(19)中华人民共和国国家知识产权局

(12)发明专利申请

(10)申请公布号 CN 113557724 A
(43)申请公布日 2021.10.26

(21)申请号 202080019967.8

(22)申请日 2020.06.04

(30)优先权数据
62/858,891 2019.06.07 US
62/906,171 2019.09.26 US
16/891,966 2020.06.03 US

(85)PCT国际申请进入国家阶段日
2021.09.09

(86)PCT国际申请的申请数据
PCT/US2020/036066 2020.06.04

(87)PCT国际申请的公布数据
WO2020/247589 EN 2020.12.10

(71)申请人 腾讯美国有限责任公司
地址 美国加利福尼亚州帕洛阿尔托公园大道2747号

(72)发明人 赵亮 赵欣 李翔 刘杉

(74)专利代理机构 北京德琦知识产权代理有限公司 11018
代理人 焦方佼 王琦

(51)Int.Cl.
H04N 19/12(2006.01)

权利要求书3页 说明书51页 附图12页

(54)发明名称

视频编解码的方法和装置

(57)摘要

本公开的各方面提供了用于视频编码/解码的方法、装置、以及非易失性计算机可读存储介质。在一个方法中，确定当前块为本地分区树结构的父节点。本地分区树结构的树深度小于或等于阈值。然后，根据本地分区树结构，对当前块进行分区。基于当前块的预测模式，重建当前块。在另一个方法中，将当前块的色度样本划分为子块。基于色度帧内预测模式的子集，预测一个子块的色度样本，其中，色度帧内预测模式包括平面，直流DC，水平，垂直，衍生模式DM，左交叉分量线性模式L_CCLM、顶部交叉分量线性模式T_CCLM，以及左和顶部交叉分量线性模式LT_CCLM。然后，基于预测的色度样本，重建当前块。

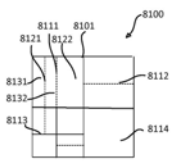

图 8A

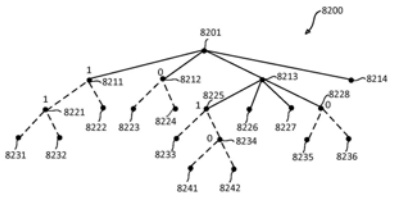

图 8B

CN 113557724 A

1.一种在解码器中进行视频解码的方法,其特征在于,包括:

解码当前图片中当前块的预测信息,其中,所述当前图片为已编码视频序列的一部分,所述预测信息指示所述当前块的单分区树结构和块大小;

基于所述当前块的单分区树结构和块大小,确定所述当前块是否是本地分区树结构的父节点,所述本地分区树结构的树深度小于或等于阈值;

当所述当前块是所述本地分区树结构的父节点时,根据所述本地分区树结构对所述当前块进行分区;以及,

基于所述当前块的预测模式重建所述当前块。

2.根据权利要求1所述的方法,其特征在于,基于(i)所述块大小为64个样本,并且所述本地分区树结构为四叉树,或(ii)所述块大小为64个样本,并且所述本地分区树结构为三叉树,或(iii)所述块大小为32个样本,并且所述本地分区树结构为二叉树,确定所述当前块的预测模式为非帧间预测。

3.根据权利要求1所述的方法,其特征在于,基于所述预测信息中包括的发信号通知的标志,以及(i)所述块大小为64个样本,并且所述本地分区树结构为二叉树,或(ii)所述块大小为128个样本,并且所述本地分区树结构为三叉树,确定所述当前块的预测模式。

4.根据权利要求1所述的方法,其特征在于,所述阈值是基于帧间编码的所述当前块的第一阈值和基于非帧间编码的所述当前块的第二阈值,所述第一阈值不同于所述第二阈值。

5.根据权利要求1所述的方法,其特征在于,进一步包括:

基于所述当前块是所述本地分区树结构的父节点,确定所述当前块为小色度帧内预测单元SCIPU。

6.根据权利要求1所述的方法,其特征在于,所述分区进一步包括:

基于是否确定所述当前块的预测模式,对所述当前块进行分区。

7.根据权利要求1所述的方法,其特征在于,所述分区进一步包括:

基于所述当前块是否被帧间编码,对所述当前块进行分区。

8.一种在解码器中进行视频解码的方法,其特征在于,包括:

解码当前图片中当前块的预测信息,其中,所述当前图片为已编码视频序列的一部分,所述预测信息指示所述当前块的色度样本的分区树结构;

基于所述分区树结构,将所述当前块的色度样本分区成多个子块;

当(i)所述子块的上相邻样本和左相邻样本中的至少一个不可用于预测所述子块,以及,(ii)所述子块的块大小小于或等于大小阈值,或所述子块的边长小于或等于长度阈值,基于色度帧内预测模式的子集,预测所述多个子块中的所述子块的色度样本;以及,

基于预测的所述色度样本,重建所述当前块。

9.根据权利要求8所述的方法,其特征在于,所述当前块的块大小大于所述大小阈值,并且所述子块的块大小小于或等于所述大小阈值,或所述当前块的边长大于所述长度阈值,并且所述子块的边长小于或等于所述长度阈值。

10.根据权利要求8所述的方法,其特征在于,所述大小阈值包括2x2,2x4和4x2之一,以及允许的最小亮度帧内编码块的块尺寸,所述长度阈值包括2、4和允许的最小亮度帧内编码块的边长之一。

11.根据权利要求8所述的方法,其特征在于,所述色度帧内预测模式包括平面模式、DC模式、水平模式、垂直模式、衍生模式DM、左交叉分量线性模式L_CCLM、顶部交叉分量线性模式T_CCLM,以及左和顶部交叉分量线性模式LT_CCLM。

12.根据权利要求8所述的方法,其特征在于,所述色度帧内预测模式的子集包括一种或两种色度帧内预测模式。

13.根据权利要求8所述的方法,其特征在于,若所述子块的上相邻样本不可用于预测所述子块并且位于所述当前块内,所述色度帧内预测模式的子集包括衍生模式DM、左交叉分量线性模式L_CCLM和垂直模式中的至少一个。

14.根据权利要求8所述的方法,其特征在于,若所述子块的左相邻样本不可用于预测所述子块并且位于所述当前块内,所述色度帧内预测模式的子集包括衍生模式DM、顶部交叉分量线性模式T_CCLM和水平模式中的至少一个。

15.根据权利要求8所述的方法,其特征在于,所述当前块是一个并行处理区域PPR,其中所有子块都被并行重建。

16.一种装置,其特征在于,包括处理电路,用于:

解码当前图片中当前块的预测信息,其中,所述当前图片为已编码视频序列的一部分,所述预测信息指示所述当前块的单分区树结构和块大小;

基于所述当前块的单分区树结构和块大小,确定所述当前块是否是本地分区树结构的父节点,所述本地分区树结构的树深度小于或等于阈值;

当所述当前块是所述本地分区树结构的父节点时,根据所述本地分区树结构对所述当前块进行分区;以及,

基于所述当前块的预测模式重建所述当前块。

17.根据权利要求16所述的装置,其特征在于,所述处理电路还被配置为以下至少一项:

基于是否已确定所述当前块的预测模式,对所述当前块进行分区;以及,

基于所述当前块是否被帧间编码,对所述当前块进行分区。

18.一种装置,其特征在于,包括处理电路,用于:

解码当前图片中当前块的预测信息,其中,所述当前图片为已编码视频序列的一部分,所述预测信息指示所述当前块的色度样本的分区树结构;

基于所述分区树结构,将所述当前块的色度样本分区成多个子块;

当(i)所述子块的上相邻样本和左相邻样本中的至少一个不可用于预测所述子块,以及,(ii)所述子块的块大小小于或等于大小阈值,或所述子块的边长小于或等于长度阈值,基于色度帧内预测模式的子集,预测所述多个子块中的所述子块的色度样本;以及,

基于预测的所述色度样本,重建所述当前块。

19.一种非易失性计算机可读存储介质,其特征在于,所述非易失性计算机可读存储介质存储程序,所述程序可由至少一个处理器运行,以执行:

解码当前图片中当前块的预测信息,其中,所述当前图片为已编码视频序列的一部分,所述预测信息指示所述当前块的单分区树结构和块大小;

基于所述当前块的单分区树结构和块大小,确定所述当前块是否是本地分区树结构的父节点,所述本地分区树结构的树深度小于或等于阈值;

当所述当前块是所述本地分区树结构的父节点时，根据所述本地分区树结构对所述当前块进行分区；以及，

基于所述当前块的预测模式重建所述当前块。

20.一种非易失性计算机可读存储介质，其特征在于，所述非易失性计算机可读存储介质存储程序，所述程序可由至少一个处理器运行，以执行：

解码当前图片中当前块的预测信息，其中，所述当前图片为已编码视频序列的一部分，所述预测信息指示所述当前块的色度样本的分区树结构；

基于所述分区树结构，将所述当前块的色度样本分区成多个子块；

当(i)所述子块的上相邻样本和左相邻样本中的至少一个不可用于预测所述子块，以及，(ii)所述子块的块大小小于或等于大小阈值，或所述子块的边长小于或等于长度阈值，基于色度帧内预测模式的子集，预测所述多个子块中的所述子块的色度样本；以及，

基于预测的所述色度样本，重建所述当前块。

视频编解码的方法和装置

[0001] 引用并入

[0002] 本申请要求于2020年6月3日提交的美国申请第16/891,966号“视频编解码的方法和装置”的优先权,该美国申请要求了2019年6月7日提交的美国临时申请第62/858,891号“小色度块大小限制”的优先权,以及2019年9月26日提交的美国临时申请第62/906,171号“本地双树的改进”的优先权。在先申请的全部公开内容通过引用整体并入本文。

技术领域

[0003] 本公开描述了与视频编解码有关的实施例。

背景技术

[0004] 本文所提供的背景描述旨在整体呈现本公开的背景。在背景技术部分以及本说明书的各个方面中所描述的目前已署名的发明人的工作所进行的程度,并不表明其在本申请提交时作为现有技术,且从未明示或暗示其被承认为本公开的现有技术。

[0005] 通过具有运动补偿的帧间图片预测技术,可以进行视频编码和解码。未压缩的数字视频可包括一系列图片,每个图片具有例如1920×1080亮度样本及相关色度样本的空间维度。所述系列图片具有固定的或可变的图片速率(也非正式地称为帧率),例如每秒60个图片或60Hz。未压缩的视频具有非常大的比特率要求。例如,每个样本8比特的1080p60 4:2:0的视频(1920x1080亮度样本分辨率,60Hz帧率)要求接近1.5Gbit/s带宽。一小时这样的视频就需要超过600GB的存储空间。

[0006] 视频编码和解码的一个目的,是通过压缩减少输入视频信号的冗余信息。视频压缩可以帮助降低对上述带宽或存储空间的要求,在某些情况下可降低两个或更多数量级。无损和有损压缩,以及两者的组合均可采用。无损压缩是指从压缩的原始信号中重建原始信号精确副本的技术。当使用有损压缩时,重建信号可能与原始信号不完全相同,但是原始信号和重建信号之间的失真足够小,使得重建信号可用于预期应用。有损压缩广泛应用于视频。容许的失真量取决于应用。例如,相比于电视应用的用户,某些消费流媒体应用的用户可以容忍更高的失真。可实现的压缩比反映出:较高的允许/容许失真可产生较高的压缩比。

[0007] 视频编码器和解码器可利用几大类技术,例如包括:运动补偿、变换、量化和熵编码。

[0008] 视频编解码器技术可包括已知的帧内编码技术。在帧内编码中,在不参考先前重建的参考图片的样本或其它数据的情况下表示样本值。在一些视频编解码器中,图片在空间上被细分为样本块。当所有的样本块都以帧内模式编码时,该图片可以为帧内图片。帧内图片及其衍生(例如独立解码器刷新图片)可用于复位解码器状态,并且因此可用作编码视频比特流和视频会话中的第一图片,或用作静止图像。帧内块的样本可用于变换,且可在熵编码之前量化变换系数。帧内预测可以是使预变换域中的样本值最小化的技术。在某些情形下,变换后的DC值越小,且AC系数越小,则在给定的量化步长尺寸下需要越少的比特来表

示熵编码之后的块。

[0009] 如同从诸如MPEG-2代编码技术中所获知的,传统帧内编码不使用帧内预测。然而,一些较新的视频压缩技术包括:试图从例如周围样本数据和/或元数据中得到数据块的技术,其中周围样本数据和/或元数据是在空间相邻的编码/解码期间、且在解码顺序之前获得的。这种技术后来被称为"帧内预测"技术。需要注意的是,至少在某些情形下,帧内预测仅使用正在重建的当前图片的参考数据,而不使用参考图片的参考数据。

[0010] 可以存在许多不同形式的帧内预测。当在给定的视频编码技术中可以使用超过一种这样的技术时,所使用的技术可以按帧内预测模式进行编码。在某些情形下,模式可具有子模式和/或参数,且这些模式可单独编码或包含在模式码字中。将哪个码字用于给定模式/子模式/参数组合会通过帧内预测影响编码效率增益,因此用于将码字转换成比特流的熵编码技术也会出现这种情况。

[0011] H.264引入了一种帧内预测模式,其在H.265中进行了改进,且在诸如联合开发模型(Joint Exploration Model,JEM),多功能视频编码(Versatile Video Coding,VVC),基准集(Benchmark Set,BMS)的更新的编码技术中进一步被改进。通过使用属于已经可用的样本的相邻样本值可以形成预测块。将相邻样本的样本值按照某一方向复制到预测块中。对所使用方向的引用可以被编码在比特流中,或者本身可以被预测。

[0012] 参照图1A,右下方描绘了来自H.265的33个(对应于35种帧内模式中的33种角度模式)可能的预测方向中已知的九个预测方向的子集。箭头会聚的点(101)表示正在被预测的样本。箭头表示样本正在被预测的方向。例如,箭头(102)表示根据右上方与水平方向成45度角的一个或多个样本,预测样本(101)。类似地,箭头(103)表示根据左下方与水平方向成22.5度角的一个或多个样本,预测样本(101)。

