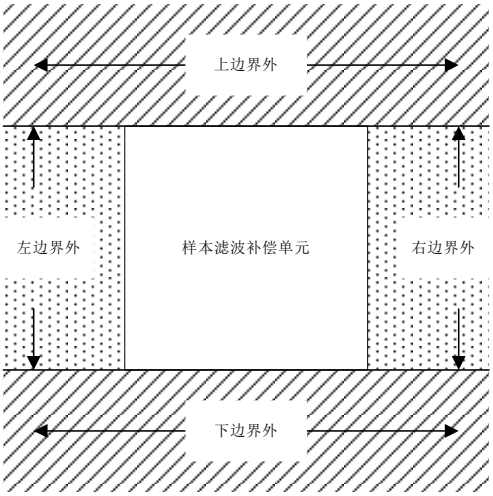


图45 自适应修正滤波单元

8.13.4 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

如果 EalfEnableFlag 为 0，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 filterIndex）。

```
xInterval = ((((PicWidthInLuma+ (1 << LcuSizeInBit) - 1) >> LcuSizeInBit) + 1) >> 2) << LcuSizeInBit
yInterval = ((((PicHeightInLuma+ (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)
filterIndex = alfCoeffIndexTab[regionTable[index]]
```

其中(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，regionTable 规定如下。

```
regionTable[16] = {0, 1, 4, 5, 15, 2, 3, 6, 14, 11, 10, 7, 13, 12, 9, 8}
```

如果 EalfEnableFlag 为 1，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 filterIndex）。

```
lcu_width = 1 << LcuSizeInBit
lcu_height = 1 << LcuSizeInBit
y_interval = ((((PicHeightInLuma + lcu_height - 1) / lcu_height) + 4) / 8 * lcu_height)
x_interval = ((((PicWidthInLuma + lcu_width - 1) / lcu_width) + 4) / 8 * lcu_width)
if (yInterval == 0)
    y_st_offset=0
else {
    y_cnt = Clip3(0, 8, (PicHeightInLuma + y_interval - 1) / y_interval)
```

T/AI 109.8-2025

```
        y_st_offset = PicHeightInLuma - y_interval * (y_cnt - 1)
        y_st_offset = (y_st_offset + lcu_height / 2) / lcu_height * lcu_height
}
if (xInterval == 0)
        x_st_offset=0
else {
        x_cnt = Clip3(0, 8, (PicWidthInLuma + x_interval - 1) / x_interval)
        x_st_offset = PicWidthInLuma - x_interval * (x_cnt - 1)
        x_st_offset = (x_st_offset + lcu_width / 2) / lcu_width * lcu_width
}
y_index = (y_interval == 0) ? 7 : Clip3(0, 7, y / y_interval)
y_index_offset = y_index <<3
y_index2 = (y_interval == 0 || y < y_st_offset) ? 0 : Clip3(-1, 6, (y - y_st_offset) / y_interval) + 1
y_index_offset2 = y_index2<<3
x_index = (x_interval == 0) ? 7: Clip3(0, 7, x / x_interval)
x_index2 = (x_interval == 0 || x < x_st_offset) ? 0 : Clip3(-1, 6, (x - x_st_offset) / x_interval) + 1
if (AlfRegionOrderIndex == 0)
        filterIndex = alfCoeffIndexTab[regionTable[0][y_index_offset + x_index]]
else if (AlfRegionOrderIndex == 1)
        filterIndex = alfCoeffIndexTab[regionTable[1][ y_index_offset + x_index2]]
else if (AlfRegionOrderIndex == 2)
        filterIndex=alfCoeffIndexTab[regionTable[2][ y_index_offset2 + x_index2]]
else if (AlfRegionOrderIndex == 3)
        filterIndex= alfCoeffIndexTab[regionTable[3][ y_index_offset2 + x_index]]
```

其中(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，regionTable 规定如下。

```
regionTable[4][64]={
{63,60,59,58,5,4,3,0,62,61,56,57,6,7,2,1,49,50,55,54,9,8,13,14,48,51,52,53,10,11,12,15,47,46,33,32,31,30,17,16,44,45,3
        4,35,28,29,18,19,43,40,39,36,27,24,23,20,42,41,38,37,26,25,22,21}
{42,43,44,47,48,49,62,63,41,40,45,46,51,50,61,60,38,39,34,33,52,55,56,59,37,36,35,32,53,54,57,58,26,27,28,31,10,9,6,5
        ,25,24,29,30,11,8,7,4,22,23,18,17,12,13,2,3,21,20,19,16,15,14,1,0}
{21,22,25,26,37,38,41,42,20,23,24,27,36,39,40,43,19,18,29,28,35,34,45,44,16,17,30,31,32,33,46,47,15,12,11,10,53,52,5
        1,48,14,13,8,9,54,55,50,49,1,2,7,6,57,56,61,62,0,3,4,5,58,59,60,63}
{0,1,14,15,16,19,20,21,3,2,13,12,17,18,23,22,4,7,8,11,30,29,24,25,5,6,9,10,31,28,27,26,58,57,54,53,32,35,36,37,59,56,5
        5,52,33,34,39,38,60,61,50,51,46,45,40,41,63,62,49,48,47,44,43,42}}
```

8.13.5 自适应修正滤波操作

如果 AlfLcuEnableFlag[compIndex][LcuIndex]等于 1，则对 compIndex 分量进行自适应修正滤波；否则不进行自适应修正滤波。

当自适应修正滤波过程中用到的样本是自适应修正滤波单元内的样本时，直接使用该样本进行滤波；当自适应修正滤波过程中用到的样本不是自适应修正滤波单元内的样本时，按以下方式进行滤波：

T/AI 109.8-2025

a) 如果该样本在图像边界外，或在片边界外且 CplfEnableFlag 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。

获取样本(x+i, y+j)（其中，i,j=-3,-2,-1,0,1,2,3）的权限值。当样本(x+i, y+j)（其中，i,j=-3,-2,-1,0,1,2,3）是自适应修正滤波单元内的样本时，直接使用该样本对应的权限存储单元的权限值作为该样本的权限值；当样本(x+i, y+j)不是自适应修正滤波单元内的样本时，按以下方式获取该样本的权限值：

a) 如果该样本在图像边界外，或在片边界外且 CplfEnableFlag 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对应的权限存储单元的权限值作为该样本的权限值；

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对应的权限存储单元的权限值作为该样本的权限值；

c) 否则，直接使用该样本对应的权限存储单元的权限值作为该样本的权限值。

如果 EalfEnableFlag 等于 0，若存在样本(x+i,y+j)（其中，i,j=-3,-2,-1,0,1,2,3）的权限值大于(x,y)的权限值，则 p’(x,y)等于 p(x,y)，否则，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[filterIndex][8] * p(x, y)
for(j=0; j<8; j++)
    ptmp += AlfCoeffLuma[filterIndex][j] * (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 145。

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] * p(x, y)
for(j=0; j<8; j++)
    ptmp += AlfCoeffChroma[i][j] * (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 145。

表145 样本补偿滤波坐标偏移值

j 值	Hor[j]值	Ver[j]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

T/AI 109.8-2025

如果 EalfEnableFlag 等于 1，若存在样本(x+i, y+j)（其中，i, j=-3, -2, -1, 0, 1, 2, 3）的权限值大于(x, y)的权限值，则 p’(x, y)等于 p(x, y)，否则，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[filterIndex][14] * p(x, y)
offset = 1 << (AlfLumaShift[filterIndex] - 1)
for (j=0; j<14; j++)
    ptmp += AlfCoeffLuma[filterIndex][j] * (p(x-Hor[j], y-Ver[j]) + p(x+Hor[j], y+Ver[j])
ptmp = (ptmp + offset) >> AlfLumaShift[filterIndex]
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=0～13）见表 146。

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][14] * p(x, y)
offset = 1 << (AlfChromaShift[i] - 1)
for (j=0; j<14; j++)
    ptmp += AlfCoeffChroma[i][j] * (p(x-Hor[j], y-Ver[j]) + p(x+Hor[j], y+Ver[j])
ptmp = (ptmp + offset) >> AlfChromaShift[i]
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=0～13）见表 146。

表146 样本补偿滤波坐标偏移值

j 值	Hor[j]值	Ver[j]值
0	0	3
1	2	2
2	1	2
3	0	2
4	1	-2
5	2	-2
6	2	1
7	1	1
8	0	1
9	1	-1
10	2	-1
11	3	0
12	2	0
13	1	0

T/AI 109.8-2025

附 录 A
（规范性）
扫描表

扫描表 InvScanCoeffInBlk[7][7][pos][2]通过以下过程生成：

```
for (yIndex = 0; yIndex < 7; yIndex++) {
    ySize = 1 << (yIndex + 1)
    for (xIndex = 0; xIndex < 7; xIndex++) {
        xSize = 1 << (xIndex + 1)
        pos = 0
        numLine = xSize + ySize - 1
        InvScanCoeffInBlk[yIndex][xIndex][0][0] = 0
        InvScanCoeffInBlk[yIndex][xIndex][0][1] = 0
        pos++
        for (l = 1; l < numLine; l++) {
            if (l % 2) {
                x = Min(l, xSize - 1)
                y = Max(0, l - (xSize - 1))
                while (x >= 0 && y < ySize) {
                    InvScanCoeffInBlk[yIndex][xIndex][pos][0] = x
                    InvScanCoeffInBlk[yIndex][xIndex][pos][1] = y
                    pos++
                    x--
                    y++
                }
            }
            else {
                y = Min(l, ySize - 1)
                x = Max(0, l - (ySize - 1))
                while (y >= 0 && x < xSize) {
                    InvScanCoeffInBlk[yIndex][xIndex][pos][0] = x
                    InvScanCoeffInBlk[yIndex][xIndex][pos][1] = y
                    pos++
                    x++
                    y--
                }
            }
        }
    }
}
```

基于扫描区域的系数扫描方法的扫描表 ScanOrder 的初始化方法如下，其中输入为扫描区域宽度 xSize，扫描区域高度 ySize，系数矩阵宽度 width：

T/AI 109.8-2025

```
initScanOrder(scanOrder, xSize, ySize, width) {
    pos = 0
    numLine = xSize + ySize - 1
    /* starting point */
    scanOrder[pos] = 0
    pos++
    /* loop */
    for (l = 1; l < numLine; l++) {
        if (l % 2) { /* decreasing loop */
            x = Min(l, xSize - 1)
            y = Max(0, l - (xSize - 1))
            while (x >= 0 && y < ySize) {
                scanOrder[pos] = y * width + x
                pos++
                x--
                y++
            }
        }
        else { /* increasing loop */
            y = Min(l, ySize - 1)
            x = Max(0, l - (ySize - 1))
            while (y >= 0 && x < xSize) {
                scanOrder[pos] = y * width + x
                pos++
                x++
                y--
            }
        }
    }
}
```

扫描表TravScan[4][4][1024][2]通过以下过程生成：

```
for (i = 0; i < 4; i++) {
    for (j = 0; j < 4; j++) {
        sPos = 0;
        for (y = 0; y < (4 << j); y+=2) {
            for (x = 0; x < (4 << i); x++) {
                TravScan[i][j][sPos + x][0] = x
                TravScan[i][j][sPos + x][1] = y
                TravScan[i][j][sPos + 2 * (4 << i) - 1 - x][0] = x
                TravScan[i][j][sPos + 2 * (4 << i) - 1 - x][1] = y + 1
            }
            sPos += (2 * (4 << i))
```

T/AI 109.8-2025

```
        }
    }
}
```

T/AI 109.8-2025

附 录 B
（规范性）
智能分割信息语法及语义补充

B.1 语法描述

智能分割参数扩展描述的 analysis_data 语法定义见表 B.1。

表B.1 智能分割信息语法表

智能分割参数扩展定义	描述符
SvacEx_aiseg_info( ) {	
AISeg_mode	u(2)
if (AISeg_mode != 1) {	
AISeg_prob_enable	u(1)
AISeg_category_num	u(5)
for (i=0; i<AISeg_category_num; i++) {	
AISeg_category	u(8)
AISeg_num	u(8)
for (j=0; j<AISeg_num; j++) {	
if (AISeg_prob_enable==1) {	
AISeg_prob	u(8)
}	
if (AISeg_mode == 0) {	
AISeg_left	u(16)
AISeg_top	u(16)
AISeg_width	u(16)
AISeg_height	u(16)
}	
if (AISeg_mode == 2) {	
point_num_minus1	u(8)
base_point_x	u(16)
base_point_y	u(16)
for (k=0; k<point_num_minus1; k++) {	
delta_x	se(v)
delta_y	se(v)
}	
}	
}	

T/AI 109.8-2025

表 B.1 智能分割信息语法表（续）

智能分割参数扩展定义	描述符
}	
}	
if (AISeg_mode==1) {	
lag_picture_order_index	u(6)
map_class_num	u(8)
for (i=0; i<map_class_num; i++) {	
map_label[i]	u(8)
map_value[i]	u(8)
}	
map_width	u(16)
map_height	u(16)
map_Data	u(v)
}	
}	

B.2 语义描述

智能分割信息，是通过智能模型对图像中的关键目标进行识别和分割，得到 ROI 区域和 ROI 类别等信息，包括矩阵表示、点表示和图表示方案。智能分割信息矩形表示、点表示方案，是将分割信息以矩形、多边形的方式来进行编码。智能分割信息图表示方案，是将智能分割信息以图的形式，编码后存入对应扩展信息载体中。

智能分割图数据编码采用本文件规范，等价于小分辨率的灰度图复用本文件规范编码器进行编码，且其采用编码器的档次和级别不高于视频编码器的档次和级别，示例见附录 O。

智能分割信息编码方式标识 AISeg_mode

2 位无符号整数。表示当前图像的智能分割信息的编码方式，为 0 表示当前图像的编码方式为矩形框方式，为 1 表示当前图像的编码方式为图表示方式，为 2 表示当前图像的编码方式为点表示方式。

智能分割概率使能标志 AISeg_prob_enable

1 位无符号整数。表示当前图像的智能分割信息是否包含概率信息。为 0 表示不包含，为 1 表示包含。

智能分割目标类别数 AISeg_category_num

5 位无符号整数。表示当前图像的智能分割目标类别数。

智能分割类别信息 AISeg_category

8 位无符号整数。表示一个智能分割区域的类别信息。一般的，使用数值 0 表示背景类别。数值 1 至 6 规定类别见 GB25724-2017 中表 F.16。为满足智能分割业务需求，增加类别规定如下{1：人员，2：人脸，3：机动车，4：非机动车，5：物品，6：场景，7：动物，8：车牌，9-63：保留，64-255：自定义}。

智能分割目标数 AISeg_num

T/AI 109.8-2025

8 位无符号整数。表示当前图像的某类智能分割目标的个数。

智能分割概率值 AISeg_prob

8 位无符号整数。表示一个智能分割区域的类别概率值。值为 0 表示不生效，无概率值信息；其他值表示的概率值可由使用者自行定义，建议 1 表示概率值 1%，100 表示概率值 100%，n 表示概率值 n%。

智能分割矩形的左上角横坐标 AISeg_left

16 位无符号整数。表示一个智能分割矩形左上角在图像中的横坐标，默认精度单位为整像素。

智能分割矩形的左上角纵坐标 AISeg_top

16 位无符号整数。表示一个智能分割矩形左上角在图像中的纵坐标，默认精度单位为整像素。

智能分割矩形的宽度 AISeg_width

16 位无符号整数。表示一个智能分割矩形在图像中的宽度，默认精度单位为整像素。

智能分割矩形的长度 AISeg_height

16 位无符号整数。表示一个智能分割矩形在图像中的长度，默认精度单位为整像素。

智能分割多边形点个数 point_num_minus1

8 位无符号整数。表示一个智能分割多边形的点个数减 1。

智能分割多边形初始点横坐标值 base_point_x

16 位无符号整数。表示一个智能分割多边形的初始点的横坐标。

智能分割多边形初始点纵坐标值 base_point_y

16 位无符号整数。表示一个智能分割多边形的初始点的纵坐标。

智能分割多边形点横坐标差值 delta_x

表示当前图像第 i 个智能分割多边形的第 j 个点对第 j-1 个点的横坐标差值。

智能分割多边形点纵坐标差值 delta_y

表示当前图像第 i 个智能分割多边形的第 j 个点对第 j-1 个点的纵坐标差值。

智能分割图像素类别数量 map_class_num

8 位无符号整数。表示当前智能分割图的类别数量。

智能分割图像素标签 map_label[i]

8 位无符号整数。表示当前智能分割图的第 i 类的标签。

智能分割图像素标签映射值 map_value[i]

8 位无符号整数。表示当前智能分割图的第 i 类的标签 map_label[i]的映射值。

智能分割图滞后帧数 lag_picture_order_index

6 位无符号整数。表示当前视频编码图像的 POI 减去当前智能分割图编码图像对应的视频编码图像的 POI 的值。

智能分割图数据 map_Data

n 位无符号整数。表示当前智能分割图的编码后的数据。map_Data 可以提取出来，并用本文件规范解码器进行解码。$n = data_length * 8 - 48 - map_class_num * 16$。

map_Data 为 POI 等于当前编码图像的 POI 减去 lag_picture_order_index 的视频编码图像对应的智能分割图编码图像的访问单元。若该智能分割图编码图像对应的视频编码图像为随机访问点图像，那么该智能分割图编码图像也应为随机访问点图像。

智能分割图宽度 map_width

16 位无符号整数。表示当前智能分割图的宽度。

智能分割图高度 map_height

16 位无符号整数。表示当前智能分割图的高度。

T/AI 109.8-2025

附 录 C
（资料性）
子位流抽取方法

C.1 概述

本附录描述了子位流抽取的方法。本附录的输入为符合本文件规范的位流 inBitStream，一个目标空域层 targetLayer，一个目标最高时域层号 targetTId，输出为符合本文件规范的子位流 outBitStream。

C.2 构建目标空域层列表

首先，解析位流中的第一个 nal_unit_type 为 7 的 NAL 单元，得到 NumOfLayers，SpsIndependentLayerFlag[i]和 RefLayerId[i]（i=0～NumOfLayers-1）。

然后，构建目标空域层列表 targetLayerList，其中 targetLayerList[k]表示空域层列表中第 k 层的 LayerId。初始化空域层列表中层数 numTargetLayers 为 0，执行以下操作

```
for ( k = targetLayer; k >= 0; ) {
    targetLayerList[numTargetLayers] = k
    numTargetLayers += 1
    if (SpsIndependentLayerFlag[k] == 0)
        k = RefLayerId[k]
    else
        break
}
```

最后，以位流 inBitStream，空域层列表 targetLayerList，空域层列表中层数 numTargetLayers，目标最高时域层号 targetTId 作为输入，调用 C.3 子位流抽取方法，得到子位流 outBitStream。

C.3 子位流抽取方法

本过程的输入为一个符合本文件规范的位流 inBitStream，一个目标空域层列表 targetLayerList，空域层列表中层数 numTargetLayers，一个目标最高时域层号 targetTId。该过程的输出是一个符合本文件规范的子位流 outBitStream。

按照下列步骤获取符合条件的子位流：

a) 复制输入位流 inBitStream，得到子位流 outBitStream；
b) 移除 outBitStream 中所有 temporal_id 大于 targetTId 的 NAL 单元；
c) 移除 outBitStream 中所有 LayerId 的值不包含在 targetLayerList 中，且 nal_unit_type 不为 5，6，7，8，9，10，11，或 16 的 NAL 单元；
d) 移除 outBitStream 中所有 LayerId 的值不包含在 targetLayerList 中，且不包含 PayloadType 为 19，25，26 或 127 的 nal_unit_type 为 6 的 NAL 单元；
e) 如果 outBitStream 中存在包含 PayloadType 为 25 的 nal_unit_type 为 6 的 NAL 单元，更新该 NAL 单元中的分层 SEI 负载，将 sei_num_layers_minus1 设置为 numTargetLayers-1，并删

T/AI 109.8-2025

除不在 targetLayerList 中的 sei_layer_id；否则，在每个序列起始图像和随机访问点图像所在的访问单元的第一个编码片 NAl 单元之前，插入包含 PayloadType 为 25 的 nal_unit_type 为 6 的 NAL 单元，其中 sei_num_layers_minus1 设置为 numTargetLayers-1，并 sei_layer_id[i] 设置为 targetLayerList[numTargetLayers-1-i]（i=0～numTargetLayers-1）。

注：子位流 outBitStream 中的访问单元中可能存在不包含编码片 NAL 单元的层单元。

T/AI 109.8-2025

附 录 D
（资料性）
视频数据加解密方法示例

D.1 位流加密

下面描述位流加密方法。

步骤1：生成安全参数集NAL单元，配置加密方法。

产生安全参数集，安全参数集的作用域为RAS或知识图像。所有使用同一个安全参数集加密的NAL单元应属于该安全参数集作用域。

a) 根据当前安全参数集的作用域设置 sec_is_library_flag；
b) 设置 encryption_enable_flag 为 1（表示启动加密）；根据配置中的加密类型，设置 encryption_type；
c) 根据配置设置 encryption_unit_mode。根据 encryption_unit_mode 确定加密基本单元。如果 encryption_unit_mode 值为‘0’表示以 NAL 为单位进行加密；值为‘1’表示以访问单元为单位加密，将访问单元中所有 NAL 单元 RBSP 需加密的部分按位流顺序拼接在一起加密；值为‘2’表示以层单元为单位加密，将层单元中所有 NAL 单元 RBSP 需加密的部分按位流顺序拼接在一起加密。需要加密的 SEI 数据，不应与 PayloadType 等于 25、26 或 127 的负载封装在同一 NAL 单元中；
d) 根据配置设置 encryption_level_mode，如果需要对编码片 NAL 单元的仅部分 RBSP 数据加密，则设置其为 1，如果需要对编码片 NAL 单元和扩展数据 NAL 单元的仅部分 RBSP 数据加密，则设置其为 2，否则设置为 0；如果设置 encryption_level_mode 为 1 或 2，需要根据配置的加密字节长度设置 encryption_num_minus1 和 encryption_base_byte；
e) 设置 vek_flag 为 1，根据配置中的密钥加密类型，设置 vek_encryption_type。基于安全随机数产生 VEK，使用 vek_encryption_type 指定的算法使用 VKEK 采用 ECB 模式对 VEK 加密，得到 EVEK，将 EVEK 的数据长度减一和数据分别写入 evek_length_minus1 和 evek 中，并将加密 VEK 过程中使用的 VKEK 版本的数据长度减一和数据分别写入 vkek_version_length_minus1 和 vkek_version 中；VEK 和 VKEK 需根据业务安全要求做更新；
f) 设置 iv_flag 为 1，将当前配置的 IV 的数据长度减一和数据分别写入 iv_length_minus1 和 iv 中。

