(19)国家知识产权局

(12)发明专利申请

(10)申请公布号 CN 117499659 A
(43)申请公布日 2024.02.02

(21)申请号 202311447045.9

(22)申请日 2020.07.24

(66)本国优先权数据
PCT/CN2019/097742 2019.07.25 CN
PCT/CN2019/109849 2019.10.07 CN

(62)分案原申请数据
202080053763.6 2020.07.24

(71)申请人 北京字节跳动网络技术有限公司
地址 100041 北京市石景山区实兴大街30号院3号楼2层B-0035房间
申请人 字节跳动有限公司

(72)发明人 许继征 张莉 张凯 刘鸿彬

(74)专利代理机构 北京市柳沈律师事务所 11105
专利代理师 张亮 刘文洁

(51)Int.Cl.
H04N 19/20(2014.01)
H04N 19/42(2014.01)
H04N 19/172(2014.01)
H04N 19/159(2014.01)
H04N 19/174(2014.01)

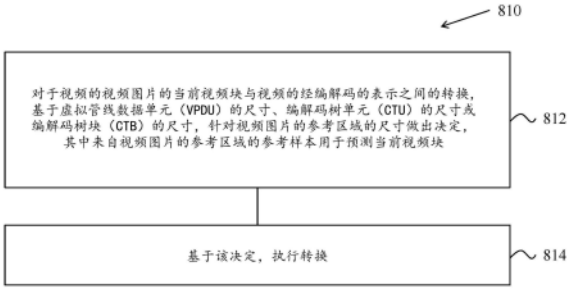
权利要求书3页 说明书26页 附图9页

(54)发明名称

帧内块复制虚拟缓冲区的尺寸限制

(57)摘要

描述了一种用于视频处理的方法。该方法包括:对于视频的视频图片的当前视频块与视频的经编解码的表示之间的转换,基于虚拟管线数据单元(VPDU)的尺寸、编解码树块(CTB)的尺寸或编解码树单元(CTU)的尺寸,针对视频图片的参考区域的尺寸做出决定,其中来自视频图片的参考区域的参考样本用于预测当前视频块;以及基于该决定,执行转换。



CN 117499659 A

1.一种处理视频数据的方法,包括:

对于视频的视频图片的当前视频块和所述视频的比特流之间的转换,确定预测模式被应用于所述当前视频块;

对于所述预测模式,维持包括从所述视频图片推导的参考样本虚拟缓冲区,其中所述虚拟缓冲区的尺寸基于包括所述当前视频块的编解码树块CTB的尺寸而确定;

在所述预测模式中,基于指向所述虚拟缓冲区中的所述参考样本的块矢量而推导所述当前视频块的预测样本;以及

基于所述预测样本进行所述转换。

2.如权利要求1所述的方法,其中,所述预测模式是帧内块复制IBC模式和所述虚拟缓冲区是IBC虚拟缓冲区。

3.如权利要求1所述的方法,其中,所述虚拟缓冲区的高度和宽度的乘积是固定的,并且所述虚拟缓冲区的所述宽度是基于所述CTB尺寸确定的。

4.如权利要求3所述的方法,其中,所述虚拟缓冲区的高度等于所述CTB的高度,并且所述虚拟缓冲区的宽度被设置为通过将所述虚拟缓冲区的面积除以所述CTB的高度所获得的值。

5.如权利要求4所述的方法,其中,所述虚拟缓冲区的面积是256*128,所述虚拟缓冲区的高度等于ctbSizeY,并且所述虚拟缓冲区的宽度是256*128/ctbSizeY,其中ctbSizeY表示包括所述当前视频块的亮度CTB的尺寸。

6.如权利要求1所述的方法,其中,所述虚拟缓冲区的宽度是虚拟单元的宽度的一倍或多倍,

其中所述虚拟单元的宽度是min(ctbSize,64),其中ctbSize表示所述CTB的尺寸。

7.如权利要求1所述的方法,其中,所述虚拟缓冲区的高度是虚拟单元的高度的一倍或多倍,

其中所述虚拟单元的高度是min(ctbSize,64),其中ctbSize表示所述CTB的尺寸。

8.如权利要求1所述的方法,还包括:

基于虚拟单元的尺寸和所述CTB的尺寸确定所述虚拟缓冲区中的一个或多个样本的有效性,

其中所述虚拟单元的尺寸是min(ctbSize,64),其中ctbSize表示所述CTB的尺寸。

9.如权利要求8所述的方法,其中,对应于所述虚拟单元的所述虚拟缓冲区中的块的样本被设置为-1。

10.如权利要求1所述的方法,其中,所述比特流满足的比特流一致性要求包括:由指向所述虚拟缓冲区中的所述参考样本的所述块矢量确定的参考块不具有等于-1的值。

11.如权利要求1所述的方法,其中,所述虚拟缓冲区在每个编解码树单元行的开始处用值-1刷新。

12.如权利要求1所述的方法,其中,所述转换包括将所述当前视频块编码为所述比特流。

13.如权利要求1所述的方法,其中,所述转换包括将所述当前视频块从所述比特流解码。

14.一种处理视频数据的装置,包括处理器和其上具有指令的非瞬时性存储器,其中,

所述指令当由所述处理器执行时,使得所述处理器:

对于视频的视频图片的当前视频块和所述视频的比特流之间的转换,确定预测模式被应用于所述当前视频块;

对于所述预测模式,维持包括从所述视频图片推导的参考样本虚拟缓冲区,其中所述虚拟缓冲区的尺寸基于包括所述当前视频块的编解码树块CTB的尺寸而确定;

在所述预测模式中,基于指向所述虚拟缓冲区中的所述参考样本的块矢量而推导所述当前视频块的预测样本;以及

基于所述预测样本进行所述转换。

15.如权利要求14所述的装置,其中所述预测模式是帧内块复制IBC模式和所述虚拟缓冲区是IBC虚拟缓冲区。

16.如权利要求14所述的装置,其中所述虚拟缓冲区的高度和宽度的乘积是固定的,并且所述虚拟缓冲区的所述宽度是基于所述CTB尺寸确定的。

17.如权利要求16所述的装置,其中所述虚拟缓冲区的高度等于所述CTB的高度,并且所述虚拟缓冲区的宽度被设置为通过将所述虚拟缓冲区的面积除以所述CTB的高度所获得的值。

18.如权利要求17所述的装置,其中所述虚拟缓冲区的面积是256*128,所述虚拟缓冲区的高度等于ctbSizeY,并且所述虚拟缓冲区的宽度是256*128/ctbSizeY,其中ctbSizeY表示包括所述当前视频块的亮度CTB的尺寸。

19.一种非瞬时性计算机可读储存介质,存储使得处理器进行以下的指令:

对于视频的视频图片的当前视频块和所述视频的比特流之间的转换,确定预测模式被应用于所述当前视频块;

对于所述预测模式,维持包括从所述视频图片的参考区域推导的参考样本虚拟缓冲区,其中所述虚拟缓冲区的尺寸基于包括所述当前视频块的编解码树块CTB的尺寸而确定;

在所述预测模式中,基于指向所述虚拟缓冲区中的所述参考样本的块矢量而推导所述当前视频块的预测样本;以及

基于所述预测样本进行所述转换。

20.一种储存视频的比特流的非瞬时性计算机可读储存介质,,所述比特流由视频处理装置执行的方法生成,其中所述方法包括:

确定预测模式被应用于所述视频的视频图片的当前视频块;

对于所述预测模式,维持包括从所述视频图片推导的参考样本虚拟缓冲区,其中所述虚拟缓冲区的尺寸基于包括所述当前视频块的编解码树块CTB的尺寸而确定;

在所述预测模式中,基于指向所述虚拟缓冲区中的所述参考样本的块矢量而推导所述当前视频块的预测样本;以及

基于所述预测样本生成所述比特流。

21.一种储存视频的比特流的方法,包括:

确定预测模式被应用于所述视频的视频图片的当前视频块;

对于所述预测模式,维持包括从所述视频图片推导的参考样本虚拟缓冲区,其中所述虚拟缓冲区的尺寸基于包括所述当前视频块的编解码树块(CTB)的尺寸而确定;

在所述预测模式中,基于指向所述虚拟缓冲区中的所述参考样本的块矢量而推导所述

当前视频块的预测样本;

基于所述预测样本生成所述比特流;以及

将所述比特流存储在非瞬时性计算机可读储存介质中。

帧内块复制虚拟缓冲区的尺寸限制

[0001] 本申请是于2022年1月25日提交的号为202080053763.6的中国发明专利申请的分案申请,该中国发明专利申请是于2020年7月24日提交的国际专利申请第PCT/CN2020/104081号的中国国家阶段,其要求于2019年7月25日提交的国际专利申请号PCT/CN2019/097742以及于2019年10月7日提交的国际专利申请号PCT/CN2019/109849的优先权和利益。上述申请的全部公开通过引用并入作为本申请的公开的一部分。

技术领域

[0002] 本专利文档涉及视频编解码技术,设备和系统。

背景技术

[0003] 尽管在视频压缩方面有了进展,但是数字视频在互联网和其他数字通信网络上的带宽使用量仍然是最大的。随着能够接收和显示视频的连接用户设备数量的增加,预期数字视频使用的带宽需求将继续增长。

发明内容

[0004] 涉及数字视频编解码的设备,系统和方法,具体地,涉及用于帧内块复制(IBC)的通用虚拟缓冲区。所描述的方法可应用于现有的视频编解码标准(例如,高效视频编解码(HEVC))和未来的视频编解码标准或视频编码器。

[0005] 在一代表性方面中,所公开的技术可用于提供一种用于视频处理的方法,包括:对于视频的视频图片的当前视频块与视频的经编解码的表示之间的转换,基于虚拟管线数据单元(VPDU)的尺寸、编解码树块(CTB)的尺寸或编解码树单元(CTU)的尺寸,针对视频图片的参考区域的尺寸做出决定,其中,来自视频图片的参考区域的参考样本用于预测当前视频块;以及基于该决定,执行转换。

[0006] 在又一代表性方面中,上述方法以处理器可执行代码的形式来实现并存储在计算机可读程序介质中。

[0007] 在又一代表性方面中,公开了一种配置为或可操作以执行上述方法的设备。该设备可包括编程为实施该方法的处理器。

[0008] 在又一代表性方面中,视频解码器装置可实现如本文中所描述的方法。

[0009] 在附图、说明书和权利要求书中更详细地描述了所公开的技术的上述和其他方面和特征。

附图说明

[0010] 图1示出了当前图片参考的示例。

[0011] 图2示出了JVET-M0407中的动态参考区域的示例。

[0012] 图3示出了具有整形的解码流的流程图。

[0013] 图4示出了当前CU(蓝色602)的示例,填充为红色的块(604)是交叉VPDU列参考块,

而填充为黄色的块(606)是交叉VPDU行参考块。每个大块指示64×64VPDU,绿色区域(608)指示可用于IBC参考的重新解释的像素。

[0014] 图5是用于实现本文档中描述的可视媒体解码或可视媒体编码技术的硬件平台的示例的框图。

[0015] 图6是可以在其中可实现所公开的技术的示例性视频处理系统的框图。

[0016] 图7A,图7B和图7C示出了基于所公开的技术的一些实现方式的用于视频处理的示例性方法的流程图。

[0017] 图8示出了基于所公开的技术的一些实现方式的用于视频处理的示例性方法的流程图。

具体实施方式

[0018] 所公开的技术的实施例可应用于现有的视频编解码标准(例如,HEVC,H.265)和未来的标准以提高压缩性能。在本文档中使用章节标题来提高说明书的可读性,而不以任何方式将讨论或实施例(和/或实现方式)仅限制于对应章节。

[0019] 2视频编解码介绍

[0020] 由于对更高分辨率视频的日益增长的需求,视频编解码方法和技术普遍存在于现代技术中。视频编码器通常包括压缩或解压缩数字视频的电子电路或软件,并且被不断改进以提供更高的编解码效率。视频编码器将未压缩的视频转换为压缩格式,反之亦然。在视频质量、用于表示视频的数据量(由比特率确定)、编码和解码算法的复杂性、对数据丢失和错误的敏感性、编辑的方便性、随机访问和端到端时延(延迟)之间存在复杂的关系。压缩格式通常符合标准视频压缩规范,例如,高效视频编解码(HEVC)标准(也称为H.265或MPEG-H第2部分),待最终确定的通用视频编解码(VVC)标准,或其他当前和/或将来的视频编解码标准。

[0021] 视频编解码标准主要是通过众所周知的ITU-T和ISO/IEC标准的发展而得以演进。ITU-T制作了H.261和H.263标准,ISO/IEC制作了MPEG-1和MPEG-4Visual标准,并且两个组织联合制作了H.262/MPEG-2视频标准和H.264/MPEG-4高级视频编解码(Advanced Video Coding,AVC)标准和H.265/HEVC标准。从H.262开始,视频编解码标准基于混合视频编解码结构,其中利用时域预测加变换编解码。为了探索HEVC之外的未来视频编解码技术,由VCEG和MPEG于2015年联合成立联合视频探索团队(JVET)。从那时起,JVET采用了许多新方法并将其纳入名为联合探索模型(JEM)的参考软件。2018年4月,VCEG(Q6/16)和ISO/IEC JTC1 SC29/WG11(MPEG)之间的联合视频专家团队(JVET)成立,致力于VVC标准,目标是与HEVC相比降低50%比特率。

[0022] 2.1HEVC H.265中的帧间预测

[0023] 每个帧间预测的PU具有对于一个或两个参考图片列表的运动参数。运动参数包括运动矢量和参考图片索引。也可以使用inter_pred_idc来信令通知对两个参考图片列表中一个的使用。运动矢量可以明确地被编解码为相对于预测符的增量。

[0024] 当使用跳过(skip)模式来编解码CU时,一个PU与该CU相关联,并且不存在显著的残差系数,不存在编解码的运动矢量增量或参考图片索引。指定Merge模式,由此从邻域PU(包括空域和时域候选)中获得当前PU的运动参数。Merge模式可以应用于任何帧间预测的

PU,而不仅适用于跳过模式。Merge模式的替代方案是运动参数的显式传输,其中,每个PU显式地信令通知:运动矢量(更确切地,相比于运动矢量预测符的运动矢量差(MVD))、每个参考图片列表的对应参考图片索引、以及参考图片列表使用。这样的模式在本公开中被命名为高级运动矢量预测(AMVP)。