[0013] 仍然参考图1A,在左上方示出了一个包括4×4个样本的正方形块(104)(由粗虚线表示)。正方形块(104)包括16个样本,每个样本用"S"、以及其在Y维度上的位置(例如,行索引)和在X纬度上的位置(例如,列索引)来标记。例如,样本S21是Y维度上的第二个样本(从上方开始)和X维度上的第一个(从左侧开始)样本。类似地,样本S44在Y维度和X维度上都是块(104)中的第四个样本。由于该块为4×4大小的样本,因此S44位于右下角。还示出了遵循类似编号方案的参考样本。参考样本用"R"、以及其相对于块(104)的Y位置(例如,行索引)和X位置(列索引)来标记。在H.264和H.265中,预测样本都与正在重建的块相邻;因此,无需使用负值。

[0014] 通过从信号通知的预测方向所占用的相邻样本来复制参考样本值,可以进行帧内图片预测。例如,假设编码视频比特流包括信令,对于该块,该信令指示与箭头(102)一致的预测方向,即,根据右上方与水平方向成45度角的一个或多个预测样本来预测样本。在这种情况下,根据同一参考样本R05,-1),预测样本S41、S32、S23和S14。根据参考样本R08,预测样本S44。

[0015] 在某些情况下,例如通过内插,可以合并多个参考样本的值,以便计算参考样本,尤其是当方向不能被45度整除时。

[0016] 随着视频编码技术的发展,可能的方向的数量已经增加了。在H.264(2003年)中,可以表示九种不同的方向。在H.265(2013年)和JEM/VVC/BMS中增加到了33个,而在此申请时,可以支持多达93个方向。已经进行了实验来识别最可能的方向,并且熵编码中的某些技

术被用于使用少量比特来表示那些可能的方向，对于较不可能的方向则接受某些代价。此外，有时可以根据在相邻的、已经解码的块中所使用的相邻方向来预测方向本身。

[0017] 图1B示出了示意图(105)，其描述了根据JEM的65种帧内预测方向，以说明随着时间的推移预测方向的数量增加。

[0018] 表示方向的编码视频比特流中的帧内预测方向比特的映射可以因视频编码技术的不同而不同，并且，例如可以从对帧内预测模式到码字的预测方向的简单直接映射，到包括最可能的模式和类似技术的复杂的自适应方案。然而，在所有情况下，视频内容中可能存在某些方向，其在统计学上比其它方向更不可能出现。由于视频压缩的目的是减少冗余，所以在运行良好的视频编码技术中，与更可能的方向相比，那些不太可能的方向将使用更多数量的比特来表示。

[0019] 运动补偿可以是一种有损压缩技术，且可涉及如下技术：来自先前重建的图片或重建图片一部分(参考图片)的样本数据块在空间上按运动矢量(下文称为MV)指示的方向移位后，用于新重建的图片或图片部分的预测。在某些情况下，参考图片可与当前正在重建的图片相同。MV可具有两个维度X和Y，或者三个维度，其中第三个维度表示使用中的参考图片(后者间接地可为时间维度)。

[0020] 在一些视频压缩技术中，应用于某个样本数据区域的MV可根据其它MV来预测，例如根据与正在重建的区域空间相邻的另一个样本数据区域相关的、且按解码顺序在该MV前面的那些MV。这样做可以大大减少编码MV所需的数据量，从而消除冗余信息并增加压缩量。MV预测可以有效地进行，例如，当对从相机导出的输入视频信号(称为自然视频)进行编码时，存在一种统计上的可能性，即面积大于单个MV适用区域的区域，会朝着类似的方向移动，因此，在某些情况下，可以用邻近区域的MV导出的相似运动矢量进行预测。这导致针对给定区域发现的MV与根据周围MV预测的MV相似或相同，并且在熵编码之后，又可以用比直接编码MV时使用的比特数更少的比特数来表示。在某些情况下，MV预测可以是对从原始信号(即样本流)导出的信号(即MV)进行无损压缩的示例。在其它情况下，MV预测本身可能是有损的，例如由于根据几个周围MV计算预测值时产生的取整误差。

[0021] H.265/HEVC(ITU-T H.265建议书，“高效视频编解码(High Efficiency Video Coding)”，2016年12月)中描述了各种MV预测机制。在H.265提供的多种MV预测机制中，本申请描述的是下文称作“空间合并”的技术。

[0022] 请参考图1C，当前块(111)可以包括在运动搜索过程期间已由编码器发现的样本，根据已产生空间偏移的相同大小的先前块，可预测所述样本。另外，可从一个或多个参考图片相关联的元数据中导出所述MV，而非对MV直接编码。例如，使用关联于A0、A1和B0、B1、B2(分别对应112到116)五个周围样本中的任一样本的MV，(按解码次序)从最近的参考图片的元数据中导出所述MV。在H.265中，MV预测可使用相邻块也正在使用的相同参考图片的预测值。

发明内容

[0023] 本公开的多个各方面提供了视频编解码的方法和装置。在一些示例中，视频解码装置包括处理电路。

[0024] 根据本公开的多个方面，提供了一种在解码器中进行视频解码的方法。在该方法

中，解码当前图片中当前块的预测信息，其中，所述当前图片为已编码视频序列的一部分。所述预测信息指示所述当前块的单分区树结构和块大小。基于所述当前块的单分区树结构和块大小，确定所述当前块是否是本地分区树结构的父节点。所述本地分区树结构的树深度小于或等于阈值。当所述当前块是所述本地分区树结构的父节点时，根据所述本地分区树结构对所述当前块进行分区。基于所述当前块的预测模式重建所述当前块。

[0025] 在一个实施例中，基于(i)所述块大小为64个样本，并且所述本地分区树结构为四叉树，或(ii)所述块大小为64个样本，并且所述本地分区树结构为三叉树，或(iii)所述块大小为32个样本，并且所述本地分区树结构为二叉树，确定所述当前块的预测模式为非帧间预测。

[0026] 在一个实施例中，基于所述预测信息中包括的发信号通知的标志，以及(i)所述块大小为64个样本，并且所述本地分区树结构为二叉树，或(ii)所述块大小为128个样本，并且所述本地分区树结构为三叉树，确定所述当前块的预测模式。

[0027] 在一个实施例中，所述阈值是基于帧间编码的所述当前块的第一阈值和基于非帧间编码的所述当前块的第二阈值，所述第一阈值不同于所述第二阈值。

[0028] 在一个实施例中，基于所述当前块是所述本地分区树结构的父节点，确定所述当前块为小色度帧内预测单元SCIPU。

[0029] 在一个实施例中，基于是否确定所述当前块的预测模式，对所述当前块进行分区。

[0030] 在一个实施例中，基于所述当前块是否被帧间编码，对所述当前块进行分区。

[0031] 本公开的各方面提供了一种装置，用于执行任一种视频解码方法、或视频解码方法的组合。在一个实施例中，该装置包括处理电路，用于解码当前图片中当前块的预测信息，其中，所述当前图片为已编码视频序列的一部分。所述预测信息指示所述当前块的单分区树结构和块大小。所述处理电路基于所述当前块的单分区树结构和块大小，确定所述当前块是否是本地分区树结构的父节点。所述本地分区树结构的树深度小于或等于阈值。当所述当前块是所述本地分区树结构的父节点时，所述处理电路根据所述本地分区树结构对所述当前块进行分区。所述处理电路基于所述当前块的预测模式重建所述当前块。

[0032] 根据本公开的各方面，提供了一种在解码器中进行视频解码的方法。在该方法中，解码当前图片中当前块的预测信息，其中，所述当前图片为已编码视频序列的一部分。所述预测信息指示所述当前块的色度样本的分区树结构。基于所述分区树结构，将所述当前块的色度样本分区成多个子块。当(i)所述子块的上相邻样本和左相邻样本中的至少一个不可用于预测所述子块，以及，(ii)所述子块的块大小小于或等于大小阈值，或所述子块的边长小于或等于长度阈值，基于色度帧内预测模式的子集，预测所述多个子块中的所述子块的色度样本。基于预测的所述色度样本，重建所述当前块。

[0033] 在一个实施例中，所述当前块的块大小大于所述大小阈值，并且所述子块的块大小小于或等于所述大小阈值，或所述当前块的边长大于所述长度阈值，并且所述子块的边长小于或等于所述长度阈值。

[0034] 在一个实施例中，所述大小阈值包括2x2，2x4和4x2之一，以及允许的最小亮度帧内编码块的块尺寸，所述长度阈值包括2、4和允许的最小亮度帧内编码块的边长之一。

[0035] 在一个实施例中，所述色度帧内预测模式包括平面模式、DC模式、水平模式、垂直模式、衍生模式DM、左交叉分量线性模式L_CCLM、顶部交叉分量线性模式T_CCLM，以及左和

顶部交叉分量线性模式LT_CCLM。

[0036] 在一个实施例中,所述色度帧内预测模式的子集包括一种或两种色度帧内预测模式。

[0037] 在一个实施例中,若所述子块的上相邻样本不可用于预测所述子块并且位于所述当前块内,所述色度帧内预测模式的子集包括衍生模式DM、左交叉分量线性模式L_CCLM和垂直模式中的至少一个。

[0038] 在一个实施例中,若所述子块的左相邻样本不可用于预测所述子块并且位于所述当前块内,所述色度帧内预测模式的子集包括衍生模式DM、顶部交叉分量线性模式T_CCLM和水平模式中的至少一个。

[0039] 在一个实施例中,所述当前块是一个并行处理区域PPR,其中所有子块都被并行重建

[0040] 本公开的各方面提供了一种装置,用于执行任一种视频解码方法、或视频解码方法的组合。在一个实施例中,该装置包括处理电路,用于解码当前图片中当前块的预测信息,其中,所述当前图片为已编码视频序列的一部分。所述预测信息指示所述当前块的色度样本的分区树结构。所述处理电路基于所述分区树结构,将所述当前块的色度样本分区成多个子块。当(i)所述子块的上相邻样本和左相邻样本中的至少一个不可用于预测所述子块,以及,(ii)所述子块的块大小小于或等于大小阈值,或所述子块的边长小于或等于长度阈值,所述处理电路基于色度帧内预测模式的子集,预测所述多个子块中的所述子块的色度样本。所述处理电路基于预测的所述色度样本,重建所述当前块。

[0041] 本公开的多个方面还提供了一种非易失性计算机可读介质,其中,所述非易失性计算机可读介质存储有指令,当所述指令由计算机执行用于视频解码时,使得所述计算机执行上述视频解码方法的一种或其组合。

[0042] 附图简要说明

[0043] 根据以下详细描述和附图,所公开的主题的其他特征、性质和各种优点将进一步明确,其中:

[0044] 图1A示出了帧内预测模式的示例性子集的示意图。

[0045] 图1B示出了示例性的帧内预测方向。

[0046] 图1C示出了在一个示例中当前块及其周围的空间合并候选的示意图。

[0047] 图2示出了根据一个实施例的通信系统的简化框图。

[0048] 图3示出了根据一个实施例的通信系统的简化框图。

[0049] 图4示出了根据一个实施例的解码器的简化框图。

[0050] 图5示出了根据一个实施例的编码器的简化框图。

[0051] 图6示出了根据另一个实施例的编码器的简化框图。

[0052] 图7示出了根据另一个实施例的解码器的简化框图。

[0053] 图8A和图8B示出了使用四叉树加二叉树(Quad-Tree plus Binary Tree,QTBT)分区结构和对应QTBT结构的块分区的示例。

[0054] 图9A示出了垂直中心侧三叉树(ternary tree,TT)分区的示例。

[0055] 图9B示出了水平中心侧三叉树分区的示例。

[0056] 图10示出了在一些示例中示例性的帧内预测方向。

[0057] 图11A-11E示出了根据实施例的可并行处理区域(Parallel-Processable Region,PPR)的示例性形状。

[0058] 图12示出了根据实施例的概述示例性过程的流程图。

[0059] 图13A和13B示出了根据实施例的一些示例性子分区。

[0060] 图14A-14D示出了根据实施例的不同的YUV格式。

[0061] 图15示出了根据实施例的示例性多参考线帧内预测。

[0062] 图16示出了根据实施例的概述示例性过程的流程图。

[0063] 图17示出了根据实施例的计算机系统的示意图。

具体实施方式

[0064] 本公开包括针对一个或多个小块大小和实现本地双树的实施例。实施例包括用于改进小块(例如,小色度块)限制和使用本地双树的方法、装置和非易失性计算机可读存储介质。另外,可以将块理解为预测块、编码块或编码单元(Coding Unit,CU)。

[0065] I.视频编码器和解码器

[0066] 图2是根据本申请公开的实施例的通信系统(200)的简化框图。通信系统(200)包括多个终端装置,所述终端装置可通过例如网络(250)彼此通信。举例来说,通信系统(200)包括通过网络(250)互连的第一终端装置(210)和第二终端装置(220)。在图2的实施例中,第一终端装置(210)和第二终端装置(220)执行单向数据传输。举例来说,第一终端装置(210)可对视频数据(例如由终端装置(210)采集的视频图片流)进行编码以通过网络(250)传输到第二端装置(220)。已编码的视频数据以至少一个已编码视频码流形式传输。第二终端装置(220)可从网络(250)接收已编码视频数据,对已编码视频数据进行解码以恢复视频数据,并根据恢复的视频数据显示视频图片。单向数据传输在媒体服务等应用中是较常见的。

[0067] 在另一实施例中,通信系统(200)包括执行已编码视频数据的双向传输的第三终端装置(230)和第四终端装置(240),所述双向传输可例如在视频会议期间发生。对于双向数据传输,第三终端装置(230)和第四终端装置(240)中的每个终端装置可对视频数据(例如由终端装置采集的视频图片流)进行编码,以通过网络(250)传输到第三终端装置(230)和第四终端装置(240)中的另一终端装置。第三终端装置(230)和第四终端装置(240)中的每个终端装置还可接收由第三终端装置(230)和第四终端装置(240)中的另一终端装置传输的已编码视频数据,且可对所述已编码视频数据进行解码以恢复视频数据,且可根据恢复的视频数据在可访问的显示装置上显示视频图片。