将安全参数集打包到安全参数集NAL单元中：

a) 设置安全参数集 NAL 单元的 encryption_idc，由于安全参数集 NAL 单元不加密，所以设置 encryption_idc 为 0；
b) 在图像序列参数集 NAL 单元前添加安全参数集 NAL 单元，插入到压缩视频位流中。

步骤2：获取对应同一安全参数集的待加密的NAL单元并对其加密。

针对该安全参数集作用域内需要加密的NAL单元：

a) 确定每一个需要加密的 NAL 单元的加密级别：如果 encryption_level_mode 为 0，则对 NAL 单元 RBSP 全加密；如果 encryption_level_mode 为 1，NAL 单元的 NAL 类型为编码片，则对其 RBSP 部分加密；如果 encryption_level_mode 为 1，NAL 单元的 NAL 类型不为编码片，则对其 RBSP 全加密；如果 encryption_level_mode 为 2，NAL 单元的 NAL 类型为编码片或扩展数据，则对其 RBSP 部分加密；如果 encryption_level_mode 为 2，NAL 单元的 NAL 类型不为编码片或扩展数据，则对其 RBSP 全加密；

T/AI 109.8-2025

b) 确定每一个需要加密的 NAL 单元的加密数据长度：对于每一个需要加密的 NAL 单元，根据 encryption_level_mode 取值和其 NAL 类型以及该 NAL 单元的 RBSP 长度计算该 NAL 单元的加密字节长度，该长度不超过 RBSP 数据长度减一。如果 NAL 单元的 RBSP 为全加密，加密数据长度为 RBSP 数据长度减一，如果 NAL 单元的 RBSP 为部分加密，对应加密数据长度由 encryption_num_minus1 和 encryption_base_byte 决定，如果该数据长度大于 RBSP 数据长度减一，则加密数据长度为 RBSP 数据长度减一；

c) 根据加密数据长度确定需加密的数据：如果加密基本单元中只有一个 NAL 单元，根据其加密字节长度获取该 NAL 单元的 RBSP 前加密数据长度个字节的数据用于加密，并保留 RBSP 剩余未加密部分数据；如果加密基本单元中有多个 NAL 单元，根据各个 NAL 单元的加密字节长度获取各个 NAL 单元的 RBSP 前加密数据长度个字节的数据，并保留每个 RBSP 剩余未加密部分数据，将这些 NAL 单元的 RBSP 加密部分数据按位流顺序拼接在一起用于加密，并记录每个 RBSP 加密部分数据长度；

d) 对于需加密的加密数据，采用安全参数集中 encryption_type 标记的加密算法使用 VEK 加密；加密时初始化 IV 为安全参数集中的 iv。

步骤3：得到加密后的NAL单元。

a) 加密后的数据恢复成 RBSP：如果加密基本单元中只有一个 NAL 单元，直接将加密后的数据与保留的 RBSP 剩余未加密部分数据拼接，得到加密后的 RBSP 数据；如果加密基本单元中有多个 NAL 单元，将加密后的数据按照记录的每个 RBSP 加密部分数据长度切分，得到每个加密部分 RBSP 加密后的部分 RBSP，依次与对应于相同 NAL 单元的 RBSP 剩余未加密部分数据拼接，得到加密后的各个 RBSP；

b) 得到加密后的 NAL 单元：将加密后的 RBSP 做添加防伪起始码处理，结合对应的 NAL 单元头，同时设置该 NAL 单元头的 encryption_idc 为 1，得到完整加密后的 NAL 单元。

D.2 位流解密

下面描述位流解密方法。

步骤 1：输入压缩视频位流，获取位流中安全参数集 NAL 单元。

获取 RAS 或知识图像的一个安全参数集 NAL 单元，从安全参数据集中获取到 encryption_enable_flag、encryption_unit_mode 、encryption_level_mode、encryption_type、vek_flag、iv_flag。如安全参数集中 encryption_enable_flag 为 0，则该安全参数集不支持对作用域内的 NAL 单元做加密，无需对这些 NAL 单元解密，可以直接解码。

a) 如果 encryption_type 为 0，使用算法 SM1 对加密位流解密；如果 encryption_type 为 1，使用算法 SM4 对加密位流解密；

b) 根据 encryption_unit_mode 确定解密基本单元。如果 encryption_unit_mode 值为‘0’表示以 NAL 为单位进行解密；值为‘1’表示以访问单元为单位解密，将访问单元中所有 NAL 单元 RBSP 需解密的部分按位流顺序拼接在一起解密；值为‘2’表示以层单元为单位解密，将层单元中所有 NAL 单元 RBSP 需解密的部分按位流顺序拼接在一起解密；

c) 如果 encryption_level_mode 为 1 或 2，获取安全参数集中的 encryption_num_minus1 和 encryption_base_byte，计算在部分加密情况下的加密字节长度；

d) 如果 vek_flag 为 1，获取当前安全参数集中的 evek_length_minus1 和 evek，得到 EVEK，获取当前安全参数集中的 vkek_version_length_minus1 和 vkek_version，基于 vkek_version 获取 VKEK，获取安全参数集中的 vek_encryption_type 标记的解密算法，使用该算法采用

T/AI 109.8-2025

VKEK 对 EVEK 解密得到 VEK；

e) 如果 iv_flag 为 1，获取安全参数集中的 iv_length_minus1 和 iv，得到 IV。

针对每个安全参数集，按照步骤 2、3 对对应的标记为加密的 NAL 单元解密。

步骤 2：针对一个安全参数集，对其对应的加密 NAL 单元解密。

以解密基本单元为单位，对一个或多个 NAL 单元解密。

a) 确定每一个需要解密的 NAL 单元的加密级别：如果 encryption_level_mode 为 0，则对 NAL 单元 RBSP 全部解密；如果 encryption_level_mode 为 1，NAL 单元的 NAL 类型为编码片，则对其 RBSP 部分解密；如果 encryption_level_mode 为 1，NAL 单元的 NAL 类型不为编码片，则对其 RBSP 全部解密；如果 encryption_level_mode 为 2，NAL 单元的 NAL 类型为编码片或扩展数据，则对其 RBSP 部分解密；如果 encryption_level_mode 为 2，NAL 单元的 NAL 类型不为编码片或扩展数据，则对其 RBSP 全部解密；

b) 确定每一个需要解密的 NAL 单元的加密数据长度：对于每一个需要解密的 NAL 单元，根据 encryption_level_mode 取值和其 NAL 类型以及该 NAL 单元的 RBSP 长度计算该 NAL 单元的加密字节长度，该长度不超过 RBSP 数据长度减一。如果 NAL 单元的 RBSP 为全部解密，加密数据长度为 RBSP 数据长度减一，如果 NAL 单元的 RBSP 为部分解密，对应加密数据长度由 encryption_num_minus1 和 encryption_base_byte 决定，如果该数据长度大于 RBSP 数据长度减一，则加密数据长度为 RBSP 数据长度减一；

c) 根据加密数据长度确定需解密的数据：如果解密基本单元中只有一个 NAL 单元，根据其加密字节长度获取该 NAL 单元的 RBSP 前加密数据长度个字节的数据用于解密，并保留 RBSP 剩余未加密部分数据；如果解密基本单元中有多个 NAL 单元，根据各个 NAL 单元的加密字节长度获取各个 NAL 单元的 RBSP 前加密数据长度个字节的数据，并保留每个 RBSP 剩余未加密部分数据，将这些 NAL 单元的 RBSP 加密部分数据按位流顺序拼接在一起用于解密，并记录每个 RBSP 加密部分数据长度；

d) 对加密数据解密：用安全参数集中 encryption_type 标记的算法和 VEK 对需解密的数据解密，得到解密后的数据，解密时采用的初始化 IV 为从安全参数集得到的 IV。

步骤 3：得到解密后的 NAL 单元。

a) 解密后的数据恢复成 RBSP：如果解密基本单元中只有一个 NAL 单元，直接将解密后的数据与保留的 RBSP 剩余未加密部分数据拼接，得到解密后的 RBSP 数据；如果解密基本单元中有多个 NAL 单元，将解密后的数据按照记录的每个 RBSP 加密部分数据长度切分，得到每个加密部分 RBSP 解密后的部分 RBSP，依次与对应于相同 NAL 单元的 RBSP 剩余未加密部分数据拼接，得到解密后的各个 RBSP；

b) 恢复出加密前的 NAL 单元：将解密后的 RBSP 做添加防伪起始码处理，结合对应的 NAL 单元头，得到完整解密后的 NAL 单元。

T/AI 109.8-2025

附 录 E
（资料性）
视频数据认证方法示例

E.1 显示图像签名

下面描述显示图像签名方法。

步骤1：生成安全参数集NAL单元，并将其插入压缩视频位流中。

产生安全参数集，安全参数集的作用域为单个 RAS，参与认证的所有访问单元不能跨 RAS，不能处于两个 RAS 中：

a) 设置 sec_is_library_flag 为 0、authentication_enable_flag 为 1（表示启动认证）、hash_type 为 0（使用 SM3 算法），根据配置设置 authentication_hash_mode 值为 0 或 1（使用连接方式或树顶方式计算二次摘要）；

b) 根据配置设置 hash_discard_nrap_pictures_flag，如需要认证非随机访问点图像设置为 0，否则设置为 1；

c) 根据配置的哈希周期HashPeriodInDoi，设置hash_period_in_doi_minus1为HashPeriodInDoi减 1；HashPeriodInDoi 为 1 表示对单个访问单元独立签名，HashPeriodInDoi 大于 1 表示对多个访问单元共同签名。

将安全参数集打包到安全参数集 NAL 单元中：

a) 设置安全参数集 NAL 单元的 authentication_idc 为 1；

b) 设置安全参数集 NAL 单元的 authentication_data_id，设置为 0 或 1，建议和上一组参与认证的 NAL 单元的 authentication_data_id 不同；

c) 在图像序列参数集 NAL 单元前添加安全参数集 NAL 单元，插入到压缩视频位流中。

步骤 2：计算 RAS 中参与签名的显示图像对应该安全参数集的摘要数据。

在计算一个访问单元的摘要时，针对访问单元中需要认证的层单元的NAL单元（包括安全参数集NAL单元（如存在）、图像序列参数集NAL单元（如存在）、图像参数集NAL单元（如存在）、图像头NAL单元、显示图像分层编码片NAL单元、扩展数据NAL单元（如存在）、补充增强信息NAL单元（如存在）等）：

a) 将这些 NAL 单元的头信息中的 authentication_idc 设置为 1；

b) 设置当前 NAL 单元的 authentication_data_id，如果存在上一组 authentication_data_id 相同的参与认证的 NAL 单元，该 authentication_data_id 应与上一组的 authentication_data_id 不同；否则，如果该 NAL 单元与安全参数集 NAL 单元一起参与认证，则 authentication_data_id 还需要和安全参数集 NAL 单元的 authentication_data_id 一致；否则，设置为 0 或 1；

c) 然后将该层单元中所有参与认证的 NAL 单元拼接在一起计算该层单元的摘要。

根据配置从编码器输出的压缩视频位流中取出访问单元，按照哈希周期hash_period_in_doi_minus1以及随机访问点图像的DOI确定一起共同签名认证的一组访问单元（见哈希周期hash_period_in_doi_minus1定义），示例如下：

如图E.1所示，在hash_period_in_doi_minus1为3、随机访问点图像的DOI为0的情况下，上面一个RAS中，在DOI连续的情况下，访问单元DOI范围为[0～3]、[4～7]、[8]的三组访问单元分别做签名认证；下面一个RAS中，DOI不连续，访问单元DOI范围为[0～3]、[5～7]、[9～11]、[12～14]的四组访问单元分别做签名认证。

T/AI 109.8-2025

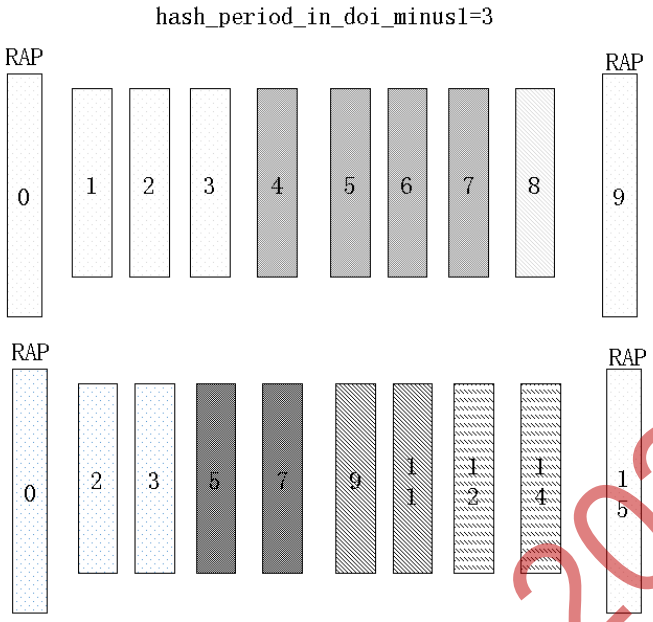


图 E.1 共同签名访问单元

依位流次序计算每个访问单元中每个层单元的摘要H1，H2，…，Hn，n等于所有访问单元中参与认证的层单元的总个数。认证数据作用范围不应跨RAS，所以每个RAS中最后一组一起认证的访问单元数量可能小于HashPeriodInDoi。

基于各个摘要值按照authentication_hash_mode指定的方式计算二次摘要。

步骤3：对二次摘要值做数字签名。

步骤4：生成显示图像对应安全参数集的认证数据NAL单元，并添加到压缩视频位流中。

产生认证数据：

a) 设置 for_current_ras_idc 为 0，auth_is_library_flag 为 0；

b) 设置当前认证数据的 authentication_data_id，该 authentication_data_id 和此次参与认证的 NAL 单元的 authentication_data_id 保持一致；

c) 根据配置设置 authentication_hash_list_flag，0 表示认证数据不含摘要列表，1 表示认证数据包含摘要列表。在 authentication_hash_list_flag 为 1 时，将 authentication_hash_number_minus1 设置为摘要的数量减 1，authentication_hash 为显示图像各摘要值{H1，H2，…，Hn}；

d) 将签名数据写入 authentication_data，将 authentication_data_length_minus1 设置为 authentication_data 的实际长度减 1。

将认证数据打包到认证数据NAL单元中，然后将认证数据NAL单元插入到此次认证的最后一个访问单元中或之后、下一个认证数据NAL单元之前及下下一个不是显示知识图像的随机访问点访问单元之前。设置认证数据NAL单元头的temporal_id与layer_id为0。每个RAS中的最后一个认证数据允许放入下一个RAS中，此情况下，设置认证数据中的for_current_ras_idc为0。认证数据和最后一个参与认证的访问单元中的层单元间隔不能超过HashPeriodInDoi（等于此次认证对应的安全参数集中的hash_period_in_doi_minus1+1）个访问单元（不含非显示知识图像访问单元）。认证数据NAL单元的authentication_idc设置为1，authentication_data_id设置为参与认证的层单元中的NAL单元头中的authentication_data_id。

T/AI 109.8-2025

显示图像的认证数据可位于最后一个显示图像编码片所在的访问单元中；也可位于下一个 RAS 的显示图像编码片的访问单元中。认证数据 NAL 单元应位于访问单元中除流结尾 NAL 单元、编码视频序列结尾 NAL 单元之外的所有其它类型的 NAL 单元之后。

E.2 显示图像认证

下面描述显示图像的认证方法。

步骤1：输入压缩视频位流，获取位流中安全参数集NAL单元。

获取RAS的一个安全参数集NAL单元，从安全参数据集中获取到authentication_enable_flag、authentication_data_id 、 hash_type 、 authentication_hash_mode 、 hash_discard_nrap_pictures_flag、hash_period_in_doi_minus1。如安全参数集中authentication_enable_flag为0，则该安全参数集不支持对显示图像做认证。如安全参数集中hash_discard_nrap_pictures_flag为1，则该安全参数集不支持对非随机访问点图像做认证。

针对每个安全参数集，按照步骤2、3对对应的参与认证的层单元做认证。

步骤2：针对一个安全参数集获取RAS中的显示图像访问单元，计算显示图像的摘要数据。

计算摘要：

接收访问单元的NAL数据，获取访问单元的DOI，按照哈希周期hash_period_in_doi_minus1以及随机访问点图像的DOI确定一起共同签名认证的一组访问单元（见哈希周期hash_period_in_doi_minus1定义），将访问单元中的参与认证的层单元的NAL单元拼接在一起计算该层单元的摘要。RAS中最后一组一起参与认证的数据的数量可能小于hash_period_in_doi_minus1+1。

按照位流顺序基于各个摘要值计算二次摘要，将二次摘要值以authentication_data_id为标识存入到本地缓存；

将连续的authentication_data_id一致的多个层单元的摘要以authentication_data_id为标识、按照位流次序生成摘要列表{H1’, H2’, …, Hm’}存入本地缓存，m等于访问单元中参与认证的层单元的总个数。如之前有未认证过的以authentication_data_id为标识的摘要列表，则生成此摘要列表的层单元认证失败，使用新的摘要列表覆盖老的摘要列表。

步骤3：获取显示图像认证数据，完成显示图像的认证。

从参与认证的最后一个显示图像编码片所在的访问单元开始计算到最大hash_period_in_doi_minus1+1个访问单元，获取认证数据。

认证数据中for_current_ras_idc为1时，表示认证数据为当前RAS的认证数据；for_current_ras_idc为0，表示认证数据为上一个RAS的最后一个认证数据。

认证数据中authentication_hash_list_flag为0时，使用二次摘要值认证：

根据认证数据中的authentication_data_id在本地缓存查找二次摘要值，如找到二次摘要值，用二次摘要值验证认证数据中的数字签名；如签名验证成功，则参与生成二次摘要值的层单元认证成功，如验证失败，则参与生成二次摘要值的层单元认证失败。如果找到且authentication_data_id和最新接收到的层单元序列的authentication_data_id一致，此时如还有其他标识不等于authentication_data_id的没有认证过的二次摘要值，则没有认证过的二次摘要值对应的认证数据丢失，这些没有认证过的二次摘要值对应的层单元认证失败。如果没有找到authentication_data_id对应的层单元的二次摘要值，则该认证数据无效，认证失败。

当 authentication_hash_list_flag 为 1 时，使用摘要列表认证：

a) 从认证数据 NAL 单元中解析得到 authentication_data_id 以及对应的摘要列表{H1, H2, …, Hn}，计算二次摘要；如果 authentication_hash_mode 为 0，使用连接方式做二次摘要计算；

T/AI 109.8-2025

如果 authentication_hash_mode 为 1，使用树顶方式做二次摘要计算；

b) 使用该二次摘要值验证从认证数据 NAL 单元中解析得到的签名数据，判断认证数据 NAL 单元中传输的摘要列表{H1, H2, …, Hn}是否通过验证。如果没有通过验证，则该认证数据中的摘要列表数据不可信，摘要列表认证失败；

c) 根据认证数据 NAL 单元中的参数确定参与签名的层单元。根据 authentication_data_id 查找本地缓存的层单元的摘要列表。如果找到，通过摘要列表可确认需要认证的层单元。如找到且 authentication_data_id 和最新接收到的层单元序列的 authentication_data_id 一致，此时如还有其他标识不等于 authentication_data_id 的没有认证过的摘要列表，则没有认证过的摘要列表对应的认证数据丢失，这些没有认证过的摘要列表对应的层单元认证失败。如果没有找到 authentication_data_id 对应的层单元的摘要列表，则该认证数据无效，认证失败；

d) 顺序对层单元的摘要列表{H1’, H2’, …, Hm’}和认证数据中的摘要列表{H1, H2, …, Hn}做匹配，实现对层单元的认证。先在认证数据中的摘要列表中查找 H1’，查找成功后记录认证数据中的摘要列表的位置，H1’对应的层单元认证成功；然后在认证数据中的摘要列表中从该位置的下一个位置开始查找 H2’， 如查找成功则更新认证数据中的摘要列表的位置，H2’对应的层单元认证成功；继续对后续 H3’,…, Hm’做查找，查找成功则对应的层单元认证成功，如查找失败则对应的层单元认证失败。

E.3 知识图像签名

安全参数集sec_is_library_flag为1指示对知识图像做独立签名认证。每个知识图像，包括显示知识图像与非显示知识图像，应独立签名认证。

操作方法如下。

步骤1：生成安全参数集NAL单元，并将其插入压缩视频位流中。

产生安全参数集，安全参数集的作用域为单个知识图像，参与认证的知识图像不能跨RAS，不能处于两个RAS中：

a) 设置 sec_is_library_flag 为 1、authentication_enable_flag 为 1（表示启动认证）、hash_type 为 0（使用 SM3 算法）、根据配置设置 authentication_hash_mode 值为 0 或 1（使用连接方式或树顶方式计算二次摘要）；

b) 根据配置设置 hash_discard_nrap_pictures_flag 设置为 0 或 1，无作用；

c) 根据配置的哈希周期HashPeriodInDoi,设置hash_period_in_doi_minus1为HashPeriodInDoi减 1；因为知识图像没有 DOI，一个知识图像针对一个安全参数集只产生一个认证数据。