[0025] 当信令通知指示要使用两个参考图片列表中的一者时,PU从样点的一个块产生。这被称为“单向预测”。单向预测可用于P条带和B条带的二者。

[0026] 当信令通知指示要使用参考图片列表中的二者时,PU从样点的两个块产生。这被称为“双向预测”。双向预测仅可用于B条带。

[0027] 以下文本提供HEVC中指定的帧间预测模式。将从Merge模式开始描述。

[0028] 2.2当前图片参考

[0029] HEVC屏幕内容编解码扩展(HEVC-SCC)和当前VVC测试模型(VTM-3.0)已经采用了当前图片参考(CPR),或曾经被命名为帧内块复制(IBC)。IBC将运动补偿的构思从帧间编解码扩展到了帧内编解码。如图1所展示,当应用CPR时,当前块由同一图片中的参考块预测。在编解码或解码当前块之前,必须已经重建了参考块中的样点。尽管CPR对于大多数相机捕获的序列效率不高,但是它示出了屏幕内容的显著编解码增益。原因在于屏幕内容图片中有很多重复的图案,诸如图标和文本字符。CPR可以有效地移除这些重复图案之间的冗余。在HEVC-SCC中,如果帧间编解码的编解码单元(CU)选择当前图片作为其参考图片,则帧间编解码的编解码单元(CU)可以应用CPR。在这种情况下,MV重命名为块矢量(BV),并且BV始终具有整数像素精度。为了与主配置文件(profile)HEVC兼容,当前图片在解码图片缓冲区(DPB)中被标记为“长期”参考图片。应当注意,类似地,在多视图/3D视频编解码标准中,视图间参考图片也被标记为“长期”参考图片。

[0030] 在BV找到其参考块之后,可以通过复制参考块来生成预测。可以通过从原始信号中减去参考像素来得到残差。然后可以如其他编解码模式下一样应用变换和量化。

[0031] 然而,当参考块在图片的外部,或与当前块重叠,或在重建区域的外部,或在受一些限制限制的有效区域的外部时,没有定义部分或所有像素值。基本上,存在处理这样的问题的两种解决方案。一种是例如在比特流一致性中不允许这样的情况。另一种是对那些未定义的像素值应用填充。以下子章节详细描述了解决方案。

[0032] 2.3HEVC屏幕内容编解码扩展中的CPR

[0033] 在HEVC的屏幕内容编解码扩展中,当块使用当前图片作为参考时,应保证整个参考块在可用的重建区域内,如以下规范文本所指示:

[0034] 变量offsetX和offsetY推导如下:

[0035] offsetX＝(ChromaArrayType＝＝0)?0:(mvCLX[0]&0x7?2:0)

[0036] (8-104)

[0037] offsetY＝(ChromaArrayType＝＝0)?0:(mvCLX[1]&0x7?2:0)

[0038] (8-105)

[0039] 比特流一致性的要求是,当参考图片为当前图片时,亮度运动矢量mvLX应遵守以下限制:

[0040] -当调用节6.4.1中指定的z扫描顺序块可用性的推导过程时,将(xCurr,yCurr)设置为等于(xCb,yCb)并将相邻的亮度位置(xNbY,yNbY)设置为等于(xPb+(mvLX[0]>>2)-

offsetX,yPb+(mvLX[1]>>2)-offsetY)作为输入,输出应等于TRUE。

[0041] -当调用节6.4.1中指定的z扫描顺序块可用性的推导过程时,将(xCurr,yCurr)设置为等于(xCb,yCb)并将相邻的亮度位置(xNbY,yNbY)设置为等于(xPb+(mvLX[0]>>2)+nPbW-1+offsetX,yPb+(mvLX[1]>>2)+nPbH-1+offsetY)作为输入,输出应等于TRUE。

[0042] -以下条件中的一者或二者应为真:

[0043] -(mvLX[0]>>2)+nPbW+xB1+offsetX的值小于或等于0。

[0044] -(mvLX[1]>>2)+nPbH+yB1+offsetY的值小于或等于0。

[0045] -以下条件应为真:

[0046] (xPb+(mvLX[0]>>2)+nPbSw-1+offsetX)/CtbSizeY-xCb/CtbSizeY<=yCb/CtbSizeY-(yPb+(mvLX[1]>>2)+nPbSh-1+offsetY)/CtbSizeY (8-106)

[0047] 因此,将不会发生参考块与当前块重叠或参考块在图片的外部的情况。无需填充参考或预测块。

[0048] 2.4VVC测试模型中的CPR/IBC

[0049] 在当前的VVC测试模型(即VTM-3.0设计)中,整个参考块应使用当前的编解码树单元(CTU),并且不与当前块重叠。因此,不需要填充参考或预测块。

[0050] 当启用双树时,分割结构从亮度CTU到色度CTU可以不同。因此,对于4:2:0颜色格式,一个色度块(例如,CU)可以对应于已被划分为多个亮度CU的一个共位的亮度区域。

[0051] 仅当以下条件为真时,色度块才可以仅使用CPR模式编解码:

[0052] (1)共位亮度块内的每个亮度CU应使用CPR模式编解码

[0053] (2)首先将亮度4×4块的BV中的每一个转换为色度块的BV,并且色度块的BV是有效的BV。

[0054] 如果两个条件中的任何一个为假,则色度块不应使用CPR模式编解码。

[0055] 注意到,“有效BV”的定义具有以下限制:

[0056] (1)由BV标识的参考块内的所有样点应在限制的搜索范围内(例如,应在当前VVC设计中的同一CTU内)。

[0057] (2)由BV标识的参考块内的所有样点均已经重建。

[0058] 2.5JVET-L0297/JVET-M0407/JVET-M0408中的CPR/IBC

[0059] 在VTM3.0中,CPR/IBC的参考区域仅限制于当前CTU,最大为128x128。JVET-L0297/JVET-M0407/JVET-M0408提出了以下方法:动态改变参考区域以将存储器重新用于存储CPR/IBC的参考样点,使得CPR/IBC块可以具有更多参考候选,而CPR/IBC的参考缓冲区可以从一个CTU保留或减少。

[0060] 图2示出了方法,其中块为64x64,并且CTU包含4个64x64块。当编解码64x64块时,可以将先前的3个64x64块用作参考。这样,解码器仅需要存储4个64x64块来支持CPR/IBC。VTM4.0中采用了上述方法。

[0061] 假设当前亮度CU相对于图片的左上角的位置是(x,y)并且块矢量是(BVx,BVy)。在当前设计中,可以通过以下判断BV是否有效:亮度位置((x+BVx)>>6<<6+(1<<7),(y+BVy)>>6<<6)尚未重建,并且((x+BVx)>>6<<6+(1<<7),(y+BVy)>>6<<6)不等于(x>>6<<6,y>>6<<6)。

[0062] 2.6JVET-O1170中提议的虚拟IBC缓冲区

[0063] 引入虚拟缓冲区构思以帮助描述用于IBC预测模式的参考区域。对于CTU尺寸为ctbSize,我们表示wIbcBuf＝128*128/ctbSize,并且限定虚拟IBC缓冲区ibcBuf,其中宽度为wIbcBuf,和高度为ctbSize。因此,

[0064] -对于CTU尺寸为128x128,ibcBuf的尺寸也是128x128。

[0065] -对于CTU尺寸为64x64,ibcBuf的尺寸为256x64。

[0066] -对于CTU尺寸为32x32,ibcBuf的尺寸为512x32。

[0067] 注意到,VPDU的宽度和高度均为min(ctbSize,64)。我们表示Wv＝min(ctbSize,64)。

[0068] 虚拟IBC缓冲区ibcBuf维持如下。

[0069] (1)在开始解码每个CTU行时,用值(-1)刷新整个ibcBuf。

[0070] (2)在开始解码相对于图片的左上角的VPDU(xVPDU,yVPDU)时,设置ibcBuf[x][y]＝-1,其中x＝xVPDU％wIbcBuf,…,xVPDU％wIbcBuf+Wv-1;y＝yVPDU％ctbSize,…,yVPDU％ctbSize+Wv-1。

[0071] (3)解码后,CU包含相对于图片的左上方的(x,y),设置

[0072] ibcBuf[x％wIbcBuf][y％ctbSize]＝recSample[x][y]

[0073] 因此,比特流限制可以简单地描述为

[0074] *比特流一致性的要求是，对于bv，ibcBuf[(x + bv[0])% wIbcBuf] [(y + bv[1]) % ctbSize]不应等于-1。*

[0075] 在IBC参考缓冲区的构思的情况下,它还通过避免参考帧间插值和运动补偿过程(包括子块过程)来简化解码过程中的文本。

[0076] 2.7VPDU

[0077] 虚拟管线数据单元(VPDU)在图片中限定为非重叠的单元。在硬件解码器中,连续VPDU同时由多个管线阶段处理。在大多数管线阶段中,VPDU尺寸大致与缓冲区尺寸成比例,因此保持VPDU的尺寸较小是重要的。在大多数硬件解码器中,可以将VPDU尺寸设置为最大变换块(TB)尺寸。但是,在VVC中,三叉树(TT)和二叉树(BT)分割可能导致VPDU尺寸增加。

[0078] 为了将VPDU尺寸保持为64x64亮度样点,在VTM5中应用了以下规范性分割限制(具有语法信令通知修改):

[0079] -对于宽度或高度等于128或者宽度和高度二者都等于128的CU,不允许TT划分。

[0080] -对于N≤64(即宽度等于128且高度小于128)的128xN CU,不允许水平BT。

[0081] -对于N≤64(即,高度等于128且宽度小于128)的Nx128 CU,不允许垂直BT。

[0082] 在VVC中,通常约定VPDU的宽度和高度为亮度样点中的min(64,CtbSizeY)。因此,对于CTB/CTU尺寸为64x64、128x128或256x256,VPDU尺寸为64x64。对于CTB/CTU尺寸为32x32,VPDU尺寸为32x32。

[0083] 2.8用于帧内块复制的缓冲区管理和块矢量编解码

[0084] 各种IBC缓冲区功能及其对应管理的细节被描述在PCT/CN2019/093552中,将其作为参考并入。

[0085] 2.9JVET-M0427中的环路整形(ILR)

[0086] 环路整形(in-loop reshaping,ILR)的基本思想是将原始(在第一域中)信号(预测/重建信号)转换为第二域(整形域)。

[0087] 环路亮度整形器实现为一对查找表(LUT),但是两个LUT中仅一个需要被信令通知,因为另一个可以从信令通知的LUT中计算出。每个LUT是一维的10比特、1024条目的映射表(1D-LUT)。一个LUT是前向LUTFwdLUT,其将输入的亮度代码值Y_i映射到更改的值Y_r:Y_r=FwdLUT[Y_i]。另一个LUT是反向LUT InvLUT,其将更改的编解码值Y_r映射到$\hat{Y}_i$:$\hat{Y}_i = InvLUT[Y_r]$。($\hat{Y}_i$表示的重建值为Y_i)。

[0088] 2.9.1PWL模型

[0089] 概念上,分段线性(PWL)通过以下方式来实现:

[0090] 令x1、x2为两个输入枢转点(pivot point),并且y1、y2为它们对应的整体输出枢转点。用于在x1和x2之间的任何输入值x的输出值y可以通过以下等式插值:

[0091] y=((y2-y1)/(x2-x1))*(x-x1)+y1

[0092] 在固定点实现方式中,等式可以重写为:

[0093] y=((m*x+2FP_PREC-1)>>FP_PREC)+c

[0094] 在此,m是标量,c是偏移量,FP_PREC是用于指定精度的恒定值。

[0095] 注意到,在CE-12软件中,PWL模型用于预先计算1024个条目的FwdLUT和InvLUT映射表;但是PWL模型还允许实现在不预先计算LUT的情况下即时(on-the-fly)计算相同的映射值。

[0096] 2.9.2测试CE12-2

[0097] 2.9.2.1亮度整形

[0098] 环路亮度整形的测试2(即提议中的CE12-2)提供了较低复杂度的管线,其还消除了帧间条带重建中逐块的帧内预测的解码等待时间。对于帧间和帧内条带二者,都在整形域中进行帧内预测。

[0099] 无论条带类型如何,总是在整形域中进行帧内预测。通过这样的布置,可以在完成先前TU重建之后立即开始帧内预测。这样的布置还可以为帧内模式提供统一过程,而无需取决于条带。图3示出了基于模式的CE12-2解码过程的框图。

[0100] CE12-2还测试了亮度和色度残差缩放的16段的分段线性(PWL)模型,而不是CE12-1的32段的PWL模型。

[0101] 在CE12-2中使用环路亮度整形器进行帧间条带重建(较浅的阴影块表示整形域中的信号:亮度残差;预测的帧内亮度;以及重建的帧内亮度)

[0102] 2.9.2.2亮度相关的色度残差缩放

[0103] 亮度相关的色度残差缩放是使用固定点整数运算实现的乘法过程。色度残差缩放补偿亮度信号与色度信号的相互作用。色度残差缩放适用于TU级别。更具体地,如下适用:

[0104] -对于帧内,将重建的亮度取平均。

[0105] -对于帧间,将预测亮度取平均。

[0106] 平均值用于标识PWL模型中的索引。该索引标识缩放因子cScaleInv。色度残差乘以该数目。

[0107] 注意到,色度缩放因子是从前向映射的预测亮度值而不是重建亮度值计算出的。

[0108] 2.9.2.3ILR辅助信息的信令通知

[0109] 参数(当前)在片组标头(类似于ALF)中发送。据报道,这些需要40-100比特。

[0110] 以下规范基于JVET-L1001的版本9。添加的语法有前缀“++”。

[0111] 在7.3.2.1序列参数集RBSP语法中

[0112]

seq_parameter_set_rbsp( ) {	描述符
sps_seq_parameter_set_id	ue(v)
…	
sps_triangle_enabled_flag	u(1)
sps_ladf_enabled_flag	u(1)
if (sps_ladf_enabled_flag) {	
sps_num_ladf_intervals_minus2	u(2)
sps_ladf_lowest_interval_qp_offset	se(v)
for(i = 0; i < sps_num_ladf_intervals_minus2 + 1; i++) {	
sps_ladf_qp_offset[i]	se(v)
sps_ladf_delta_threshold_minus1[i]	ue(v)
}	
}	
++ **sps_reshaper_enabled_flag**	u(1)
rbsp_trailing_bits( )	
}	

[0113] 在7.3.3.1通用片组标头语法中

[0114]

tile_group_header( ) {	描述符
…	
if(num_tiles_in_tile_group_minus1 > 0) {	
offset_len_minus1	ue(v)
for(i = 0; i < num_tiles_in_tile_group_minus1; i++)	
entry_point_offset_minus1[i]	u(v)
}	
++ if (sps_reshaper_enabled_flag) {	
++ **tile_group_reshaper_model_present_flag**	u(1)
++ if (tile_group_reshaper_model_present_flag)	
++ tile_group_reshaper_model ( )	
++ **tile_group_reshaper_enable_flag**	u(1)
++ if (tile_group_reshaper_enable_flag && (!(qtbtt_dual_tree_intra_flag && tile_group_type == I)))	
++ **tile_group_reshaper_chr oma_residual_scale_flag**	u(1)
++ }	
byte_alignment( )	
}	

[0115] 添加新的语法表片组整形器模型：

[0116]