[0068] 在图2的实施例中,第一终端装置(210)、第二终端装置(220)、第三终端装置(230)和第四终端装置(240)可为服务器、个人计算机和智能电话,但本申请公开的原理可不限于此。本申请公开的实施例适用于膝上型计算机、平板电脑、媒体播放器和/或专用视频会议设备。网络(250)表示在第一终端装置(210)、第二终端装置(220)、第三终端装置(230)和第四终端装置(240)之间传送已编码视频数据的任何数目的网络,包括例如有线(连线的)和/或无线通信网络。通信网络(250)可在电路交换和/或分组交换信道中交换数据。该网络可包括电信网络、局域网、广域网和/或互联网。出于本申请的目的,除非在下文中有所解释,否则网络(250)的架构和拓扑对于本申请公开的操作来说可能是无关紧要的。

[0069] 作为实施例,图3示出视频编码器和视频解码器在流式传输环境中的放置方式。本申请所公开主题可同等地适用于其它支持视频的应用,包括例如视频会议、数字TV、在包括CD、DVD、存储棒等的数字介质上存储压缩视频等等。

[0070] 流式传输系统可包括采集子系统(313),所述采集子系统可包括数码相机等视频源(301),所述视频源创建未压缩的视频图片流(302)。在实施例中,视频图片流(302)包括由数码相机拍摄的样本。相较于已编码的视频数据(304)(或已编码的视频码流),视频图片流(302)被描绘为粗线以强调高数据量的视频图片流,视频图片流(302)可由电子装置(320)处理,所述电子装置(320)包括耦接到视频源(301)的视频编码器(303)。视频编码器(303)可包括硬件、软件或软硬件组合以实现或实施如下文更详细地描述的所公开主题的各方面。相较于视频图片流(302),已编码的视频数据(304)(或已编码的视频码流(304))被描绘为细线以强调较低数据量的已编码的视频数据(304)(或已编码的视频码流(304)),其可存储在流式传输服务器(305)上以供将来使用。至少一个流式传输客户端子系统,例如图3中的客户端子系统(306)和客户端子系统(308),可访问流式传输服务器(305)以检索已编码的视频数据(304)的副本(307)和副本(309)。客户端子系统(306)可包括例如电子装置(330)中的视频解码器(310)。视频解码器(310)对已编码的视频数据的传入副本(307)进行解码,且产生可在显示器(312)(例如显示屏)或另一呈现装置(未描绘)上呈现的输出视频图片流(311)。在一些流式传输系统中,可根据某些视频编码/压缩标准对已编码的视频数据(304)、视频数据(307)和视频数据(309)(例如视频码流)进行编码。该些标准的实施例包括ITU-T H.265。在实施例中,正在开发的视频编码标准非正式地称为下一代视频编码(Versatile Video Coding,VVC),本申请可用于VVC标准的上下文中。

[0071] 应注意,电子装置(320)和电子装置(330)可包括其它组件(未示出)。举例来说,电子装置(320)可包括视频解码器(未示出),且电子装置(330)还可包括视频编码器(未示出)。

[0072] 图4是根据本申请公开的实施例的视频解码器(410)的框图。视频解码器(410)可设置在电子装置(430)中。电子装置(430)可包括接收器(431)(例如接收电路)。视频解码器(410)可用于代替图3实施例中的视频解码器(310)。

[0073] 接收器(431)可接收将由视频解码器(410)解码的至少一个已编码视频序列;在同一实施例或另一实施例中,一次接收一个已编码视频序列,其中每个已编码视频序列的解码独立于其它已编码视频序列。可从信道(401)接收已编码视频序列,所述信道可以是通向存储已编码的视频数据的存储装置的硬件/软件链路。接收器(431)可接收已编码的视频数据以及其它数据,例如,可转发到它们各自的使用实体(未标示)的已编码音频数据和/或辅助数据流。接收器(431)可将已编码视频序列与其它数据分开。为了防止网络抖动,缓冲存储器(415)可耦接在接收器(431)与熵解码器/解析器(420)(此后称为“解析器(420)”)之间。在某些应用中,缓冲存储器(415)是视频解码器(410)的一部分。在其它情况下,所述缓冲存储器(415)可设置在视频解码器(410)外部(未标示)。而在其它情况下,视频解码器(410)的外部设置缓冲存储器(未标示)以例如防止网络抖动,且在视频解码器(410)的内部可配置另一缓冲存储器(415)以例如处理播出定时。而当接收器(431)从具有足够带宽和可控性的存储/转发装置或从等时同步网络接收数据时,也可能不需要配置缓冲存储器(415),或可以将所述缓冲存储器做得较小。当然,为了在互联网等业务分组网络上使用,也

可能需要缓冲存储器(415),所述缓冲存储器可相对较大且可具有自适应性大小,且可至少部分地实施于操作系统或视频解码器(410)外部的类似元件(未标示)中。

[0074] 视频解码器(410)可包括解析器(420)以根据已编码视频序列重建符号(421)。这些符号的类别包括用于管理视频解码器(410)的操作的信息,以及用以控制显示装置(412)(例如,显示屏)等显示装置的潜在信息,所述显示装置不是电子装置(430)的组成部分,但可耦接到电子装置(430),如图4中所示。用于显示装置的控制信息可以是辅助增强信息(Supplemental Enhancement Information,SEI消息)或视频可用性信息(Video Usability Information,VUI)的参数集片段(未标示)。解析器(420)可对接收到的已编码视频序列进行解析/熵解码。已编码视频序列的编码可根据视频编码技术或标准进行,且可遵循各种原理,包括可变长度编码、霍夫曼编码(Huffman coding)、具有或不具有上下文灵敏度的算术编码等等。解析器(420)可基于对应于群组的至少一个参数,从已编码视频序列提取用于视频解码器中的像素的子群中的至少一个子群的子群参数集。子群可包括图片群组(Group of Pictures,GOP)、图片、图块、切片、宏块、编码单元(Coding Unit,CU)、块、变换单元(Transform Unit,TU)、预测单元(Prediction Unit,PU)等等。解析器(420)还可从已编码视频序列提取信息,例如变换系数、量化器参数值、运动矢量等等。

[0075] 解析器(420)可对从缓冲存储器(415)接收的视频序列执行熵解码/解析操作,从而创建符号(421)。

[0076] 取决于已编码视频图片或一部分已编码视频图片(例如:帧间图片和帧内图片、帧间块和帧内块)的类型以及其它因素,符号(421)的重建可涉及多个不同单元。涉及哪些单元以及涉及方式可由解析器(420)从已编码视频序列解析的子群控制信息控制。为了简洁起见,未描述解析器(420)与下文的多个单元之间的此类子群控制信息流。

[0077] 除已经提及的功能块以外,视频解码器(410)可在概念上细分成如下文所描述的数个功能单元。在商业约束下运行的实际实施例中,这些单元中的许多单元彼此紧密交互并且可以彼此集成。然而,出于描述所公开主题的目的,概念上细分成下文的功能单元是适当的。

[0078] 第一单元是缩放器/逆变换单元(451)。缩放器/逆变换单元(451)从解析器(420)接收作为符号(421)的量化变换系数以及控制信息,包括使用哪种变换方式、块大小、量化因子、量化缩放矩阵等。缩放器/逆变换单元(451)可输出包括样本值的块,所述样本值可输入到聚合器(455)中。

[0079] 在一些情况下,缩放器/逆变换单元(451)的输出样本可属于帧内编码块;即:不使用来自先前重建的图片的预测性信息,但可使用来自当前图片的先前重建部分的预测性信息的块。此类预测性信息可由帧内图片预测单元(452)提供。在一些情况下,帧内图片预测单元(452)采用从当前图片缓冲器(458)提取的已重建信息生成大小和形状与正在重建的块相同的周围块。举例来说,当前图片缓冲器(458)缓冲部分重建的当前图片和/或完全重建的当前图片。在一些情况下,聚合器(455)基于每个样本,将帧内预测单元(452)生成的预测信息添加到由缩放器/逆变换单元(451)提供的输出样本信息中。

[0080] 在其它情况下,缩放器/逆变换单元(451)的输出样本可属于帧间编码和潜在运动补偿块。在此情况下,运动补偿预测单元(453)可访问参考图片存储器(457)以提取用于预测的样本。在根据符号(421)对提取的样本进行运动补偿之后,这些样本可由聚合器(455)

添加到缩放器/逆变换单元(451)的输出(在这种情况下被称作残差样本或残差信号),从而生成输出样本信息。运动补偿预测单元(453)从参考图片存储器(457)内的地址获取预测样本可受到运动矢量控制,且所述运动矢量以所述符号(421)的形式而供运动补偿预测单元(453)使用,所述符号(421)例如是包括X、Y和参考图片分量。运动补偿还可包括在使用子样本精确运动矢量时,从参考图片存储器(457)提取的样本值的内插、运动矢量预测机制等等。

[0081] 聚合器(455)的输出样本可在环路滤波器单元(456)中被各种环路滤波技术采用。视频压缩技术可包括环路内滤波器技术,所述环路内滤波器技术受控于包括在已编码视频序列(也称作已编码视频码流)中的参数,且所述参数作为来自解析器(420)的符号(421)可用于环路滤波器单元(456)。然而,在其他实施例中,视频压缩技术还可响应于在解码已编码图片或已编码视频序列的先前(按解码次序)部分期间获得的元信息,以及响应于先前重建且经过环路滤波的样本值。

[0082] 环路滤波器单元(456)的输出可以是样本流,所述样本流可输出到显示装置(412)以及存储在参考图片存储器(457),以用于后续的帧间图片预测。

[0083] 一旦完全重建,某些已编码图片就可用作参考图片以用于将来预测。举例来说,一旦对应于当前图片的已编码图片被完全重建,且已编码图片(通过例如解析器(420))被识别为参考图片,则当前图片缓冲器(458)可变为参考图片存储器(457)的一部分,且可在开始重建后续已编码图片之前重新分配新的当前图片缓冲器。

[0084] 视频解码器(410)可根据例如ITU-T H.265标准中的预定视频压缩技术执行解码操作。在已编码视频序列遵循视频压缩技术或标准的语法以及视频压缩技术或标准中记录的配置文件的意义上,已编码视频序列可符合所使用的视频压缩技术或标准指定的语法。具体地说,配置文件可从视频压缩技术或标准中可用的所有工具中选择某些工具作为在所述配置文件下可供使用的仅有工具。对于合规性,还要求已编码视频序列的复杂度处于视频压缩技术或标准的层级所限定的范围内。在一些情况下,层级限制最大图片大小、最大帧率、最大重建取样率(以例如每秒兆(mega)个样本为单位进行测量)、最大参考图片大小等。在一些情况下,由层级设定的限制可通过假想参考解码器(Hypothetical Reference Decoder,HRD)规范和在已编码视频序列中用信号表示的HRD缓冲器管理的元数据来进一步限定。

[0085] 在实施例中,接收器(431)可连同已编码视频一起接收附加(冗余)数据。所述附加数据可以是已编码视频序列的一部分。所述附加数据可由视频解码器(410)用以对数据进行适当解码和/或较准确地重建原始视频数据。附加数据可呈例如时间、空间或信噪比(signal noise ratio,SNR)增强层、冗余切片、冗余图片、前向纠错码等形式。

[0086] 图5是根据本申请公开的实施例的视频编码器(503)的框图。视频编码器(503)设置于电子装置(520)中。电子装置(520)包括传输器(540)(例如传输电路)。视频编码器(503)可用于代替图3实施例中的视频编码器(303)。

[0087] 视频编码器(503)可从视频源(501)(并非图5实施例中的电子装置(520)的一部分)接收视频样本,所述视频源可采集将由视频编码器(503)编码的视频图像。在另一实施例中,视频源(501)是电子装置(520)的一部分。

[0088] 视频源(501)可提供将由视频编码器(503)编码的呈数字视频样本流形式的源视

频序列,所述数字视频样本流可具有任何合适位深度(例如:8位、10位、12位……)、任何色彩空间(例如BT.601Y CrCB、RGB……)和任何合适取样结构(例如Y CrCb 4:2:0、Y CrCb 4:4:4)。在媒体服务系统中,视频源(501)可以是存储先前已准备的视频的存储装置。在视频会议系统中,视频源(501)可以是采集本地图像信息作为视频序列的相机。可将视频数据提供为多个单独的图片,当按顺序观看时,这些图片被赋予运动。图片自身可构建为空间像素阵列,其中取决于所用的取样结构、色彩空间等,每个像素可包括至少一个样本。所属领域的技术人员可以很容易理解像素与样本之间的关系。下文侧重于描述样本。

[0089] 根据实施例,视频编码器(503)可实时或在由应用所要求的任何其它时间约束下,将源视频序列的图片编码且压缩成已编码视频序列(543)。施行适当的编码速度是控制器(550)的一个功能。在一些实施例中,控制器(550)控制如下文所描述的其它功能单元且在功能上耦接到这些单元。为了简洁起见,图中未标示耦接。由控制器(550)设置的参数可包括速率控制相关参数(图片跳过、量化器、率失真优化技术的λ值等)、图片大小、图片群组(group of pictures,GOP)布局,最大运动矢量允许参考区域等。控制器(550)可用于具有其它合适的功能,这些功能涉及针对某一系统设计优化的视频编码器(503)。

[0090] 在一些实施例中,视频编码器(503)在编码环路中进行操作。作为简单的描述,在实施例中,编码环路可包括源编码器(530)(例如,负责基于待编码的输入图片和参考图片创建符号,例如符号流)和嵌入于视频编码器(503)中的(本地)解码器(533)。解码器(533)以类似于(远程)解码器创建样本数据的方式重建符号以创建样本数据(因为在本申请所考虑的视频压缩技术中,符号与已编码视频码流之间的任何压缩是无损的)。将重建的样本流(样本数据)输入到参考图片存储器(534)。由于符号流的解码产生与解码器位置(本地或远程)无关的位精确结果,因此参考图片存储器(534)中的内容在本地编码器与远程编码器之间也是按比特位精确对应的。换句话说,编码器的预测部分“看到”的参考图片样本与解码器将在解码期间使用预测时所“看到”的样本值完全相同。这种参考图片同步性基本原理(以及在例如因信道误差而无法维持同步性的情况下产生的漂移)也用于一些相关技术。