将安全参数集打包到安全参数集 NAL 单元中：

a) 设置安全参数集 NAL 单元的 authentication_idc 为 1；

b) 设置安全参数集 NAL 单元的 authentication_data_id，设置为 0 或 1，建议和上一个组参与认证的 NAL 单元的 authentication_data_id 不同；

c) 在图像序列参数集 NAL 单元前添加安全参数集 NAL 单元，插入到压缩视频位流中。

步骤2：计算知识图像对应该安全参数集的摘要数据。

在计算一个知识图像访问单元的摘要时，针对知识图像访问单元中需要认证的层单元NAL单元（包括安全参数集NAL单元（如存在）、图像序列参数集NAL单元（如存在）、图像参数集NAL单元（如存在）、图像头NAL单元、知识图像编码片NAL单元、扩展数据NAL单元（如存在）、补充增强信息NAL单元（如存在）等）：

a) 将这些 NAL 单元的头信息中的 authentication_idc 设置为 1；

T/AI 109.8-2025

b) 设置当前NAL单元的authentication_data_id，如果存在上一个组authentication_data_id相同的参与认证的NAL单元，该authentication_data_id应与上一组的authentication_data_id不同；否则，如果该NAL单元与安全参数集NAL单元一起参与认证，则authentication_data_id还需要和安全参数集NAL单元的authentication_data_id一致；否则，设置为0或1；

c) 然后将该层单元中所有参与认证的NAL单元拼接在一起计算该层单元的摘要。

若知识图像为显示知识图像时，该知识图像的编码片NAL单元包含在单个访问单元中，按位流顺序计算该访问单元的参与认证的层单元的摘要数据H1，H2，…，Hn，n为知识图像访问单元中层单元的总个数；

若知识图像为非显示知识图像时，该知识图像的编码片NAL单元可包含在多个访问单元中，对这多个访问单元的参与认证的层单元按位流顺序做摘要H1，H2，…，Hn，n为知识图像访问单元中层单元的总个数。

基于各个摘要值H1，H2，…，Hn按照authentication_hash_mode指定的方式计算二次摘要。

步骤3：使用二次摘要值做数字签名。

步骤4：生成该知识图像对应该安全参数集的认证数据NAL单元，并添加到压缩视频位流中。

产生认证数据：

a) 设置auth_is_library_flag为1、authenticaion_library_picture_index为该知识图像的library_picture_index；

b) 设置当前认证数据的authentication_data_id，该authentication_data_id和此次参与认证的NAL单元的authentication_data_id保持一致；

c) 根据配置设置authentication_hash_list_flag，0表示认证数据不含摘要列表，1表示认证数据包含摘要列表。在authentication_hash_list_flag为1时，设置authentication_hash_number_minus1为层单元摘要的数量减1，authentication_hash为层单元摘要的数据值。

将签名数据写入authentication_data，并根据authentication_data的实际长度减1设置authentication_data_length_minus1。

将认证数据打包到认证数据NAL单元中，然后将认证数据NAL单元插入到此次认证的最后一个访问单元中或之后、下一个认证数据NAL单元之前及下下一个随机访问点访问单元之前。设置认证数据NAL单元头的temporal_id与layer_id为0。认证数据NAL单元的authentication_idc设置为1，authentication_data_id设置为参与认证的层单元中的NAL单元头中的authentication_data_id。

显示知识图像的认证数据和显示知识图像访问单元间隔应不超过HashPeriodInDoi（等于此次认证对应的安全参数集中的hash_period_in_doi_minus1+1）个访问单元。显示知识图像的认证数据可位于最后一个显示知识图像编码片所在的访问单元中；也可位于下一个RAS的显示图像编码片的访问单元中。

非显示知识图像的认证数据位于最后一个非显示知识图像编码片所在的访问单元中。

认证数据NAL单元应位于访问单元中除流结尾NAL单元、编码视频序列结尾NAL单元之外的所有其它类型的NAL单元之后。

E.4 知识图像认证

每个知识图像，包括显示知识图像与非显示知识图像，需要独立完成认证。操作方法如下。

步骤1：输入压缩视频位流，获取位流中安全参数集NAL单元。

T/AI 109.8-2025

获取到知识图像的一个安全参数集NAL单元，获取sec_is_library_flag、authentication_enable_flag、authentication_data_id、hash_type、authentication_hash_mode、hash_period_in_doi_minus1。如安全参数集中的authentication_enable_flag为0或hash_discard_library_pictures_flag为1，该安全参数集不支持对知识图像做独立签名认证。

针对每个安全参数集，按照步骤2、3对对应的参与认证的层单元做认证。

步骤2：针对一个安全参数集获取知识图像访问单元，计算知识图像的摘要数据。

若知识图像为显示知识图像时，该知识图像的编码片NAL单元包含在单个访问单元中，按照位流顺序计算该访问单元的参与认证的层单元的摘要数据H1’，H2’，…，Hm’，m为知识图像访问单元中层单元的总个数；

若知识图像为非显示知识图像时，该知识图像的编码片NAL单元可包含在多个访问单元中，对这多个访问单元的参与认证的层单元按位流顺序做摘要H1’，H2’，…，Hm’，m为知识图像访问单元中层单元的总个数。

基于各个摘要值H1’，H2’，…，Hm’计算二次摘要，将二次摘要值以authentication_data_id为标识存入到本地缓存。如果authentication_hash_mode为0，使用连接方式做二次摘要计算；如果authentication_hash_mode为1，使用树顶方式做二次摘要计算。

以authentication_data_id为标识生成摘要列表{H1’，H2’，…，Hm’}存入本地缓存。

步骤3：获取知识图像认证数据，完成知识图像的认证。

如果认证的是显示知识图像，从参与认证的最后一个显示知识图像编码片所在的层单元开始计算到最大hash_period_in_doi_minus1+1个访问单元，接收认证数据；如果认证的是非显示知识图像，在最后一个非显示知识图像编码片所在的访问单元中、layer_id为0的层单元中的非显示知识图像编码片之后获取认证数据。

如auth_is_library_flag为1时，为知识图像的认证数据。

认证数据中for_current_ras_idc为1时，表示认证数据为当前RAS的认证数据；for_current_ras_idc为0，表示认证数据为上一个RAS的最后一个认证数据。

判断认证数据中的authenticaion_library_picture_index和当前知识图像的library_picture_index是否一致，如不一致，则认证失败，结束认证。

认证数据中authentication_hash_list_flag为0时，使用二次摘要值认证：

根据认证数据中的authentication_data_id在本地缓存查找二次摘要值，如找到二次摘要值，用二次摘要值验证认证数据中的数字签名；如签名验证成功，则参与生成二次摘要值的层单元认证成功，如验证失败，则参与生成二次摘要值的层单元认证失败。如果找到且authentication_data_id和最新接收到的层单元序列的authentication_data_id一致，此时如还有其他标识不等于authentication_data_id的没有认证过的二次摘要值，则没有认证过的二次摘要值对应的认证数据丢失，这些没有认证过的二次摘要值对应的层单元认证失败。如果没有找到authentication_data_id对应的层单元的二次摘要值，则该认证数据无效，认证失败。

当authentication_hash_list_flag为1时，使用摘要列表认证：

a) 从认证数据NAL单元中解析得到authentication_data_id以及对应的摘要列表{H1，H2，…，Hn}，计算各摘要值的二次摘要；如果authentication_hash_mode为0，使用连接方式做二次摘要计算；如果authentication_hash_mode为1，使用树顶方式做二次摘要计算；

b) 使用该二次摘要值验证从认证数据NAL单元中解析得到的签名数据，判断认证数据NAL单元中传输的摘要列表{H1，H2，…，Hn}是否通过验证。如果没有通过验证，则该认证数据中的摘要列表数据不可信，摘要列表认证失败；

c) 根据认证数据NAL单元中的参数确定参与签名的层单元。根据authentication_data_id查找

T/AI 109.8-2025

本地缓存的层单元的摘要列表。如果找到，通过摘要列表可确认需要认证的层单元。如找到且authentication_data_id 和最新接收到的层单元序列的 authentication_data_id 一致，此时如还有其他标识不等于 authentication_data_id 的没有认证过的摘要列表，则没有认证过的摘要列表对应的认证数据丢失，这些没有认证过的摘要列表对应的层单元认证失败。如果没有找到 authentication_data_id 对应的层单元的摘要列表，则该认证数据无效，认证失败；

d) 顺序对层单元的摘要列表{H1’，H2’，…，Hm’}和认证数据中的摘要列表{H1，H2，…，Hn}做匹配，实现对层单元的认证。先在认证数据中的摘要列表中查找 H1’，查找成功后记录认证数据中的摘要列表的位置，H1’对应的层单元认证成功；然后在认证数据中的摘要列表中从该位置的下一个位置开始查找 H2’，如查找成功则更新认证数据中的摘要列表的位置，H2’对应的层单元认证成功；继续对后续 H3’，…，Hm’做查找，查找成功则对应的层单元认证成功，如查找失败则对应的层单元认证失败。

T/AI 109.8-2025

附 录 F
（规范性）
视频档次和级别

F.1 概述

视频档次和级别提供了一种规定本文件的语法和语义的子集的手段。视频档次和级别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。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规定的子集。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。如果一个位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，则认为此位流在这个档次和级别上符合本文件。

profile_id 和 level_id 规定了位流的档次和级别。

F.2 档次

本文件规定的视频档次见表 F.1。

表F.1 视频档次规定

profile_id 值	档次
0x00	禁止
0x30	主要档次（Mainprofile）
0x40	高级 8 位档次（High 8bit profile）
0x42	高级 10 位档次（High 10bit 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。

主要档次的位流应满足以下条件：

——profile_id 的值应为 0x30。

——chroma_format 的值应为‘01’。

——sample_precision 的值应为‘001’。

——编码视频序列起始图像与随后第一个编码视频序列结尾 NAL 单元之间，或编码视频序列起始图像与下一个编码视频序列起始图像之间所有编码图像的 PictureStructure 的值均应相同。

——MiniSize 的值应为 8。

——log2_lcu_size_minus2 的取值范围应为 3～5。

——log2_min_cu_size_minus2 的值应为 0。

——stable_patch_flag 的值应为‘1’。

——ref_colocated_patch_flag 的值应为‘0’。

T/AI 109.8-2025

——uniform_patch_flag 的值应为‘1’。
——属于同一图像的片应按 patch_index 的值从小到大依次出现。
——如果编码块的任一边长大于 64，该编码块只应选择跳过模式或帧间预测模式。如果选择帧间预测模式，ctp_zero_flag 的值应为‘1’。
——如果当前编码单元的亮度样本的数量小于 64，当前编码单元不应使用跳过模式和直接模式，且当前编码单元的预测单元的预测参考模式不应为‘PRED_List01’。
——MaxSplitTimes 的值应为 6。
——MaxPartRatio 的值应为 8。
——如果 I 图像的最大编码单元的宽度和高度均等于 128，则该最大编码单元应使用四叉树划分且 qt_split_flag 的值应为‘1’。
——如果编码树节点的宽度等于 128 且高度等于 64 且该编码树节点被划分，则该编码树节点应使用垂直二叉树划分且 bet_split_dir_flag 的值应为‘1’。
——如果编码树节点的宽度等于 64 且高度等于 128 且该编码树节点被划分，则该编码树节点应使用水平二叉树划分且 bet_split_dir_flag 的值应为‘0’。
——如果当前序列的 library_stream_flag 为‘1’，则该序列的所有图像均应为 I 图像。
——如果位流中存在语法元素 duplicate_sequence_header_flag，则该语法元素的值应为‘1’。
——解码图像缓冲区中最多只应存在 1 幅知识图像。
——library_picture_mode_index 的取值范围为 0~1。
——NumOfUpdatedLibrary 的值应为 1。
——MinLibraryUpdatePeriod 的值应为 1。
——num_of_nn_filter_minus1 的取值应为 0~3。
——MAX_TEMPORAL_ID 的值应为 4。
——MAX_LAYER 的值应为 4。
——MAX_HASH_PERIDO 的值应为 256。
——low_delay 的值应为 1。
——asr_enable_flag 的值应为 0。
——smvd_enable_flag 的值应为 0。
——emvr_enable_flag 的值应为 0。
——obmc_enable_flag 的值应为 0。
——ibc_enable_flag 的值应为 0。
——isc_enable_flag 的值应为 0。
——fimc_enable_flag 的值应为 0。
——bio_enable_flag 的值应为 0。
——dmvr_enable_flag 的值应为 0。
——pmc_enable_flag 的值应为 0。
——nn_tools_set_hook 的值应为‘00000000’。
——layer_id 的值应为 0。
——num_of_layers_minus1 的值应为 0。
——ssvc_enable_flag 的值应为 0。
——privacy_enable_flag 的值应为 0。
——reference_picture_resampling_enable_flag 的值应为 0。
——F.3.2 规定的级别限制。

T/AI 109.8-2025

——支持的级别包括：2.0.15、2.0.30、2.0.60、4.0.30、4.0.60、6.0.30、6.2.30、6.4.30、6.6.30、6.0.60、6.2.60、6.4.60、6.6.60、6.0.120、6.2.120、6.4.120、6.6.120、8.0.30、8.2.30、8.4.30、8.6.30、8.0.60、8.2.60、8.4.60、8.6.60、8.0.120、8.2.120、8.4.120、8.6.120、10.0.30、10.2.30、10.4.30、10.6.30、10.0.60、10.2.60、10.4.60、10.6.60、10.0.120、10.2.120、10.4.120 和 10.6.120。

高级 8 位档次的位流应满足以下条件：

——profile_id 的值应为 0x40。

——chroma_format 的值应为‘01’。

——sample_precision 的值应为‘001’。

——编码视频序列起始图像与随后第一个编码视频序列结尾 NAL 单元之间，或编码视频序列起始图像与下一个编码视频序列起始图像之间所有编码图像的 PictureStructure 的值均应相同。

——MiniSize 的值应为 8。

——log2_lcu_size_minus2 的取值范围应为 3～5。

——log2_min_cu_size_minus2 的值应为 0。

——stable_patch_flag 的值应为‘1’。

——ref_colocated_patch_flag 的值应为‘0’。

——uniform_patch_flag 的值应为‘1’。

——属于同一图像的片应按 patch_index 的值从小到大依次出现。

——如果编码块的任一边长大于 64，该编码块只应选择跳过模式或帧间预测模式。如果选择帧间预测模式，ctp_zero_flag 的值应为‘1’。

——如果当前编码单元的亮度样本的数量小于 64，当前编码单元不应使用跳过模式和直接模式，且当前编码单元的预测单元的预测参考模式不应为‘PRED_List01’。

——MaxSplitTimes 的值应为 6。

——MaxPartRatio 的值应为 8。

——如果 I 图像的最大编码单元的宽度和高度均等于 128，则该最大编码单元应使用四叉树划分且 qt_split_flag 的值应为‘1’。

——如果最大编码单元的宽度和高度均等于 128，并且该图像的 PictureRprFlag 的值为 1，则该最大编码单元应使用四叉树划分且 qt_split_flag 的值应为‘1’。

——如果编码树节点的宽度等于 128 且高度等于 64 且该编码树节点被划分，则该编码树节点应使用垂直二叉树划分且 bet_split_dir_flag 的值应为‘1’。

——如果编码树节点的宽度等于 64 且高度等于 128 且该编码树节点被划分，则该编码树节点应使用水平二叉树划分且 bet_split_dir_flag 的值应为‘0’。

——如果当前序列的 library_stream_flag 为‘1’，则该序列的所有图像均应为 I 图像。

——如果位流中存在语法元素 duplicate_sequence_header_flag，则该语法元素的值应为‘1’。

——解码图像缓冲区中最多只应存在 1 幅知识图像。

——library_picture_mode_index 的取值范围为 0～1。

——NumOfUpdatedLibrary 的值应为 1。

——MinLibraryUpdatePeriod 的值应为 1。

——num_of_nn_filter_minus1 的取值应为 0～3。

——MAX_TEMPORAL_ID 的值应为 4。

——MAX_LAYER 的值应为 4。

T/AI 109.8-2025

——MAX_HASH_PERIDO 的值应为 256。
——asr_enable_flag 的值应为 0。
——smvd_enable_flag 的值应为 0。
——emvr_enable_flag 的值应为 0。
——obmc_enable_flag 的值应为 0。
——ibc_enable_flag 的值应为 0。
——isc_enable_flag 的值应为 0。
——fimc_enable_flag 的值应为 0。
——bio_enable_flag 的值应为 0。
——dmvr_enable_flag 的值应为 0。
——pmc_enable_flag 的值应为 0。
——nn_tools_set_hook 的值应为‘00000000’。
——F.3.2 规定的级别限制。
——支持的级别包括：2.0.15、2.0.30、2.0.60、4.0.30、4.0.60、6.0.30、6.2.30、6.4.30、6.6.30、6.0.60、6.2.60、6.4.60、6.6.60、6.0.120、6.2.120、6.4.120、6.6.120、8.0.30、8.2.30、8.4.30、8.6.30、8.0.60、8.2.60、8.4.60、8.6.60、8.0.120、8.2.120、8.4.120、8.6.120、10.0.30、10.2.30、10.4.30、10.6.30、10.0.60、10.2.60、10.4.60、10.6.60、10.0.120、10.2.120、10.4.120 和 10.6.120。

高级 10 位档次的位流应满足以下条件：
——profile_id 的值应为 0x42。
——chroma_format 的值应为‘01’。
——sample_precision 的值应为‘001’或‘010’。
——encoding_precision 的值应为‘001’或‘010’。
——编码视频序列起始图像与随后第一个编码视频序列结尾 NAL 单元之间，或编码视频序列起始图像与下一个编码视频序列起始图像之间所有编码图像的 PictureStructure 的值均应相同。
——MiniSize 的值应为 8。
——log2_lcu_size_minus2 的取值范围应为 3～5。
——log2_min_cu_size_minus2 的值应为 0。
——stable_patch_flag 的值应为‘1’。
——ref_colocated_patch_flag 的值应为‘0’。
——uniform_patch_flag 的值应为‘1’。
——属于同一图像的片应按 patch_index 的值从小到大依次出现。
——如果编码块的任一边长大于 64，该编码块只应选择跳过模式或帧间预测模式。如果选择帧间预测模式，ctp_zero_flag 的值应为‘1’。
——如果当前编码单元的亮度样本的数量小于 64，当前编码单元不应使用跳过模式和直接模式，且当前编码单元的预测单元的预测参考模式不应为‘PRED_List01’。
——MaxSplitTimes 的值应为 6。
——MaxPartRatio 的值应为 8。
——如果 I 图像的最大编码单元的宽度和高度均等于 128，则该最大编码单元应使用四叉树划分且 qt_split_flag 的值应为‘1’。
——如果最大编码单元的宽度和高度均等于 128，并且该图像的 PictureRprFlag 的值为 1，则该最

T/AI 109.8-2025

大编码单元应使用四叉树划分且 qt_split_flag 的值应为‘1’。

——如果编码树节点的宽度等于 128 且高度等于 64 且该编码树节点被划分，则该编码树节点应使用垂直二叉树划分且 bet_split_dir_flag 的值应为‘1’。

——如果编码树节点的宽度等于 64 且高度等于 128 且该编码树节点被划分，则该编码树节点应使用水平二叉树划分且 bet_split_dir_flag 的值应为‘0’。

——如果当前序列的 library_stream_flag 为‘1’，则该序列的所有图像均应为 I 图像。

——如果位流中存在语法元素 duplicate_sequence_header_flag，则该语法元素的值应为‘1’。

——解码图像缓冲区中最多只应存在 1 幅知识图像。

——library_picture_mode_index 的取值范围为 0～1。

——NumOfUpdatedLibrary 的值应为 1。

——MinLibraryUpdatePeriod 的值应为 1。

——num_of_nn_filter_minus1 的取值应为 0～3。

——MAX_TEMPORAL_ID 的值应为 4。

——MAX_LAYER 的值应为 4。

——MAX_HASH_PERIDO 的值应为 256。

——asr_enable_flag 的值应为 0。

——smvd_enable_flag 的值应为 0。

——emvr_enable_flag 的值应为 0。

——obmc_enable_flag 的值应为 0。

——ibc_enable_flag 的值应为 0。

——isc_enable_flag 的值应为 0。

——fimc_enable_flag 的值应为 0。

——bio_enable_flag 的值应为 0。

——dmvr_enable_flag 的值应为 0。

——pmc_enable_flag 的值应为 0。

——nn_tools_set_hook 的值应为‘00000000’。

——F.3.2 规定的级别限制。

——支持的级别包括：2.0.15、2.0.30、2.0.60、4.0.30、4.0.60、6.0.30、6.2.30、6.4.30、6.6.30、6.0.60、6.2.60、6.4.60、6.6.60、6.0.120、6.2.120、6.4.120、6.6.120、8.0.30、8.2.30、8.4.30、8.6.30、8.0.60、8.2.60、8.4.60、8.6.60、8.0.120、8.2.120、8.4.120、8.6.120、10.0.30、10.2.30、10.4.30、10.6.30、10.0.60、10.2.60、10.4.60、10.6.60、10.0.120、10.2.120、10.4.120 和 10.6.120。

F.3 级别

F.3.1 本文件规定的级别

本文件规定的级别见表 F.2。

T/AI 109.8-2025

表F.2 级别

level_id 值	级别
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
0x41	6.4.30
0x43	6.6.30
0x44	6.0.60
0x46	6.2.60
0x45	6.4.60
0x47	6.6.60
0x48	6.0.120
0x4A	6.2.120
0x49	6.4.120
0x4B	6.6.120
0x50	8.0.30
0x52	8.2.30
0x51	8.4.30
0x53	8.6.30
0x54	8.0.60
0x56	8.2.60
0x55	8.4.60
0x57	8.6.60
0x58	8.0.120
0x5A	8.2.120
0x59	8.4.120
0x5B	8.6.120
0x60	10.0.30
0x62	10.2.30
0x61	10.4.30
0x63	10.6.30
0x64	10.0.60
0x66	10.2.60
0x65	10.4.60
0x67	10.6.60