++tile_group_reshaper_model () {	描述符
++ **reshaper_model_min_bin_idx**	ue(v)
++ **reshaper_model_delta_max_bin_idx**	ue(v)
++ **reshaper_model_bin_delta_abs_cw_prec_minus1**	ue(v)
++ for (i = reshaper_model_min_bin_idx; i <= reshaper_model_max_bin_idx; i++) {	
++ **reshape_model_bin_delta_abs_CW** [i]	u(v)
++ if (reshaper_model_bin_delta_abs_CW[i]) > 0)	
++ **reshaper_model_bin_delta_sign_CW_flag**[i]	u(1)
++ }	
++}	

[0117] ++在一般序列参数集RBSP语义中,添加以下语义:

[0118] **sps_reshaper_enabled_flag** 等于1指定整形器用在编解码视频序列(CVS)中。sps_reshaper_enabled_flag等于0表示整形器没有用在CVS中。

[0119] ++在片组头语法中,添加以下语义

[0120] **tile_group_reshaper_model_present_flag** 等于1指定tile_group_reshaper_model()存在于片组头中。tile_group_reshaper_model_present_flag等于0指定tile_group_reshaper_model()不存在于片组头中。当不存在tile_group_reshaper_model_present_flag时,则将其推断为等于0。

[0121] **tile_group_reshaper_enabled_flag** 等于1指定为当前片组启用整形器。tile_group_reshaper_enabled_flag等于0指定为当前片组没有启用整形器。当不存在tile_group_reshaper_enable_flag时,则将其推断为等于0。

[0122] **tile_group_reshaper_chroma_residual_scale_flag** 等于1指定为当前片组启用色度残差缩放。tile_group_reshaper_chroma_residual_scale_flag等于0指定为当前片组没有启用色度残差缩放。当不存在tile_group_reshaper_chroma_residual_scale_flag时,则将其推断为等于0。

[0123] ++添加tile_group_reshaper_model()语法

[0124] **reshape_model_min_bin_idx** 指定在整形器构建过程中要使用的最小二进制数(bin)(或段)索引。reshape_model_min_bin_idx的值应在0到MaxBinIdx的范围中(含端值)。MaxBinIdx的值应等于15。

[0125] **reshape_model_delta_max_bin_idx** 指定最大允许的二进制数(或段)索引MaxBinIdx减去在整形器构建过程中要使用的最大二进制数索引。reshape_model_max_bin_idx的值设置为等于MaxBinIdx-reshape_model_delta_max_bin_idx。

[0126] **reshaper_model_bin_delta_abs_cw_prec_minus1** 加1指定用于表示语法reshape_model_bin_delta_abs_CW[i]的比特数目。

[0127] **reshape_model_bin_delta_abs_CW[i]** 指定第i个二进制数的绝对差量码字值。

[0128] **reshaper_model_bin_delta_sign_CW_flag[i]** 指定reshape_model_bin_delta_abs_CW[i]的符号,如下:

[0129] -如果reshape_model_bin_delta_sign_CW_flag[i]等于0,则对应的变量RspDeltaCW[i]为正值。

[0130] -否则(reshape_model_bin_delta_sign_CW_flag[i]不等于0),对应的变量RspDeltaCW[i]为负值。

[0131] 当不存在reshape_model_bin_delta_sign_CW_flag[i]时,则将其推断为等于0。

[0132] 变量RspDeltaCW[i]＝(1 2*reshape_model_bin_delta_sign_CW[i])*reshape_model_bin_delta_abs_CW[i];

[0133] 变量RspCW[i]按如下步骤推导:

[0134] 变量OrgCW设置为等于(1<<$BitDepth_Y$)/(MaxBinIdx+1)。

[0135] -如果reshaper_model_min_bin_idx<=i<=reshaper_model_max_bin_idx

[0136] 则RspCW[i]=OrgCW+RspDeltaCW[i]。

[0137] -否则,RspCW[i]=0。

[0138] 如果$BitDepth_Y$的值等于10,则RspCW[i]的值应在32到2*OrgCW-1的范围中。

[0139] 变量InputPivot[i]其中i的范围在0到MaxBinIdx+1内(含端值)推导如下:

[0140] InputPivot[i]=i*OrgCW

[0141] 变量ReshapePivot[i],其中i的范围在0到MaxBinIdx+1内(含端值),变量ScaleCoef[i]和InvScaleCoeff[i],其中i的范围在0到MaxBinIdx内(含端值),推导如下:

```
        shiftY = 14
    ReshapePivot[ 0 ] = 0;
        for( i = 0; i <= MaxBinIdx ; i++) {
      ReshapePivot[ i + 1 ] = ReshapePivot[ i ] + RspCW[ i ]
        ScaleCoef[ i ] = ( RspCW[ i ] * (1 << shiftY) + (1 << (Log2(OrgCW) - 1)))
```

[0142] `>> (Log2(OrgCW))`

```
      if ( RspCW[ i ] == 0 )
          InvScaleCoeff[ i ] = 0
      else
          InvScaleCoeff[ i ] = OrgCW * (1 << shiftY) / RspCW[ i ]
    }
```

[0143] 变量ChromaScaleCoef[i],其中i的范围在0到MaxBinIdx内(含端值),推导如下:

[0144] ChromaResidualScaleLut[64]={16384,16384,16384,16384,16384,16384,16384,8192,8192,8192,8192,5461,5461,5461,5461,4096,4096,4096,4096,3277,3277,3277,3277,2731,2731,2731,2731,2341,2341,2341,2048,2048,2048,1820,1820,1820,1638,1638,1638,1638,1489,1489,1489,1489,1365,1365,1365,1365,1260,1260,1260,1260,1170,1170,1170,1170,1092,1092,1092,1092,1024,1024,1024,1024};

[0145] shiftC=11

[0146] 如果(RspCW[i]==0)

[0147] 则ChromaScaleCoef[i]=(1<<shiftC)

[0148] -否则(RspCW[i]!=0),则ChromaScaleCoef[i]=ChromaResidualScaleLut[RspCW[i]>>1]

[0149] 2.9.2.4ILR的使用

[0150] 在编码器侧,首先将每个图片(或片组)转换为整形域。并且所有编解码过程都在整形域中进行。对于帧内预测,邻域块是在整形域中;对于帧间预测,首先将参考块(从解码图片缓冲区的原始域生成的)转换为整形域。然后生成残差并将其编解码到比特流。

[0151] 在整个图片(或片组)完成编码/解码之后,将整形域中的样点转换为原始域,然后应用去块(deblocking)滤波器和其他滤波器。

[0152] 在以下情况下,对预测信号的前向整形被禁用:

[0153] ○当前块是帧内编解码的

[0154] ○当前块被编解码为CPR(当前图片参考,又称帧内块复制,IBC)

[0155] ○当前块被编解码为组合帧间帧内模式(CIIP),并且为帧内预测块禁用前向整形

[0156] 3现有实现方式的缺点

[0157] 在当前的IBC虚拟缓冲区设计中,存在一些问题。

[0158] (1)当CTU尺寸大于128×128时如何保持IBC虚拟缓冲区未被定义。

[0159] (2)虚拟缓冲区尺寸与参考样本尺寸之间的关系不清楚。

[0160] (3)可以减小用于IBC模式的行缓冲区和用于CTU行的BV。

[0161] (4)子图片可能过于受限。

[0162] (5)色度QP表可能没有以正确的方式设计。

[0163] 4用于IBC的通用虚拟缓冲区的示例性方法

[0164] 由vSize表示VPDU的宽度和高度,例如vSize=min(64,ctbSizeY),其中ctbSizeY是亮度CTB/CTU宽度/高度。

[0165] 本公开的技术的实施例克服了现有实现方式的缺点,从而提供具有更高编解码效率的视频编解码。基于公开的技术的用于IBC的一般虚拟缓冲区的方法可以提高现有视频编解码标准和未来视频编解码标准二者,并且在下文中被阐明为各种实现方式所描述的示例。下文提供的本公开的技术的示例解释了总体构思,并且不意味着解释为限制。在示例中,除非明显相反地指示,否则可以组合这些示例中描述的各种特征。

[0166] 与IBC缓冲区相关的

[0167] 1.IBC虚拟缓冲区的尺寸(例如,其用于决定块矢量还是映射样本的有效性)可取决于VPDU尺寸,CTB/CTU尺寸。

[0168] a.在一个示例中,虚拟缓冲区的宽度乘以高度可以是固定的,然而,虚拟缓冲区的宽度和/或高度可以取决于VPDU尺寸和/或CTB/CTU尺寸。

[0169] b.在一个示例中,虚拟缓冲区的高度可以等于CTB/CTU的高度。

[0170] i.此外,可替代地,还可以将虚拟缓冲区的宽度设置为(IBC虚拟缓冲区尺寸/CTB的高度)。

[0171] c.在一个示例中,虚拟缓冲区的宽度可以等于CTB/CTU的宽度。

[0172] d.在一个示例中,虚拟缓冲区的宽度可以是VPDU宽度的一倍或多倍

[0173] e.在一个示例中,虚拟缓冲区的高度可以是VPDU高度的一倍或多倍

[0174] 2.建议与IBC BV搜索区域所需的存储器尺寸相比,分配更大的IBC虚拟缓冲区尺寸。

[0175] a.在一个示例中,IBC虚拟缓冲区尺寸可大于用于IBC的VPDU存储器的总尺寸。

[0176] i.在一个示例中,可以将一个或多个CTU分配给IBC虚拟缓冲区。

[0177] b.在一个示例中,IBC虚拟缓冲区尺寸的宽度可以是(128*128/ctbSizeY+ctbSizeY)。

[0178] c.在一个示例中,IBC虚拟缓冲区尺寸的宽度可以是(128*128/ctbSizeY+ctbSizeY),并且IBC虚拟缓冲区尺寸的高度可以是ctbSizeY。

[0179] d.在一个示例中,IBC虚拟缓冲区的宽度可以是(256*128/ctbSizeY)。可替代地,

IBC虚拟缓冲区的高度可以是ctbSizeY。

[0180] e.在一个示例中,可以为较大的CTU/CTB分配较大的IBC虚拟缓冲区。

[0181] i.在一个示例中,当CTU尺寸不小于K(例如,K＝128)时,IBC虚拟缓冲区的宽度可以是(128*128/ctbSizeY+ctbSizeY)。可替代地,IBC虚拟缓冲区的高度可以是ctbSizeY。

[0182] ii.在一个示例中,当CTU尺寸小于K(例如,K＝128)时,IBC虚拟缓冲区的宽度可以是(128*128/ctbSizeY)。可替代地,IBC虚拟缓冲区的高度可以是ctbSizeY。

[0183] 3.用于IBC块的参考块可以被限制为完全在某个VPDU行或VPDU列中

[0184] a.在一个示例中,可禁止参考块跨越不同的VPDU行。

[0185] b.在一个示例中,可禁止参考块越过不同的VPDU列。

[0186] c.在一个示例中,上述VPDU行或列可以与图片相关。

[0187] d.在一个示例中,上述VPDU行或列可以与IBC虚拟缓冲区相关。

[0188] e.此外,可替代地,当由BV指向的参考块跨越两个或更多个CTU/CTB时,可以调用上述方法。

[0189] 4.用于IBC块的参考块可以跨越多个VPDU/跨越不同的VPDU行/VPDU列。然而,可能需要额外的操作来填充参考块中的一些预测值。

[0190] a.在一个示例中,可利用一些默认值来填充一些预测值。

[0191] 5.可以将范围限制应用于IBC模式中使用的块矢量(BV)和/或块矢量差(BVD)。

[0192] a.在一个示例中,BV/BVD的允许范围可取决于当前IBC经编解码块的位置,例如相对于覆盖当前块的CTU/CTB的坐标。

[0193] b.在一个示例中,可将块矢量限制在$[-2^m,2^m-1]$的范围内。

[0194] c.在一个示例中,精度转换后的块矢量差可以限制在$[-2^n,2^n-1]$的范围内。

[0195] d.在一个示例中,精度转换后的块矢量差可以限制在$[-2^n+1,2^n-1]$的范围内。

[0196] e.在一个示例中,在比特流中信令通知的块矢量差可以限制在$[-2^n,2^n-1]$的范围内。

[0197] f.在一个示例中,在比特流中信令通知的块矢量差可以限制在$[-2^n+1,2^n-1]$的范围内。

[0198] g.在一个示例中,m设置为18或17或15。

[0199] h.在一个示例中,n设置为17或16或14。

[0200] i.在一个示例中,m和/或可取决于BV/运动矢量存储的精度和/或与块矢量差相关联的精度。

[0201] j.在一个示例中,可将块矢量限制在与用于帧间预测模式的运动矢量相同的范围内。

[0202] k.在一个示例中,可将块矢量差限制在与用于帧间预测模式的运动矢量相同的范围内。

[0203] l.在一个示例中,一致性比特流应当满足上述子项目被满足。

[0204] i.可替代地,在利用BV/BVD对块进行编码/解码之前,可对经解码的BV/BVD应用用于BV/BVD的裁剪过程。

[0205] 6.可以限制映射到IBC虚拟缓冲区的可用样本的数量。

[0206] a.在一个示例中,映射到缓冲区的可用样本的最大数目可以小于IBC虚拟缓冲区

尺寸。

[0207] b.在一个示例中,当CTB/CTU尺寸大于64×64时,映射到IBC虚拟缓冲区的可用样本的最大数目可以是固定的。

[0208] c.在一个示例中,映射到IBC虚拟缓冲区的可用样本的数目可被限制为小于或等于VPDU中样本的数目的一倍或多倍。

[0209] i.在一个示例中,当CTU/CTB尺寸大于64×64时,映射到IBC虚拟缓冲区的可用样本的数目可被限制为小于或等于VPDU中样本的数目的三倍。

[0210] 7.映射到IBC虚拟缓冲区的IBC参考样本的不可用性标记可以以VPDU的粒度执行

[0211] a.在一个示例中,当样本需要被标记为不可用时,相同VPDU内的样本也可以被标记为不可用。

[0212] b.在一个示例中,一个或多个VPDU可以被同时标记为不可用。

[0213] c.在一个示例中,哪个VPDU的样本被标记为不可用可以取决于当前VPDU的位置。

[0214] d.在一个示例中,哪个VPDU的样本被标记为不可用可以取决于先前或最近解码的VPDU的位置。

[0215] 8.当CTU/CTB尺寸大于64×64时,IBC参考可以是当前VPDU和三个最近解码的VPDU。

[0216] a.在一个示例中,可以为映射到虚拟IBC缓冲区的每个VPDU保持索引,以跟踪每个VPDU的解码顺序。

[0217] 9.可保持计数器以跟踪映射到缓冲区的可用VPDU的数目。

[0218] a.在一个示例中,计数器在开始解码每一CTU行时被重置为0,且在映射到缓冲区的一个VPDU已被解码时,计数器增加1。

[0219] b.在一个示例中,当计数器大于某一值(例如,3)时,映射到缓冲区的一个VPDU的样本可被标记为不可用,且计数器可减小1。

[0220] 10.当CTU/CTB尺寸为128×128时,对应的IBC虚拟缓冲区的尺寸可以是256×128。

[0221] a.可替代地,IBC虚拟缓冲区可以具有尺寸(k*64)×128,其中k>2。

[0222] 11.当CTU/CTB尺寸是256×256时,对应的IBC虚拟缓冲区可以具有64×256的尺寸以跟踪参考样本的可用性,即ibcBufW=64,ibcBufH=256。