[0091] “本地”解码器(533)的操作可与例如已在上文结合图4详细描述视频解码器(410)的“远程”解码器相同。然而,另外简要参考图4,当符号可用且熵编码器(545)和解析器(420)能够无损地将符号编码/解码为已编码视频序列时,包括缓冲存储器(415)和解析器(420)在内的视频解码器(410)的熵解码部分,可能无法完全在本地解码器(533)中实施。

[0092] 此时可以观察到,除存在于解码器中的解析/熵解码之外的任何解码器技术,也必定以基本上相同的功能形式存在于对应的编码器中。出于此原因,本申请侧重于解码器操作。可简化编码器技术的描述,因为编码器技术与全面地描述的解码器技术互逆。仅在某些区域中需要更详细的描述,并且在下文提供。

[0093] 在操作期间,在一些实施例中,源编码器(530)可执行运动补偿预测编码。参考来自视频序列中被指定为“参考图片”的至少一个先前已编码图片,所述运动补偿预测编码对输入图片进行预测性编码。以此方式,编码引擎(532)对输入图片的像素块与参考图片的像素块之间的差异进行编码,所述参考图片可被选作所述输入图片的预测参考。

[0094] 本地视频解码器(533)可基于源编码器(530)创建的符号,对可指定为参考图片的图片的已编码视频数据进行解码。编码引擎(532)的操作可为有损过程。当已编码视频数据可在视频解码器(图5中未示)处被解码时,重建的视频序列通常可以是带有一些误差的源

视频序列的副本。本地视频解码器(533)复制解码过程,所述解码过程可由视频解码器对参考图片执行,且可使重建的参考图片存储在参考图片高速缓存(534)中。以此方式,视频编码器(503)可在本地存储重建的参考图片的副本,所述副本与将由远端视频解码器获得的重建参考图片具有共同内容(不存在传输误差)。

[0095] 预测器(535)可针对编码引擎(532)执行预测搜索。即,对于将要编码的新图片,预测器(535)可在参考图片存储器(534)中搜索可作为所述新图片的适当预测参考的样本数据(作为候选参考像素块)或某些元数据,例如参考图片运动矢量、块形状等。预测器(535)可基于样本块逐像素块操作,以找到合适的预测参考。在一些情况下,根据预测器(535)获得的搜索结果,可确定输入图片可具有从参考图片存储器(534)中存储的多个参考图片取得的预测参考。

[0096] 控制器(550)可管理源编码器(530)的编码操作,包括例如设置用于对视频数据进行编码的参数和子群参数。

[0097] 可在熵编码器(545)中对所有上述功能单元的输出进行熵编码。熵编码器(545)根据例如霍夫曼编码、可变长度编码、算术编码等技术对各种功能单元生成的符号进行无损压缩,从而将所述符号转换成已编码视频序列。

[0098] 传输器(540)可缓冲由熵编码器(545)创建的已编码视频序列,从而为通过通信信道(560)进行传输做准备,所述通信信道可以是通向将存储已编码的视频数据的存储装置的硬件/软件链路。传输器(540)可将来自视频编码器(503)的已编码视频数据与要传输的其它数据合并,所述其它数据例如是已编码音频数据和/或辅助数据流(未示出来源)。

[0099] 控制器(550)可管理视频编码器(503)的操作。在编码期间,控制器(550)可以为每个已编码图片分配某一已编码图片类型,但这可能影响可应用于相应的图片的编码技术。例如,通常可将图片分配为以下任一种图片类型:

[0100] 帧内图片(I图片),其可以是不将序列中的任何其它图片用作预测源就可被编码和解码的图片。一些视频编解码器容许不同类型的帧内图片,包括例如独立解码器刷新(Independent Decoder Refresh,“IDR”)图片。所属领域的技术人员了解I图片的变体及其相应的应用和特征。

[0101] 预测性图片(P图片),其可以是可使用帧内预测或帧间预测进行编码和解码的图片,所述帧内预测或帧间预测使用至多一个运动矢量和参考索引来预测每个块的样本值。

[0102] 双向预测性图片(B图片),其可以是可使用帧内预测或帧间预测进行编码和解码的图片,所述帧内预测或帧间预测使用至多两个运动矢量和参考索引来预测每个块的样本值。类似地,多个预测性图片可使用多于两个参考图片和相关联元数据以用于重建单个块。

[0103] 源图片通常可在空间上细分成多个样本块(例如,4×4、8×8、4×8或16×16个样本的块),且逐块进行编码。这些块可参考其它(已编码)块进行预测编码,根据应用于块的相应图片的编码分配来确定所述其它块。举例来说,I图片的块可进行非预测编码,或所述块可参考同一图片的已经编码的块来进行预测编码(空间预测或帧内预测)。P图片的像素块可参考一个先前编码的参考图片通过空间预测或通过时域预测进行预测编码。B图片的块可参考一个或两个先前编码的参考图片通过空间预测或通过时域预测进行预测编码。

[0104] 视频编码器(503)可根据例如ITU-T H.265建议书的预定视频编码技术或标准执行编码操作。在操作中,视频编码器(503)可执行各种压缩操作,包括利用输入视频序列中

的时间和空间冗余的预测编码操作。因此，已编码视频数据可符合所用视频编码技术或标准指定的语法。

[0105] 在实施例中，传输器(540)可在传输已编码的视频时传输附加数据。源编码器(530)可将此类数据作为已编码视频序列的一部分。附加数据可包括时间/空间/SNR增强层、冗余图片和切片等其它形式的冗余数据、SEI消息、VUI参数集片段等。

[0106] 采集到的视频可作为呈时间序列的多个源图片(视频图片)。帧内图片预测(常常简化为帧内预测)利用给定图片中的空间相关性，而帧间图片预测则利用图片之间的(时间或其它)相关性。在实施例中，将正在编码/解码的特定图片分割成块，正在编码/解码的特定图片被称作当前图片。在当前图片中的块类似于视频中先前已编码且仍被缓冲的参考图片中的参考块时，可通过称作运动矢量的矢量对当前图片中的块进行编码。所述运动矢量指向参考图片中的参考块，且在使用多个参考图片的情况下，所述运动矢量可具有识别参考图片的第三维度。

[0107] 在一些实施例中，双向预测技术可用于帧间图片预测中。根据双向预测技术，使用两个参考图片，例如按解码次序都在视频中的当前图片之前(但按显示次序可能分别是过去和将来)第一参考图片和第二参考图片。可通过指向第一参考图片中的第一参考块的第一运动矢量和指向第二参考图片中的第二参考块的第二运动矢量对当前图片中的块进行编码。具体来说，可通过第一参考块和第二参考块的组合来预测所述块。

[0108] 此外，合并模式技术可用于帧间图片预测中以改善编码效率。

[0109] 根据本申请公开的一些实施例，帧间图片预测和帧内图片预测等预测的执行以块为单位。举例来说，根据HEVC标准，将视频图片序列中的图片分割成编码树单元(coding tree unit，CTU)以用于压缩，图片中的CTU具有相同大小，例如64×64像素、32×32像素或16×16像素。一般来说，CTU包括三个编码树块(coding tree block，CTB)，所述三个编码树块是一个亮度CTB和两个色度CTB。更进一步的，还可将每个CTU以四叉树拆分为至少一个编码单元(coding unit，CU)。举例来说，可将64×64像素的CTU拆分为一个64×64像素的CU，或4个32×32像素的CU，或16个16×16像素的CU。在实施例中，分析每个CU以确定用于CU的预测类型，例如帧间预测类型或帧内预测类型。此外，取决于时间和/或空间可预测性，将CU拆分为至少一个预测单元(prediction unit，PU)。通常，每个PU包括亮度预测块(prediction block，PB)和两个色度PB。在实施例中，编码(编码/解码)中的预测操作以预测块为单位来执行。以亮度预测块作为预测块为例，预测块包括像素值(例如，亮度值)的矩阵，例如8×8像素、16×16像素、8×16像素、16×8像素等等。

[0110] 图6是根据本申请公开的另一实施例的视频编码器(603)的图。视频编码器(603)用于接收视频图片序列中的当前视频图片内的样本值的处理块(例如预测块)，且将所述处理块编码到作为已编码视频序列的一部分的已编码图片中。在本实施例中，视频编码器(603)用于代替图3实施例中的视频编码器(303)。

[0111] 在HEVC实施例中，视频编码器(603)接收用于处理块的样本值的矩阵，所述处理块为例如8×8样本的预测块等。视频编码器(603)使用例如率失真(rate-distortion，RD)优化来确定是否使用帧内模式、帧间模式或双向预测模式来编码所述处理块。当在帧内模式中编码处理块时，视频编码器(603)可使用帧内预测技术以将处理块编码到已编码图片中；且当在帧间模式或双向预测模式中编码处理块时，视频编码器(603)可分别使用帧间预测

或双向预测技术将处理块编码到已编码图片中。在某些视频编码技术中，合并模式可以是帧间图片预测子模式，其中，在不借助预测值外部的已编码运动矢量分量的情况下，从至少一个运动矢量预测值导出运动矢量。在某些其它视频编码技术中，可存在适用于主题块的运动矢量分量。在实施例中，视频编码器(603)包括其它组件，例如用于确定处理块模式的模式决策模块(未示出)。

[0112] 在图6的实施例中，视频编码器(603)包括如图6所示的耦接到一起的帧间编码器(630)、帧内编码器(622)、残差计算器(623)、开关(626)、残差编码器(624)、通用控制器(621)和熵编码器(625)。

[0113] 帧间编码器(630)用于接收当前块(例如处理块)的样本、比较所述块与参考图片中的至少一个参考块(例如先前图片和后来图片中的块)、生成帧间预测信息(例如根据帧间编码技术的冗余信息描述、运动矢量、合并模式信息)、以及基于帧间预测信息使用任何合适的技术计算帧间预测结果(例如已预测块)。在一些实施例中，参考图片是基于已编码的视频信息解码的已解码参考图片。

[0114] 帧内编码器(622)用于接收当前块(例如处理块)的样本、在一些情况下比较所述块与同一图片中已编码的块、在变换之后生成量化系数、以及在一些情况下还(例如根据至少一个帧内编码技术的帧内预测方向信息)生成帧内预测信息。在实施例中，帧内编码器(622)还基于帧内预测信息和同一图片中的参考块计算帧内预测结果(例如已预测块)。

[0115] 通用控制器(621)用于确定通用控制数据，且基于所述通用控制数据控制视频编码器(603)的其它组件。在实施例中，通用控制器(621)确定块的模式，且基于所述模式将控制信号提供到开关(626)。举例来说，当所述模式是帧内模式时，通用控制器(621)控制开关(626)以选择供残差计算器(623)使用的帧内模式结果，且控制熵编码器(625)以选择帧内预测信息且将所述帧内预测信息添加在码流中；以及当所述模式是帧间模式时，通用控制器(621)控制开关(626)以选择供残差计算器(623)使用的帧间预测结果，且控制熵编码器(625)以选择帧间预测信息且将所述帧间预测信息添加在码流中。

[0116] 残差计算器(623)用于计算所接收的块与选自帧内编码器(622)或帧间编码器(630)的预测结果之间的差(残差数据)。残差编码器(624)用于基于残差数据操作，以对残差数据进行编码以生成变换系数。在实施例中，残差编码器(624)用于将残差数据从空间域转换到频域，且生成变换系数。变换系数接着经由量化处理以获得量化的变换系数。在各种实施例中，视频编码器(603)还包括残差解码器(628)。残差解码器(628)用于执行逆变换，且生成已解码残差数据。已解码残差数据可适当地由帧内编码器(622)和帧间编码器(630)使用。举例来说，帧间编码器(630)可基于已解码残差数据和帧间预测信息生成已解码块，且帧内编码器(622)可基于已解码残差数据和帧内预测信息生成已解码块。适当处理已解码块以生成已解码图片，且在一些实施例中，所述已解码图片可在存储器电路(未示出)中缓冲并用作参考图片。

[0117] 熵编码器(625)用于将码流格式化以产生已编码的块。熵编码器(625)根据HEVC标准等合适标准产生各种信息。在实施例中，熵编码器(625)用于获得通用控制数据、所选预测信息(例如帧内预测信息或帧间预测信息)、残差信息和码流中的其它合适的信息。应注意，根据所公开的主题，当在帧间模式或双向预测模式的合并子模式中对块进行编码时，不存在残差信息。

[0118] 图7是根据本申请公开的另一实施例的视频解码器(710)的图。视频解码器(710)用于接收作为已编码视频序列的一部分的已编码图像,且对所述已编码图像进行解码以生成重建的图片。在实施例中,视频解码器(710)用于代替图3实施例中的视频解码器(310)。

[0119] 在图7实施例中,视频解码器(710)包括如图7中所示耦接到一起的熵解码器(771)、帧间解码器(780)、残差解码器(773)、重建模块(774)和帧内解码器(772)。

[0120] 熵解码器(771)可用于根据已编码图片来重建某些符号,这些符号表示构成所述已编码图片的语法元素。此类符号可包括例如用于对所述块进行编码的模式(例如帧内模式、帧间模式、双向预测模式、后两者的合并子模式或另一子模式)、可分别识别供帧内解码器(772)或帧间解码器(780)用以进行预测的某些样本或元数据的预测信息(例如帧内预测信息或帧间预测信息)、呈例如量化的变换系数形式的残差信息等等。在实施例中,当预测模式是帧间或双向预测模式时,将帧间预测信息提供到帧间解码器(780);以及当预测类型是帧内预测类型时,将帧内预测信息提供到帧内解码器(772)。残差信息可经由逆量化并提供到残差解码器(773)。

[0121] 帧间解码器(780)用于接收帧间预测信息,且基于所述帧间预测信息生成帧间预测结果。

[0122] 帧内解码器(772)用于接收帧内预测信息,且基于所述帧内预测信息生成预测结果。

[0123] 残差解码器(773)用于执行逆量化以提取解量化的变换系数,且处理所述解量化的变换系数,以将残差从频域转换到空间域。残差解码器(773)还可能需要某些控制信息(用以获得量化器参数QP),且所述信息可由熵解码器(771)提供(未标示数据路径,因为这仅仅是低量控制信息)。

[0124] 重建模块(774)用于在空间域中组合由残差解码器(773)输出的残差与预测结果(可由帧间预测模块或帧内预测模块输出)以形成重建的块,所述重建的块可以是重建的图片的一部分,所述重建的图片继而可以是重建的视频的一部分。应注意,可执行解块操作等其它合适的操作来改善视觉质量。