T/AI 109.8-2025

表 F.3 级别（续）

level_id 值	级别
0x68	10.0.120
0x6A	10.2.120
0x69	10.4.120
0x6B	10.6.120
其他	保留

F.3.2 与档次无关的级别限制

对于所有档次，每个最大编码单元编码后最大二进制位数的限制见表 F.3。

表F.3 最大编码单元编码后最大二进制位数

图像格式	最大编码单元编码后最大二进制位数
4:2:0	2(LcuSizeInBit-1) + 2(LcuSizeInBit×2)×BitDepth×1.5

级别 2.0.15、2.0.30 和 2.0.60 的参数限制见表 F.4。

表F.4 级别 2.0.15、2.0.30 和 2.0.60 的参数限制

参 数	级别		
	2.0.15	2.0.30	2.0.60
每行最大样本数	352	352	352
每帧最大行数	288	288	288
每秒最大帧数	15	30	60
每帧最大片数	16	16	16
最大亮度样本速率（样本每秒）	1 520 640	3 041 280	6 082 560
最大位速率（位每秒）	1 500 000	2 000 000	2 500 000
每帧最大位数（位）	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（图像）	16	16	16

级别 4.0.30 和 4.0.60 的参数限制见表 F.5。

表F.5 级别 4.0.30 和 4.0.60 的参数限制

参 数	级 别	
	4.0.30	4.0.60
每行最大样本数	720	720
每帧最大行数	576	576
每秒最大帧数	30	60
每帧最大片数	32	32

T/AI 109.8-2025

表 F.5 级别 4.0.30 和 4.0.60 的参数限制（续）

参　数	级　别	
	4.0.30	4.0.60
最大亮度样本速率（样本每秒）	12 441 600	24 883 200
最大位速率（位每秒）	6 000 000	10 000 000
每帧最大位数（位）	2 500 000	2 500 000
BBV缓冲区大小（位）	6 012 928	10 010 624
解码图像缓冲区大小（图像）	16	16

级别 6.0.30 和 6.2.30 的参数限制见表 F.6。

表F.6　级别 6.0.30 和 6.2.30 的参数限制

参　数	级　别	
	6.0.30	6.2.30
每行最大样本数	2 048	2 048
每帧最大行数	1 152	1 152
每秒最大帧数	30	30
每帧最大片数	64	64
最大亮度样本速率（样本每秒）	66 846 720	66 846 720
最大位速率（位每秒）	12 000 000	30 000 000
每帧最大位数（位）	2 500 000	2 500 000
BBV缓冲区大小（位）	12 009 472	30 015 488
解码图像缓冲区大小（图像）	Min(14622720/(SpsPicWidthInLuma[NumOfLayers-1] × SpsPicHeightInLuma[NumOfLayers-1]),16)	

级别 6.4.30 和 6.6.30 的参数限制见表 F.7。

表F.7　级别 6.4.30 和 6.6.30 的参数限制

参　数	级　别	
	6.4.30	6.6.30
每行最大样本数	2 048	2 048
每帧最大行数	1 152	1 152
每秒最大帧数	30	30
每帧最大片数	64	64
最大亮度样本速率（样本每秒）	66 846 720	66 846 720
最大位速率（位每秒）	12 000 000	30 000 000
每帧最大位数（位）	2 500 000	2 500 000
BBV缓冲区大小（位）	12 009 472	30 015 488
解码图像缓冲区大小（图像）	Min(16711680/(SpsPicWidthInLuma[NumOfLayers-1] × SpsPicHeightInLuma[NumOfLayers-1]),16)	

T/AI 109.8-2025

级别 6.0.60 和 6.2.60 的参数限制见表 F.8。

表F.8 级别 6.0.60 和 6.2.60 的参数限制

参 数	级 别	
	6.0.60	6.2.60
每行最大样本数	2 048	2 048
每帧最大行数	1 152	1 152
每秒最大帧数	60	60
每帧最大片数	64	64
最大亮度样本速率（样本每秒）	133 693 440	133 693 440
最大位速率（位每秒）	20 000 000	50 000 000
每帧最大位数（位）	2 500 000	2 500 000
BBV缓冲区大小（位）	20 004 864	50 003 968
解码图像缓冲区大小（图像）	Min(14622720/(SpsPicWidthInLuma[NumOfLayers-1] × SpsPicHeightInLuma[NumOfLayers-1]),16)	

级别 6.4.60 和 6.6.60 的参数限制见表 F.9。

表F.9 级别 6.4.60 和 6.6.60 的参数限制

参 数	级 别	
	6.4.60	6.6.60
每行最大样本数	2 048	2 048
每帧最大行数	1 152	1 152
每秒最大帧数	60	60
每帧最大片数	64	64
最大亮度样本速率（样本每秒）	133 693 440	133 693 440
最大位速率（位每秒）	20 000 000	50 000 000
每帧最大位数（位）	2 500 000	2 500 000
BBV缓冲区大小（位）	20 004 864	50 003 968
解码图像缓冲区大小（图像）	Min(16711680/(SpsPicWidthInLuma[NumOfLayers-1] × SpsPicHeightInLuma[NumOfLayers-1]),16)	

级别 6.0.120 和 6.2.120 的参数限制见表 F.10。

T/AI 109.8-2025

表F.10 级别 6.0.120 和 6.2.120 的参数限制

参 数	级 别	
	6.0.120	6.2.120
每行最大样本数	2 048	2 048
每帧最大行数	1 152	1 152
每秒最大帧数	120	120
每帧最大片数	64	64
最大亮度样本速率（样本每秒）	267 386 880	267 386 880
最大位速率（位每秒）	25 000 000	100 000 000
每帧最大位数（位）	2 500 000	2 500 000
BBV缓冲区大小（位）	25 001 984	100 007 936
解码图像缓冲区大小（图像）	Min(14622720/(SpsPicWidthInLuma[NumOfLayers-1] × SpsPicHeightInLuma[NumOfLayers-1]),16)	

级别 6.4.120 和 6.6.120 的参数限制见表 F.11。

表F.11 级别 6.4.120 和 6.6.120 的参数限制

参 数	级 别	
	6.4.120	6.6.120
每行最大样本数	2 048	2 048
每帧最大行数	1 152	1 152
每秒最大帧数	120	120
每帧最大片数	64	64
最大亮度样本速率（样本每秒）	267 386 880	267 386 880
最大位速率（位每秒）	25 000 000	100 000 000
每帧最大位数（位）	2 500 000	2 500 000
BBV缓冲区大小（位）	25 001 984	100 007 936
解码图像缓冲区大小（图像）	Min(16711680/(SpsPicWidthInLuma[NumOfLayers-1] × SpsPicHeightInLuma[NumOfLayers-1]),16)	

级别 8.0.30 和 8.2.30 的参数限制见表 F.12。

表F.12 级别 8.0.30 和 8.2.30 的参数限制

参 数	级 别	
	8.0.30	8.2.30
每行最大样本数	4 096	4 096
每帧最大行数	2 304	2 304
每秒最大帧数	30	30
每帧最大片数	128	128
最大亮度样本速率（样本每秒）	283 115 520	283 5 520

T/AI 109.8-2025

表 F.12 级别 8.0.30 和 8.2.30 的参数限制（续）

参 数	级 别	
	8.0.30	8.2.30
最大位速率（位每秒）	25 000 000	100 000 000
每帧最大位数（位）	10 000 000	10 000 000
BBV缓冲区大小（位）	25 001 984	100 007 936
解码图像缓冲区大小（图像）	Min(58490880/(SpsPicWidthInLuma[NumOfLayers-1] × SpsPicHeightInLuma[NumOfLayers-1]),16)	

级别 8.4.30 和 8.6.30 的参数限制见表 F.13。

表F.13 级别 8.4.30 和 8.6.30 的参数限制

参 数	级 别	
	8.4.30	8.6.30
每行最大样本数	4 096	4 096
每帧最大行数	2 304	2 304
每秒最大帧数	30	30
每帧最大片数	128	128
最大亮度样本速率（样本每秒）	283 115 520	283 115 520
最大位速率（位每秒）	25 000 000	100 000 000
每帧最大位数（位）	10 000 000	10 000 000
BBV缓冲区大小（位）	25 001 984	100 007 936
解码图像缓冲区大小（图像）	Min(66846720/(SpsPicWidthInLuma[NumOfLayers-1] × SpsPicHeightInLuma[NumOfLayers-1]),16)	

级别 8.0.60 和 8.2.60 的参数限制见表 F.14。

表F.14 级别 8.0.60 和 8.2.60 的参数限制

参 数	级 别	
	8.0.60	8.2.60
每行最大样本数	4 096	4 096
每帧最大行数	2 304	2 304
每秒最大帧数	60	60
每帧最大片数	128	128
最大亮度样本速率（样本每秒）	566 231 040	566 231 040
最大位速率（位每秒）	40 000 000	160 000 000
每帧最大位数（位）	10 000 000	10 000 000
BBV缓冲区大小（位）	40 009 728	160 006 144
解码图像缓冲区大小（图像）	Min(58490880/(SpsPicWidthInLuma[NumOfLayers-1] × SpsPicHeightInLuma[NumOfLayers-1]),16)	

T/AI 109.8-2025

级别 8.4.60 和 8.6.60 的参数限制见表 F.15。

表F.15 级别 8.4.60 和 8.6.60 的参数限制

参 数	级 别	
	8.4.60	8.6.60
每行最大样本数	4 096	4 096
每帧最大行数	2 304	2 304
每秒最大帧数	60	60
每帧最大片数	128	128
最大亮度样本速率（样本每秒）	566 231 040	566 231 040
最大位速率（位每秒）	40 000 000	160 000 000
每帧最大位数（位）	10 000 000	10 000 000
BBV缓冲区大小（位）	40 009 728	160 006 144
解码图像缓冲区大小（图像）	Min(66846720/(SpsPicWidthInLuma[NumOfLayers-1] × SpsPicHeightInLuma[NumOfLayers-1]),16)	

级别 8.0.120 和 8.2.120 的参数限制见表 F.16。

表F.16 级别 8.0.120 和 8.2.120 的参数限制

参 数	级 别	
	8.0.120	8.2.120
每行最大样本数	4 096	4 096
每帧最大行数	2 304	2 304
每秒最大帧数	120	120
每帧最大片数	128	128
最大亮度样本速率（样本每秒）	1 132 462 080	1 132 462 080
最大位速率（位每秒）	60 000 000	240 000 000
每帧最大位数（位）	10 000 000	10 000 000
BBV缓冲区大小（位）	60 014 592	240 009 216
解码图像缓冲区大小（图像）	Min(58490880/(SpsPicWidthInLuma[NumOfLayers-1] × SpsPicHeightInLuma[NumOfLayers-1]),16)	

级别 8.4.120 和 8.6.120 的参数限制见表 F.17。

表F.17 级别 8.4.120 和 8.6.120 的参数限制

参 数	级 别	
	8.4.120	8.6.120
每行最大样本数	4 096	4 096
每帧最大行数	2 304	2 304
每秒最大帧数	120	120

T/AI 109.8-2025

表 F.17 级别 8.4.120 和 8.6.120 的参数限制（续）

参 数	级 别	
	8.4.120	8.6.120
每帧最大片数	128	128
最大亮度样本速率（样本每秒）	1 132 462 080	1 132 462 080
最大位速率（位每秒）	60 000 000	240 000 000
每帧最大位数（位）	10 000 000	10 000 000
BBV缓冲区大小（位）	60 014 592	240 009 216
解码图像缓冲区大小（图像）	Min(66846720/(SpsPicWidthInLuma[NumOfLayers-1] × SpsPicHeightInLuma[NumOfLayers-1]),16)	

级别 10.0.30 和 10.2.30 的参数限制见表 F.18。

表F.18 级别 10.0.30 和 10.2.30 的参数限制

参 数	级 别	
	10.0.30	10.2.30
每行最大样本数	8 192	8 192
每帧最大行数	4 608	4 608
每秒最大帧数	30	30
每帧最大片数	128	128
最大亮度样本速率（样本每秒）	1 069 547 520	1 069 547 520
每片最大亮度样本速率（样本每秒）	267 386 880	267 386 880
最大位速率（位每秒）	60 000 000	240 000 000
每帧最大位数（位）	30 000 000	30 000 000
BBV缓冲区大小（位）	60 014 592	240 009 216
解码图像缓冲区大小（图像）	Min(226492416/(SpsPicWidthInLuma[NumOfLayers-1] × SpsPicHeightInLuma[NumOfLayers-1]),16)	

级别 10.4.30 和 10.6.30 的参数限制见表 F.19。

表F.19 级别 10.4.30 和 10.6.30 的参数限制

参 数	级 别	
	10.4.30	10.6.30
每行最大样本数	8 192	8 192
每帧最大行数	4 608	4 608
每秒最大帧数	30	30
每帧最大片数	128	128
最大亮度样本速率（样本每秒）	1 069 547 520	1 069 547 520
每片最大亮度样本速率（样本每秒）	267 386 880	267 386 880
最大位速率（位每秒）	60 000 000	240 0 000

T/AI 109.8-2025

表 F.19 级别 10.4.30 和 10.6.30 的参数限制（续）

参 数	级 别	
	10.4.30	10.6.30
每帧最大位数（位）	30 000 000	30 000 000
BBV缓冲区大小（位）	60 014 592	240 009 216
解码图像缓冲区大小（图像）	Min(301989888/(SpsPicWidthInLuma[NumOfLayers-1] × SpsPicHeightInLuma[NumOfLayers-1]),16)	

级别 10.0.60 和 10.2.60 的参数限制见表 F.20。

表F.20 级别 10.0.60 和 10.2.60 的参数限制

参 数	级 别	
	10.0.60	10.2.60
每行最大样本数	8 192	8 192
每帧最大行数	4 608	4 608
每秒最大帧数	60	60
每帧最大片数	128	128
最大亮度样本速率（样本每秒）	2 139 095 040	2 139 095 040
每片最大亮度样本速率（样本每秒）	534 773 760	534 773 760
最大位速率（位每秒）	120 000 000	480 000 000
每帧最大位数（位）	30 000 000	30 000 000
BBV缓冲区大小（位）	120 012 800	480 002 048
解码图像缓冲区大小（图像）	Min(226492416/(SpsPicWidthInLuma[NumOfLayers-1] × SpsPicHeightInLuma[NumOfLayers-1]),16)	

级别 10.4.60 和 10.6.60 的参数限制见表 F.21。

表F.21 级别 10.4.60 和 10.6.60 的参数限制

参 数	级 别	
	10.4.60	10.6.60
每行最大样本数	8 192	8 192
每帧最大行数	4 608	4 608
每秒最大帧数	60	60
每帧最大片数	128	128
最大亮度样本速率（样本每秒）	2 139 095 040	2 139 095 040
每片最大亮度样本速率（样本每秒）	534 773 760	534 773 760
最大位速率（位每秒）	120 000 000	480 000 000
每帧最大位数（位）	30 000 000	30 000 000
BBV缓冲区大小（位）	120 012 800	480 002 048
解码图像缓冲区大小（图像）	Min(301989888/(SpsPicWidthInLuma[NumOfLayers-1] × SpsPicHeightInLuma[NumOfLayers-1]),16)	

T/AI 109.8-2025

级别 10.0.120 和 10.2.120 的参数限制见表 F.22。

表F.22 级别 10.0.120 和 10.2.120 的参数限制

参 数	级 别	
	10.0.120	10.2.120
每行最大样本数	8 192	8 192
每帧最大行数	4 608	4 608
每秒最大帧数	120	120
每帧最大片数	128	128
最大亮度样本速率（样本每秒）	4 278 190 080	4 278 190 080
每片最大亮度样本速率（样本每秒）	1 069 547 520	1 069 547 520
最大位速率（位每秒）	240 000 000	800 000 000
每帧最大位数（位）	30 000 000	30 000 000
BBV缓冲区大小（位）	240 009 216	800 014 336
解码图像缓冲区大小（图像）	Min(226492416/(SpsPicWidthInLuma[NumOfLayers-1] × SpsPicHeightInLuma[NumOfLayers-1]),16)	

级别 10.4.120 和 10.6.120 的参数限制见表 F.23。

表F.23 级别 10.4.120 和 10.6.120 的参数限制

参 数	级 别	
	10.4.120	10.6.120
每行最大样本数	8 192	8 192
每帧最大行数	4 608	4 608
每秒最大帧数	120	120
每帧最大片数	128	128
最大亮度样本速率（样本每秒）	4 278 190 080	4 278 190 080
每片最大亮度样本速率（样本每秒）	1 069 547 520	1 069 547 520
最大位速率（位每秒）	240 000 000	800 000 000
每帧最大位数（位）	30 000 000	30 000 000
BBV缓冲区大小（位）	240 009 216	800 014 336
解码图像缓冲区大小（图像）	Min(301989888/(SpsPicWidthInLuma[NumOfLayers-1] × SpsPicHeightInLuma[NumOfLayers-1]),16)	

T/AI 109.8-2025

某一级别下 BBV 缓冲区大小是 16384 位的倍数。

与表 F.4 到表 F.23 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：

当前图像编码后位流的二进制位数×1.8 + ((PicHorizontalSize × PicVerticalSize × (BitDepth + (BitDepth>>1)))>>5)

式中，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。

T/AI 109.8-2025

附 录 G
（资料性）
隐私保护单层编码模式低权限解码填充方法

G.1 概述

本附录描述了开启隐私保护单层编码模式（PrivacyModeIndex 的值等于 3），且为低权限解码（UserPermission 为 0），对于高权限的编码单元（PrivacyLevel 的值等于 1 的编码单元）的像素值进行填充的参考实现方法。

本附录为隐私保护单层编码模式进行低权限解码的高权限编码单元设置一个预测模式，用于隐私保护区域低权限解码的填充。该模式可以通过用户指定，是一种用户可以自行选择的方法，还可以根据用户需求拓展至任意预测模式。

G.2 方法

G.2.1 概述

本方法利用本文件规范的帧内预测模式来实现高权限的编码单元（PrivacyLevel 的值等于 1 的编码单元）在低权限时的解码操作，调用本文件规范的编码单元解码实现（见 8.6）。本附录只具体描述四种方法，其他所有预测模式与之类似，可以很方便地延伸拓展，用户可以根据实际需求效果自行设置和拓展。

G.2.2 方法1

若 PrivacyModeIndex 的值等于 3 且编码单元 PrivacyLevel 的值等于 1 且 UserPermission 的值等于 0，则按照以下分别进行预测样本解码和残差样本解码操作。

预测样本解码包括：将编码单元的 IntraCuFlag 设置为 1，DtSplitFlag 设置为 0，IbcCuFlag 设置为 0，IscCuFlag 设置为 0，SawpFlag 设置为 0，IntraPfFlag 设置为 0，IipFlag 设置为 0；确定编码单元类型和相关信息（见 8.6.3）；如果编码单元包括亮度分量，将编码单元的 IntraLumaPredMode 的值设置为 0，亮度预测模式设置为‘Intra_Luma_DC’，并进行普通帧内预测（见 8.8.1）；如果编码单元包括色度分量，将编码单元的 IntraChromaPredMode 的值设置为 1，色度预测模式设置为‘Intra_Chroma_DC’，并进行普通帧内预测（见 8.8.1）。

残差样本解码包括：将编码单元的残差样值矩阵 ResidueMatrix 中的值置为 0。

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 8.10）。

G.2.3 方法2

若 PrivacyModeIndex 的值等于 3 且编码单元 PrivacyLevel 的值等于 1 且 UserPermission 的值等于 0，则按照以下分别进行预测样本解码和残差样本解码操作。

预测样本解码包括：将编码单元的 IntraCuFlag 设置为 1，DtSplitFlag 设置为 0，IbcCuFlag 设置为 0，IscCuFlag 设置为 0，SawpFlag 设置为 0，IntraPfFlag 设置为 0，IipFlag 设置为 0；确定编码单元类型和相关信息（见 8.6.3）；如果编码单元包括亮度分量，将编码单元的 IntraLumaPredMode 的值设置为 1，亮度预测模式设置为‘Intra_Luma_Plane’，并进行普通帧内预测（见 8.8.1）；如果编码单元包括色度分量，将编码单元的 IntraChromaPredMode 的值设置为 1，色度预测模式设置为

T/AI 109.8-2025

‘Intra_Chroma_DC’，并进行普通帧内预测（见 8.8.1）。

残差样本解码包括：将编码单元的残差样值矩阵 ResidueMatrix 中的值置为 0。

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 8.10）。

G.2.4 方法3

若 PrivacyModeIndex 的值等于 3 且编码单元 PrivacyLevel 的值等于 1 且 UserPermission 的值等于 0，则按照以下分别进行预测样本解码和残差样本解码操作。

预测样本解码包括：将编码单元的 IntraCuFlag 设置为 1，DtSplitFlag 设置为 0，IbcCuFlag 设置为 0，IscCuFlag 设置为 0，SawpFlag 设置为 0，IntraPfFlag 设置为 0，IipFlag 设置为 0；确定编码单元类型和相关信息（见 8.6.3）；如果编码单元包括亮度分量，将编码单元的 IntraLumaPredMode 的值设置为 2，亮度预测模式设置为‘Intra_Luma_Bilinear’，并进行普通帧内预测（见 8.8.1）；如果编码单元包括色度分量，将编码单元的 IntraChromaPredMode 的值设置为 4，色度预测模式设置为‘Intra_Chroma_Bilinear’，并进行普通帧内预测（见 8.8.1）。

残差样本解码包括：将编码单元的残差样值矩阵 ResidueMatrix 中的值置为 0。

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 8.10）。

G.2.5 方法4

若 PrivacyModeIndex 的值等于 3 且编码单元 PrivacyLevel 的值等于 1 且 UserPermission 的值等于 0，则按照以下分别进行预测样本解码和残差样本解码操作。

预测样本解码包括：将编码单元的 IntraCuFlag 设置为 1，DtSplitFlag 设置为 0，IbcCuFlag 设置为 0，IscCuFlag 设置为 0，SawpFlag 设置为 0，IntraPfFlag 设置为 0，IipFlag 设置为 0；确定编码单元类型和相关信息（见 8.6.3）；如果编码单元包括亮度分量，将编码单元的 IntraLumaPredMode 的值设置为 13，亮度预测模式设置为‘Intra_Luma_Angular’，并进行普通帧内预测（见 8.8.1）；如果编码单元包括色度分量，将编码单元的 IntraChromaPredMode 的值设置为 1，色度预测模式设置为‘Intra_Chroma_DC’，并进行普通帧内预测（见 8.8.1）。