[0223] a.在一个示例中,在解码具有左上位置(x0,y0)的VPDU之前,IBC缓冲区中的对应VPDU行(0,y0%256)设置为-1。

[0224] 12.当CTU/CTB尺寸是256×256时,对应的IBC虚拟缓冲区可以具有128×256的尺寸以跟踪参考样本的可用性,即ibcBufW=128,ibcBufH=256。

[0225] a.在一个示例中,除了某个VPDU行之外,可以为缓冲区中的每个VPDU行仅保留一个VPDU(不包括当前的VPDU)。

[0226] i.在一个示例中,除了最后的VPDU行之外,可以为缓冲区中的每个VPDU行仅保留一个VPDU(不包括当前的VPDU)。

[0227] 13.在CTU行的开始时可不重置IBC缓冲区。

[0228] a.在一个示例中,从上述CTU行继承的IBC虚拟缓冲区可用作当前CTU行的初始状态。

[0229] b.可替代地,在CTU行的开始时可部分地重置IBC虚拟缓冲区。

[0230] i.在一个示例中,可将上述CTU行中的VPDU继承到当前IBC缓冲区中,并且可重置其他缓冲区区域。

[0231] ii.在一个示例中,可将上述CTU行的最左下VPDU继承到当前IBC缓冲区中,并且可重置其他缓冲区区域。

[0232] 14.是否和/或如何将缓冲区中的样本标记为不可用可以独立于色度块位置。

[0233] a.在一个示例中,只有当亮度块是VPDU中的第一块时,IBC缓冲区中的对应样本才可标记为不可用。

[0234] b.在一个示例中,当解码色度编解码单元时,可能不允许将缓冲区中的样本重置或标记为不可用。

[0235] 15.提出了在开始对每个VPDU进行解码时,可以基于当前的VPDU位置来重置对应的IBC虚拟缓冲区。

[0236] a.在一个示例中,将重置对应于(xVPDU+ctbSizeY,yVPDU)的IBC虚拟缓冲区,其中(xVPDU,yVPDU)表示当前VPDU相对于图片左上角的位置。

[0237] 16.是否和/或如何标记IBC虚拟缓冲区中的样本可以取决于最近解码的VPDU的位置和VPDU尺寸。

[0238] a.在一个示例中,是否和/或如何将IBC虚拟缓冲区中的样本标记为不可用可取决于最近解码的VPDU的位置,即砖块/条带/片/图片。

[0239] b.在一个示例中,是否和/或如何将IBC虚拟缓冲区中的样本标记为不可用可取决于最近解码的VPDU在砖块/条带/片/图片的CTU行中的位置。

[0240] c.在一个示例中,IBC虚拟缓冲区中将被标记为不可用的样本可独立于当前块的位置。

[0241] 与IBC行缓冲区相关

[0242] 17.可以禁止跨越CTU行的块矢量预测

[0243] a.在一个示例中,当块矢量预测来自与当前CTU行相比不同的CTU行时,其可被认为是不可用的。

[0244] 18.用于跨越CTU行的当前IBC块的去块决定可以独立于其他块的块矢量或运动矢量。

[0245] a.在一个示例中,当两个块具有不同的CTU行并且一个块以IBC模式编解码,而另一个块以IBC或帧间模式编解码时,去块边界强度可以总是设置为等于1。

[0246] b.在一个示例中,当两个块具有不同的CTU行并且一个块以IBC模式编解码,而另一个块以IBC或帧间模式编解码时,去块边界强度可以总是被设置为等于0。

[0247] 19.用于跨越CTU行的当前IBC块的去块决定可以独立于另一个块是以IBC模式还是以帧间模式编解码。

[0248] a.在一个示例中,当两个块具有不同的CTU行并且一个块以IBC模式编解码而另一个块不以帧内模式编解码时,去解块边界强度可以总是被设置为等于1。

[0249] 与子图片相关

[0250] 20.提出了允许在某些条件下在两个子图片之间进行预测(例如,运动/样本预测)。

[0251] a.在一个示例中,如果第一子图片的边界与图片边界(或一致性窗口边界)重合,

则可以允许使用来自第二子图片的信息。

[0252] i.此外,可替代地,子图片边界是左子图片边界或右子图片边界。

[0253] ii.此外,可替代地,图片边界是左图片边界或右图片边界。

[0254] iii.在一个示例中,第二子图片的左(或右)边界可以与图片的左(或右)边界重合。

[0255] b.只有在允许图片环绕(picture wrapping)(例如,sps_ref_wraparound_enabled_flag等于1)的情况下,条件才成立。

[0256] 21.图片环绕排除了子图片。

[0257] a.在一个示例中,如果使用了子图片,则启用图片环绕。

[0258] b.在一个示例中,如果启用了图片环绕,则不能使用子图片。

[0259] c.可替代地,当使用了子图片时,可以启用图片环绕。

[0260] iv.在一个示例中,可以为具有与图片边界重合的边界的子图片启用图片环绕。

[0261] 与色度QP表相关

[0262] 22.色度QP表的最小索引可以独立于用于色度分量的比特深度。

[0263] a.在一个示例中,色度QP表的最小索引可取决于用于亮度分量的比特深度。

[0264] b.在一个示例中,色度QP表的最小索引可以是$QpBdOffset_Y$,即6*bit_depth_luma_minus8。

[0265] 5所公开技术的示例性实现方式

[0266] 5.1实施例#1

[0267] 当CTU尺寸为256×256时,保持64×256IBC虚拟缓冲区ibcBuf,即ibcBufW=64,ibcBufH=256。VPDU尺寸是64×64,并且除了当前的VPDU之外,使用3个附加的VPDU的片上存储器来存储IBC参考样本。

[0268] 缓冲区ibcBuf在开始解码CTU行时被重置为-1。

[0269] 在开始解码相对于图片的左上角的左上位置(x0,y0)的新VPDU时,应用以下

[0270] 1)对于x=x0..x0+63,y=y0..y0+63,ibcBuf[x%ibcBufW][y%ibcBufH]=-1

[0271] 2)在解码CU之后,对于该CU中相对于图片的左上角的(x,y),将ibcBuf[x%ibcBufW][y%ibcBufH]设置为环路滤波(例如SAO,去块,ALF)之前的样本(x,y)的重建值。

[0272] 3)给定bv,用于(x,y)的参考为ibcBuf[(x+bv[0])%ibcBufW][(y+bv[1])%ibcBufH]

[0273] 比特流限制为以下两个条件应为真

[0274] 1)对于相对于图片的左上角的左上位置(x,y)的WxH块,(y%ibcBufH)+H<=ibcBufH

[0275] 2)对于x=0..W-1,y=0..,H-1,ibcBuf[(x+bv[0])%ibcBufW][(y+bv[1])%ibcBufH]不应包含无效像素值,例如,-1。

[0276] 5.2实施例#2

[0277] 当CTU尺寸是256×256时,保持128×256IBC虚拟缓冲区ibcBuf,即ibcBufW=128,ibcBufH=256。VPDU尺寸是64×64,并且除了当前的VPDU之外,使用3个附加的VPDU的片上存储器来存储IBC参考样本。

[0278] 缓冲区ibcBuf在开始解码CTU行时被重置为-1。xPrevVPDU=0以及yPrevVPDU=0。

[0279] 在开始解码相对于图片的左上角的左上位置(x0,y0)的新VPDU时,应用以下

[0280] 1)如果(yPrevVPDU+64)%ibcBufH不等于0,

[0281] 对于x=x0..x0+63,y=y0..y0+63,ibcBuf[(x+xPrevVPDU-64)%ibcBufW][(y+yPrevVPDU)%ibcBufH]=-1

[0282] 2)否则((yPrevVPDU+64)%ibcBufH等于0),

[0283] 对于x=x0…x0+63,y=y0…y0+63,

[0284] ibcBuf[(x+xPrevVPDU)%ibcBufW][(y+yPrevVPDU)%ibcBufH]=-1

[0285] 3)xPrevVPDU=x0以及yPrevVPDU=y0

[0286] 比特流限制为以下两个条件应为真

[0287] 1)对于相对于图片的左上角的左上位置(x,y)的WxH块,(y%ibcBufH)+H<=ibcBufH

[0288] 2)对于x=0..W-1,y=0..,H-1,ibcBuf[(x+bv[0])%ibcBufW][(y+bv[1])%ibcBufH]不应包含无效像素值,例如,-1。

[0289] 5.3实施例#3

[0290] 该实施例反映了项目2。以黑体斜体标记的改变是基于VVC草案6文档JVET-O2001-vE。

[0291] **log2_min_luma_coding_block_size_minus2**加2指定最小亮度编解码块尺寸。

[0292] 变量CtbLog2SizeY,CtbSizeY,MinCbLog2SizeY,MinCbSizeY,IbcBufWidthY,IbcBufWidthC和Vsize推导如下:

[0293] CtbLog2SizeY=log2_ctu_size_minus5+5 (7-15)

[0294] CtbSizeY=1<<CtbLog2SizeY (7-16)

[0295] MinCbLog2SizeY=log2_min_luma_coding_block_size_minus2+2 (7-17)

[0296] MinCbSizeY=1<<MinCbLog2SizeY (7-18)

[0297] IbcBufWidthY=128*128/CtbSizeY+CtbSizeY (7-19)

[0298] IbcBufWidthC=IbcBufWidthY/SubWidthC (7-20)

[0299] Vsize=min(64,CtbSizeY) (7-21)

[0300] 5.4实施例#4

[0301] 该实施例反映了项目3。

[0302] 将(xCb,yCb)表示为当前块相对于图片左上角的左上位置。块宽度和高度分别为W和H。块的块矢量是(xBv,yBv)

[0303] VPDU行相对于图片的限制:

[0304] 比特流限制为(xCb+xBv)/vSize应当等于(xCb+xBv+W-1/vSize)。

[0305] VPDU列相对于图片的限制:

[0306] 比特流限制为(yCB+yBv)/vSize应当等于(yCB+yBv+H-1/vSize)。

[0307] VPDU行相对于IBC缓冲区的限制:

[0308] 比特流限制为((xCb+xBv)%IbcBufWidthY)/vSize应该等于(((xCb+xBv+W-1)%IbcBufWidthY)/vSize)。

[0309] VPDU列相对于IBC缓冲区的限制:

[0310] 比特流限制为((yCb+yBv)%IbcBufHeightY)/vSize应当等于(((yCb+yBv+H-1)%IbcBufHeightY)/vSize)。

[0311] 5.5实施例#5

[0312] 该实施例反映了将IBC虚拟缓冲区中的样本标记为不可用应该独立于色度块。

[0313] 以黑体和斜体标记的改变基于JVET-O2001-vE。删除的文本用双括号标记(例如,[[a]]表示删除了字符“a”)。

[0314] 7.4.9.5编解码单元语义

[0315] 当ResetIbcBuf等于1时,应用以下:

[0316] -对于x=0..IbcBufWidthY-1和y=0..CtbSizeY-1,进行以下赋值:

[0317] IbcVirBuf[0][x][y]=-1 (7-153)

[0318] -变量ResetIbcBuf设置为等于0。

[0319] 当x0%VSize等于0且y0%VSize等于0***且 cIdx 等于 0***时,对x=x0..x0+VSize-1且y=y0..y0+VSize-1进行以下赋值:

[0320] IbcVirBuf[0][x%IbcBufWidthY][y%CtbSizeY]=-1(7-154)

[0321] 5.6实施例#6

[0322] 7.4.9.5编解码单元语义

[0323] ...

[0324] 当ResetIbcBuf等于1时,应用以下:

[0325] -对于x=0..IbcBufWidthY-1和y=0..CtbSizeY-1,进行以下赋值:

[0326] IbcVirBuf[0][x][y]=-1 (7-153)

[0327] ***xPrevV = 0***

[0328] ***yPrevV = 0***

[0329] -变量ResetIbcBuf设置为等于0。

[0330] 当x0%VSize等于0且y0%VSize等于0且***cIdx 等于 0 时，应用以下：[[对于 x = x0..x0 + VSize − 1 且 y = y0..y0 + VSize − 1 进行赋值:]]***

– 如果yPrevV% VSize 等于0,则xP 设置为等于xPrevV − (CtbLog2SizeY == 7) ? 128 : (−VSize)，否则xP 设置为等于xPrevV − (CtbLog2SizeY == 7) ? 64 : (−VSize)。

– 对于 x = xP..xP + max(VSize, cbWidth) −1 和 y = yPrevV.. yPrevV + max(VSize, cbHeight) −1，进行以下赋值

[0331] IbcVirBuf[0][x%IbcBufWidthY][y%CtbSizeY]=-1(7-154)

[0332] ***– 变量xPrevV 设置为等于 (x0 + cbWidth − 1) / VSize * VSize 并且yPrevV 设置为等于(y0 + cbHeight − 1) / VSize * VSize。***

[0333] 5.7实施例#7

[0334] 本实施例给出了允许子图片预测的示例。与JVET-O2001-vE相比的改变以黑体和斜体突出显示。删除的文本用双括号标记(例如,[[a]]表示删除了字符“a”)。

[0335] 8.5.6.3.3Luma整数样本获取过程

[0336] 此过程的输入为：

[0337] -全样本单元($xInt_L$,$yInt_L$)中的亮度位置，

[0338] -亮度参考样本阵列$refPicLX_L$。

[0339] 该过程的输出是预测的亮度样本值predSampleLXL

[0340] 变量移位设置为等于Max(2,14-$BitDepth_Y$)。

[0341] 变量picW设置为等于pic_width_in_luma_samples,变量picH设置为等于pic_height_in_luma_samples。

[0342] 全样本单元(xInt,yInt)中的亮度位置推导如下：

– 如果 subpic_treated_as_pic_flag [SubPicIdx] 等于 1，则应用以下：

[0343] ***xInt = Clip3(SubPicLeftBoundaryPos, SubPicRightBoundaryPos, sps_ref_wraparound_enabled_flag ? ClipH((sps_ref_wraparound_offset_minus1 + 1) * MinCbSizeY, picW, xIntL) :xInt) (8-782)***

[0344] ***yInt = Clip3(SubPicTopBoundaryPos, SubPicBotBoundaryPos, yInt) (8-782)***

– 否则：

[0345] xInt＝Clip3(0,picW-1,sps_ref_wraparound_enabled_flag? (8-782)

[0346] ClipH((sps_ref_wraparound_offset_minus1+1)*MinCbSizeY,picW,$xInt_L$):$xInt_L$)

[0347] yInt＝Clip3(0,picH-1,$yInt_L$) (8-783)

[0348] 预测的亮度样本值predSampleLXL推导如下：

[0349] PredSampleLXL＝refPicLXL[Xint][Yint]<<shift3

[0350] (8-784)

[0351] 5.8实施例#8

[0352] 本实施例示出了色度QP表的示例性设计。以黑体和斜体标记的改变基于JVET-O2001-vE。

[0353] 7.4.3.3序列参数集RBSP语义

[0354] ...