[0125] 应注意,可使用任何合适的技术来实施视频编码器(303)、视频编码器(503)和视频编码器(603)以及视频解码器(310)、视频解码器(410)和视频解码器(710)。在实施例中,可使用至少一个集成电路来实施视频编码器(303)、视频编码器(503)和视频编码器(603)以及视频解码器(310)、视频解码器(410)和视频解码器(710)。在另一实施例中,可使用执行软件指令的至少一个处理器来实施视频编码器(303)、视频编码器(503)和视频编码器(603)以及视频解码器(310)、视频解码器(410)和视频解码器(710)。

[0126] II.HEVC块分区结构

[0127] 在HEVC中,通过使用表示为编码树的四叉树结构,将编码树单元(Coding Tree Unit,CTU)分割成CU,以适应各种局部特性。在CU级作出是否使用图片间(时间)或图片内(空间)预测来对图片区域进行编码的决策。根据预测单元(Prediction Unit,PU)分割类型,可以将每个CU进一步分割成一个、两个或四个PU。在一个PU内,应用相同的预测过程,并且基于PU将相关信息传输到解码器。在通过应用基于PU分割类型的预测过程获得残差块之后,可以根据另一QT结构(如CU的编码树)将CU分区成变换单元(Transform Unit,TU)。

[0128] HEVC结构的一个关键特征是它具有包括CU、PU和TU的多个分区概念。在HEVC中,CU

或TU只能是正方形，而对于帧间预测块，PU可以是正方形或矩形形状。对于所提出的用于帧内预测和变换的矩形PU，可以扩展以用于联合开发模型(Joint Exploration Model，JEM)。

[0129] 在图片边界处，HEVC施加了一个隐式的四叉树分割，使得块可以保持四叉树分割，直到大小适合图片边界。

[0130] III.使用QTBT的块分区结构

[0131] 在VVC测试模型(VVC test model，VTM)中，应用四叉树(quad-tree，QT)加二叉树(binary-tree，BT)加三叉树(ternary tree，TT)分区结构。四叉树加二叉树(Quad-Tree plus Binary Tree，QTBT)结构去除了多个分区类型的概念，即，去除了CU、PU和TU概念，并且支持更多灵活的CU分区形状。

[0132] 在QTBT结构中，CU可以具有正方形或矩形形状。如图8A和图8B所示，首先通过QT结构对CTU进行分区。通过BT结构，对QT叶节点进行进一步分区。在BT分割中，可以有两种分割类型，即对称水平分割和对称垂直分割。BT叶节点为CU，并且两个CU之间的分割用于预测和变换处理而无需进一步分区。因此，在QTBT结构中，CU、PU和TU可以具有相同的块大小。

[0133] 在JEM中，CU有时可以包括不同颜色分量的编码块(Coding Block，CB)，例如，在4:2:0色度格式的P分片和B分片的情况下，一个CU可以包含一个亮度CB和两个色度CB。或者，CU可以包括单个分量的CB，例如，在I分片的情况下，一个CU可仅包含一个亮度CB或仅包含两个色度CB。

[0134] 为QTBT分区方案定义了以下参数：

[0135] -编码树单元尺寸(CTU size)：例如在HEVC中，QT的根节点大小，

[0136] -最小四叉树尺寸(MinQTSize)：允许的最小QT叶节点大小

[0137] -最大二叉树尺寸(MaxBTSize)：允许的最大BT根节点大小

[0138] -最大二叉树深度(MaxBTDepth)：允许的最大BT深度

[0139] -最小二叉树尺寸(MinQTSize)：允许的最小BT叶节点大小

[0140] 在QTBT分区结构的一个示例中，将CTU size设定为具有两个对应64×64色度样本块的128×128亮度样本，将MinQTSize设定为16×16，将MaxBTSize设定为64×64，将MinBTSize(针对宽度和高度)设定为4×4，并且将MaxBTDepth设定为4。首先将QT分区应用于CTU，以产生QT叶节点。QT叶节点可以具有从16×16(即，MinQTSize)到128×128(即，CTU大小)的大小。如果QT叶节点的尺寸是128×128，则该QT叶节点不会被BT进一步分割，因为其大小超过MaxBTSize(即64×64)。否则，可以由BT树进一步对QT叶节点分区。因此，QT叶节点也是BT的根节点，并且，QT叶节点的BT深度为0。当BT深度达到MaxBTDepth(即，4)时，不考虑进一步的分割。当BT节点具有等于MinBTSize(即，4)的宽度时，不考虑进一步的水平分割。类似地，当BT节点的高度等于MinBTSize时，不考虑进一步的垂直分割。通过预测和变换处理进一步处理BT的叶节点，而无需任何进一步的分区。在JEM中，最大CTU大小是256×256亮度样本。

[0141] 图8A示出了使用QTBT分区结构(8200)的块分区(8100)的示例，并且图8B示出了对应的QTBT结构(8200)。实线指示QT分割，并且虚线指示二叉树BT分割。在每个非叶BT分割节点中，用信号通知标志以指示分割类型(即，对称水平分割或对称垂直分割)。例如，在图8B示例中，“0”指示对称水平分割，并且“1”指示对称垂直分割。然而，对于QT分割，不指示或用信号通知分割类型标志，因为QT分割水平地和垂直地分割非叶节点，以产生尺寸相同的四

个小块。

[0142] 参考图8B,在QTBT结构(8200)中,首先通过QT结构,将根节点(8201)分区成QT节点(8211)-(8214)。因此,如图8A所示,通过实线,将编码树块(8101)分区成四个尺寸相同的块(8111)-(8114)。

[0143] 返回参考图8B,分别通过两个BT分割,对QT节点(8211)和(8212)进一步分割。如上所述,BT分割包括两个分割类型,即,对称水平分割和对称垂直分割。将QT非叶节点(8211)指示为"1",并且因此可以使用对称垂直分割,将QT非叶节点(8211)分割成两个节点(8221)和(8222)。将QT非叶节点(8212)指示为"0",并且可以使用对称水平分割,将QT非叶节点(8212)分割成两个节点(8223)和(8224)。使用另一QTBT结构,将QT非叶节点(8213)进一步分割成四个节点(8225)-(8228)。不对节点(8214)进一步分割,因此,节点(8214)是叶节点。因此,如图8A所示,将块(8111)垂直地分区,形成两个相等大小的块(8121)和(8122);将块(8112)水平地分区,形成两个相等大小的块;对块(8113)分区,形成四个相等大小的块,并且,不对块(8114)进一步分区。

[0144] 返回参考图8B,在更深的层次中,进一步分割一些节点(例如,节点(8221)-(8228)),而不分割其它节点。例如,通过对称垂直分割,将BT非叶节点(8221)进一步分割成两个叶节点(8231)和(8232),而不对叶节点(8222)进一步分割。因此,如图8A所示,将块(8121)分区,形成两个相等大小的块(8131)和(8132),而不对块(8122)进一步分区。

[0145] 在完成QTBT结构(8200)的分割之后,未被进一步分割的叶节点是用于预测和变换处理的CU。因此,在QTBT结构中,CU、与CU相关联的PU以及与CU相关联的TU可具有相同的块大小。另外,在QTBT结构中,CU可包括不同颜色分量的CB。例如,在4:2:0格式下,在P分片或B分片中,一个CU可包括一个亮度CB和两个色度CB。然而,在一些其它实施例中,CU可包括单分量的CB。例如,在I分片中,一个CU可包括一个亮度CB或两个色度CB。也就是说,QTBT结构支持亮度和色度具有不同分区结构的能力。

[0146] 此外,QTBT方案支持亮度和色度具有单独的QTBT结构的灵活性。当前,对于P分片和B分片,一个CTU中的亮度CTB和色度CTB共享相同的QTBT结构。然而,对于I分片,通过QTBT结构将亮度CTB分区成CU,并且通过另一QTBT结构,将色度CTB分区成色度CU。因此,I分片中的CU包括亮度分量的编码块、或两个色度分量的编码块,且P分片或B分片中的CU包括所有三个颜色分量的编码块。

[0147] 在HEVC中,限制用于小块的帧间预测,以减少运动补偿的存储器访问,使得4×8块和8×4块不支持双向预测,并且4×4块不支持帧间预测。在JEM-7.0实施的QTBT中,去除这些限制。

[0148] 例如,在VTM6中,编码树方案支持亮度和色度具有分离的块树结构的能力。例如,对于P分片和B分片,一个CTU中的亮度CTB和色度CTB必须共享相同的编码树结构。然而,对于I分片,亮度和色度可以具有分离的块树结构。当应用分离的块树模式时,通过一个编码树结构,将亮度CTB分区成CU,并且,通过另一编码树结构,将色度CTB分区成色度CU。这意味着I分片中的CU可包括亮度分量的编码块、或两个色度分量的编码块,并且P分片或B分片中的CU总是包括所有三个颜色分量的编码块,除非视频是单色的。

[0149] IV.使用三叉树(Triple-Trees,TT)的块分区结构

[0150] 除了上述QTBT结构之外,称为多类型树(Multi-Type-Tree,MTT)结构的另一分割

结构可以比QTBT结构更灵活。在MTT结构中，除了QT和BT以外，引入了水平和垂直中心侧TT，如图9A和图9B所示。

[0151] 图9A示出了垂直中心侧TT分区的示例。例如，将块(910)垂直地分割成三个子块(911)-(913)，其中子块(912)位于块(910)的中间。

[0152] 图9B示出了水平中心侧TT分区的示例。例如，将块(920)水平地分割成三个子块(921)-(923)，其中子块(922)位于块(920)的中间。

[0153] 类似于BT分割，在TT分割中，用信号通知标志，以指示分割类型(即，对称水平分割或对称垂直分割)。在示例中，“0”指示对称水平分割，并且“1”指示对称垂直分割。

[0154] TT分区的一个益处在于，TT分区可以是QT分区和BT分区的补充。例如，TT分区能够捕获位于块中心的对象，而QT分区和BT分区总是沿着块中心进行分割。TT分区的另一益处在于，通过TT分区形成的分区的宽度和高度总是2的幂，因此不需要额外的变换。

[0155] 将包括四叉树、二叉树和三叉树分割类型的MTT结构称为QTBTTT结构。类似于QTBT结构，QTBTTT结构也支持具有不同结构的亮度和色度。例如，在I分片中，用于对亮度CTB进行分区的QTBTTT结构可不同于用于对色度CTB进行分区的QTBTTT结构。这意味着当启用分离的树结构时，CU包括一个亮度CB、或两个色度CB。然而，在P分片或B分片中，亮度CTB可以与一个CTU中的色度CTB共享相同的QTBTTT结构。这意味着当禁用分离的树结构时，CU包括所有三个CB(即，一个亮度CB和两个色度CB)。

[0156] 在VVC中，将双树或分离树用于I分片。即，一棵树用于亮度分量，并且另一棵树用于色度分量。对于B分片和P分片，亮度分量和色度分量共享一棵树。

[0157] 两级树的设计主要是为了降低复杂性。在示例中，遍历树的复杂性是T^D，其中，T表示分割类型的数量，并且，D表示树的深度。

[0158] V.亮度分量的帧内预测

[0159] 图10示出了在一些示例(例如，VVC)中的示例性帧内预测方向的图示。在图10中，总共有95个帧内预测模式(模式-14至模式80)，其中模式0是平面模式(称为帧内平面(INTRA_PLANAR))，模式1是DC模式(称为帧内DC(INTRA_DC))，并且其它模式(模式-14至模式-1和模式2至模式80)是角度(或方向)模式(称为帧内角度(INTRA_ANGULAR))。在角度(或方向)模式中，模式18(称为INTRA_ANGULAR18)是水平模式，模式50(称为INTRA_ANGULAR50)是垂直模式，并且模式2(称为INTRA_ANGULAR2)是指向左下方向的对角线模式，模式34(称为INTRA_ANGULAR34)是指向左上方向的对角线模式，并且模式66(称为INTRA_ANGULAR66)是指向右上方向的对角线模式。模式-14至模式-1和模式67至模式80称为宽角度帧内预测(Wide-Angle Intra Prediction,WAIP)模式。

[0160] VI.色度分量的帧内预测

[0161] 对于帧内PU的色度分量(例如，在VTM中)，编码器从8种模式中选择最佳色度预测模式，其中，该8种模式包括平面模式、DC模式、水平模式、垂直模式、来自亮度分量的帧内预测模式的直接副本(Direct copy of the intra prediction Mode,DM)、左和顶交叉分量线性模式(Left and Top Cross-Component Linear Mode,LT_CCLM)、左交叉分量线性模式(Left Cross-Component Linear Mode,L_CCLM)和顶交叉分量线性模式(Top Cross-Component Linear Mode,T_CCLM)。可将表1中分别由色度预测模式索引4、5和6表示的LT_CCLM、L_CCLM和T_CCLM分类成一组交叉分量线性模式(Cross-Component Linear Mode,

CCLM)。

[0162] 表1显示了在启用CCLM时从亮度模式到色度预测模式的示例性推导。平面模式、垂直模式、水平模式、DC模式、LT_CCLM、L_CCLM、T_CCLM、DM用色度预测模式索引从0到7表示。对于除DM之外的这些色度预测模式，对应的默认色度帧内预测方向为0、50、18、1、81、82和83。

[0163] 表1

[0164]

色度预测模式索引	对应的亮度帧内预测方向				
	0	50	18	1	X (0 <= X <= 66)
0	66	0	0	0	0
1	50	66	50	50	50
2	18	18	66	18	18
3	1	1	1	66	1
4	81	81	81	81	81
5	82	82	82	82	82
6	83	83	83	83	83
7	0	50	18	1	X

[0165] 当色度分量的色度预测模式索引号为表1中的7时，亮度分量的帧内预测方向可以用于生成色度分量的帧内预测样本。例如，对于色度预测模式索引7(即DM)，当对应的亮度帧内预测方向为50(即，垂直方向)时，色度帧内预测方向也为50，即，用于色度分量的垂直模式。