残差样本解码包括：将编码单元的残差样值矩阵 ResidueMatrix 中的值置为 0。

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 8.10）。

T/AI 109.8-2025

附 录 H
（规范性）
位流参考缓冲区管理

H.1 概述

本附录规定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 H.3.1 规定的方式进入 BBV 缓冲区，按照 H.3.2 规定的方式移出 BBV 缓冲区。符合本文件的位流不应导致 BBV 缓冲区上溢。如果 low_delay 的值为‘0’，编码数据应按 H.3.2.2 规定的方式移出 BBV 缓冲区；如果 low_delay 的值为‘1’，编码数据应按 H.3.2.3 规定的方式移出 BBV 缓冲区。符合本文件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。

H.2 约定

H.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。

H.2.2 约定二

BBV 缓冲区的大小为 BBS，单位为二进制位。

H.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）按下式计算：

$$R_{max} = BitRate \times 400 \quad \cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots \quad (H.1)$$

式中：

BitRate——以 400bit/s 为单位计算视频位流的位率；

R_{max} ——编码数据输入 BBV 缓冲区的最大速率。

H.3 基本操作

H.3.1 数据输入

H.3.1.1 概述

本附录规定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。

H.3.1.2 方法一

H.3.1.2.1 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）：

——序列参数集、跟在序列参数集之后的补充增强信息。这些数据记为 b(n)，b(n)与本幅图像起

T/AI 109.8-2025

始码之间不应有其他图像的数据。

——本幅图像的所有编码数据。

——跟在本幅图像参数集后的认证数据和安全参数集。

——跟在本幅图像后的填充数据、图像分界数据。

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)按下式计算

$$R(n) = d_n^* \div (\tau(n) - \tau(n+1) + t(n+1) - t(n)) \quad \cdots\cdots\cdots\cdots\cdots\cdots \quad (H.2)$$

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）；

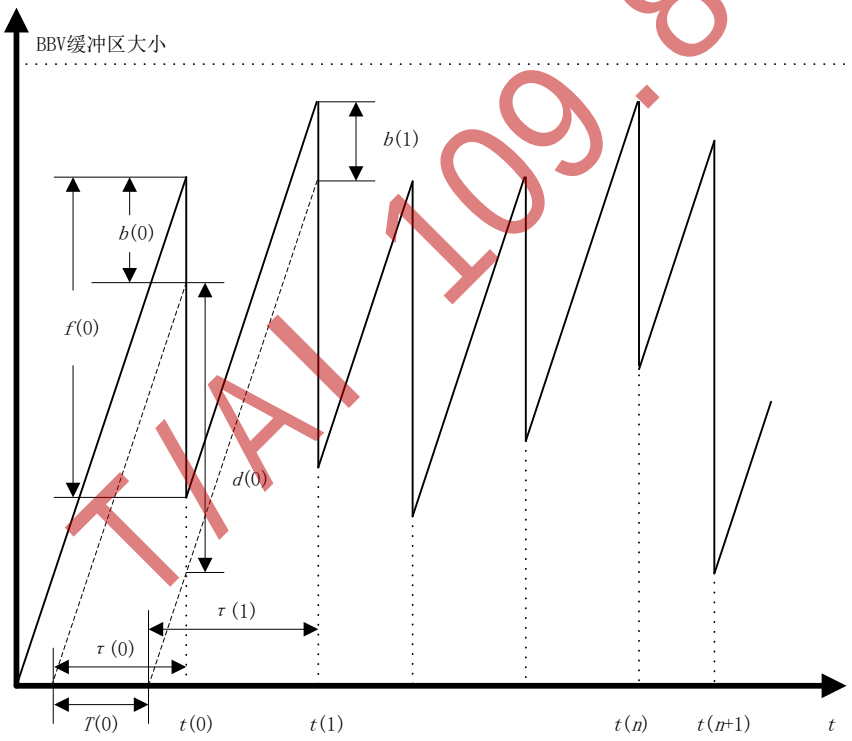
$t(n)$ ——第 n 幅图像的编码数据从 BBV 缓冲区移出的时间，单位为秒（s）；

$t(n+1)-t(n)$ ——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。

在编码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 H.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H.1 BBV 缓冲区占用情况一

H.3.1.2.2 编码视频序列开始时的歧义性

对编码视频序列进行随机访问时，序列参数集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。

T/AI 109.8-2025

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序列参数集后 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。

H.3.1.2.3 编码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 RBSP 是流结尾 RBSP，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参数集中规定的最大速率。

视频序列的第 1 幅图像的 NAL 单元头前的所有数据和这个 NAL 单元头输入 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（H.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。

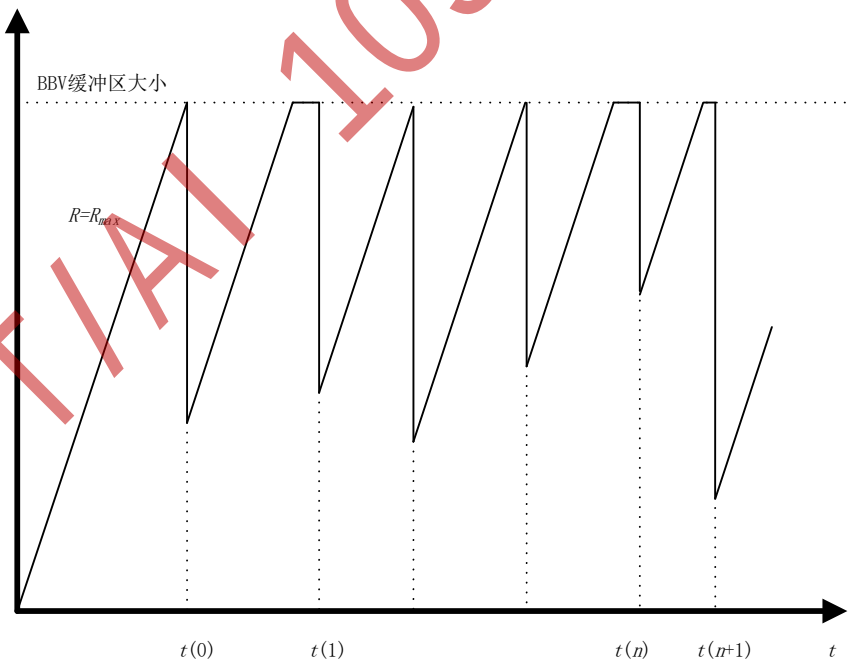
H.3.1.3 方法二

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

——如果 BBV 缓冲区没有充满，数据以速率 R_{max} 输入缓冲区；

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

视频序列的第 1 幅图像的 NAL 单元头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 H.2。



图H.2 BBV 缓冲区占用情况二

H.3.2 数据移出

T/AI 109.8-2025

H.3.2.1 概述

位流中第 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 幅图像的图像集参数 NAL 单元头进入 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 H.3.2.2 和 H.3.2.3 分别确定。

H.3.2.2 非低延迟

本条规定 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。

对于不包含知识图像的解码位流，第 n 幅图像的解码时刻 t(n)是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)。

对于包含知识图像的解码位流，如果第 n 幅图像是知识图像，则该图像的参考解码时刻 t'(n)＝t'(n-1)，其中 t'(0)＝t(0)；该图像的解码时刻 t(n)是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)。

对于包含知识图像的解码位流，如果第 n 幅图像不是知识图像，则该图像的参考解码时刻 t'(n)＝t'(n-1)+delta'(n)，其中 t'(0)＝t(0)；该图像的解码时刻 t(n)如下：

——如果第 n-1 幅图像不是知识图像且 t(n-1)+delta(n)小于参考解码时刻 t'(n)-delta'(n)-delta'(n-1)，则 t(n)是前一幅图像的解码时刻 t(n-1)加上两倍的检测时间间隔 delta(n)；

——如果第 n-1 幅图像是知识图像且 t(n-1)+delta(n-1)小于参考解码时刻 t'(n)-delta'(n-1)-delta'(n-2)，则 t(n)是前一幅图像的解码时刻 t(n-1)加上两倍的检测时间间隔 delta(n-1)；

——否则，则 t(n)是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 delta(n)和假设检测时间间隔 delta'(n)在 H.4 规定。

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，符合本文件的位流不应发生下溢。

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。

H.3.2.3 低延迟

本条规定 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。

对于不包含知识图像的解码位流，第 n 幅图像的解码时刻 t(n)是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(n)乘以图像周期的积。

对于包含知识图像的解码位流，如果第 n 幅图像是知识图像，则该图像的参考解码时刻 t'(n)＝t'(n-1)，其中 t'(0)＝t(0)；该图像的解码时刻 t(n)是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)。

对于包含知识图像的解码位流，如果第 n 幅图像不是知识图像，则该图像的参考解码时刻 t'(n)＝t'(n-1)+delta'(n)+(BbvCheckTimes(n)×图像周期)，其中 t'(0)＝t(0)；该图像的解码时刻 t(n)如下：

——如果第 n-1 幅图像不是知识图像且 t(n-1)+delta(n)+(BbvCheckTimes(n)×图像周期)小于参考解码时刻 t'(n)-delta'(n)-(BbvCheckTimes(n)×图像周期)-delta'(n-1)-(BbvCheckTimes(n-1)×图像周期)，则 t(n)是前一幅图像的解码时刻 t(n-1)加上两倍的检测时间间隔 delta(n)，再加上 BbvCheckTimes 乘以图像周期的积；

T/AI 109.8-2025

——如果第 n-1 幅图像是知识图像且 t(n-1)+delta(n-1)+(BbvCheckTimes(n)×图像周期)小于参考解码时刻 t'(n)-delta'(n-1)-(BbvCheckTimes(n)×图像周期)-delta'(n-2)-(BbvCheckTimes(n-2)×图像周期)，则 t(n)是前一幅图像的解码时刻 t(n-1)加上两倍的检测时间间隔 delta(n-1)，再加上 BbvCheckTimes(n)乘以图像周期的积；

——否则，t(n)是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(n)乘以图像周期的积。

检测时间间隔 delta(n)和假设检测时间间隔 delta'(n)在 H.4 规定。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(n)大于 0，当前解码图像描述为大图像。一个编码视频序列的最后一幅图像不应是大图像。

H.4 缓冲区检测时间间隔

记帧率的倒数为 T，记帧率加 1 的和的倒数为 D。如果解码位流中不包含知识图像，则 P 等于 T；如果解码位流中包含知识图像，则 P 等于 D。

BBV 缓冲区检测时间间隔 delta(n)和假设检测时间间隔 delta'(n)由解码完第 n-1 幅图像后输出的显示图像（见 8.3.4 ）确定。

T/AI 109.8-2025

附 录 I
（规范性）
加权量化矩阵

本附录规定序列参数集位流中加载的 4×4 和 8×8 变换块的默认加权量化矩阵 $WeightQuantMatrix_{4x4}$ 和 $WeightQuantMatrix_{8x8}$。

$$WeightQuantMatrix_{4x4} = \begin{bmatrix} 64 & 64 & 64 & 68 \\ 64 & 64 & 68 & 72 \\ 64 & 68 & 76 & 80 \\ 72 & 76 & 84 & 96 \end{bmatrix}$$

$$WeightQuantMatrix_{8x8} = \begin{bmatrix} 64 & 64 & 64 & 64 & 68 & 68 & 72 & 76 \\ 64 & 64 & 64 & 68 & 72 & 76 & 84 & 92 \\ 64 & 64 & 68 & 72 & 76 & 80 & 88 & 100 \\ 64 & 68 & 72 & 80 & 84 & 92 & 100 & 112 \\ 68 & 72 & 80 & 84 & 92 & 104 & 112 & 128 \\ 76 & 80 & 84 & 92 & 104 & 116 & 132 & 152 \\ 96 & 100 & 104 & 116 & 124 & 140 & 164 & 188 \\ 104 & 108 & 116 & 128 & 152 & 172 & 192 & 216 \end{bmatrix}$$

T/AI 109.8-2025

附 录 J
（资料性）
解密正确性验证功能和解码正确性验证功能参考实现方法示例

J.1 概述

本附录说明了一种利用用户数据实现解密正确性验证功能和解码正确性验证功能的参考实现方法。本附录为解密正确性验证功能提供的参考方法为，在补充增强信息中的用户数据中提供加密前位流的验证信息，用于对支持加密功能的编码器产生的位流在解密后验证其正确性和完整性。本附录为解码正确性验证功能提供的参考方法为，在补充增强信息中的用户数据中提供编码重建图像的验证信息，用于解码后验证解码重建图像的正确性和完整性。

本附录具体给出的解密正确性验证功能参考方法的输入为加密前或解密后的位流，输出为验证值信息。本附录具体给出的解码正确性验证功能参考方法的输入为经过编解码后的重建图像，输出为验证值信息。

本附录具体采用哈希方法计算输入数据的验证值。

J.2 语法和语义

J.2.1 语法描述

在用户数据中传送位流或重建图验证值信息参数负载定义，见表 J.1。

表J.1 用户数据参数负载定义

用户数据定义	描述符
user_data (PayloadSize) {	
data_type	u(1)
for (size = 0; size < 8; size++) {	
hash_data[size]	u(16)
bit_equal_to_1	u(1)
}	
reserved_bits	u(7)
}	

J.2.2 语义描述

数据类型 data_type

二值变量。其值为“0”表示由位流计算哈希数据，后续验证值数据为位流验证信息；其值为“1”表示由重建图像计算哈希数据，后续验证值数据为重建图验证信息。用于计算哈希的位流具体应为当前帧中除了补充增强信息、安全参数集、认证数据的所有 NAL 在加密前或解密后的 NAL 数据（不包含起始码）按照解码顺序拼接在一起后得到的位流。用于计算哈希的重建图像具体应为当前帧在编解码时所产生的重建图像所有颜色分量的图像数据。

注：可以在任何一帧图像的位流的补充增强信息 NAL 中包含此附录所规定的用户数据，每个补充增强信息 NAL 的用户数据中最多包含一帧图像对应的验证信息。本附录规定包含位流验证值信息的补充增强信息 NAL 的加密标志

T/AI 109.8-2025

encryption_idc 应为 0，即该 RBSP 不加密。

哈希验证值数据 hash_data[size]

16 位无符号整数。表示一个验证值数据中的一段 2 个字节的数据，该 2 个字节在[2*size，2*size+2]字节区间之内。本附录提供参考方法具体使用哈希信息作为验证值数据，验证值数据总共 16 字节，对应 size 取值范围为[0,7]。

防竞争位 bit_equal_to_1

1 位无符号整数，其取值必须为 1。验证值数据每两个字节后跟一个 1 位整数 1，为了避免与 NAL 单元起始码冲突。验证值数据包含此防竞争位后共 17 字节。

J.3 位流验证值计算过程

在计算哈希验证值前，加密前或解密后的位流被处理以及拼接，形成验证值计算的输入数据,具体表示为数组 rawdata，之后经过计算哈希过程，具体过程如函数 process_stream_hash( )所示。

```
process_stream_hash( ) {
    iLen = 0
    for( nal = 0; nal < totalNalNum; nal++) {
        totalNalBytes = Len( nalData[nal])
        for (nalbyte = 0; nalbyte < totalNalBytes; nalbyte++) {
            rawdata[iLen++] = nalData[nal] [nalbyte] & 0xFF
        }
    }
    dataLen = iLen
    hashInit (context )
    hashUpdate(context, rawdata, dataLen )
    hashFinal(digestVal, context )
}
```

上述过程中，totalNalNum 为当前帧中用于计算验证值的 NAL 个数；nalData[nal]为某个当前帧第 nal 个 NAL 在加密前（该 NAL 中的加密标志按照最终是否加密标记）或解密后的数据（不包含起始码）；Len( )函数用于计算 nalData[nal]数据的总字节数得到 totalNalBytes；rawdata 的每个字节数据记录的是每个 NAL 数据 nalData[nal]中的每个字节的数据；dataLen 记录了用于哈希输入的位流数据总字节长度。

context 为计算哈希所需要的上下文；rawdata 是哈希计算的输入数据，dataLen 为其长度；digestVal 为哈希计算结果，即 16 字节验证值，在句法 hash_data 中以上述 J.2 规定方式传输。

函数 hashInit( )、hashUpdate( )和 hashFinal( )表示计算哈希值的步骤中使用的函数。

J.4 重建图验证值计算过程

重建图像的所有颜色分量图像被处理并用于计算哈希。其具体过程如函数 process_reconstruction_hash( )所示。

```
process_reconstruction_hash( ) {
    hashInit(context)
    for (comp = 0; comp < totalComponentNum; comp++) {
```

T/AI 109.8-2025

```
            for (row = 0; row < height[comp]; row++ ) {
                iLen = 0
                for (col = 0; col < width[comp]; col++) {
                    rawdata[iLen++] = ReconData[comp][row][col] & 0xFF
                    if (bitdepth[comp] > 8)
                        rawdata[iLen++] = ReconData[comp][row][col] >> 8 & 0xFF
                }
                dataLen = iLen
                hashUpdate(context, rawdata, dataLen)
            }
        }
        hashFinal(digestVal, context)
}
```

上述过程中，totalComponentNum 为当前帧的图像颜色分量个数；width[comp]和 height[comp]为第 comp 个颜色分量图像的宽高;ReconData[comp][row][col]为第 comp 个颜色分量图像第 row 行第 col 列像素的像素值；bitdepth[comp]表示第 comp 个颜色分量图像的位深度，其中如果位深度大于 8，需要将其 2 个字节数据都写入 rawdata 中。rawdata 的每个字节数据记录的是所有颜色分量图像每一行的每个字节的数据；dataLen 记录了用于哈希输入的重建图数据总字节长度。

context 为计算哈希所需要的上下文；rawdata 是由加密前或解密后位流或重建图像每行的数据，dataLen 为其长度，将 rawdata 作为 hashUpdate 函数的输入数据；digestVal 为哈希计算的最终结果，即 16 字节验证值，在句法 hash_data 中以上述 J.2 规定方式传输。

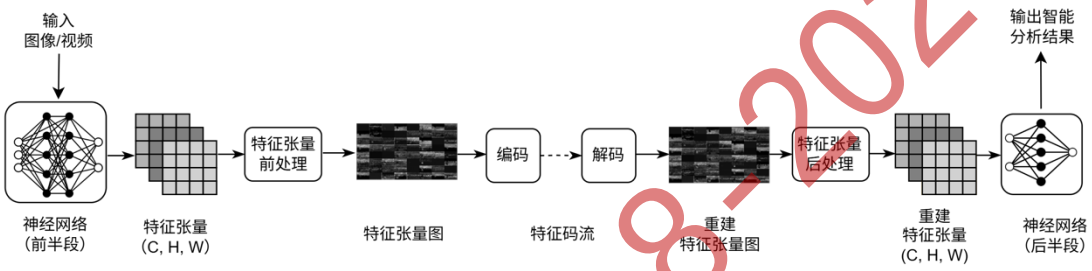
函数 hashInit( )、hashUpdate( )和 hashFinal( )表示计算哈希值的步骤中使用的函数。

T/AI 109.8-2025

附 录 K
（资料性）
特征张量图前处理、后处理和编码模式示例

K.1 概述

本附录描述了通用的机器视觉特征编解码方法，其流程图见图 K.1。机器视觉特征编解码技术将用于智能任务的深度神经网络一分为二，从网络中间层提取包含智能分析所需的特征张量数据，并通过特征张量图前处理对该特征张量数据进行初步的整理和压缩，从而得到特征张量图。特征张量图进一步通过本文件规范编码器进行压缩，得到特征位流，然后在解码端通过本文件规范解码器重建特征张量图。随后将重建的特征张量图经过特征张量图后处理，还原神经网络中间层数据，即特征张量数据，输入到神经网络后半段，从而完成智能任务分析。



图K.1 通用的机器视觉特征编解码方法流程图

在通用的机器视觉特征编解码方法中，特征张量图前处理的输入为特征张量数据，其功能包括特征张量数据尺寸缩减、特征量化、特征扫描排列，输出为 YUV 格式的特征张量图；特征张量图后处理的输入为重建的 YUV 格式的特征张量图，其功能包括特征反扫描重排、特征反量化、特征张量数据尺寸的还原，输出为重建的特征张量数据。

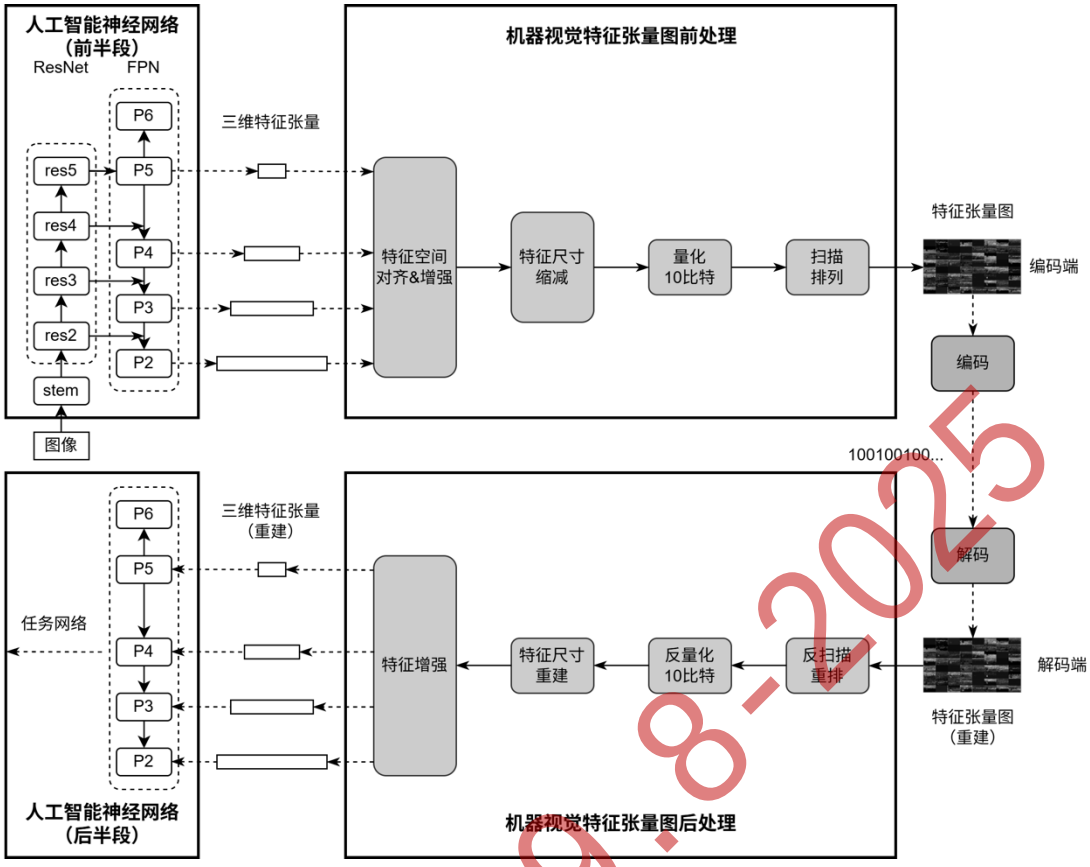
建议每个随机访问点视频编码图像之后，编码的第一个特征张量图同样是随机访问点图像。