[0355] 对于i＝0..same_qp_table_for_chroma?0:2的第i个色度QP映射表ChromaQpTable[i]推导如下：

[0356]
```
qpInVal[ i ][ 0 ] = [[QpBdOffsetC ]]QpBdOffsetY + delta_qp_in_val_minus1[ i ][ 0 ]
qpOutVal[ i ][ 0 ] = −QpBdOffsetC + delta_qp_out_val[ i ][ 0 ]
for( j = 1; j <= num_points_in_qp_table_minus1[ i ]; j++ ) {
    qpInVal[ i ][ j ] = qpInVal[ i ][ j − 1 ] + delta_qp_in_val_minus1[ i ][ j ] + 1
    qpOutVal[ i ][ j ] = qpOutVal[ i ][ j − 1 ] + delta_qp_out_val[ i ][ j ]
}
ChromaQpTable[ i ][ qpInVal[ i ][ 0 ] ] = qpOutVal[ i ][ 0 ]
for( k = qpInVal[ i ][ 0 ] − 1; k >= [[QpBdOffsetC ]] QpBdOffsetY; k − − )
    ChromaQpTable[ i ][ k ] = Clip3( −QpBdOffsetC, 63, ChromaQpTable[ i ][ k + 1 ] − 1 )  (7-31)
for( j = 0; j < num_points_in_qp_table_minus1[ i ]; j++ ) {
    sh = ( delta_qp_in_val_minus1[ i ][j + 1 ] + 2 ) >> 1
    for( k = qpInVal[ i ][ j ] + 1, m = 1; k <= qpInval[ i ][ j + 1 ]; k++, m++ )
        ChromaQpTable[ i ][ k ] = ChromaQpTable[ i ][ qpInVal[ i ][ j ] ] +
                    ( delta_qp_out_val[ i ][j + 1]  *  m  +  sh  )  /
( delta_qp_in_val_minus1[ i ][j + 1] + 1 )
}
for( k = qpInVal[ i ][ num_points_in_qp_table_minus1[ i ] ] + 1; k <= 63; k++ )
    ChromaQpTable[ i ][ k ] = Clip3( −QpBdOffsetC, 63, ChromaQpTable[ i ][ k − 1 ] + 1 )
```

[0357] 当same_qp_table_for_chroma等于1时,对于k = [[QpBdOffsetC]] QpBdOffsetY..63,将ChromaQpTable[1][k]和ChromaQpTable[2][k]设置为等于ChromaQpTable[0][k]。

[0358] 比特流一致性的要求为对于i=0..same_qp_table_for_chroma?0:2和j=0..num_points_in_qp_table_minus1[i],qpInVal[i][j]的值应在-QpBdOffsetY到63的范围内,并且qpOutVal[i][j]应在-QpBdOffset$_C$到63的范围内(包括端值)。

[0359] ...

[0360] 5.9实施例#9

[0361] 本实施例示出了扩大的IBC虚拟缓冲区的示例性设计。以黑体和斜体标记的改变基于JVET-O2001-vE。删除的文本用双括号标记(例如,[[a]]表示删除了字符“a”)。

[0362] 7.4.3.3序列参数集RBSP语义

[0363] ...

[0364] 变量CtbLog2SizeY,CtbSizeY,MinCbLog2SizeY,MinCbSizeY,IbcBufWidthY,IbcBufWidthC和Vsize推导如下:

[0365] CtbLog2SizeY=

[0366] log2_ctu_size_minus5+5

[0367] (7-15)

[0368] CtbSizeY=

[0369] 1<<CtbLog2SizeY

[0370] (7-16)

[0371] MinCbLog2SizeY=log2_min_luma_coding_block_size_minus2+2(7-17)

[0372] MinCbSizeY＝

[0373] 1<<MinCbLog2SizeY

[0374] (7-18)

[0375] IbcBufWidthY＝128*128/CtbSizeY+CtbSizeY (7-19)

[0376] IbcBufWidthC＝IbcBufWidthY/SubWidthC (7-20)

[0377] Vsize＝min(64,CtbSizeY)

[0378] (7-21)

[0379] ...

[0380] 7.4.9.5编解码单元语义

[0381] ...

[0382] 当x0％VSize等于0且y0％VSize等于0时,对x＝x0..x0+VSize-1且y＝y0..y0+VSize-1进行以下赋值:

[0383] $$IbcVirBuf[\,0\,][\,(\,x\,+\,\boldsymbol{CtbSizeY}\,)\,\%\,IbcBufWidthY\;\;][\,y\,\%\,CtbSizeY\,]=-1$$ (7-154)

[0384] 5.10实施例#10

[0385] 本实施例示出了扩大的IBC虚拟缓冲区的示例性设计。以黑体和斜体标记的改变基于JVET-O2001-vE。删除的文本用双括号标记(例如,[[a]]表示删除了字符“a”)。

[0386] 7.4.3.3序列参数集RBSP语义

[0387] ...

[0388] 变量CtbLog2SizeY,CtbSizeY,MinCbLog2SizeY,MinCbSizeY,IbcBufWidthY,IbcBufWidthC和Vsize推导如下:

[0389] CtbLog2SizeY＝

[0390] log2_ctu_size_minus5+5

[0391] (7-15)

[0392] CtbSizeY＝

[0393] 1<<CtbLog2SizeY

[0394] (7-16)

[0395] MinCbLog2SizeY＝log2_min_luma_coding_block_size_minus2+2(7-17)

[0396] MinCbSizeY＝

[0397] 1<<MinCbLog2SizeY

[0398] (7-18)

[0399] $$\boldsymbol{W=CtbSizeY==128?\ 256:\ CtbSizeY}$$

[0400] IbcBufWidthY＝W[[128]]*128/CtbSizeY (7-19)

[0401] IbcBufWidthC＝IbcBufWidthY/SubWidthC (7-20)

[0402] Vsize＝min(64,CtbSizeY)

[0403] (7-21)

[0404] ...

[0405] 7.4.9.5编解码单元语义

[0406] ...

[0407] 当x0％VSize等于0且y0％VSize等于0时,对x＝x0..x0+VSize-1且y＝y0..y0+VSize-1进行以下赋值:

[0408] IbcVirBuf[0][(x+CtbSizeY)％IbcBufWidthY][y％CtbSizeY]＝-1

[0409] (7-154)

[0410] 5.11实施例#11

[0411] 以黑体和斜体标记的改变基于JVET-O2001-vE。删除的文本用双括号标记(例如,[[a]]表示删除了字符“a”)。

[0412] 7.4.3.3序列参数集RBSP语义

[0413] ...

[0414] 变量CtbLog2SizeY,CtbSizeY,MinCbLog2SizeY,MinCbSizeY,IbcBufWidthY,IbcBufWidthC和Vsize推导如下:

[0415] CtbLog2SizeY＝

[0416] log2_ctu_size_minus5+5

[0417] (7-15)

[0418] CtbSizeY＝

[0419] 1<<CtbLog2SizeY

[0420] (7-16)

[0421] MinCbLog2SizeY＝log2_min_luma_coding_block_size_minus2+2(7-17)

[0422] MinCbSizeY＝

[0423] 1<<MinCbLog2SizeY

[0424] (7-18)

[0425] IbcBufWidthY＝[[128]]256*128/CtbSizeY (7-19)

[0426] IbcBufWidthC＝IbcBufWidthY/SubWidthC (7-20)

[0427] Vsize＝min(64,CtbSizeY)

[0428] (7-21)

[0429] ...

[0430] 7.4.9.5编解码单元语义

[0431] ...

[0432] 当x0％VSize等于0且y0％VSize等于0时,对x＝x0..x0+VSize-1且y＝y0..y0+VSize-1进行以下赋值:

[0433] $$IbcVirBuf[\,0\,][\,(x + (\,\boldsymbol{IbcBufWidthY} >> \boldsymbol{1}\,)\,)\,\%\,IbcBufWidthY\,][\,y\,\%\,CtbSizeY\,] = -1 \quad (7\text{-}154)$$

[0434] 图4示出了跨越VPDU列和VPDU行的参考块的示例。如图4所示,对于当前的CU(蓝色602中),以红色填充的块(604)是跨越VPDU列参考块,而以黄色填充的块(606)是跨越VPDU行参考块。每个大块指示64×64VPDU,绿色区域(608)指示可用于IBC参考的重建的像素。

[0435] 图5是视频处理装置500的框图。设备500可以用于实现本文中描述的一个或多个方法。装置500可以实施在智能电话、平板计算机、计算机、物联网(IoT)接收器等中。装置500可以包括一个或多个处理器502、一个或多个存储器504和视频处理硬件506。处理器502

可配置为实现本文档中描述的一个或多个方法(包括但不限于方法400)。一个或多个存储器504可用于存储用于实现本文中描述的方法和技术的数据和代码。视频处理硬件506可用于在硬件电路中实现本文档中描述的一些技术。

[0436] 图6是示出可在其中实现本文中公开的各种技术的示例性视频处理系统1900的框图。各种实现方式可以包括系统1900的一些或全部组件。系统1900可以包括用于接收视频内容的输入1902。视频内容可以以原始或未压缩格式(例如,8或10比特多分量像素值)来接收,或者可以以经压缩或经编码格式来接收。输入1902可表示网络接口、外围总线接口或存储接口。网络接口的示例包括诸如以太网、无源光网络(PON)等的有线接口,以及诸如Wi-Fi或蜂窝接口的无线接口。

[0437] 系统1900可以包括编解码组件1904,其可以实现本文档中描述的各种编解码或编码的方法。编解码组件1904可将来自输入1902的视频的平均比特率降低到编解码组件1904的输出,以产生视频的经编解码的表示。因此,编解码技术有时被称为视频压缩或视频代码转换技术。编解码组件1904的输出可以被存储,或者通过连接的通信组件(如组件1906所示)被传输。在输入1902处接收的视频的存储的或传送的比特流(或经过编解码的)表示可以由组件1908用于生成被发送到显示接口1910的像素值或可显示视频。从比特流表示生成用户可视视频的过程有时被称为视频解压缩。此外,虽然某些视频处理操作被称为“编解码”操作或工具,但是应理解,编解码工具或操作在编码器处使用,并且反转编解码结果的对应解码工具或操作将由解码器执行。

[0438] 外围总线接口或显示器接口的示例可包括通用串行总线(USB)或高清晰度多媒体接口(HDMI)或显示器端口(Displayport)等。存储接口的示例包括SATA(串行高级技术附件)、PCI、IDE接口等。本文档中描述的技术可以实施在各种电子设备中,例如移动电话、膝上型计算机、智能电话或能够进行数字数据处理和/或视频显示的其它设备。

[0439] 在一些实施例中,可以使用在参照图5或6描述的硬件平台上实现的设备来实现视频编解码方法。

[0440] 上文中描述的示例可以并入在下文中描述的方法(例如,可在视频解码器或视频编码器处实现的方法710、720、730、810)的上下文中。

[0441] 图7A示出了用于视频处理的示例性方法710的流程图。方法710包括:在步骤712,基于与当前块相关联的虚拟缓冲区的尺寸,针对映射到虚拟缓冲区的一个或多个样本或块矢量的有效性做出决定,其中当前块基于参考块中的像素进行了编解码,当前图片包括当前块和参考块,以及虚拟缓冲区的尺寸基于虚拟管线数据单元(VPDU)的尺寸、编解码树块(CTB)的尺寸或编解码树单元(CTU)的尺寸。方法700包括:在步骤714,基于该决定,执行当前块和当前块的比特流表示之间的转换。

[0442] 图7B示出了用于视频处理的示例性方法720的流程图。方法720包括:在步骤722,对于基于参考块中的像素进行编解码的当前块,指定参考块的一个或多个参考样本为不可用,该一个或多个参考样本中的每个都被映射到虚拟缓冲区并且至少在与参考块相关联的当前虚拟管线数据单元(VPDU)中具有对应的样本,以及当前图片包括当前块和参考块。方法720包括:在步骤724,基于该指定,执行当前块和当前块的比特流表示之间的转换。

[0443] 图7C示出了用于视频处理的示例性方法730的流程图。方法730包括:在步骤732,对于基于参考块中的像素进行编解码的当前块,基于当前块的编解码树单元(CTU)的尺寸

或编解码树块(CTB)的尺寸,确定与参考块相关联的虚拟缓冲区的尺寸,其中当前图片包括当前块和参考块。方法730包括:在步骤734,基于该确定,执行当前块和当前块的比特流表示之间的转换。

[0444] 图8示出了用于视频处理的示例性方法810的流程图。方法810包括:在步骤812,对于视频的视频图片的当前视频块与视频的经编解码的表示之间的转换,基于虚拟管线数据单元(VPDU)的尺寸、编解码树单元(CTU)的尺寸或编解码树块(CTB)的尺寸,针对视频图片的参考区域的尺寸做出决定,其中来自视频图片的参考区域的参考样本用于预测当前视频块。方法810还包括:在步骤814,基于该决定,执行转换。

[0445] 所公开技术的一些实施例包括决定或确定启用视频处理工具或模式。在一个示例中,当视频处理工具或模式被启用时,编码器将在视频块的处理中使用或实现该工具或模式,但是可以不必基于工具或模式的使用来修改所得到的比特流。也就是说,从视频的块到视频的比特流表示的转换将在决定或确定启用视频处理工具或模式时使用视频处理工具或模式。在另一示例中,当启用视频处理工具或模式时,解码器将基于视频处理工具或模式利用比特流已被修改的知识来处理比特流。也就是说,将使用决定或确定所启用的视频处理工具或模式来执行从视频的比特流表示到视频的块的转换。

[0446] 所公开的技术的一些实施例包括决定或确定禁用视频处理工具或模式。在一个示例中,当视频处理工具或模式被禁用时,编码器将不在视频的块到视频的比特流表示的转换中使用工具或模式。在另一示例中,当视频处理工具或模式被禁用时,解码器将利用这样的知识来处理比特流,即没有使用决定或确定所禁用的视频处理工具或模式来修改比特流。

[0447] 在本文档中,术语“视频处理”可以指的是视频编码、视频解码、视频压缩或视频解压缩。例如,视频压缩算法可以在从视频的像素表示到对应的比特流表示的转换过程中应用,反之亦然。如语法所定义的,当前视频块的比特流表示例如可以对应于在比特流内并置的或分布在比特流内的不同位置的比特。例如,宏块可以按照变换并经编码的误差残差值来编解码,并且还可以使用比特流的标头和其他字段中的比特进行编解码。