[0166] 当色度分量的色度预测模式索引号不是表1中的7时，色度分量的帧内预测方向可以取决于默认色度帧内预测方向是否与对应的亮度帧内预测相同。

[0167] 当默认色度帧内预测方向与对应的亮度帧内预测相同时，帧内预测方向66可用于生成色度分量的帧内预测样本。例如，对于色度预测模式索引1(即，垂直模式)，默认色度帧内预测方向为50，当对应的亮度帧内预测方向也是50(即，垂直方向)时，帧内预测方向66用于生成色度分量的帧内预测样本。

[0168] 当默认色度帧内预测方向与对应的亮度帧内预测方向不同时，默认的色度帧内预测方向可以用于生成色度分量的帧内预测样本。例如，对于色度预测模式索引1(即，垂直模式)，默认的色度帧内预测方向是50，并且当相应的亮度帧内预测方向不是50(例如，18)，默认的色度帧内预测50可以用于生成色度分量的帧内预测样本。

[0169] 表2示出了当启用CCLM时，帧内色度预测模式的示例性信令(例如，二值化)过程。结果表明，在所有色度帧内预测模式中，DM(即，模式7)具有最少的码字。

[0170] 表2

[0171]

帧内色度预测模式的值	二进制串 (Bin string)
7	0
4	10
5	1110
6	1111
0	11000
1	11001
2	11010
3	11011

[0172] VII.小块大小限制

[0173] 包括2x2样本、2x4样本和4x2样本在内的小块大小在双树中受到限制。此限制通过禁止以下分割来执行：不允许分割4x4块、2x8块和8x2块，不允许对8x4块和4x8块进行三元分割。

[0174] 在单树的情况下，当色度块的块大小(例如，区域大小或样本数目)大于或等于阈值，并且任何色度块分割可以导致色度块的子块大小小于阈值或者不允许对色度块进一步分割时，可将色度块视为可并行处理区域(PPR)。阈值可以是任何正整数，比如，16或32。在一个示例中，将阈值设置为等于16个样本，以实现最坏情况下的处理吞吐量为VTM的4倍。

[0175] 图11A至图11E示出了根据本公开实施例的PPR的示例性形状。每个块(或子块)中的标记值指示该块(或子块)中的样本数目。例如，图11A示出了将4x4色度块(16个样本)分割成四个2x2子块，并且每个子块具有4个样本。可以将该4x4色度块视为一个PPR。

[0176] 通过将PPR内的多个块(或子块)标记为“不可用”，以预测此PPR中包括的所有块(或子块)，可以消除16样本PPR中的块(或子块)之间的帧内预测依赖性。注意，当相邻块不可用时，可以通过VVC中的参考替换过程自动执行参考样本的替换。另外，该方法不需要对帧内预测的其它部分进行任何改变。

[0177] VIII.小块大小限制的改进

[0178] 本公开的实施例可以改进使用一个或多个小块大小。对于单树中的小色度块限制，其中，不同颜色分量共享相同的块分区结构，可能将一个PPR内的多个块(或子块)标记为不可用。例如，对于PPR中的一些小色度块，比如，分别在图11A、图11B和图11C中的右下块(1101)至(1103)，这些小色度块的左相邻样本和上相邻样本均标记为不可用。然而，在一些相关情况下，允许所有色度预测模式(比如，表1中列出的模式)用于预测PPR内的所有块(或子块)，从而导致一些冗余信令比特。

[0179] 根据本公开的各方面，该方法可应用于一个或多个小块。可以通过块的块区域大小或边长(例如，块高度或块宽度)识别小块。一个或多个小块的尺寸可以例如是，2的倍数。在一个示例中，一个或多个小块可以包括2×2块。在另一示例中，小块可以包括2x2块、2×4块和4×2块。

[0180] 一个或多个小块可以具有一个固定尺寸。例如，小块可以包括2×N和N×2块，其中N是正整数。即，小块的一侧限于有2个样本。在另一示例中，小块可以包括4×N和N×4色度块，其中N是正整数。即，小块的一侧限于有4个样本。

[0181] 一个或多个小块可以基于亮度帧内编码块，其中，亮度帧内编码块具有最小亮度帧内编码块大小。例如，块区域大小、块宽度和/或块高度小于具有最小亮度帧内编码块大小的帧内编码块。即，对于小块，小块的块大小小于最小亮度帧内编码块的块大小，和/或，小块的块宽度小于最小亮度帧内编码块的块宽度，和/或，块高度小于最小亮度帧内编码块的块高度。

[0182] 一个或多个小块可以基于阈值。例如，小块可以包括2xN或Nx2个像素，并且小块中的像素数目小于阈值ThrePixels，其中N是正整数。可以将阈值ThrePixels设置为预定值，比如，16或32。可在比特流中，比如，在序列参数集(Sequence Parameter Set，SPS)、图片参数集(Picture Parameter Set，PPS)或切片头中等，发信号通知ThrePixels。

[0183] 根据本公开的各方面，对于一个当前PPR中的帧内编码色度(或亮度)块(或子块)，如果其上相邻样本和/或左相邻样本未用于预测当前块(例如，将这些相邻块中的至少一个标记为“不可用”)，只有某些色度帧内预测模式可用于预测当前块，而不是可应用于其它块(例如，非小块)的所有色度帧内预测模式。

[0184] 例如，可以使用三种不同的分区模式分割4x4块，如图11A至图11C所示。对于每个分区模式，仅具有4个样本的右下2x2块的上相邻样本和左相邻样本可标记为“不可用”，并且不用于预测右下2x2块。

[0185] 根据本公开的各方面，可将块的允许色度预测模式约束为可用色度预测模式的子集，其中，可用色度预测模式的子集可包括，例如，表1中列出的一个或多个模式。可以基于块的顶和/或左相邻样本的可用性，约束允许的色度预测模式。

[0186] 在一个实施例中，对于一个当前PPR中的一个或任何小色度块(或子块)，如果小色度块(或子块)的上相邻样本和/或左相邻样本不可用于预测小色度块(或子块)，则可约束小色度块(或子块)的允许色度预测模式。允许的色度预测模式可限于表1中列出的模式的子集。例如，允许的色度预测模式可以仅限于一种模式(例如，DM)。当仅限于一种模式时，不需要用信号发送比特，以指示当前色度块的色度预测模式。

[0187] 在一个实施例中，对于一个当前PPR中的一个或任何小色度块(或子块)，如果小色度块(或子块)的上相邻样本和/或左相邻样本在当前PPR内，并且不可用于预测小色度块(或子块)，则可约束小色度块(或子块)的允许色度预测模式。例如，允许的色度预测模式可以仅限于一种模式，比如，表1中列出的多个模式中的一种模式(例如，DM)。当仅限于一种模式时，不需要用信号发送比特，以指示当前色度块的色度预测模式。

[0188] 在一个实施例中，对于一个当前PPR中的一个或任何小色度块，如果小色度块(或子块)的上和/或左相邻样本在当前图片(或CTU行或CTU，或切片，或图块，或图块组)之外，则可约束小色度块(或子块)的允许色度预测模式。例如，允许的色度预测模式可以仅限于一种模式，比如，表1中列出的多个模式中的一个模式(例如，DM)。当仅限于一种模式时，不需要用信号发送比特，以指示当前色度块的色度预测模式。

[0189] 根据本公开的各方面，对于一个当前PPR中的一个或任何小色度块，如果小色度块(或子块)的上相邻样本或左相邻样本不可用于预测小色度块(或子块)，则可约束小色度块

(或子块)的允许色度预测模式仅为两种模式,比如,DM和L_CCLM(或T_CCLM)。根据上述内容和相邻样本中的哪一个不可用,可以应用不同的约束。

[0190] 在一个实施例中,对于一个当前PPR中的一个或任何小色度块,如果小色度块(或子块)的上相邻样本在当前PPR内,并且不可用于预测小色度块(或子块),则可约束小色度块(或子块)的允许色度预测模式仅为DM和/或L_CCLM模式。

[0191] 在一个实施例中,对于一个当前PPR中的一个或任何小色度块,如果小色度块(或子块)的左相邻样本在当前PPR内,则可约束小色度块(或子块)的允许色度预测模式仅为DM和/或T_CCLM模式。

[0192] 根据本公开的各方面,对于一个当前PPR中的一个或任何小色度块,如果小色度块(或子块)的上相邻样本或左相邻样本在当前PPR内,并且不可用于预测小色度块(或子块),则可约束小色度块(或子块)的允许色度预测模式仅为两种模式,比如,DM和垂直(或水平)模式。根据以上内容和相邻样本中的哪一个不可用,可以应用不同的约束。

[0193] 在一个实施例中,对于一个当前PPR中的一个或任何小色度块,如果小色度块(或子块)的上相邻样本在当前PPR内并且不可用于预测小色度块(或子块),则可约束小色度块(或子块)的允许色度预测模式仅为DM和/或水平模式。

[0194] 在一个实施例中,对于一个当前PPR中的一个或任何小色度块,如果小色度块(或子块)的左相邻样本在当前PPR内,并且不可用于预测小色度块(或子块),则可约束小色度块(或子块)的允许色度预测模式仅为DM和/或垂直模式。

[0195] 根据本公开的各方面,本公开中呈现的上述方法可应用于单树和/或双树情况。即,无论色度样本和亮度样本是否使用相同的分区树结构,均可以使用这些方法。

[0196] IX.小块大小限制的改进流程图

[0197] 图12示出了根据本公开实施例的概述示例过程(1200)的流程图。在各种实施例中,过程(1200)由处理电路执行,比如,终端设备(210)、终端设备(220)、终端设备(230)和终端设备(240)中的处理电路、执行视频编码器(303)功能的处理电路、执行视频解码器(310)功能的处理电路、执行视频解码器(410)功能的处理电路、执行帧内预测模块(452)功能的处理电路、执行视频编码器(503)功能的处理电路、执行预测器(535)功能的处理电路、执行帧内编码器(622)功能的处理电路、执行帧内解码器(772)功能的处理电路等等。在一些实施例中,过程(1200)以软件指令实现,因此当处理电路执行软件指令时,处理电路执行过程(1200)。

[0198] 过程(1200)可通常在步骤(S1210)处开始。过程(1200)对当前图片中当前块的预测信息进行解码,其中,当前图片是已编码视频序列的一部分。预测信息可以指示当前块的色度样本的分区树结构。然后,过程(1200)进行到步骤(S1220)。

[0199] 在步骤(S1220)处,过程(1200)基于分区树结构,将当前块的色度样本分区成多个子块。然后,过程(1200)进行到步骤(S1230)。

[0200] 在步骤(S1230)处,当满足以下两个条件时,过程(1200)基于色度帧内预测模式的子集,预测多个子块中一个子块的色度样本:(i)该子块的上相邻样本和左相邻样本中的至少一个不可用于预测子块;以及(ii)子块的块大小小于或等于大小阈值,或子块的边长小于或等于长度阈值。然后,过程(1200)进行到步骤(S1240)。

[0201] 在步骤(S1240)处,过程(1200)基于预测的色度样本重建当前块。

[0202] 在重建当前块之后,过程(1200)终止。

[0203] 在实施例中,当前块的块大小大于大小阈值,并且子块的块大小小于或等于大小阈值。大小阈值可以包括2x2、2x4和4x2中的一个。大小阈值可以包括允许的最小亮度帧内编码块的块大小。

[0204] 在实施例中,当前块的边长大于长度阈值,并且子块的边长小于或等于长度阈值。长度阈值包括2和4中的一个。长度阈值可以包括允许的最小亮度帧内编码块的边长。

[0205] 在实施例中,色度帧内预测模式包括平面模式、DC模式、水平模式、垂直模式、导出模式(Derived Mode,DM)、左交叉分量线性模式(Left Cross-Component Linear Mode,L_CCLM)、顶交叉分量线性模式(Top Cross-Component Linear Mode,T_CCLM)以及左和顶交叉分量线性模式(Left and Top Cross-Component Linear Mode,LT_CCLM)。

[0206] 在实施例中,色度帧内预测模式的子集包括色度帧内预测模式中的一个或两个。例如,色度帧内预测模式的子集仅包括DM。

[0207] 在实施例中,当子块的上相邻样本不可用于预测子块并位于当前块内时,色度帧内预测模式的子集包括DM、L_CCLM和垂直模式中的至少一个。

[0208] 在实施例中,当子块的左相邻样本不可用于预测子块并位于当前块内时,色度帧内预测模式的子集包括DM、T_CCLM和水平模式中的至少一个。

[0209] 在实施例中,当前块是PPR,其中,PPR中的所有子块是并行重建。

[0210] X.帧内子分区(Intra Sub-Partition,ISP)编解码模式

[0211] 以帧内子分区(ISP)编解码模式编码的亮度编码块可被垂直地(在垂直ISP模式下)或水平地(在水平ISP模式下)分区成多个子分区(例如,2个或4个),其取决于块的块大小,如表3所示。表3示出了取决于块大小的子分区的示例。

[0212] 表3

[0213]

块大小	子分区的数目
4x4	未划分
4x8和8x4	2
所有其它情况	4

[0214] 在一些示例中,将CU分割为2个还是4个子分区取决于CU的块大小。图13A和图13B示出了一些示例性子分区。在一些示例中,所有子分区均满足具有至少16个样本的条件。对于色度分量,可以不应用ISP模式。在一些示例中,发信号通知ISP时,最多使用两个二进制数。第一个二进制数指示是否使用ISP。如果使用ISP,则进一步发信号通知另一个二进制数以指示ISP的方向,除非只有一个方向可用。

[0215] 图13A示出了根据本公开实施例的在ISP模式下编码的编码块(1300)的示例性水平ISP模式和示例性垂直ISP模式。在图13A的示例中,编码块(1300)的块大小是W1xH1,例如是4x8个样本或8x4个样本。因此,将编码块(1300)分区成2个子分区。如图13所示,可将编码块(1300)水平地分区成两个子分区(1311)-(1312),其中,每个子分区具有水平ISP模式的W1xH1/2个样本的大小,或者将编码块(1300)垂直地分区成两个子分区(1321)-(1322),其中,每个子分区具有垂直ISP模式的W1/2xH1个样本的大小。

[0216] 图13B示出了根据本公开实施例的在ISP模式下编码的另一编码块(1350)的示例性水平ISP模式和示例性垂直ISP模式。在图13B的示例中,编码块(1350)的块大小是