此外，本附录描述了特征张量图前处理、后处理和编码模式的一种实现方法，并具体地给出了一种特征张量图前处理、后处理和编码模式举例，所用技术方案为非对称特征编码（AFC，Asymmetrical Feature Coding），将在 K.2 小节详细介绍。

K.2 非对称特征编码技术

AFC 技术方案流程图见图 K.2。

T/AI 109.8-2025



图K.2 AFC技术方案流程图

AFC是由深度神经网络和编码器共同构建的。在编码端，AFC输入特征金字塔（Feature Pyramid Networks，FPN），通过深度神经网络执行特征空间对齐、特征增强以及通道降维。然后经过10位的线性量化、扫描排列之后，特征张量图被输入到本文件规范编码器进行高效编码，从而生成特征位流。在解码端，AFC执行本文件规范解码、反扫描重排、10位的线性反量化、特征尺寸重建和特征增强，其输出为重建的特征金字塔。

针对JDE以及Mask/Fast R-CNN，AFC分别设计了相应的网络架构，分别可以用来实现目标追踪、实例分割和目标检测智能任务。

K.2.1 面向JDE的网络架构AFC-JDE

面向JDE的网络架构AFC-JDE见图K.3，包括AFC前处理过程、编解码过程、AFC后处理过程，该网络架构用于实现目标追踪智能任务。注意，本节中所描述的特征尺寸的参数C、H、W与智能任务网络的分割点选择有关，表示当前分割点处的特征尺寸。

K.2.1.1 AFC前处理过程

步骤1：取特征金字塔作为AFC前处理的输入，三层特征分别是尺寸为（C,4H,4W）的P1特征，尺寸为（2C,2H,2W）的P2特征，尺寸为（4C,H,W）的P3特征。

步骤2：进入特征空间对齐阶段，使用卷积层和下采样层处理P1特征得到尺寸为（C,H,W）的特征P'_1，该卷积层的步长为2、卷积核尺寸为3×3、卷积核数量为2，该下采样层的下采样倍率为4；使用卷积层和下采样层处理P2特征得到尺寸为（2C,H,W）的特征P'_2，该卷积层的步长为2、卷积核

T/AI 109.8-2025

尺寸为 3×3、卷积核数量为 1，该下采样层的下采样倍率为 2；使用卷积层处理 P3 特征得到尺寸为（4C,H,W）的特征$P'3$，该卷积层的步长为 1、卷积核尺寸为 1×1、卷积核数量为 1；

步骤 3：沿着通道维度将对齐过的所有输入特征连接起来，得到尺寸为（7C,H,W）的特征 P4；

步骤 4：进入特征增强阶段，使用注意力模块 ECA 处理特征 P4 得到尺寸为（7C,H,W）的特征 P5；

步骤 5：进入通道降维阶段，使用卷积层处理特征 P5 得到尺寸为（64,H,W）的特征 P6，该卷积层的步长为 1、卷积核尺寸为 1×1、卷积核数量为 1；

步骤 6：进入特征量化阶段，使用 10 位的线性量化处理特征 P6 得到尺寸为（64,H,W）特征 P7，具体操作为：

获取当前特征张量数据中的最大值ftm_iquant_max和最小值ftm_iquant_min，根据公式（K.1）对每个特征值进行 10 位的线性量化：

$$\hat{X} = \frac{(2^{10}-1)*(X-ftm_iquant_min)}{(ftm_iquant_max-ftm_iquant_min)} \quad \text{（K.1）}$$

其中，$\hat{X}$为量化后的特征值，当前特征张量数据中的最大值ftm_iquant_max和最小值ftm_iquant_min以特征张量图信息扩展语法表的图像参数集中规定方式传输到解码端；

步骤 7：进入特征扫描排列阶段，将特征 P7 按水平扫描顺序打包为特征张量图 Z，这里的特征张量图 Z 格式为 YUV444、YUV422、YUV420 中的任一种。

K.2.1.2 编解码过程

步骤 1：使用本文件规范编码器编码特征张量图 Z 得到特征位流 binZ；

步骤 2：使用本文件规范解码器解码特征位流 binZ 得到重建的特征张量图Z'。

K.2.1.3 AFC后处理过程

步骤 1：进入特征反扫描重排阶段，使用重建的特征张量图Z'按照水平扫描的反操作得到尺寸为（64,H,W）的量化特征 P8；

步骤 2：进入特征反量化阶段，使用 10 位的线性反量化处理特征 P8 得到尺寸为（64,H,W）的特征 P9，具体操作为：

从图像参数集中获取当前特征张量数据的最大值ftm_iquant_max和最小值ftm_iquant_min，根据公式（K.2）对每个量化后的特征值进行 10 位的线性反量化：

$$X' = \frac{\hat{X}'*(ftm_iquant_max-ftm_iquant_min)}{(2^{10}-1)} + ftm_iquant_min \quad \text{（K.2）}$$

其中，$\hat{X}'$为解码得到的量化特征值，X'为重建特征值；

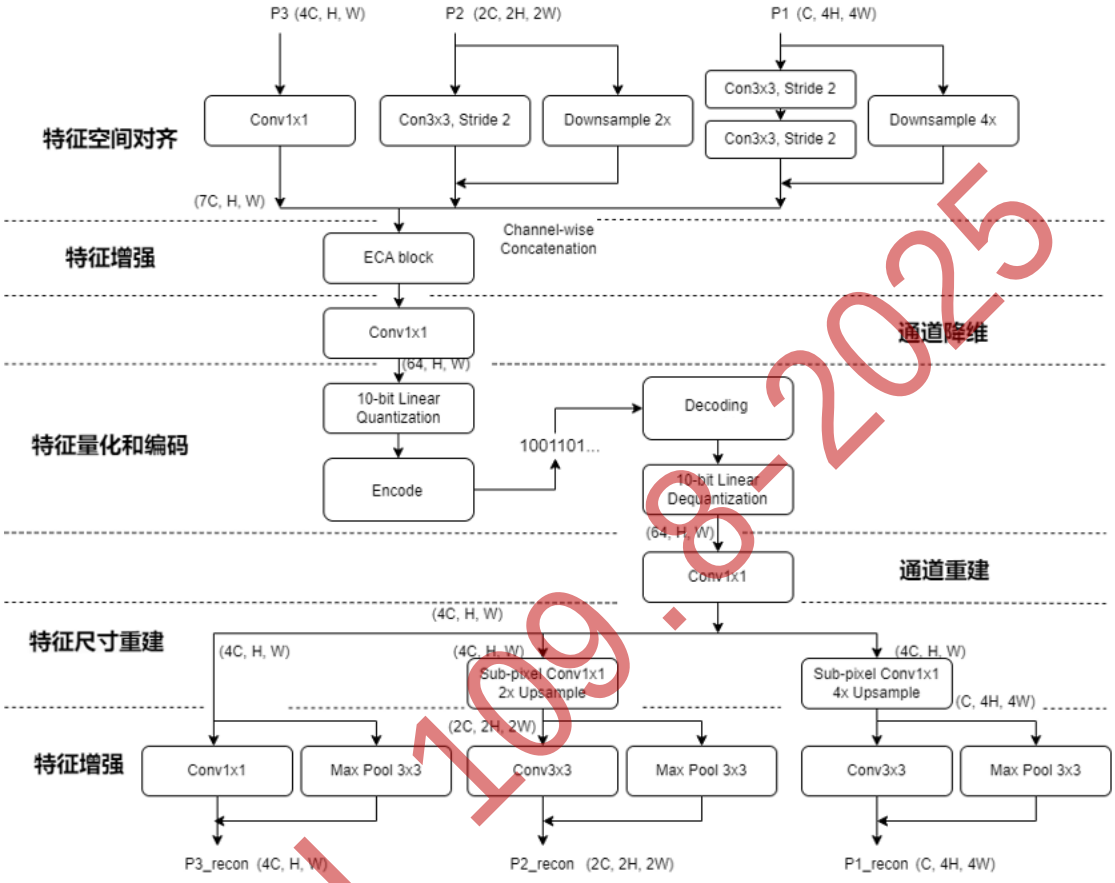
步骤 3：进入通道重建阶段，使用卷积层处理特征 P9 得到尺寸为（4C,H,W）的特征 P10，该卷积层的步长为 1、卷积核尺寸为 1×1、卷积核数量为 1；

步骤 4：进入特征尺寸重建阶段，此阶段分三个支路，第一个支路使用卷积层和上采样层处理特征 P10 得到尺寸为（C,4H,4W）的特征 P11，该卷积层的步长为 1、卷积核尺寸为 1×1、卷积核数量为 1，该上采样层的上采样率为 4；第二个支路使用卷积层和上采样层处理特征 P10 得到尺寸为（2C,2H,2W）的特征 P12，该卷积层的步长为 1、卷积核尺寸为 1×1、卷积核数量为 1，该上采样层的上采样率为 2；第三个支路对特征 P10 不作处理，此阶段得到尺寸为（4C,H,W）的特征 P13；

步骤 5：进入特征增强阶段，此阶段分三个支路重建三层金字塔特征，第一路使用卷积层和最大池化层处理特征 P11 得到尺寸为（C,4H,4W）的重建特征 P1_recon，该卷积层的步长为 1、卷积核尺寸为 3×3、卷积核数量为 1，该最大池化层的步长为 1、池化核为 3×3、池化核数量为 1；第二路使用

T/AI 109.8-2025

卷积层和最大池化层处理 P12 得到尺寸为（2C,2H,2W）的重建特征 P2_recon，该卷积层的步长为 1、卷积核尺寸为 3×3、卷积核数量为 1，该最大池化层的步长为 1、池化核为 3×3、池化核数量为 1；第三路使用卷积层和最大池化层处理 P13 得到尺寸为（4C,H,W）的重建特征 P3_recon，该卷积层的步长为 1、卷积核尺寸为 1×1、卷积核数量为 1，该最大池化层的步长为 1、池化核为 3×3、池化核数量为 1。



图K.3　面向 JDE 的网络架构 AFC-JDE

K.2.2　面向Faster/Mask-RCNN的网络架构AFC-RCNN

面向 Faster/Mask-RCNN 的网络架构 AFC-RCNN 见图 K.4，包括 AFC 前处理过程、编解码过程、AFC 后处理过程。这基本上与面向 JDE 的网络架构相同，只是增加了一个新的分支来适应四个输入特征层。该网络架构用于实现实例分割和目标检测智能任务。注意，本节中所描述的特征尺寸的参数 C、H、W 与智能任务网络的分割点选择有关，表示当前分割点处的特征尺寸。

K.2.2.1　AFC前处理过程

步骤 1：取特征金字塔作为 AFC 前处理的输入，四层特征分别是尺寸为（C,8H,8W）的 K1 特征，尺寸为（C,4H,4W）的 K2 特征，尺寸为（C,2H,2W）的 K3 特征，尺寸为（C,H,W）的 K4 特征。

步骤 2：进入特征空间对齐阶段，使用卷积层和下采样层处理 K1 特征得到尺寸为（C,H,W）的特征K′1，该卷积层的步长为 2、卷积核尺寸为 3×3、卷积核数量为 3，该下采样层的下采样倍率为 8；使用卷积层和下采样层处理 K2 特征得到尺寸为（C,H,W）的特征K′2，该卷积层的步长为 2、卷积核

T/AI 109.8-2025

尺寸为 3×3、卷积核数量为 2，该下采样层的下采样倍率为 4；使用卷积层和下采样层处理 K3 特征得到尺寸为（C,H,W）的特征K'_3，该卷积层的步长为 2、卷积核尺寸为 3×3、卷积核数量为 1，该下采样层的下采样倍率为 2；使用卷积层处理 K4 特征得到尺寸为（C,H,W）的特征K'_4，该卷积层的步长为 1、卷积核尺寸为 1×1、卷积核数量为 1；

步骤 3：沿着通道维度将对齐过的所有输入特征连接起来，得到尺寸为（4C,H,W）的特征 K5；

步骤 4：进入特征增强阶段，使用注意力模块 ECA 处理特征 K5 得到尺寸为（4C,H,W）的特征 K6；

步骤 5：进入通道降维阶段，使用卷积层处理特征 K6 得到尺寸为（64,H,W）的特征 K7，该卷积层的步长为 1、卷积核尺寸为 1×1、卷积核数量为 1；

步骤 6：进入特征量化阶段，使用 10 位的线性量化处理特征 K7 得到尺寸为（64,H,W）特征 K8，具体操作为：

获取当前特征张量数据中的最大值ftm_iquant_max和最小值ftm_iquant_min，根据公式（K.3）对每个特征值进行 10 位的线性量化：

$$\hat{X} = \frac{(2^{10}-1)*(X-ftm_iquant_min)}{(ftm_iquant_max-ftm_iquant_min)} \quad \text{（K.3）}$$

其中，$\hat{X}$为量化后的特征值，当前特征张量数据中的最大值ftm_iquant_max和最小值ftm_iquant_min以特征张量图信息扩展语法表的图像参数集中规定方式传输到解码端；

步骤 7：进入特征扫描重排阶段，将特征 K8 按水平扫描顺序打包为特征张量图 Z，这里的特征张量图 Z 格式为 YUV444、YUV422、YUV420 中的任一种。

K.2.2.2 编解码过程

步骤 1：使用本文件规范编码器编码特征张量图 Z 得到特征位流 binZ；

步骤 2：使用本文件规范解码器解码特征位流 binZ 得到重建的特征张量图Z'。

K.2.2.3 AFC后处理过程

步骤 1：进入特征反扫描重排阶段，使用重建的特征张量图Z'按照水平扫描的反操作得到尺寸为（64,H,W）的量化特征 K9；

步骤 2：进入特征反量化阶段，使用 10 位的线性反量化处理特征 K9 得到尺寸为（64,H,W）的特征 K10，具体操作为：

从图像参数集中获取当前特征张量数据的最大值ftm_iquant_max和最小值ftm_iquant_min，根据公式（K.4）对每个量化后的特征值进行 10 位的线性反量化：

$$X' = \frac{\hat{X}'*(ftm_iquant_max-ftm_iquant_min)}{(2^{10}-1)} + ftm_iquant_min \quad \text{（K.4）}$$

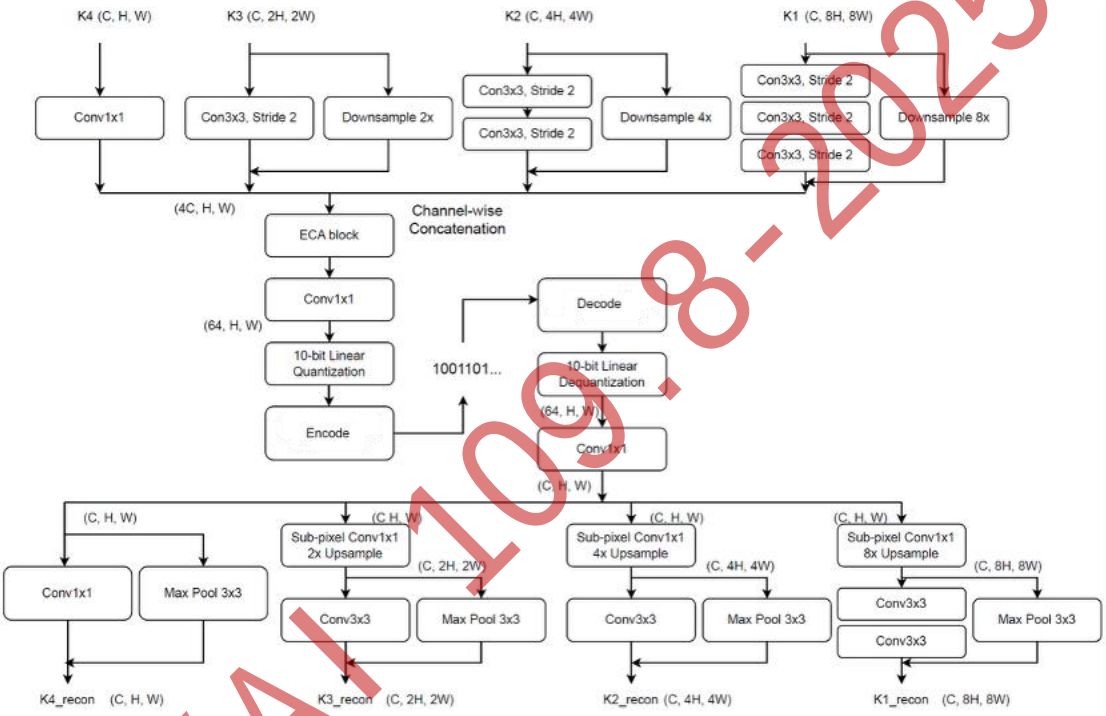
其中，$\hat{X}'$为解码得到的量化特征值，X'为重建特征值；

步骤 3：进入通道重建阶段，使用卷积层处理特征 K10 得到尺寸为（C,H,W）的特征 K11，该卷积层的步长为 1、卷积核尺寸为 1×1、卷积核数量为 1；

步骤 4：进入特征尺寸重建阶段，此阶段分四个支路，第一个支路使用卷积层和上采样层处理特征 K11 得到尺寸为（C,8H,8W）的特征 K12，该卷积层的步长为 1、卷积核尺寸为 1×1、卷积核数量为 1，该上采样层的上采样率为 8；第二个支路使用卷积层和上采样层处理特征 K11 得到尺寸为（C,4H,4W）的特征 K13，该卷积层的步长为 1、卷积核尺寸为 1×1、卷积核数量为 1，该上采样层的上采样率为 4；第三个支路使用卷积层和上采样层处理特征 K11 得到尺寸为（C,2H,2W）的特征 K14，该卷积层的步长为 1、卷积核尺寸为 1×1、卷积核数量为 1，该上采样层的上采样率为 2；第四个支路对特征 K11 不作处理，此阶段得到尺寸为（C,H,W）的特征 K15；

T/AI 109.8-2025

步骤 5：进入特征增强阶段，此阶段分四个支路重建四层金字塔特征，第一路使用卷积层和最大池化层处理特征 K12 得到尺寸为（C,8H,8W）的重建特征 K1_recon，该卷积层的步长为 1、卷积核尺寸为 3×3、卷积核数量为 2，该最大池化层的步长为 1、池化核为 3×3、池化核数量为 1；第二路使用卷积层和最大池化层处理特征 K13 得到尺寸为（C,4H,4W）的重建特征 K2_recon，该卷积层的步长为 1、卷积核尺寸为 3×3、卷积核数量为 1，该最大池化层的步长为 1、池化核为 3×3、池化核数量为 1；第三路使用卷积层和最大池化层处理特征 K14 得到尺寸为（C,2H,2W）的重建特征 K3_recon，该卷积层的步长为 1、卷积核尺寸为 3×3、卷积核数量为 1，该最大池化层的步长为 1、池化核为 3×3、池化核数量为 1；第四路使用卷积层和最大池化层处理特征 K15 得到尺寸为（C,H,W）的重建特征 K4_recon，该卷积层的步长为 1、卷积核尺寸为 1×1、卷积核数量为 1，该最大池化层的步长为 1、池化核为 3×3、池化核数量为 1。



图K.4　面向 Faster/Mask-RCNN 的网络架构

T/AI 109.8-2025

附 录 L
（规范性）
系数编码使用的数组

L.1 prefix_ctx

prefix_ctx[8] = {0,0,0,3,6,10,15,21}

L.2 min_in_group

min_in_group[10] = {0,1,2,3,4,6,8,12,16,24}

L.3 group_index

group_index[32] = {0,1,2,3,4,4,5,5,6,6,6,6,7,7,7,7,8,8,8,8,8,8,8,8,9,9,9,9,9,9,9,9}

T/AI 109.8-2025

附 录 M
（资料性）
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器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 {
```

T/AI 109.8-2025