[0448] 使用条款列表进一步描述在本文档中描述的各种技术方案和实施例。第一组条款描述了前面章节中公开的技术的某些特征和方面。

[0449] 1.一种用于视频处理的方法,包括:基于与基于参考块中的像素进行编解码的当前块相关联的虚拟缓冲区的尺寸,做出关于映射至所述虚拟缓冲区的一个或多个样本或块矢量的有效性的决定,其中当前图片包括该当前块和该参考块,并且其中虚拟缓冲区的尺寸基于虚拟管线数据单元(VPDU)的尺寸、编解码树块(CTB)的尺寸或编解码树单元(CTU)的尺寸;以及基于该决定,执行当前块与当前块的比特流表示之间的转换。

[0450] 2.如条款1所述的方法,其中,虚拟缓冲区的高度与宽度的乘积是固定的,以及该高度或该宽度是基于VPDU的尺寸、CTB的尺寸或CTU的尺寸。

[0451] 3.如条款1所述的方法,其中,虚拟缓冲区的宽度等于CTB的宽度或CTU的宽度。

[0452] 4.如条款1所述的方法,其中,虚拟缓冲区的宽度或高度分别是VPDU的宽度或高度的N倍,并且其中N≥1且是整数。

[0453] 5.如条款1所述的方法,其中,一个或多个样本的最大数目小于虚拟缓冲区的尺寸。

[0454] 6.如条款1所述的方法,其中,在确定CTB的尺寸或CTU的尺寸大于64×64时,固定一个或多个样本的最大数目。

[0455] 7.如条款1所述的方法,其中,一个或多个样本的数目小于或等于VPDU中样本的数目的N倍,并且其中N≥1且是整数。

[0456] 8.如条款7所述的方法,其中,在确定CTB的尺寸或CTU的尺寸大于64×64时,N=3。

[0457] 9.一种用于视频处理的方法,包括:对于基于参考块中的像素进行编解码的当前块,指定参考块的一个或多个参考样本为不可用,其中一个或多个参考样本中的每个被映射到虚拟缓冲区并且至少在与参考块相关联的当前虚拟管线数据单元(VPDU)中具有对应的样本,并且其中当前图片包括当前块和参考块;以及基于该指定,执行当前块与当前块的比特流表示之间的转换。

[0458] 10.根据条款9所述的方法,还包括:将当前VPDU中的对应样本指定为不可用。

[0459] 11.如条款10所述的方法,其中,基于当前VPDU的位置,将当前VPDU中的对应样本指定为不可用。

[0460] 12.如条款10所述的方法,其中,基于先前VPDU或最近解码的VPDU的位置,将当前VPDU中的对应样本指定为不可用。

[0461] 13.如条款9所述的方法,其中,一个或多个参考样本中的每个在三个最近解码的VPDU中的每个中具有对应样本。

[0462] 14.一种用于视频处理的方法,包括:对于基于参考块中的像素进行编解码的当前块,基于当前块的编解码树块(CTB)的尺寸或编解码树单元(CTU)的尺寸来确定与参考块相关联的虚拟缓冲区的尺寸,其中当前图片包括当前块和参考块;以及基于该确定,执行当前块与当前块的比特流表示之间的转换。

[0463] 15.如条款14所述的方法,其中,在确定CTB或CTU的尺寸为128×128时,虚拟缓冲区的尺寸为256×128。

[0464] 16.如条款14所述的方法,其中,在确定CTB或CTU的尺寸为256×256时,虚拟缓冲区的尺寸为64×256。

[0465] 17.如条款14所述的方法,其中,在确定CTB或CTU的尺寸为256×256时,虚拟缓冲区的尺寸为128×256。

[0466] 18.如条款1至17中任一项所述的方法,其中,基于包括当前块的当前图片中的参考块中的像素对当前块进行编解码是帧内块复制(IBC)操作,以及虚拟缓冲区是IBC虚拟缓冲区。

[0467] 19.一种视频处理方法,包括:为当前视频块的比特流表示与当前块之间的转换分配块内编解码(IBC)虚拟缓冲区尺寸,其中该IBC虚拟缓冲区尺寸大于用于该转换的块矢量搜索区域的最小尺寸;以及基于该分配,执行转换。

[0468] 20.如条款19所述的方法,其中,IBC缓冲区尺寸大于用于转换的虚拟管线数据单元存储器的总尺寸。

[0469] 21.如条款19-20中任一项所述的方法,其中,IBC虚拟缓冲区尺寸的宽度是(128*128/ctbSizeY+ctbSizeY)。

[0470] 22.一种视频处理方法,包括:对于当前视频块的比特流表示与当前块之间的转换,基于根据规则的帧内块预测编解码来确定用于转换的参考块的尺寸;以及基于该确定,

执行转换,其中,该规则将参考块限制在虚拟管线数据单元(VPDU)列或虚拟管线数据单元行内。

[0471] 23.如条款22所述的方法,其中,该规则不允许参考块跨越不同的VPDU行。

[0472] 24.如条款22所述的方法,其中,该规则不允许参考块跨越不同的VPDU列。

[0473] 25.如条款22-24中任一项所述的方法,其中,VPDU列或VPDU行与包括当前块的图片相关。

[0474] 26.如条款22-24中任一项所述的方法,其中,VPDU列或VPDU行与帧内块复制虚拟缓冲区相关。

[0475] 27.如条款1-26中任一项所述的方法,其中,转换包括视频编解码以通过当前块生成比特流表示。

[0476] 28.如条款1-26中任一项所述的方法,其中,转换包括视频解码以通过比特流表示生成当前块。

[0477] 29.一种视频系统中的设备,包括处理器和其上具有指令的非瞬态存储器,其中指令在由处理器执行时使得处理器实现条款1到28中任一项所述的方法。

[0478] 30.一种计算机程序产品,其存储在非瞬态计算机可读介质上,并且该计算机程序产品包括用于执行条款1至28中任一项所述的方法的程序代码。

[0479] 31.本文中描述的方法,装置或系统。

[0480] 第二组条款描述了前面章节中公开的技术的某些特征和方面,例如,示例性实现方式1,2,6-8。

[0481] 1、一种用于视频处理的方法,包括:对于视频的视频图片的当前视频块与视频的经编解码的表示之间的转换,基于虚拟管线数据单元(VPDU)的尺寸、编解码树块(CTB)的尺寸或编解码树单元(CTU)的尺寸,针对视频图片的参考区域的尺寸做出决定,其中,来自视频图片的参考区域的参考样本用于预测当前视频块;以及基于该决定,执行转换。

[0482] 2、如条款1所述的方法,其中,使用帧内块复制(IBC)模式对当前视频块进行编解码,其中IBC模式至少使用指向包括当前视频块的视频帧的块矢量来生成预测块;并且参考区域为IBC虚拟缓冲区。

[0483] 3、如条款1或2所述的方法,还包括:基于参考区域的尺寸,确定映射至参考区域的一个或多个样本或块矢量的有效性。

[0484] 4、如条款1或2所述的方法,其中,参考区域的高度和宽度的乘积是固定的,并且其中高度或宽度是基于VPDU的尺寸、CTB的尺寸或CTU的尺寸。

[0485] 5、如条款1或2所述的方法,其中,参考区域的高度等于CTB的高度或CTU的高度。

[0486] 6、如条款1或2所述的方法,其中,参考区域的宽度设置为通过将参考区域的尺寸除以CTB的高度而获得的值。

[0487] 7、如条款1或2所述的方法,其中,参考区域的宽度等于CTB的宽度或CTU的宽度。

[0488] 8、如条款1或2所述的方法,其中,参考区域的宽度为VPDU的宽度的N倍,并且其中N≥1且是整数。

[0489] 9、如条款1或2所述的方法,其中,参考区域的高度为VPDU的高度的N倍,并且其中N≥1且是整数。

[0490] 10、如条款1或2所述的方法,其中,参考区域的宽度为256*128/ctbSizeY,其中

ctbSizeY指示亮度CTB或CTU的宽度或高度。

[0491] 11、如条款1或2所述的方法,其中,参考区域的高度为CtbSizeY,其中ctbSizeY指示亮度CTB或CTU的宽度或高度。

[0492] 12、如条款1或2所述的方法,其中,参考区域的尺寸确定为使得更大的参考区域分配给更大的CTU或更大的CTB。

[0493] 13、如条款1或2所述的方法,其中,参考区域的宽度为(128*128/ctbSizeY+ctbSizeY),其中ctbSizeY指示亮度CTB或CTU的宽度或高度。

[0494] 14、如条款1或2所述的方法,其中,在CTB的尺寸或CTU的尺寸具有128x128亮度样本的尺寸的情况下,参考区域的尺寸对应于256x128亮度样本的尺寸。

[0495] 15、如条款1或2所述的方法,其中,参考区域的尺寸对应于(k x 64)x128亮度样本的尺寸,其中k大于2。

[0496] 16、如条款1或2所述的方法,其中,在CTB的尺寸或CTU的尺寸对应于256x256亮度样本的尺寸的情况下,参考区域的尺寸对应于64x256亮度样本的尺寸。

[0497] 17、如条款16所述的方法,其中,在对具有左上位置(x0,y0)的VPDU进行解码之前,参考区域中对应VPDU行设置为-1。

[0498] 18、如条款1或2所述的方法,其中,在CTB的尺寸或CTU的尺寸具有256x256亮度样本的尺寸的情况下,参考区域的尺寸对应于128x256亮度样本的尺寸。

[0499] 19、如条款18所述的方法,其中,对于参考区域中除特定VPDU行之外的每个VPDU行,仅保持除当前VPDU之外的一个VPDU。

[0500] 20、如条款1至19中任一项所述的方法,其中,转换包括:将当前视频块编码为经编解码的表示。

[0501] 21、如条款1至19中任一项所述的方法,其中,转换包括:将所述经编解码的表示解码为所述当前视频块。

[0502] 22、一种视频处理设备,其中,该视频处理设备包括处理器,该处理器配置为实现条款1至21中的任一项或多项所述的方法。

[0503] 23、一种计算机可读介质,其中,该计算机可读介质存储程序代码,该程序代码在被执行时致使处理器实现如条款1至21中的任一项或多项所述的方法。

[0504] 从上文可以理解,这里为了说明的目的已经描述了当前公开的技术的具体实施例,但是可以在不脱离本发明的范围的情况下进行各种修改。因此,除了所附权利要求之外,本公开的技术不受限制。

[0505] 在本专利文档中描述的主题和功能操作的实现方式可以实现在各种系统、数字电子电路中,或者计算机软件、固件或硬件中,包括在本说明书中公开的结构及其等同结构,或者上述的一个或多个的组合。本说明书中描述的主题的实现方式可以被实现为一个或多个计算机程序产品,即,在有形的非瞬态计算机可读介质上编码的计算机程序指令的一个或多个模块,用于由数据处理设备执行或控制数据处理设备的操作。计算机可读介质可以是机器可读存储设备、机器可读存储基板、存储器设备、影响机器可读传播信号的物质组合物,或上述的一个或多个的组合。术语“数据处理单元”或“数据处理装置”包括用于处理数据的所有装置、设备和机器,包括例如可编程处理器、计算机或多个处理器或计算机。除了硬件之外,该装置还可以包括为所讨论的计算机程序创建执行环境的代码,例如构成处理

器固件、协议栈、数据库管理系统、操作系统或上述的一个或多个的组合的代码。

[0506] 计算机程序(也称为程序、软件、软件应用、脚本或代码)可以以任何形式的编程语言(包括编译或解释语言)来编写,并且可以以任何形式来部署,包括作为独立的程序或作为模块、组件、子例程或其它适合在计算环境中使用的单元。计算机程序不必对应于文件系统中的文件。程序可以存储在保持其它程序或数据(例如,存储在标记语言文档中的一个或多个脚本)的文件的一部分,存储在专用于所讨论的程序的单个文件中,或者存储在多个协同文件(例如,存储一个或多个模块、子程序或代码部分的文件)中。计算机程序可被部署为在一个计算机上执行或在位于一个站点或分布在多个站点上并通过通信网络互连的多个计算机上执行。

[0507] 本说明书中描述的过程和逻辑流程可以由一个或多个可编程处理器来执行,这些可编程处理器执行一个或多个计算机程序,以通过对输入数据进行操作并生成输出来执行功能。过程和逻辑流程也可以由专用逻辑电路(例如,FPGA(现场可编程门阵列)或ASIC(专用集成电路))来执行,并且装置也可以实现为专用逻辑电路。

[0508] 例如,适于执行计算机程序的处理器包括通用和专用微处理器,以及任何类型的数字计算机的任何一个或多个处理器。通常,处理器将从只读存储器或随机存取存储器或只读存储器和随机存取存储器接收指令和数据。计算机的必要元件是用于执行指令的处理器和用于存储指令和数据的一个或多个存储器设备。通常,计算机还将包括或被可操作地耦合以从一个或多个大容量存储设备接收数据或向一个或多个大容量存储设备传送数据,所述大容量存储设备用于存储数据,例如磁盘、磁光盘或光盘。然而,计算机不需要具有这样的设备。适于存储计算机程序指令和数据的计算机可读介质包括所有形式的非易失性存储器、介质和存储器设备,包括例如半导体存储器设备,例如EPROM、EEPROM和闪存设备。处理器和存储器可以由专用逻辑电路补充或结合在专用逻辑电路中。

[0509] 本说明书以及附图旨在被认为仅仅是示例性的,其中示例性意味着示例。如在本文中所使用的,除非上下文另外清楚地指示,否则“或”的使用旨在包括“和/或”。

[0510] 虽然本专利文档包括许多细节,但这些不应被解释为对任何发明的范围或所要求保护的范围的限制,而应被解释为对具体发明的具体实施例可能特定的特征的描述。在本专利文档中在单独实施例的上下文中描述的某些特征也可以在单个实施例中组合实现。相反,在单个实施例的上下文中描述的各种特征也可以在多个实施例中单独地或以任何合适的子组合来实现。此外,尽管上面可以将特征描述为在某些组合中起作用,并且甚至最初如此要求保护,但是在一些情况下,可以从组合中去除要求保护的组合中的一个或多个特征,并且要求保护的组合可以针对子组合或子组合的变型。

[0511] 类似地,虽然在附图中以特定顺序描述了操作,但这不应被理解为要求以所示的特定顺序或以顺序的顺序执行这些操作,或者要求执行所有示出的操作以获得期望的结果。此外,在本专利文献中描述的实施例中的各种系统组件的分离不应被理解为在所有实施例中需要这种分离。