W2xH2，，例如，高于4x8个样本或8x4个样本。因此，将编码块(1350)分区成4个子分区。如图13B所示，可将编码块(1350)水平地分区成四个子分区(1361)-(1364)，其中，每个子分区具有W2xH2/4个样本的大小，或者将编码块(1350)垂直地分区成四个子分区(1371)-(1374)，其中每个子分区具有W2/4xH2个样本的大小。

[0217] 在一些示例中，对于这些子分区中的每一个子分区，通过对编码器发送的系数进行熵解码，然后对这些系数进行逆量化和逆变换，生成残差信号。然后，对于可以称为当前子分区的多个子分区中的一个子分区，可以通过对当前子分区执行帧内预测，生成预测信号。最后，可以通过将残差信号添加到预测信号，获得当前子分区的重建样本。因此，当前子分区的重建样本可用于预测另一子分区，例如，紧接当前子分区的另一子分区。该过程可以重复到其它子分区。所有子分区可以共享相同的帧内模式。

[0218] 在一些示例中，可以仅使用作为最可能模式(Most Probable Mode MPM)列表的一部分的帧内模式测试ISP模式。因此，如果以ISP模式对CU进行编码，则可以推断MPM列表的MPM标志为，例如，一个或为真。在一些情况下，可以修改MPM列表以排除DC模式，并且对用于水平ISP模式的水平帧内预测模式和用于垂直ISP模式的垂直帧内预测模式进行优先级排序。

[0219] 在一些示例中，因为每个子分区单独执行变换和重建，可以将CU的每个子分区看作是一个子变换单元。

[0220] 表4示出了根据本公开实施例的具有ISP模式的编码单元的示例性规范。

[0221] 表4

[0222]

coding_unit(x0, y0, cbWidth, cbHeight, cqtDepth, treeType, modeType) {	描 述 符 (Descriptor)
chType = treeType = = DUAL_TREE_CHROMA? 1 : 0	
if(slice_type != I | | sps_ibc_enabled_flag | | sps_palette_enabled_flag) {	
if(treeType != DUAL_TREE_CHROMA && !((cbWidth = = 4 &&	

[0223]

cbHeight = = 4) | | modeType = = MODE_TYPE_INTRA) && !sps_ibc_enabled_flag))	
cu_skip_flag[x0][y0]	ae(v)
if(cu_skip_flag[x0][y0] = = 0 && slice_type != I && !(cbWidth = = 4 && cbHeight = = 4) && modeType = = MODE_TYPE_ALL)	
pred_mode_flag	ae(v)
if(((slice_type = = I && cu_skip_flag[x0][y0] = =0) | | (slice_type != I && (CuPredMode[chType][x0][y0] != MODE_INTRA | | (cbWidth = = 4 && cbHeight = = 4 && cu_skip_flag[x0][y0] = = 0)))) && cbWidth <= 64 && cbHeight <= 64 && modeType != MODE_TYPE_INTER && sps_ibc_enabled_flag && treeType != DUAL_TREE_CHROMA)	
pred_mode_ibc_flag	ae(v)
if((((slice_type = = I | | (cbWidth = = 4 && cbHeight = = 4) | | sps_ibc_enabled_flag) && CuPredMode[x0][y0] = = MODE_INTRA) | | (slice_type != I && !(cbWidth = = 4 && cbHeight = = 4) && !sps_ibc_enabled_flag && CuPredMode[x0][y0] != MODE_INTRA)) && sps_palette_enabled_flag && cbWidth <= 64 && cbHeight <= 64 && cu_skip_flag[x0][y0] = = 0 && modeType != MODE_INTER)	
pred_mode_plt_flag	ae(v)
}	
if(CuPredMode[chType][x0][y0] = = MODE_INTRA | | CuPredMode[chType][x0][y0] = = MODE_PLT) {	
if(treeType = = SINGLE_TREE | | treeType = = DUAL_TREE_LUMA) {	
if(pred_mode_plt_flag) {	
if(treeType = = DUAL_TREE_LUMA)	
palette_coding(x0, y0, cbWidth, cbHeight, 0, 1)	
else /* SINGLE_TREE */	
palette_coding(x0, y0, cbWidth, cbHeight, 0, 3)	
} else {	
if(sps_bdpcm_enabled_flag && cbWidth <= MaxTsSize && cbHeight <= MaxTsSize)	

[0224]

intra_bdpcm_flag	ae(v)
if(intra_bdpcm_flag)	
intra_bdpcm_dir_flag	ae(v)
else {	
if(sps_mip_enabled_flag && (Abs(Log2(cbWidth) − Log2(cbHeight)) <= 2) && cbWidth <= MaxTbSizeY && cbHeight <= MaxTbSizeY)	
intra_mip_flag[x0][y0]	ae(v)
if(intra_mip_flag[x0][y0])	
intra_mip_mode[x0][y0]	ae(v)
else {	
if(sps_mrl_enabled_flag && ((y0 % CtbSizeY) > 0))	
intra_luma_ref_idx[x0][y0]	ae(v)
if (sps_isp_enabled_flag && intra_luma_ref_idx[x0][y0] = = 0 && (cbWidth <= MaxTbSizeY && cbHeight <= MaxTbSizeY) && (cbWidth * cbHeight > MinTbSizeY * MinTbSizeY))	
intra_subpartitions_mode_flag[x0][y0]	ae(v)
if(intra_subpartitions_mode_flag[x0][y0] = = 1)	
intra_subpartitions_split_flag[x0][y0]	ae(v)
if(intra_luma_ref_idx[x0][y0] = = 0)	
intra_luma_mpm_flag[x0][y0]	
if(intra_luma_mpm_flag[x0][y0]) {	
if(intra_luma_ref_idx[x0][y0] = = 0)	
intra_luma_not_planar_flag[x0][y0]	ae(v)
if(intra_luma_not_planar_flag[x0][y0])	
intra_luma_mpm_idx[x0][y0]	ae(v)
} else	
intra_luma_mpm_remainder[x0][y0]	ae(v)
}	

[0225]

}	
}	
}	
}	
}	

[0226] 在表4中,当变量intra_subpartitions_mode_flag[x0][y0]等于1时,它指定将当前帧内CU分区为NumIntraSubPartitions[x0][y0]个矩形变换块子分区。当变量intra_subpartitions_mode_flag[x0][y0]等于0时,它指定不将当前帧内编码单元分区为矩形变换块子分区。当变量intra_subpartitions_mode_flag[x0][y0]不存在时,推断变量intra_subpartitions_mode_flag[x0][y0]等于0。

[0227] 变量intra_subpartitions_split_flag[x0][y0]指定帧内子分区分割类型是水平或垂直。当变量intra_subpartitions_split_flag[x0][y0]不存在时,推导如下:

[0228] -若cbHeight大于MaxTbSizeY,推断intra_subpartitions_split_flag[x0][y0]等于0。

[0229] -否则,若变量cbWidth大于MaxTbSizeY,推断intra_subpartitions_split_flag[x0][y0]等于1。

[0230] 变量IntraSubPartitionsSplitType指定用于当前亮度编码块的分割类型如表5所示。表5示出了与IntraSubPartitionsSplitType相关联的示例性名称。IntraSubPartitionsSplitType推导如下。

[0231] -若变量intra_subpartitions_mode_flag[x0][y0]等于0,将变量IntraSubPartitionsSplitType设为等于0。

[0232] -否则,将变量IntraSubPartitionsSplitType设为等于1+intra_subpartitions_split_flag[x0][y0]。

[0233] 表5

[0234]

帧内子分区分割类型（IntraSubPartitionsSplitType）	帧内子分区分割类型（IntraSubPartitionsSplitType）的名称
0	ISP_NO_SPLIT
1	ISP_HOR_SPLIT
2	ISP_VER_SPLIT

[0235] 变量NumIntraSubPartitions指定了将帧内亮度编码块划分成的变换块子分区的数量。NumIntraSubPartitions推导如下。

[0236] -若变量IntraSubPartitionsSplitType等于ISP_NO_SPLIT,将变量NumIntraSubPartitions设为等于1。

[0237] -否则,如果以下条件之一为真,将NumIntraSubPartitions设为等于2:

[0238] -cbWidth等于4,并且cbHeight等于8,

[0239] -cbWidth等于8,并且cbHeight等于4;

[0240] -否则,将变量NumIntraSubPartitions设为等于4。

[0241] XI.缩放变换系数的变换过程

[0242] 缩放变换系数的变换过程的输入可以包括(i)亮度位置(xTbY,yTbY),指定当前亮度变换块相对于当前图片左上角亮度样本的左上角样本,(ii)变量nTbW,指定当前变换块的宽度;(iii)变量nTbH,指定当前变换块的高度;(iv)变量cIdx,指定当前块的颜色分量,以及(v)缩放变换系数的一个(nTbW)x(nTbH)数组d[x][y],其中,x=0..nTbW-1,y=0..nTbH-1。

[0243] 这个过程的输出为残差样本的(nTbW)x(nTbH)数组r[x][y],其中,x=0..nTbW-1,y=0..nTbH-1。

[0244] 当lfnst_idx[xTbY][yTbY]不等于0,nTbW和nTbH都大于或等于4,以下适用:

[0245] -变量predModeIntra,nLfnstOutSize,log2LfnstSize,nLfnstSize和nonZeroSize推导如下:

[0246] predModeIntra=(cIdx==0)?IntraPredModeY[xTbY][yTbY]:IntraPredModeC[xTbY][yTbY] 等式(1)

[0247] nLfnstOutSize=(nTbW>=8&&nTbH>=8)?48:16 等式(2)

[0248] log2LfnstSize=(nTbW>=8&&nTbH>=8)?3:2 等式(3)

[0249] nLfnstSize=1<<log2LfnstSize 等式(4)

[0250] nonZeroSize=((nTbW==4&&nTbH==4)||(nTbW==8&&nTbH==8))?8:16 等式(5)

[0251] -当intra_mip_flag[xTbComp][yTbComp]等于1,并且cIdx等于0,将predModeIntra设为等于INTRA_PLANAR。

[0252] -当predModeIntra等于INTRA_LT_CCLM,INTRA_L_CCLM,或INTRA_T_CCLM,将predModeIntra设为等于IntraPredModeY[xTbY+nTbW/2][yTbY+nTbH/2]。

[0253] -以predModeIntra,nTbW,nTbH和cIdx为输入,修改后的predModeIntra为输出,调用广角帧内预测模式映射过程。

[0254] -x=0..nonZeroSize-1的列表u[x]的值推导如下:

[0255] xC=DiagScanOrder[2][2][x][0] 等式(6)

[0256] yC=DiagScanOrder[2][2][x][1] 等式(7)

[0257] u[x]=d[xC][yC] 等式(8)

[0258] -调用一维低频不可分变换,使用以下内容作为输入:缩放变换系数nonZeroSize的输入长度,设置为等于nLfnstOutSize的变换输出长度nTrS,x=0..nonZeroSize-1的已缩放非零变换系数u[x]的列表,LFNST集selectionpredModeIntra的帧内预测模式,以及,在选择的LFNST集lfnst_idx[xTbY][yTbY]中用于变换选择的LFNST索引,使用x=0..nLfnstOutSize-1的列表v[x]作为输出。

[0259] -x=0..nLfnstSize-1,y=0..nLfnstSize-1的数组d[x][y]推导如下:

[0260] -若predModeIntra小于或等于34,以下适用:

[0261] d[x][y]=(y<4)?v[x+(y<<log2LfnstSize)]:((x<4)?v[32+x+((y-4)<<2)]:d[x][y]) 等式(9)

[0262] -否则,以下适用:

[0263] d[x][y]=(x<4)?v[y+(x<<log2LfnstSize)]:((y<4)?v[32+y+((x-4)<<2)]:d[x][y]) 等式(10)

[0264] 变量implicitMtsEnabled推导如下:

[0265] -若sps_mts_enabled_flag等于1,并且以下条件之一为真,将implicitMtsEnabled设置为等于1:

[0266] -IntraSubPartitionsSplitType不等于ISP_NO_SPLIT;

[0267] -cu_sbt_flag等于1,并且Max(nTbW,nTbH)小于或等于32;

[0268] -sps_explicit_mts_intra_enabled_flag等于0,CuPredMode[0][xTbY][yTbY]等于MODE_INTRA,lfnst_idx[x0][y0]等于0,并且intra_mip_flag[x0][y0]等于0。

[0269] -否则,将implicitMtsEnabled设置为等于0。

[0270] 指定水平变换内核的变量trTypeHor和指定垂直变换内核的变量trTypeVer推导如下:

[0271] -若cIdx大于0,将trTypeHor和trTypeVer设置为等于0。

[0272] -否则,如果implicitMtsEnabled等于1,以下适用:

[0273] -若IntraSubPartitionsSplitType不等于ISP_NO_SPLIT,orsps_explicit_mts_intra_enabled_flag等于0,并且CuPredMode[0][xTbY][yTbY]等于MODE_INTRA,trTypeHor和trTypeVer推导如下:

[0274] trTypeHor=(nTbW>=4&&nTbW<=16)?1:0 等式(11)

[0275] trTypeVer=(nTbH>=4&&nTbH<=16)?1:0 等式(12)

[0276] -否则(cu_sbt_flag等于1),trTypeHor和trTypeVer依赖于cu_sbt_horizontal_flag和cu_sbt_pos_flag。