```
                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)
}
```

T/AI 109.8-2025

附 录 N
（规范性）
反变换矩阵

N.1 DCT2 型反变换矩阵

N.1.1 4×4 DCT2反变换矩阵

4×4 DCT2 反变换矩阵规定如下：

$DCT2_4$ = {
{ 32 32 32 32 }
{ 42 17 -17 -42 }
{ 32 -32 -32 32 }
{ 17 -42 42 -17 }}

N.1.2 8×8 DCT2反变换矩阵

8×8 DCT2 反变换矩阵规定如下：

$DCT2_8$ = {
{ 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 }}

N.1.3 16×16 DCT2反变换矩阵

16×16 DCT2 反变换矩阵规定如下：

$DCT2_{16}$ = {
{ 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 }

T/AI 109.8-2025

{ 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 }}

N.1.4 32×32 DCT2反变换矩阵

32×32 DCT2 反变换矩阵规定如下：

$DCT2_{32}$ = {
{ T_{11} T_{12} }
{ T_{21} T_{22} }}

其中，T_{11} = {
{ 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 }}

T_{12} = {
{ 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 }

T/AI 109.8-2025

{-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 }}
T_{21} = {
{ 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 }}
T_{22} = {
{ 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 }}

N.1.5 64×64 DCT2反变换矩阵

64×64 DCT2 反变换矩阵规定如下：

$DCT2_{64}$ = {
{ T_{11} T_{12} T_{13} T_{14} }
{ T_{21} T_{22} T_{23} T_{24} }

T/AI 109.8-2025

{ T_{31} T_{32} T_{33} T_{34} }
{ T_{41} T_{42} T_{43} T_{44} }}
其中，T_{11} = {
{ 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 }
{ 45 45 45 45 44 44 43 42 41 40 39 38 37 36 34 33 }
{ 45 45 44 43 41 39 36 34 30 27 23 19 15 11 7 2 }
{ 45 44 42 39 36 31 26 20 14 8 1 -6 -12 -18 -24 -30 }
{ 45 43 40 35 29 21 13 4 -4 -13 -21 -29 -35 -40 -43 -45 }
{ 45 42 37 30 20 10 -1 -12 -22 -31 -38 -43 -45 -45 -41 -36 }
{ 45 41 34 23 11 -2 -15 -27 -36 -43 -45 -44 -39 -30 -19 -7 }
{ 45 39 30 16 1 -14 -28 -38 -44 -45 -40 -31 -18 -3 12 26 }
{ 44 38 25 9 -9 -25 -38 -44 -44 -38 -25 -9 9 25 38 44 }
{ 44 36 20 1 -18 -34 -44 -45 -37 -22 -3 16 33 43 45 38 }
{ 44 34 15 -7 -27 -41 -45 -39 -23 -2 19 36 45 43 30 11 }
{ 44 31 10 -14 -34 -45 -42 -28 -6 18 37 45 40 24 1 -22 }
{ 43 29 4 -21 -40 -45 -35 -13 13 35 45 40 21 -4 -29 -43 }
{ 43 26 -1 -28 -44 -42 -24 3 30 44 41 22 -6 -31 -45 -40 }
{ 43 23 -7 -34 -45 -36 -11 19 41 44 27 -2 -30 -45 -39 -15 }
{ 42 20 -12 -38 -45 -28 3 33 45 34 6 -26 -44 -39 -14 18 }}
T_{12} = {
{ 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 }
{ 31 30 28 26 24 22 20 18 16 14 12 10 8 6 3 1 }
{ -2 -7 -11 -15 -19 -23 -27 -30 -34 -36 -39 -41 -43 -44 -45 -45 }
{-34 -38 -41 -44 -45 -45 -45 -43 -40 -37 -33 -28 -22 -16 -10 -3 }
{-45 -43 -40 -35 -29 -21 -13 -4 4 13 21 29 35 40 43 45 }
{-28 -18 -8 3 14 24 33 39 44 45 44 40 34 26 16 6 }
{ 7 19 30 39 44 45 43 36 27 15 2 -11 -23 -34 -41 -45 }
{ 37 44 45 41 33 20 6 -10 -24 -36 -43 -45 -42 -34 -22 -8 }
{ 44 38 25 9 -9 -25 -38 -44 -44 -38 -25 -9 9 25 38 44 }
{ 24 6 -14 -31 -42 -45 -39 -26 -8 12 30 41 45 40 28 10 }
{-11 -30 -43 -45 -36 -19 2 23 39 45 41 27 7 -15 -34 -44 }
{-39 -45 -38 -20 3 26 41 45 36 16 -8 -30 -43 -44 -33 -12 }
{-43 -29 -4 21 40 45 35 13 -13 -35 -45 -40 -21 4 29 43 }
{-20 8 33 45 39 18 -10 -34 -45 -38 -16 12 36 45 37 14 }
{ 15 39 45 30 2 -27 -44 -41 -19 11 36 45 34 7 -23 -43 }
{ 41 43 22 -10 -37 -45 -30 1 31 45 36 8 -24 -44 -40 -16 }}
T_{13} = {
{ 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 }
{ -1 -3 -6 -8 -10 -12 -14 -16 -18 -20 -22 -24 -26 -28 -30 -31 }
{-45 -45 -44 -43 -41 -39 -36 -34 -30 -27 -23 -19 -15 -11 -7 -2 }
{ 3 10 16 22 28 33 37 40 43 45 45 45 44 41 38 34 }
{ 45 43 40 35 29 21 13 4 -4 -13 -21 -29 -35 -40 -43 -45 }

T/AI 109.8-2025

{ -6 -16 -26 -34 -40 -44 -45 -44 -39 -33 -24 -14 -3 8 18 28 }
{-45 -41 -34 -23 -11 2 15 27 36 43 45 44 39 30 19 7 }
{ 8 22 34 42 45 43 36 24 10 -6 -20 -33 -41 -45 -44 -37 }
{ 44 38 25 9 -9 -25 -38 -44 -44 -38 -25 -9 9 25 38 44 }
{-10 -28 -40 -45 -41 -30 -12 8 26 39 45 42 31 14 -6 -24 }
{-44 -34 -15 7 27 41 45 39 23 2 -19 -36 -45 -43 -30 -11 }
{ 12 33 44 43 30 8 -16 -36 -45 -41 -26 -3 20 38 45 39 }
{ 43 29 4 -21 -40 -45 -35 -13 13 35 45 40 21 -4 -29 -43 }
{-14 -37 -45 -36 -12 16 38 45 34 10 -18 -39 -45 -33 -8 20 }
{-43 -23 7 34 45 36 11 -19 -41 -44 -27 2 30 45 39 15 }
{ 16 40 44 24 -8 -36 -45 -31 -1 30 45 37 10 -22 -43 -41 }}

T_{14} = {
{ 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 }
{-33 -34 -36 -37 -38 -39 -40 -41 -42 -43 -44 -44 -45 -45 -45 -45 }
{ 2 7 11 15 19 23 27 30 34 36 39 41 43 44 45 45 }
{ 30 24 18 12 6 -1 -8 -14 -20 -26 -31 -36 -39 -42 -44 -45 }
{-45 -43 -40 -35 -29 -21 -13 -4 4 13 21 29 35 40 43 45 }
{ 36 41 45 45 43 38 31 22 12 1 -10 -20 -30 -37 -42 -45 }
{ -7 -19 -30 -39 -44 -45 -43 -36 -27 -15 -2 11 23 34 41 45 }
{-26 -12 3 18 31 40 45 44 38 28 14 -1 -16 -30 -39 -45 }
{ 44 38 25 9 -9 -25 -38 -44 -44 -38 -25 -9 9 25 38 44 }
{-38 -45 -43 -33 -16 3 22 37 45 44 34 18 -1 -20 -36 -44 }
{ 11 30 43 45 36 19 -2 -23 -39 -45 -41 -27 -7 15 34 44 }
{ 22 -1 -24 -40 -45 -37 -18 6 28 42 45 34 14 -10 -31 -44 }
{-43 -29 -4 21 40 45 35 13 -13 -35 -45 -40 -21 4 29 43 }
{ 40 45 31 6 -22 -41 -44 -30 -3 24 42 44 28 1 -26 -43 }
{-15 -39 -45 -30 -2 27 44 41 19 -11 -36 -45 -34 -7 23 43 }
{-18 14 39 44 26 -6 -34 -45 -33 -3 28 45 38 12 -20 -42 }}

T_{21} = {
{ 42 17 -17 -42 -42 -17 17 42 42 17 -17 -42 -42 -17 17 42 }
{ 41 14 -22 -44 -37 -6 30 45 31 -3 -36 -45 -24 12 40 42 }
{ 41 11 -27 -45 -30 7 39 43 15 -23 -45 -34 2 36 44 19 }
{ 40 8 -31 -45 -22 18 44 34 -3 -38 -42 -12 28 45 26 -14 }
{ 40 4 -35 -43 -13 29 45 21 -21 -45 -29 13 43 35 -4 -40 }
{ 39 1 -38 -40 -3 37 41 6 -36 -42 -8 34 43 10 -33 -44 }
{ 39 -2 -41 -36 7 43 34 -11 -44 -30 15 45 27 -19 -45 -23 }
{ 38 -6 -43 -31 16 45 22 -26 -45 -12 34 41 1 -40 -36 10 }
{ 38 -9 -44 -25 25 44 9 -38 -38 9 44 25 -25 -44 -9 38 }
{ 37 -12 -45 -18 33 40 -6 -44 -24 28 43 1 -42 -30 22 45 }
{ 36 -15 -45 -11 39 34 -19 -45 -7 41 30 -23 -44 -2 43 27 }
{ 36 -18 -45 -3 43 24 -31 -39 12 45 10 -40 -30 26 42 -6 }
{ 35 -21 -43 4 45 13 -40 -29 29 40 -13 -45 -4 43 21 -35 }

T/AI 109.8-2025

{ 34 -24 -41 12 45 1 -45 -14 40 26 -33 -36 22 42 -10 -45 }
{ 34 -27 -39 19 43 -11 -45 2 45 7 -44 -15 41 23 -36 -30 }
{ 33 -30 -36 26 38 -22 -40 18 42 -14 -44 10 45 -6 -45 1 }}

T_{22} = {
{ 42 17 -17 -42 -42 -17 17 42 42 17 -17 -42 -42 -17 17 42 }
{ 16 -20 -44 -38 -8 28 45 33 -1 -34 -45 -26 10 39 43 18 }
{-19 -44 -36 -2 34 45 23 -15 -43 -39 -7 30 45 27 -11 -41 }
{-43 -37 -1 36 44 16 -24 -45 -30 10 41 39 6 -33 -45 -20 }
{-40 -4 35 43 13 -29 -45 -21 21 45 29 -13 -43 -35 4 40 }
{-12 31 44 14 -30 -45 -16 28 45 18 -26 -45 -20 24 45 22 }
{ 23 45 19 -27 -45 -15 30 44 11 -34 -43 -7 36 41 2 -39 }
{ 44 28 -20 -45 -18 30 44 8 -37 -39 3 42 33 -14 -45 -24 }
{ 38 -9 -44 -25 25 44 9 -38 -38 9 44 25 -25 -44 -9 38 }
{ 8 -39 -34 16 45 14 -36 -38 10 45 20 -31 -41 3 44 26 }
{-27 -43 2 44 23 -30 -41 7 45 19 -34 -39 11 45 15 -36 }
{-45 -16 37 34 -20 -44 -1 44 22 -33 -38 14 45 8 -41 -28 }
{-35 21 43 -4 -45 -13 40 29 -29 -40 13 45 4 -43 -21 35 }
{ -3 44 16 -39 -28 31 37 -20 -43 8 45 6 -44 -18 38 30 }
{ 30 36 -23 -41 15 44 -7 -45 -2 45 11 -43 -19 39 27 -34 }
{ 45 3 -45 -8 44 12 -43 -16 41 20 -39 -24 37 28 -34 -31 }}

T_{23} = {
{ 42 17 -17 -42 -42 -17 17 42 42 17 -17 -42 -42 -17 17 42 }
{-18 -43 -39 -10 26 45 34 1 -33 -45 -28 8 38 44 20 -16 }
{-41 -11 27 45 30 -7 -39 -43 -15 23 45 34 -2 -36 -44 -19 }
{ 20 45 33 -6 -39 -41 -10 30 45 24 -16 -44 -36 1 37 43 }
{ 40 4 -35 -43 -13 29 45 21 -21 -45 -29 13 43 35 -4 -40 }
{-22 -45 -24 20 45 26 -18 -45 -28 16 45 30 -14 -44 -31 12 }
{-39 2 41 36 -7 -43 -34 11 44 30 -15 -45 -27 19 45 23 }
{ 24 45 14 -33 -42 -3 39 37 -8 -44 -30 18 45 20 -28 -44 }
{ 38 -9 -44 -25 25 44 9 -38 -38 9 44 25 -25 -44 -9 38 }
{-26 -44 -3 41 31 -20 -45 -10 38 36 -14 -45 -16 34 39 -8 }
{-36 15 45 11 -39 -34 19 45 7 -41 -30 23 44 2 -43 -27 }
{ 28 41 -8 -45 -14 38 33 -22 -44 1 44 20 -34 -37 16 45 }
{ 35 -21 -43 4 45 13 -40 -29 29 40 -13 -45 -4 43 21 -35 }
{-30 -38 18 44 -6 -45 -8 43 20 -37 -31 28 39 -16 -44 3 }
{-34 27 39 -19 -43 11 45 -2 -45 -7 44 15 -41 -23 36 30 }
{ 31 34 -28 -37 24 39 -20 -41 16 43 -12 -44 8 45 -3 -45 }}

T_{24} = {
{ 42 17 -17 -42 -42 -17 17 42 42 17 -17 -42 -42 -17 17 42 }
{-42 -40 -12 24 45 36 3 -31 -45 -30 6 37 44 22 -14 -41 }
{ 19 44 36 2 -34 -45 -23 15 43 39 7 -30 -45 -27 11 41 }
{ 14 -26 -45 -28 12 42 38 3 -34 -44 -18 22 45 31 -8 -40 }

T/AI 109.8-2025

{-40 -4 35 43 13 -29 -45 -21 21 45 29 -13 -43 -35 4 40 }
{ 44 33 -10 -43 -34 8 42 36 -6 -41 -37 3 40 38 -1 -39 }
{-23 -45 -19 27 45 15 -30 -44 -11 34 43 7 -36 -41 -2 39 }
{-10 36 40 -1 -41 -34 12 45 26 -22 -45 -16 31 43 6 -38 }
{ 38 -9 -44 -25 25 44 9 -38 -38 9 44 25 -25 -44 -9 38 }
{-45 -22 30 42 -1 -43 -28 24 44 6 -40 -33 18 45 12 -37 }
{ 27 43 -2 -44 -23 30 41 -7 -45 -19 34 39 -11 -45 -15 36 }
{ 6 -42 -26 30 40 -10 -45 -12 39 31 -24 -43 3 45 18 -36 }
{-35 21 43 -4 -45 -13 40 29 -29 -40 13 45 4 -43 -21 35 }
{ 45 10 -42 -22 36 33 -26 -40 14 45 -1 -45 -12 41 24 -34 }
{-30 -36 23 41 -15 -44 7 45 2 -45 -11 43 19 -39 -27 34 }
{ -1 45 6 -45 -10 44 14 -42 -18 40 22 -38 -26 36 30 -33 }}

T_{31} = {
{ 32 -32 -32 32 32 -32 -32 32 32 -32 -32 32 32 -32 -32 32 }
{ 31 -34 -28 37 24 -39 -20 41 16 -43 -12 44 8 -45 -3 45 }
{ 30 -36 -23 41 15 -44 -7 45 -2 -45 11 43 -19 -39 27 34 }
{ 30 -38 -18 44 6 -45 8 43 -20 -37 31 28 -39 -16 44 3 }
{ 29 -40 -13 45 -4 -43 21 35 -35 -21 43 4 -45 13 40 -29 }
{ 28 -41 -8 45 -14 -38 33 22 -44 -1 44 -20 -34 37 16 -45 }
{ 27 -43 -2 44 -23 -30 41 7 -45 19 34 -39 -11 45 -15 -36 }
{ 26 -44 3 41 -31 -20 45 -10 -38 36 14 -45 16 34 -39 -8 }
{ 25 -44 9 38 -38 -9 44 -25 -25 44 -9 -38 38 9 -44 25 }
{ 24 -45 14 33 -42 3 39 -37 -8 44 -30 -18 45 -20 -28 44 }
{ 23 -45 19 27 -45 15 30 -44 11 34 -43 7 36 -41 2 39 }
{ 22 -45 24 20 -45 26 18 -45 28 16 -45 30 14 -44 31 12 }
{ 21 -45 29 13 -43 35 4 -40 40 -4 -35 43 -13 -29 45 -21 }
{ 20 -45 33 6 -39 41 -10 -30 45 -24 -16 44 -36 -1 37 -43 }
{ 19 -44 36 -2 -34 45 -23 -15 43 -39 7 30 -45 27 11 -41 }
{ 18 -43 39 -10 -26 45 -34 1 33 -45 28 8 -38 44 -20 -16 }}

T_{32} = {
{ 32 -32 -32 32 32 -32 -32 32 32 -32 -32 32 32 -32 -32 32 }
{ -1 -45 6 45 -10 -44 14 42 -18 -40 22 38 -26 -36 30 33 }
{-34 -27 39 19 -43 -11 45 2 -45 7 44 -15 -41 23 36 -30 }
{-45 10 42 -22 -36 33 26 -40 -14 45 1 -45 12 41 -24 -34 }
{-29 40 13 -45 4 43 -21 -35 35 21 -43 -4 45 -13 -40 29 }
{ 6 42 -26 -30 40 10 -45 12 39 -31 -24 43 3 -45 18 36 }
{ 36 15 -45 11 39 -34 -19 45 -7 -41 30 23 -44 2 43 -27 }
{ 45 -22 -30 42 1 -43 28 24 -44 6 40 -33 -18 45 -12 -37 }
{ 25 -44 9 38 -38 -9 44 -25 -25 44 -9 -38 38 9 -44 25 }
{-10 -36 40 1 -41 34 12 -45 26 22 -45 16 31 -43 6 38 }
{-39 -2 41 -36 -7 43 -34 -11 44 -30 -15 45 -27 -19 45 -23 }
{-44 33 10 -43 34 8 -42 36 6 -41 37 3 -40 38 1 -39 }

T/AI 109.8-2025

{-21 45 -29 -13 43 -35 -4 40 -40 4 35 -43 13 29 -45 21 }
{ 14 26 -45 28 12 -42 38 -3 -34 44 -18 -22 45 -31 -8 40 }
{ 41 -11 -27 45 -30 -7 39 -43 15 23 -45 34 2 -36 44 -19 }
{ 42 -40 12 24 -45 36 -3 -31 45 -30 -6 37 -44 22 14 -41 }}

T_{33} = {
{ 32 -32 -32 32 32 -32 -32 32 32 -32 -32 32 32 -32 -32 32 }
{-33 -30 36 26 -38 -22 40 18 -42 -14 44 10 -45 -6 45 1 }
{-30 36 23 -41 -15 44 7 -45 2 45 -11 -43 19 39 -27 -34 }
{ 34 24 -41 -12 45 -1 -45 14 40 -26 -33 36 22 -42 -10 45 }
{ 29 -40 -13 45 -4 -43 21 35 -35 -21 43 4 -45 13 40 -29 }
{-36 -18 45 -3 -43 24 31 -39 -12 45 -10 -40 30 26 -42 -6 }
{-27 43 2 -44 23 30 -41 -7 45 -19 -34 39 11 -45 15 36 }
{ 37 12 -45 18 33 -40 -6 44 -24 -28 43 -1 -42 30 22 -45 }
{ 25 -44 9 38 -38 -9 44 -25 -25 44 -9 -38 38 9 -44 25 }
{-38 -6 43 -31 -16 45 -22 -26 45 -12 -34 41 -1 -40 36 10 }
{-23 45 -19 -27 45 -15 -30 44 -11 -34 43 -7 -36 41 -2 -39 }
{ 39 -1 -38 40 -3 -37 41 -6 -36 42 -8 -34 43 -10 -33 44 }
{ 21 -45 29 13 -43 35 4 -40 40 -4 -35 43 -13 -29 45 -21 }
{-40 8 31 -45 22 18 -44 34 3 -38 42 -12 -28 45 -26 -14 }
{-19 44 -36 2 34 -45 23 15 -43 39 -7 -30 45 -27 -11 41 }
{ 41 -14 -22 44 -37 6 30 -45 31 3 -36 45 -24 -12 40 -42 }}

T_{34} = {
{ 32 -32 -32 32 32 -32 -32 32 32 -32 -32 32 32 -32 -32 32 }
{-45 3 45 -8 -44 12 43 -16 -41 20 39 -24 -37 28 34 -31 }
{ 34 27 -39 -19 43 11 -45 -2 45 -7 -44 15 41 -23 -36 30 }
{ -3 -44 16 39 -28 -31 37 20 -43 -8 45 -6 -44 18 38 -30 }
{-29 40 13 -45 4 43 -21 -35 35 21 -43 -4 45 -13 -40 29 }
{ 45 -16 -37 34 20 -44 1 44 -22 -33 38 14 -45 8 41 -28 }
{-36 -15 45 -11 -39 34 19 -45 7 41 -30 -23 44 -2 -43 27 }
{ 8 39 -34 -16 45 -14 -36 38 10 -45 20 31 -41 -3 44 -26 }
{ 25 -44 9 38 -38 -9 44 -25 -25 44 -9 -38 38 9 -44 25 }
{-44 28 20 -45 18 30 -44 8 37 -39 -3 42 -33 -14 45 -24 }
{ 39 2 -41 36 7 -43 34 11 -44 30 15 -45 27 19 -45 23 }
{-12 -31 44 -14 -30 45 -16 -28 45 -18 -26 45 -20 -24 45 -22 }
{-21 45 -29 -13 43 -35 -4 40 -40 4 35 -43 13 29 -45 21 }
{ 43 -37 1 36 -44 16 24 -45 30 10 -41 39 -6 -33 45 -20 }
{-41 11 27 -45 30 7 -39 43 -15 -23 45 -34 -2 36 -44 19 }
{ 16 20 -44 38 -8 -28 45 -33 -1 34 -45 26 10 -39 43 -18 }}

T_{41} = {
{ 17 -42 42 -17 -17 42 -42 17 17 -42 42 -17 -17 42 -42 17 }
{ 16 -40 44 -24 -8 36 -45 31 -1 -30 45 -37 10 22 -43 41 }
{ 15 -39 45 -30 2 27 -44 41 -19 -11 36 -45 34 -7 -23 43 }

T/AI 109.8-2025

{ 14 -37 45 -36 12 16 -38 45 -34 10 18 -39 45 -33 8 20 }
{ 13 -35 45 -40 21 4 -29 43 -43 29 -4 -21 40 -45 35 -13 }
{ 12 -33 44 -43 30 -8 -16 36 -45 41 -26 3 20 -38 45 -39 }
{ 11 -30 43 -45 36 -19 -2 23 -39 45 -41 27 -7 -15 34 -44 }
{ 10 -28 40 -45 41 -30 12 8 -26 39 -45 42 -31 14 6 -24 }
{ 9 -25 38 -44 44 -38 25 -9 -9 25 -38 44 -44 38 -25 9 }
{ 8 -22 34 -42 45 -43 36 -24 10 6 -20 33 -41 45 -44 37 }
{ 7 -19 30 -39 44 -45 43 -36 27 -15 2 11 -23 34 -41 45 }
{ 6 -16 26 -34 40 -44 45 -44 39 -33 24 -14 3 8 -18 28 }
{ 4 -13 21 -29 35 -40 43 -45 45 -43 40 -35 29 -21 13 -4 }
{ 3 -10 16 -22 28 -33 37 -40 43 -45 45 -45 44 -41 38 -34 }
{ 2 -7 11 -15 19 -23 27 -30 34 -36 39 -41 43 -44 45 -45 }
{ 1 -3 6 -8 10 -12 14 -16 18 -20 22 -24 26 -28 30 -31 }}

T_{42} = {
{ 17 -42 42 -17 -17 42 -42 17 17 -42 42 -17 -17 42 -42 17 }
{-18 -14 39 -44 26 6 -34 45 -33 3 28 -45 38 -12 -20 42 }
{-43 23 7 -34 45 -36 11 19 -41 44 -27 -2 30 -45 39 -15 }
{-40 45 -31 6 22 -41 44 -30 3 24 -42 44 -28 1 26 -43 }
{-13 35 -45 40 -21 -4 29 -43 43 -29 4 21 -40 45 -35 13 }
{ 22 1 -24 40 -45 37 -18 -6 28 -42 45 -34 14 10 -31 44 }
{ 44 -34 15 7 -27 41 -45 39 -23 2 19 -36 45 -43 30 -11 }
{ 38 -45 43 -33 16 3 -22 37 -45 44 -34 18 1 -20 36 -44 }
{ 9 -25 38 -44 44 -38 25 -9 -9 25 -38 44 -44 38 -25 9 }
{-26 12 3 -18 31 -40 45 -44 38 -28 14 1 -16 30 -39 45 }
{-45 41 -34 23 -11 -2 15 -27 36 -43 45 -44 39 -30 19 -7 }
{-36 41 -45 45 -43 38 -31 22 -12 1 10 -20 30 -37 42 -45 }
{ -4 13 -21 29 -35 40 -43 45 -45 43 -40 35 -29 21 -13 4 }
{ 30 -24 18 -12 6 1 -8 14 -20 26 -31 36 -39 42 -44 45 }
{ 45 -45 44 -43 41 -39 36 -34 30 -27 23 -19 15 -11 7 -2 }
{ 33 -34 36 -37 38 -39 40 -41 42 -43 44 -44 45 -45 45 -45 }}

T_{43} = {
{ 17 -42 42 -17 -17 42 -42 17 17 -42 42 -17 -17 42 -42 17 }
{-42 20 12 -38 45 -28 -3 33 -45 34 -6 -26 44 -39 14 18 }
{-15 39 -45 30 -2 -27 44 -41 19 11 -36 45 -34 7 23 -43 }
{ 43 -26 -1 28 -44 42 -24 -3 30 -44 41 -22 -6 31 -45 40 }
{ 13 -35 45 -40 21 4 -29 43 -43 29 -4 -21 40 -45 35 -13 }
{-44 31 -10 -14 34 -45 42 -28 6 18 -37 45 -40 24 -1 -22 }
{-11 30 -43 45 -36 19 2 -23 39 -45 41 -27 7 15 -34 44 }
{ 44 -36 20 -1 -18 34 -44 45 -37 22 -3 -16 33 -43 45 -38 }
{ 9 -25 38 -44 44 -38 25 -9 -9 25 -38 44 -44 38 -25 9 }
{-45 39 -30 16 -1 -14 28 -38 44 -45 40 -31 18 -3 -12 26 }
{ -7 19 -30 39 -44 45 -43 36 -27 15 -2 -11 23 -34 41 -45 }

T/AI 109.8-2025

{ 45 -42 37 -30 20 -10 -1 12 -22 31 -38 43 -45 45 -41 36 }
{ 4 -13 21 -29 35 -40 43 -45 45 -43 40 -35 29 -21 13 -4 }
{-45 44 -42 39 -36 31 -26 20 -14 8 -1 -6 12 -18 24 -30 }
{ -2 7 -11 15 -19 23 -27 30 -34 36 -39 41 -43 44 -45 45 }
{ 45 -45 45 -45 44 -44 43 -42 41 -40 39 -38 37 -36 34 -33 }}

T_{44} = {
{ 17 -42 42 -17 -17 42 -42 17 17 -42 42 -17 -17 42 -42 17 }
{-41 43 -22 -10 37 -45 30 1 -31 45 -36 8 24 -44 40 -16 }
{ 43 -23 -7 34 -45 36 -11 -19 41 -44 27 2 -30 45 -39 15 }
{-20 -8 33 -45 39 -18 -10 34 -45 38 -16 -12 36 -45 37 -14 }
{-13 35 -45 40 -21 -4 29 -43 43 -29 4 21 -40 45 -35 13 }
{ 39 -45 38 -20 -3 26 -41 45 -36 16 8 -30 43 -44 33 -12 }
{-44 34 -15 -7 27 -41 45 -39 23 -2 -19 36 -45 43 -30 11 }
{ 24 -6 -14 31 -42 45 -39 26 -8 -12 30 -41 45 -40 28 -10 }
{ 9 -25 38 -44 44 -38 25 -9 -9 25 -38 44 -44 38 -25 9 }
{-37 44 -45 41 -33 20 -6 -10 24 -36 43 -45 42 -34 22 -8 }
{ 45 -41 34 -23 11 2 -15 27 -36 43 -45 44 -39 30 -19 7 }
{-28 18 -8 -3 14 -24 33 -39 44 -45 44 -40 34 -26 16 -6 }
{ -4 13 -21 29 -35 40 -43 45 -45 43 -40 35 -29 21 -13 4 }
{ 34 -38 41 -44 45 -45 45 -43 40 -37 33 -28 22 -16 10 -3 }
{-45 45 -44 43 -41 39 -36 34 -30 27 -23 19 -15 11 -7 2 }
{ 31 -30 28 -26 24 -22 20 -18 16 -14 12 -10 8 -6 3 -1 }}

N.2 DCT8 型反变换矩阵

N.2.1 4×4 DCT8反变换矩阵

4×4 DCT8 反变换矩阵规定如下：

$DCT8_4$ = {
{ 42 37 27 15 }
{ 37 0 -37 -37 }
{ 27 -37 -15 42 }
{ 15 -37 42 -27 }}

N.2.2 8×8 DCT8反变换矩阵

8×8 DCT8 反变换矩阵规定如下：

$DCT8_8$ = {
{ 44 42 39 35 30 23 16 8 }
{ 42 30 8 -16 -35 -44 -39 -23 }
{ 39 8 -30 -44 -23 16 42 35 }
{ 35 -16 -44 -8 39 30 -23 -42 }
{ 30 -35 -23 39 16 -42 -8 44 }

T/AI 109.8-2025

{ 23 -44 16 30 -42 8 35 -39 }
{ 16 -39 42 -23 -8 35 -44 30 }
{ 8 -23 35 -42 44 -39 30 -16 }}

N.2.3 16×16 DCT8反变换矩阵

16×16 DCT8 反变换矩阵规定如下：

$DCT8_{16}$ = {
{ 45 44 43 42 41 39 36 34 31 28 24 20 17 13 8 4 }
{ 44 41 34 24 13 0 -13 -24 -34 -41 -44 -44 -41 -34 -24 -13 }
{ 43 34 17 -4 -24 -39 -45 -41 -28 -8 13 31 42 44 36 20 }
{ 42 24 -4 -31 -44 -39 -17 13 36 45 34 8 -20 -41 -43 -28 }
{ 41 13 -24 -44 -34 0 34 44 24 -13 -41 -41 -13 24 44 34 }
{ 39 0 -39 -39 0 39 39 0 -39 -39 0 39 39 0 -39 -39 }
{ 36 -13 -45 -17 34 39 -8 -44 -20 31 41 -4 -43 -24 28 42 }
{ 34 -24 -41 13 44 0 -44 -13 41 24 -34 -34 24 41 -13 -44 }
{ 31 -34 -28 36 24 -39 -20 41 17 -42 -13 43 8 -44 -4 45 }
{ 28 -41 -8 45 -13 -39 31 24 -42 -4 44 -17 -36 34 20 -43 }
{ 24 -44 13 34 -41 0 41 -34 -13 44 -24 -24 44 -13 -34 41 }
{ 20 -44 31 8 -41 39 -4 -34 43 -17 -24 45 -28 -13 42 -36 }
{ 17 -41 42 -20 -13 39 -43 24 8 -36 44 -28 -4 34 -45 31 }
{ 13 -34 44 -41 24 0 -24 41 -44 34 -13 -13 34 -44 41 -24 }
{ 8 -24 36 -43 44 -39 28 -13 -4 20 -34 42 -45 41 -31 17 }
{ 4 -13 20 -28 34 -39 42 -44 45 -43 41 -36 31 -24 17 -8 }}

N.3 DST7 型反变换矩阵

N.3.1 4×4 DST7反变换矩阵

4×4 DST7 反变换矩阵规定如下：

$DST7_4$ = {
{ 15 27 37 42 }
{ 37 37 0 -37 }
{ 42 -15 -37 27 }
{ 27 -42 37 -15 }}

N.3.2 8×8 DST7反变换矩阵

8×8 DST7 反变换矩阵规定如下：

$DST7_8$ = {
{ 8 16 23 30 35 39 42 44 }
{ 23 39 44 35 16 -8 -30 -42 }
{ 35 42 16 -23 -44 -30 8 39 }
{ 42 23 -30 -39 8 44 16 -35 }

T/AI 109.8-2025

{ 44 -8 -42 16 39 -23 -35 30 }
{ 39 -35 -8 42 -30 -16 44 -23 }
{ 30 -44 35 -8 -23 42 -39 16 }
{ 16 -30 39 -44 42 -35 23 -8 }}

N.3.3 16×16 DST7反变换矩阵

16×16 DST7 反变换矩阵规定如下：

$DST7_{16}$ = {
{ 4 8 13 17 20 24 28 31 34 36 39 41 42 43 44 45 }
{ 13 24 34 41 44 44 41 34 24 13 0 -13 -24 -34 -41 -44 }
{ 20 36 44 42 31 13 -8 -28 -41 -45 -39 -24 -4 17 34 43 }
{ 28 43 41 20 -8 -34 -45 -36 -13 17 39 44 31 4 -24 -42 }
{ 34 44 24 -13 -41 -41 -13 24 44 34 0 -34 -44 -24 13 41 }
{ 39 39 0 -39 -39 0 39 39 0 -39 -39 0 39 39 0 -39 }
{ 42 28 -24 -43 -4 41 31 -20 -44 -8 39 34 -17 -45 -13 36 }
{ 44 13 -41 -24 34 34 -24 -41 13 44 0 -44 -13 41 24 -34 }
{ 45 -4 -44 8 43 -13 -42 17 41 -20 -39 24 36 -28 -34 31 }
{ 43 -20 -34 36 17 -44 4 42 -24 -31 39 13 -45 8 41 -28 }
{ 41 -34 -13 44 -24 -24 44 -13 -34 41 0 -41 34 13 -44 24 }
{ 36 -42 13 28 -45 24 17 -43 34 4 -39 41 -8 -31 44 -20 }
{ 31 -45 34 -4 -28 44 -36 8 24 -43 39 -13 -20 42 -41 17 }
{ 24 -41 44 -34 13 13 -34 44 -41 24 0 -24 41 -44 34 -13 }
{ 17 -31 41 -45 42 -34 20 -4 -13 28 -39 44 -43 36 -24 8 }
{ 8 -17 24 -31 36 -41 43 -45 44 -42 39 -34 28 -20 13 -4 }}

N.3.4 32×32 DST7反变换矩阵

32×32 DST7 反变换矩阵规定如下：

$DST7_{32}$ = {
{T_{11} T_{12}}
{T_{21} T_{22}}}
其中，T_{11} = {
{ 2 4 6 9 11 13 15 17 19 21 23 25 26 28 30 31}
{ 6 13 19 25 30 34 38 41 43 45 45 44 43 40 37 33}
{ 11 21 30 37 42 45 45 42 37 30 21 11 0 -11 -21 -30}
{ 15 28 38 44 45 40 31 19 4 -11 -25 -36 -43 -45 -42 -34}
{ 19 34 43 44 37 23 4 -15 -31 -42 -45 -39 -26 -9 11 28}
{ 23 39 45 38 21 -2 -25 -40 -45 -37 -19 4 26 41 45 36}
{ 26 43 43 26 0 -26 -43 -43 -26 0 26 43 43 26 0 -26}
{ 30 45 37 11 -21 -42 -42 -21 11 37 45 30 0 -30 -45 -37}
{ 33 45 28 -6 -37 -44 -23 13 40 42 17 -19 -43 -39 -11 25}
{ 36 43 17 -23 -45 -31 6 39 41 11 -28 -45 -26 13 42 38}
{ 38 40 4 -36 -42 -9 33 43 13 -30 -44 -17 26 45 21 -23}

T/AI 109.8-2025

{ 40 36 -9 -43 -30 17 45 23 -25 -45 -15 31 43 6 -37 -39}
{ 42 30 -21 -45 -11 37 37 -11 -45 -21 30 42 0 -42 -30 21}
{ 43 23 -31 -39 11 45 13 -38 -33 21 44 2 -43 -25 30 40}
{ 44 15 -39 -28 30 38 -17 -44 2 45 13 -40 -26 31 37 -19}
{ 45 6 -44 -13 42 19 -39 -25 36 30 -31 -34 26 38 -21 -41}}

T_{12} = {
{ 33 34 36 37 38 39 40 41 42 43 43 44 44 45 45 45}
{ 28 23 17 11 4 -2 -9 -15 -21 -26 -31 -36 -39 -42 -44 -45}
{-37 -42 -45 -45 -42 -37 -30 -21 -11 0 11 21 30 37 42 45}
{-23 -9 6 21 33 41 45 43 37 26 13 -2 -17 -30 -39 -44}
{ 40 45 41 30 13 -6 -25 -38 -45 -43 -33 -17 2 21 36 44}
{ 17 -6 -28 -42 -44 -34 -15 9 30 43 44 33 13 -11 -31 -43}
{-43 -43 -26 0 26 43 43 26 0 -26 -43 -43 -26 0 26 43}
{-11 21 42 42 21 -11 -37 -45 -30 0 30 45 37 11 -21 -42}
{ 44 36 4 -30 -45 -31 2 34 45 26 -9 -38 -43 -21 15 41}
{ 4 -33 -44 -21 19 44 34 -2 -37 -43 -15 25 45 30 -9 -40}
{-45 -25 19 45 28 -15 -44 -31 11 43 34 -6 -41 -37 2 39}
{ 2 41 34 -11 -44 -28 19 45 21 -26 -44 -13 33 42 4 -38}
{ 45 11 -37 -37 11 45 21 -30 -42 0 42 30 -21 -45 -11 37}
{ -9 -45 -15 37 34 -19 -44 -4 42 26 -28 -41 6 45 17 -36}
{-43 4 45 11 -41 -25 33 36 -21 -43 6 45 9 -42 -23 34}
{ 15 43 -9 -45 2 45 4 -44 -11 43 17 -40 -23 37 28 -33}}

T_{21} = {
{ 45 -2 -45 4 45 -6 -44 9 44 -11 -43 13 43 -15 -42 17}
{ 45 -11 -42 21 37 -30 -30 37 21 -42 -11 45 0 -45 11 42}
{ 44 -19 -36 34 21 -43 -2 44 -17 -37 33 23 -43 -4 45 -15}
{ 43 -26 -26 43 0 -43 26 26 -43 0 43 -26 -26 43 0 -43}
{ 41 -33 -15 45 -21 -28 43 -6 -38 37 9 -44 26 23 -45 13}
{ 39 -38 -2 40 -37 -4 41 -36 -6 42 -34 -9 43 -33 -11 43}
{ 37 -42 11 30 -45 21 21 -45 30 11 -42 37 0 -37 42 -11}
{ 34 -44 23 15 -42 39 -9 -28 45 -30 -6 38 -43 17 21 -44}
{ 31 -45 33 -2 -30 45 -34 4 28 -45 36 -6 -26 44 -37 9}
{ 28 -44 40 -19 -11 36 -45 34 -9 -21 41 -43 26 2 -30 44}
{ 25 -41 44 -33 11 15 -36 45 -39 21 4 -28 43 -43 30 -6}
{ 21 -37 45 -42 30 -11 -11 30 -42 45 -37 21 0 -21 37 -45}
{ 17 -31 41 -45 42 -33 19 -2 -15 30 -40 45 -43 34 -21 4}
{ 13 -25 34 -41 45 -44 40 -33 23 -11 -2 15 -26 36 -42 45}
{ 9 -17 25 -31 37 -41 44 -45 44 -42 38 -33 26 -19 11 -2}
{ 4 -9 13 -17 21 -25 28 -31 34 -37 39 -41 43 -44 45 -45}}

T_{22} = {
{ 41 -19 -40 21 39 -23 -38 25 37 -26 -36 28 34 -30 -33 31}
{-21 -37 30 30 -37 -21 42 11 -45 0 45 -11 -42 21 37 -30}

T/AI 109.8-2025