[0512] 仅描述了一些实现方式和示例,并且可以基于在本专利文档中描述和示出的内容来进行其它实现方式、增强和变化。

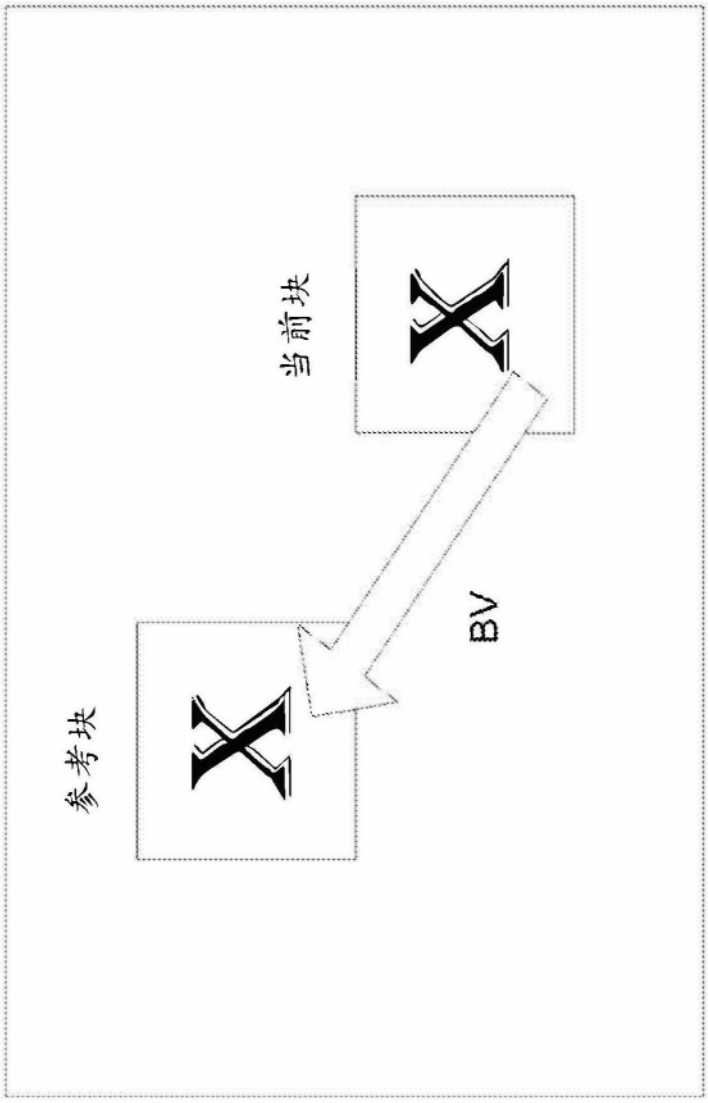


图1

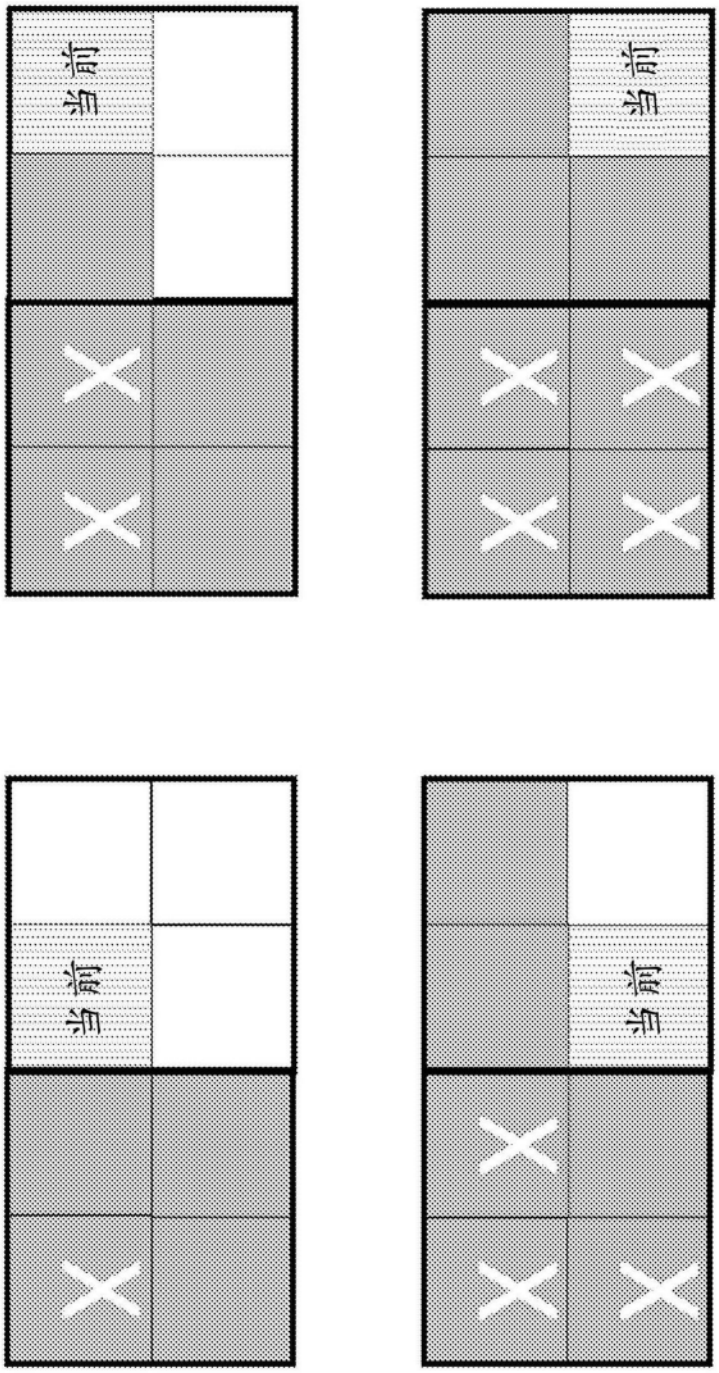


图2

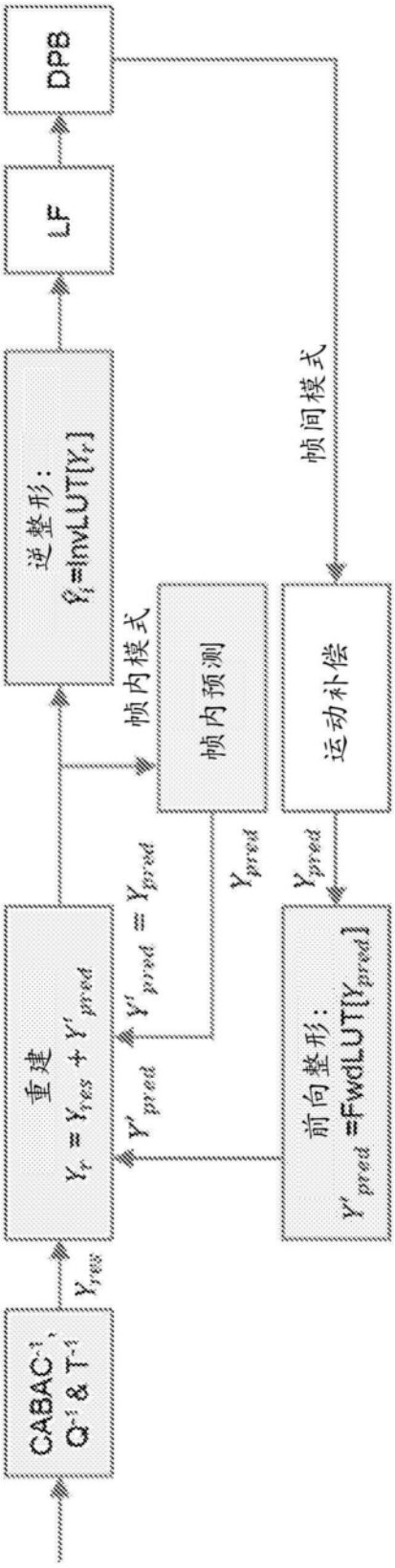

DPB
LF
逆整形：
$\hat{Y}_r$=invLUT[Y_r]
帧间模式
帧内模式
帧内预测
运动补偿
重建
$Y_r = Y_{res} + Y'_{pred}$
$Y'_{pred} = Y_{pred}$
Y_{pred}
Y'_{pred}
Y_{pred}
前向整形：
Y'_{pred}=FwdLUT[Y_{pred}]
Y_{res}
CABAC^{-1},
Q^{-1} & T^{-1}

图3

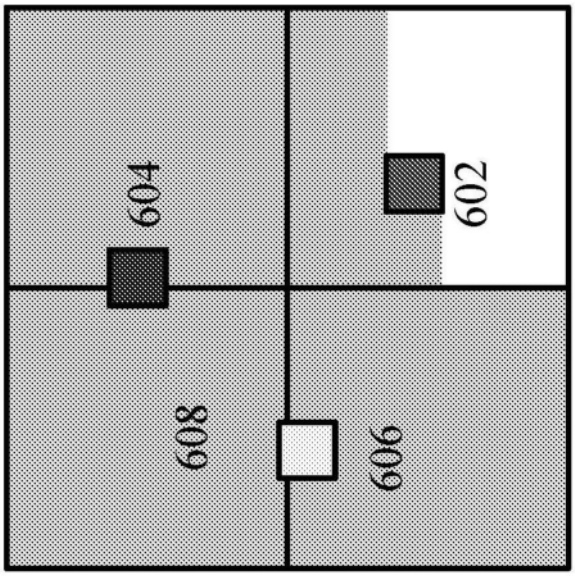


图4

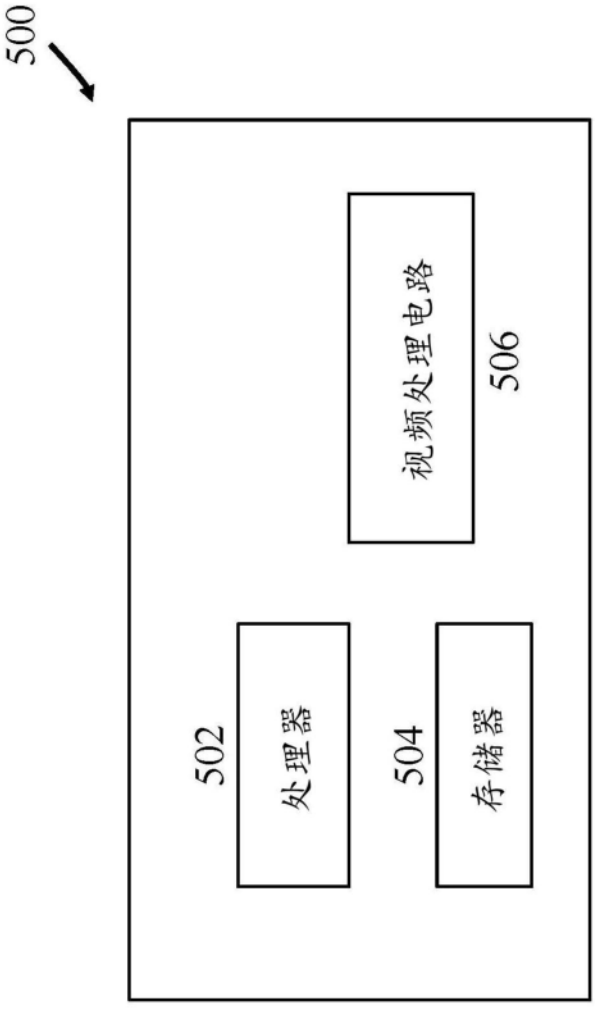


图5

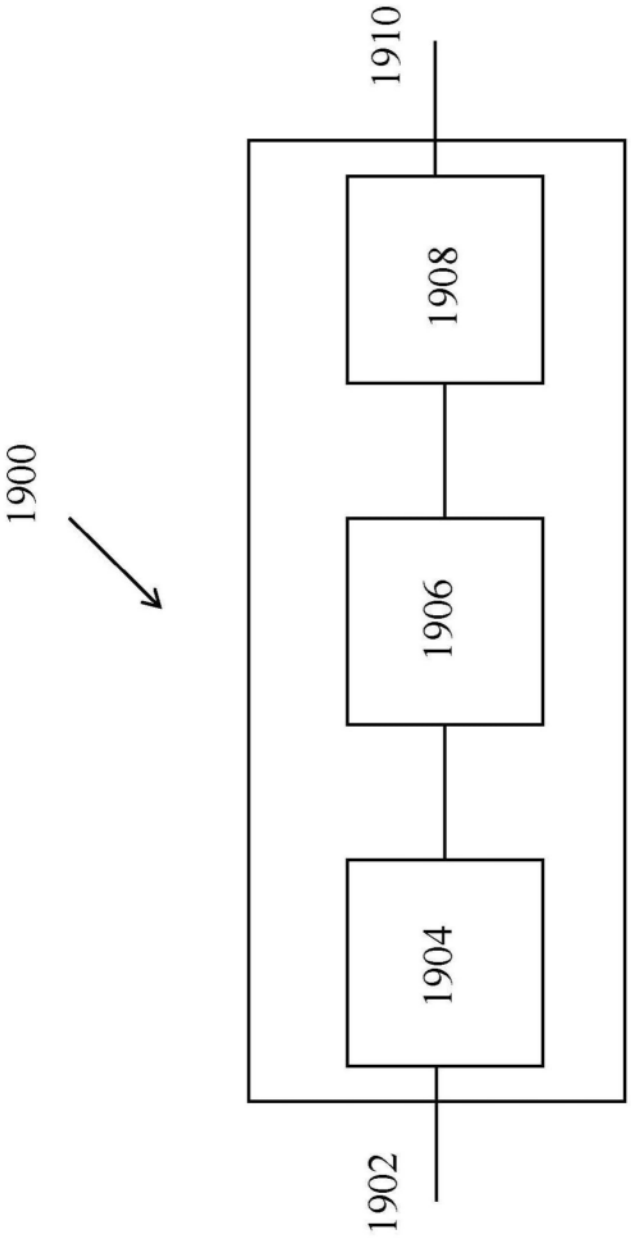


图6

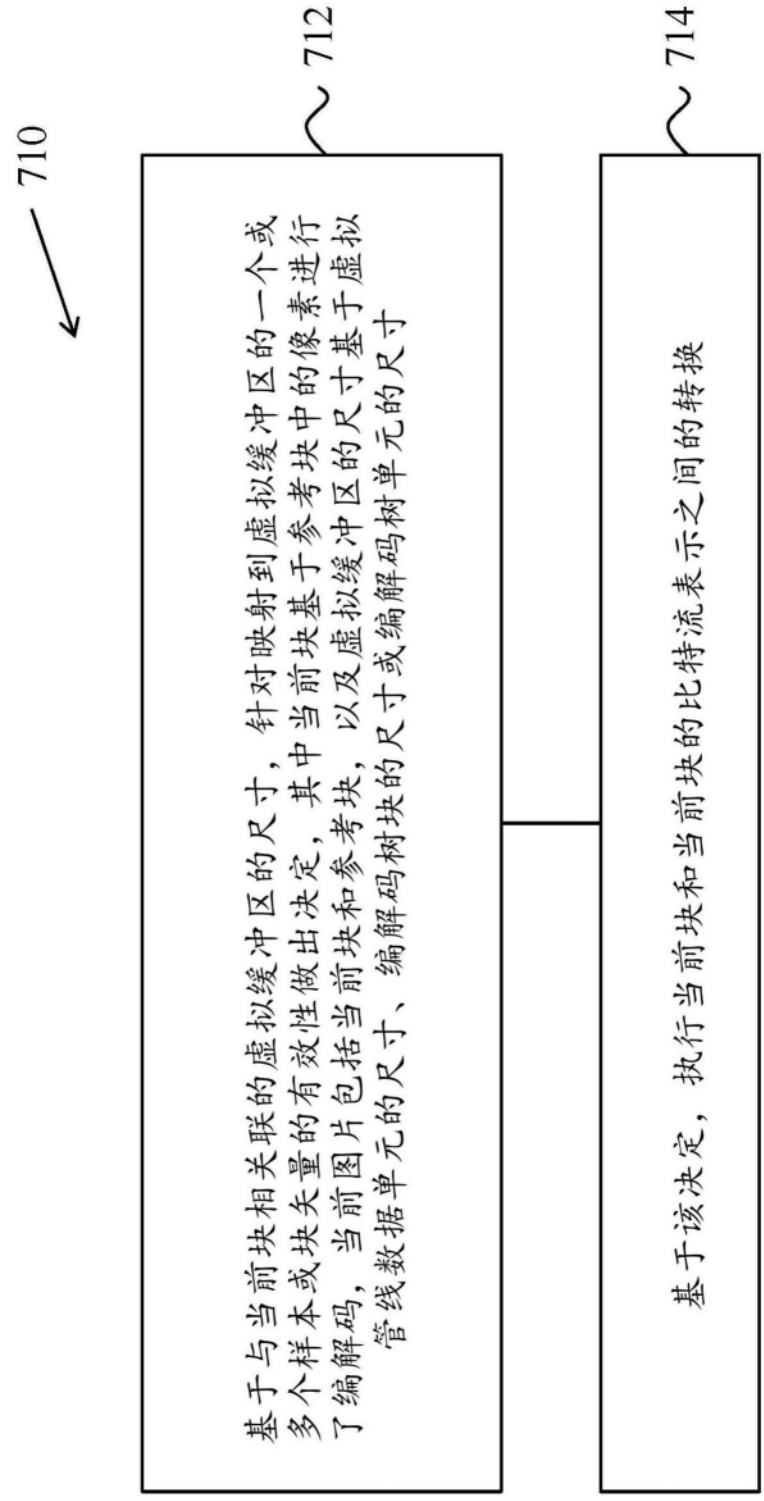

710
712
基于与当前块相关联的虚拟缓冲区的尺寸，针对映射到虚拟缓冲区的一个或多个样本或块矢量的有效性做出决定，其中当前块基于参考块中的像素进行了编解码，当前图片包括当前块和参考块，以及虚拟缓冲区的尺寸基于虚拟管线数据单元的尺寸、编解码树块的尺寸或编解码树单元的尺寸
714
基于该决定，执行当前块和当前块的比特流表示之间的转换

图7A

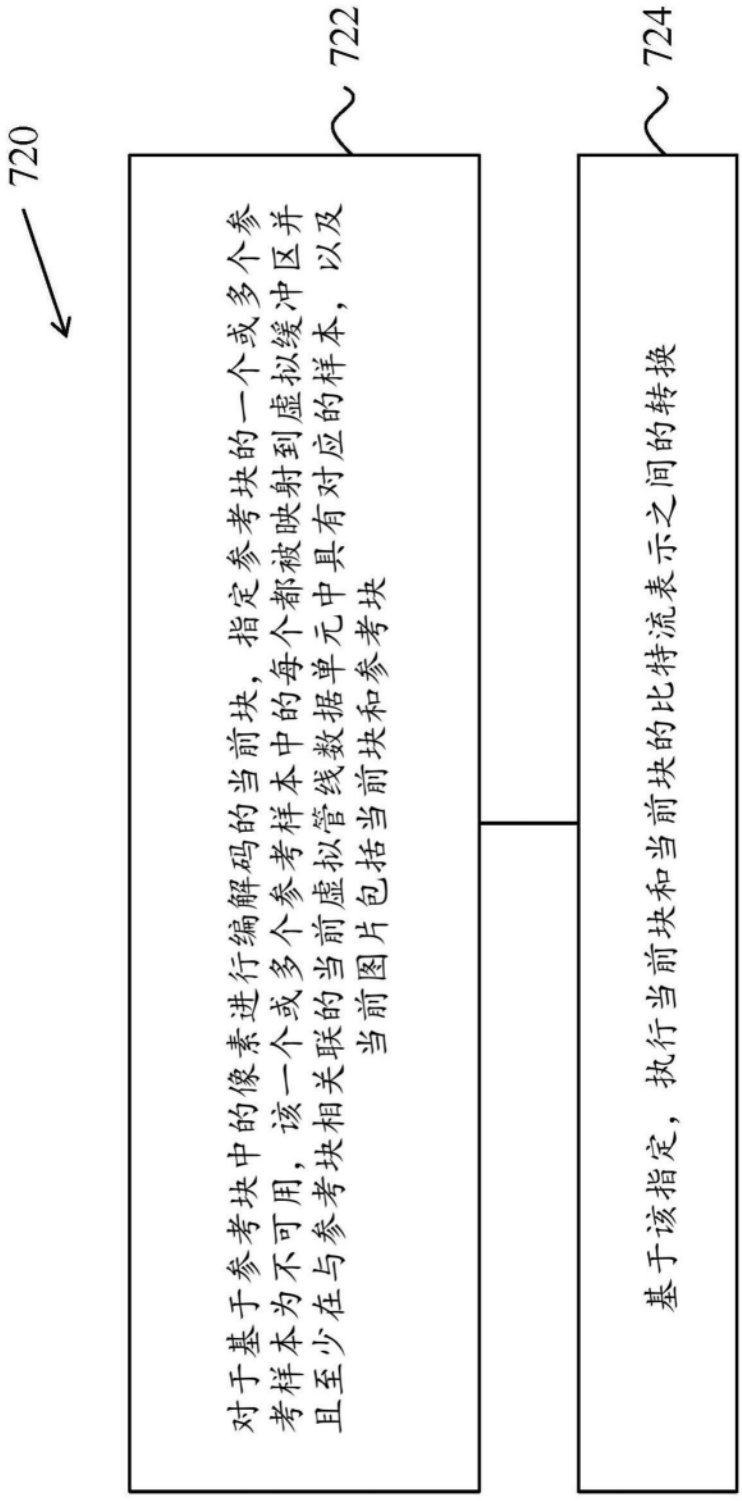

720
722
对于基于参考块中的像素进行编解码的当前块，指定参考块的一个或多个参考样本为不可用，该一个或多个参考样本中的每个都被映射到虚拟缓冲区并且至少在与参考块相关联的当前虚拟管线数据单元中具有对应的样本，以及当前图片包括当前块和参考块
724
基于该指定，执行当前块和当前块的比特流表示之间的转换

图7B

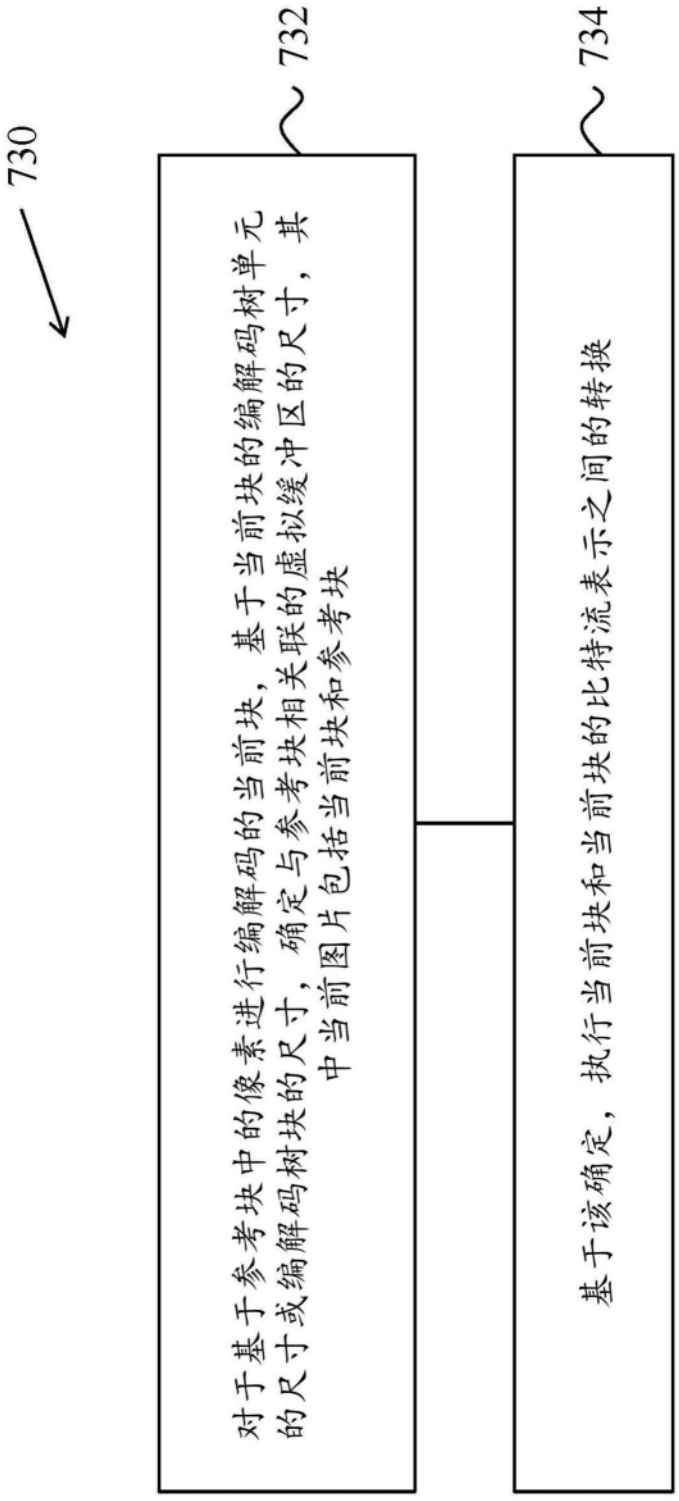

730
732
对于基于参考块中的像素进行编解码的当前块，基于当前块的编解码树单元的尺寸或编解码树块的尺寸，确定与参考块相关联的虚拟缓冲区的尺寸，其中当前图片包括当前块和参考块
734
基于该确定，执行当前块和当前块的比特流表示之间的转换

图7C

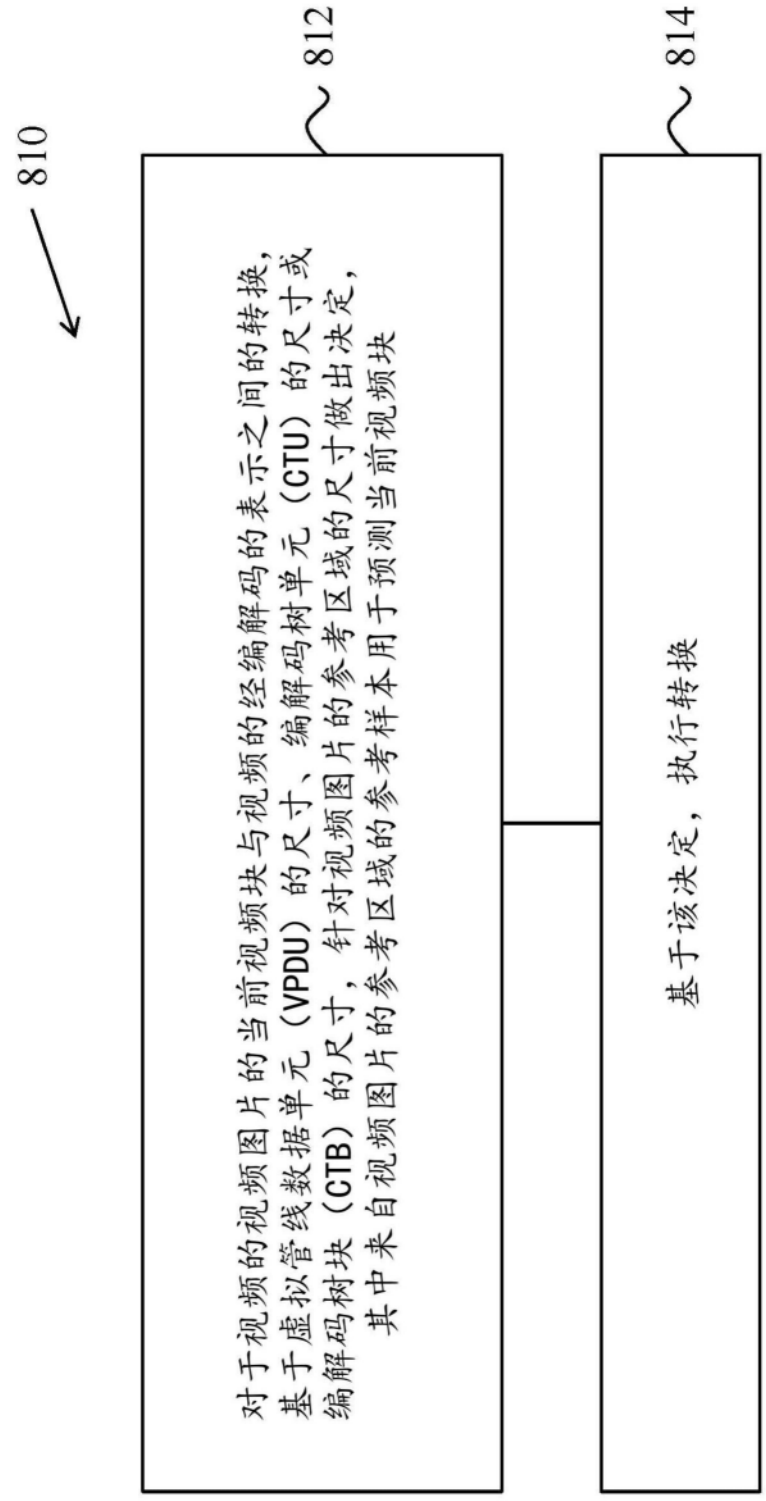

810
812
对于视频的视频图片的当前视频块与视频的经编解码的表示之间的转换，基于虚拟管线数据单元（VPDU）的尺寸、编解码树单元（CTU）的尺寸或编解码树块（CTB）的尺寸，针对视频图片的参考区域的尺寸做出决定，其中来自视频图片的参考区域的参考样本用于预测当前视频块
814
基于该决定，执行转换

图8