```
{-38  31  25 -42  -6  45 -13 -39  30  26 -41  -9  45 -11 -40  28}
{ 26  26 -43   0  43 -26 -26  43   0 -43  26  26 -43   0  43 -26}
{ 34 -40  -2  42 -31 -17  45 -19 -30  43  -4 -39  36  11 -44  25}
{-31 -13  44 -30 -15  44 -28 -17  45 -26 -19  45 -25 -21  45 -23}
{-30  45 -21 -21  45 -30 -11  42 -37   0  37 -42  11  30 -45  21}
{ 36  -2 -33  45 -25 -13  41 -40  11  26 -45  31   4 -37  43 -19}
{ 25 -44  38 -11 -23  43 -39  13  21 -43  40 -15 -19  42 -41  17}
{-39  17  13 -37  45 -33   6  23 -42  43 -25  -4  31 -45  38 -15}
{-19  38 -45  37 -17  -9  31 -44  42 -26   2  23 -40  45 -34  13}
{ 42 -30  11  11 -30  42 -45  37 -21   0  21 -37  45 -42  30 -11}
{ 13 -28  39 -45  43 -36  23  -6 -11  26 -38  44 -44  37 -25   9}
{-44  39 -31  21  -9  -4  17 -28  37 -43  45 -43  38 -30  19  -6}
{ -6  15 -23  30 -36  40 -43  45 -45  43 -39  34 -28  21 -13   4}
{ 45 -44  43 -42  40 -38  36 -33  30 -26  23 -19  15 -11   6  -2}}
```

T/AI 109.8-2025

附 录 O
（资料性）
智能分割图表示编码模式示例

O.1 概述

本附录是智能分割信息中图表示方案（AISeg_mode 的值等于 1）的参考实现举例。一般情况下，智能分割模型输出的智能分割结果为单通道的带类别标签的像素图。在图表示方案下，用户需要将该像素图转为编码器支持的图的型式。为提高编码效率，本模式可对类别标签进行映射（从 map_label 到 map_value）与反映射（从 map_value 到 map_label）。用户可以自行规定映射关系。

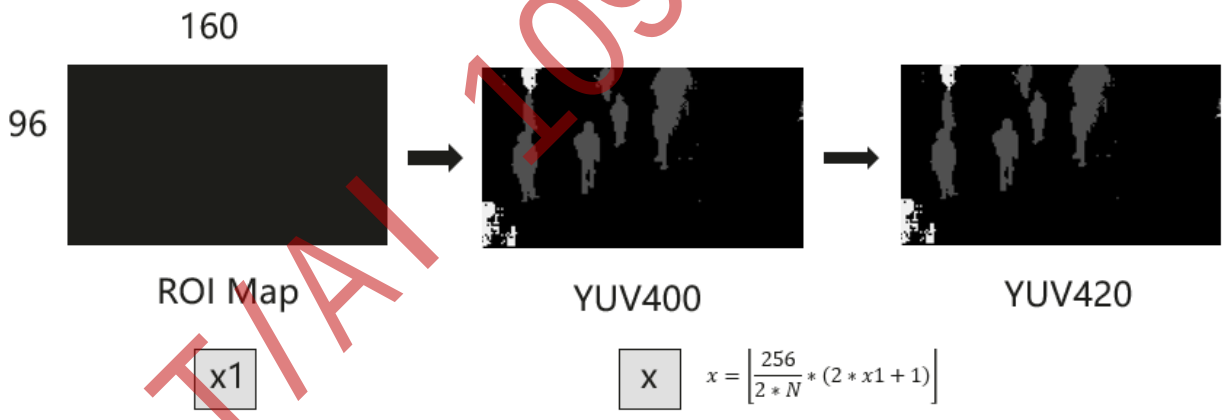
O.2 方法

智能分割图的编码前处理流程如图 S.1 所示，具体步骤为：

步骤 1：将智能分割图进行缩放到指定尺寸，如 160x90。

步骤 2：规定 map_label 到 map_value 的映射关系，对智能分割图映射的像素值进行映射。推荐采用均匀映射的方式，将 map_label 值均匀映射至像素取值区间（8bit 值对应区间为[0,255]）。假定像不同像素值数为 N，可映射为$\left\lfloor \frac{256}{2*N} * (2 * i + 1) \right\rfloor$，其中 i 为 0 到 N-1 的整数，此处$\lfloor \rfloor$表示向下取整。如 map_label={1,2,3}，则 N=3，可映射为 map_value={43，128，213}。

步骤 3：通道填充，将原先单通道的分割图，转换为 YUV420 图。填充值为 128 或 0 的 UV 通道。



图S.1 智能分割图前处理流程

智能分割图的编码后处理流程，具体步骤为：

步骤 1：取 YUV420 图的 Y 通道为初始分割图。

步骤 2：反映射，将初始分割图按 map_value 到 map_label。由于编解码存在一定的损失，故初始分割图的像素值不一定与 map_value 中的值一致，应匹配与 map_value 中最接近的值。即对于像素值x，求$\underset{i}{\operatorname{argmin}}(x - map_value[i])$。如 map_label={1,2,3}，map_value={43，128，213}的情况，对于像素值{36，100，180，15}，则其对应i的值为{0,1,2,0}，对应映射的 map_label 值为{1,2,3,1}。

T/AI 109.8-2025