[0277] -否则,trTypeHor和trTypeVer依赖于tu_mts_idx[xTbY][yTbY]。

[0278] 表6示出了根据本公开实施例的变换树的示例性规范。

[0279] 表7示出了根据本公开实施例的变换单元的示例性规范。

[0280] 表6

[0281]

transform_tree(x0, y0, tbWidth, tbHeight , treeType, chType) {	描 述 符 (Descriptor)
InferTuCbfLuma = 1	
if(IntraSubPartitionsSplitType = = ISP_NO_SPLIT && !cu_sbt_flag) {	
if(tbWidth > MaxTbSizeY | | tbHeight > MaxTbSizeY) {	
verSplitFirst = (tbWidth > MaxTbSizeY && tbWidth > tbHeight) ? 1 : 0	
trafoWidth = verSplitFirst ? (tbWidth / 2) : tbWidth	
trafoHeight = !verSplitFirst ? (tbHeight / 2) : tbHeight	
transform_tree(x0, y0, trafoWidth, trafoHeight, chType)	
if(verSplitFirst)	
transform_tree(x0 + trafoWidth, y0, trafoWidth, trafoHeight, treeType, chType)	
else	
transform_tree(x0, y0 + trafoHeight, trafoWidth, trafoHeight, treeType, chType)	
} else {	
transform_unit(x0, y0, tbWidth, tbHeight, treeType, 0, chType)	
}	
} else if(cu_sbt_flag) {	
if(!cu_sbt_horizontal_flag) {	
trafoWidth = tbWidth * SbtNumFourthsTb0 / 4	
transform_unit(x0, y0, trafoWidth, tbHeight, treeType , 0, 0)	
transform_unit(x0 + trafoWidth, y0, tbWidth − trafoWidth, tbHeight, treeType, 1, 0)	
} else {	
trafoHeight = tbHeight * SbtNumFourthsTb0 / 4	
transform_unit(x0, y0, tbWidth, trafoHeight, treeType , 0, 0)	
transform_unit(x0, y0 + trafoHeight, tbWidth, tbHeight − trafoHeight, treeType, 1, 0)	

[0282]

}	
} else if(IntraSubPartitionsSplitType = = ISP_HOR_SPLIT) {	
trafoHeight = tbHeight / NumIntraSubPartitions	
for(partIdx = 0; partIdx < NumIntraSubPartitions; partIdx++)	
transform_unit(x0, y0 + trafoHeight * partIdx, tbWidth, trafoHeight, treeType, partIdx, 0)	
} else if(IntraSubPartitionsSplitType = = ISP_VER_SPLIT) {	
trafoWidth = tbWidth / NumIntraSubPartitions	
for(partIdx = 0; partIdx < NumIntraSubPartitions; partIdx++)	
transform_unit(x0 + trafoWidth * partIdx, y0, trafoWidth, tbHeight, treeType, partIdx, 0)	
}	
}	

[0283] 表7

[0284]

transform_unit(x0, y0, tbWidth, tbHeight, treeType, subTuIndex, chType) {	描 述 符 (Descriptor)
if((treeType = = SINGLE_TREE | | treeType = = DUAL_TREE_CHROMA) &&ChromaArrayType != 0) {	
if((IntraSubPartitionsSplitType = = ISP_NO_SPLIT && !(cu_sbt_flag &&((subTuIndex = = 0 && cu_sbt_pos_flag) | |(subTuIndex = = 1 && !cu_sbt_pos_flag)))) | |(IntraSubPartitionsSplitType != ISP_NO_SPLIT &&(subTuIndex = = NumIntraSubPartitions − 1))) {	
tu_cbf_cb[x0][y0]	ae(v)
tu_cbf_cr[x0][y0]	ae(v)
}	
}	
if(treeType = = SINGLE_TREE | | treeType = = DUAL_TREE_LUMA) {	
if((IntraSubPartitionsSplitType = = ISP_NO_SPLIT && !(cu_sbt_flag &&((subTuIndex = = 0 && cu_sbt_pos_flag) | |(subTuIndex = = 1 && !cu_sbt_pos_flag))) &&(CuPredMode[chType][x0][y0] = = MODE_INTRA | |tu_cbf_cb[x0][y0] | | tu_cbf_cr[x0][y0] | |CbWidth[chType][x0][y0] > MaxTbSizeY | | CbHeight[chType][x0][y0] > MaxTbSizeY)) |	

[0285]

|(IntraSubPartitionsSplitType != ISP_NO_SPLIT && (subTuIndex < NumIntraSubPartitions − 1 | | !InferTuCbfLuma)))	
tu_cbf_luma[x0][y0]	ae(v)
if (IntraSubPartitionsSplitType != ISP_NO_SPLIT)	
InferTuCbfLuma = InferTuCbfLuma && !tu_cbf_luma[x0][y0]	
}	
if(IntraSubPartitionsSplitType != ISP_NO_SPLIT &&treeType = = SINGLE_TREE && subTuIndex = = NumIntraSubPartitions − 1)) {	
xC = CbPosX[chType][x0][y0]	
yC = CbPosY[chType][x0][y0]	
wC = CbWidth[chType][x0][y0] / SubWidthC	
hC = CbHeight[chType][x0][y0] / SubHeightC	
} else {	
xC = x0	
yC = y0	
wC = tbWidth / SubWidthC	
hC = tbHeight / SubHeightC	
}	
if((CbWidth[chType][x0][y0] > 64 | | CbHeight[chType][x0][y0] > 64 | |tu_cbf_luma[x0][y0] | | tu_cbf_cb[x0][y0] | | tu_cbf_cr[x0][y0]) &&treeType != DUAL_TREE_CHROMA) {	
if(cu_qp_delta_enabled_flag && !IsCuQpDeltaCoded) {	
cu_qp_delta_abs	ae(v)
if(cu_qp_delta_abs)	
cu_qp_delta_sign_flag	ae(v)
}	
}	
if((tu_cbf_cb[x0][y0] | | tu_cbf_cr[x0][y0]) {	
if(cu_chroma_qp_offset_enabled_flag && !IsCuChromaQpOffsetCoded) {	
cu_chroma_qp_offset_flag	ae(v)
if(cu_chroma_qp_offset_flag && chroma_qp_offset_list_len_minus1 > 0)	
cu_chroma_qp_offset_idx	
}	

[0286]

}	
if(sps_joint_cbcr_enabled_flag && ((CuPredMode[chType][x0][y0] ==MODE_INTRA&& (tu_cbf_cb[x0][y0] || tu_cbf_cr[x0][y0])) ||(tu_cbf_cb[x0][y0] && tu_cbf_cr[x0][y0])))	
tu_joint_cbcr_residual_flag[x0][y0]	ae(v)
if(tu_cbf_luma[x0][y0] && treeType != DUAL_TREE_CHROMA&& (tbWidth <= 32) && (tbHeight <= 32)&& (IntraSubPartitionsSplit[x0][y0] = = ISP_NO_SPLIT) && (!cu_sbt_flag)) {	
if(sps_transform_skip_enabled_flag && !BdpcmFlag[x0][y0] &&tbWidth <= MaxTsSize && tbHeight <= MaxTsSize)	
transform_skip_flag[x0][y0]	ae(v)
if(((CuPredMode[chType][x0][y0] = = MODE_INTER &&sps_explicit_mts_inter_enabled_flag) || (CuPredMode[chType][x0][y0] = = MODE_INTRA &&sps_explicit_mts_intra_enabled_flag)) && (!transform_skip_flag[x0][y0]))	
tu_mts_idx[x0][y0]	ae(v)
}	
if(tu_cbf_luma[x0][y0]) {	
if(!transform_skip_flag[x0][y0])	
residual_coding(x0, y0, Log2(tbWidth), Log2(tbHeight), 0)	
else	
residual_ts_coding(x0, y0, Log2(tbWidth), Log2(tbHeight), 0)	
}	
if(tu_cbf_cb[x0][y0])	
residual_coding(xC, yC, Log2(wC), Log2(hC), 1)	
if(tu_cbf_cr[x0][y0] &&!(tu_cbf_cb[x0][y0] && tu_joint_cbcr_residual_flag[x0][y0])) {	
residual_coding(xC, yC, Log2(wC), Log2(hC), 2)	
}	
}	

[0287] XII.小型色度帧内预测单元(Small Chroma Intra Prediction Unit,SCIPU)

[0288] 比如在HEVC中,最小帧内CU的大小是8x8个亮度样本。可以将最小帧内CU的亮度分量进一步分割成四个4x4亮度帧内PU,但是不能对最小帧内CU的色度分量进一步分割。因此,当处理4x4色度帧内块或4x4亮度帧内块时,可能出现最坏情况下的硬件处理吞吐量。

[0289] 在单个编码树中,比如在VTM5.0中,由于色度分区遵循相关联的亮度分区,并且最小的帧内CU具有4x4个亮度样本,因此最小的色度帧内CB是2x2。因此,VVC解码时最坏情况

下的硬件处理吞吐量仅为HEVC解码时最坏情况下的硬件处理吞吐量的1/4。

[0290] 比如在VTM6.0中，小色度帧内预测单元(small chroma intra prediction unit，SCIPU)旨在通过在单树中创建本地双树，禁止对小于16个色度样本的色度帧内CB进行分区。在进入本地双树之前，色度分区与相关联的亮度分区具有相同的分割。每个CU都可以灵活的决定是进行帧内编码、帧间编码还是IBC编码。在进入本地双树之后，所有CU应该是帧间编码或非帧间编码(帧内或IBC)。当对当前CU进行帧间编码时，当前CU的色度分量也遵循与其相关联的亮度块相同的分割。否则，若当前CU是非帧间编码时，可以进一步分割亮度块，而不能对相关联的色度帧内块进一步分割。确定当前CU是否是本地双树的父节点的示例性条件包括：(a)CU大小为64，并且对当前CU进行进一步QT分割，(b)CU大小为64，并且对当前CU进行进一步TT分割，(c)CU大小为32，并且对当前CU进行进一步BT分割，(d)CU大小为64，并且对当前CU进行进一步BT分割，以及(e)CU大小为128，并且对当前CU进行进一步TT分割。

[0291] 当上述条件之一为真时，将当前CU确定为本地双树的父节点。另外，当满足条件(a)、(b)或(c)时，将该区域(即，当前CU)内的所有CU(或子CU)约束为全部帧内编码或全部IBC编码。当满足条件d)或e)时，需要发信号通知另一个标志，以指示该区域(即，当前CU)内的所有CU(或子CU)是帧间编码还是非帧间编码。

[0292] 表8示出了根据本公开实施例的SCIPU模式的示范性规范。

[0293] 表8

[0294]

coding_tree(x0, y0, cbWidth, cbHeight, qgOnY, qgOnC, cbSubdiv, cqtDepth, mttDepth, dept hOffset, partIdx, treeTypeCurr, modeTypeCurr) {	描 述 符 (Descriptor)
if((allowSplitBtVer | | allowSplitBtHor | | allowSplitTtVer | | allowSplitTtHor | | allowSplitQT) &&(x0 + cbWidth <= pic_width_in_luma_samples) && (y0 + cbHeight <= pic_height_in_luma_samples))	

[0295]

split_cu_flag	ae(v)
if(cu_qp_delta_enabled_flag && qgOnY && cbSubdiv <= cu_qp_delta_subdiv) {	
IsCuQpDeltaCoded = 0	
CuQpDeltaVal = 0	
CuQgTopLeftX = x0	
CuQgTopLeftY = y0	
}	
if(cu_chroma_qp_offset_enabled_flag && qgOnC && cbSubdiv <= cu_chroma_qp_offset_subdiv)	
IsCuChromaQpOffsetCoded = 0	
if(split_cu_flag) {	
if((allowSplitBtVer | | allowSplitBtHor | | allowSplitTtVer | | allowSplitTtHor) && allowSplitQT)	
split_qt_flag	ae(v)
if(!split_qt_flag) {	
if((allowSplitBtHor | | allowSplitTtHor) && (allowSplitBtVer | | allowSplitTtVer))	
mtt_split_cu_vertical_flag	
if((allowSplitBtVer && allowSplitTtVer && mtt_split_cu_vertical_flag) | | (allowSplitBtHor && allowSplitTtHor && !mtt_split_cu_vertical_flag))	
mtt_split_cu_binary_flag	
}	
if(modeTypeCondition = = 1)	
modeType = MODE_TYPE_INTRA	
else if(modeTypeCondition = = 2) {	
mode_constraint_flag	
modeType = mode_constraint_flag ? MODE_TYPE_INTRA : MODE_TYPE_INTER	
} else {	
modeType = modeTypeCurr	

[0296]

}	
treeType = (modeType = = MODE_TYPE_INTRA) ? DUAL_TREE_LUMA : treeTypeCurr	
if(!split_qt_flag) {	
if(MttSplitMode[x0][y0][mttDepth] = = SPLIT_BT_VER) {	
depthOffset += (x0 + cbWidth > pic_width_in_luma_samples) ? 1 : 0	
x1 = x0 + (cbWidth / 2)	
coding_tree(x0, y0, cbWidth / 2, cbHeight, qgOnY, qgOnC, cbSubdiv + 1, cqtDepth, mttDepth + 1, depthOffset, 0, treeType, modeType)	
if(x1 < pic_width_in_luma_samples)	
coding_tree(x1, y0, cbWidth / 2, cbHeightY, qgOnY, qgOnC, cbSubdiv + 1, cqtDepth, mttDepth + 1, depthOffset, 1, treeType, modeType)	
} else if(MttSplitMode[x0][y0][mttDepth] = = SPLIT_BT_HOR) {	
depthOffset += (y0 + cbHeight > pic_height_in_luma_samples) ? 1 : 0	
y1 = y0 + (cbHeight / 2)	
coding_tree(x0, y0, cbWidth, cbHeight / 2, qgOnY, qgOnC, cbSubdiv + 1, cqtDepth, mttDepth + 1, depthOffset, 0, treeType, modeType)	
if(y1 < pic_height_in_luma_samples)	
coding_tree(x0, y1, cbWidth, cbHeight / 2, qgOnY, qgOnC, cbSubdiv + 1, cqtDepth, mttDepth + 1, depthOffset, 1, treeType, modeType)	
} else if(MttSplitMode[x0][y0][mttDepth] = = SPLIT_TT_VER) {	
x1 = x0 + (cbWidth / 4)	
x2 = x0 + (3 * cbWidth / 4)	
qgOnY = qgOnY && (cbSubdiv + 2 <= cu_qp_delta_subdiv)	
qgOnC = qgOnC && (cbSubdiv + 2 <= cu_chroma_qp_offset_subdiv)	
coding_tree(x0, y0, cbWidth / 4, cbHeight, qgOnY, qgOnC, cbSubdiv + 2, cqtDepth, mttDepth + 1, depthOffset, 0, treeType, modeType)	

[0297]

coding_tree(x1, y0, cbWidth / 2, cbHeight, qgOnY, qgOnC, cbSubdiv + 1, cqtDepth, mttDepth + 1, depthOffset, 1, treeType, modeType)	
coding_tree(x2, y0, cbWidth / 4, cbHeight, qgOnY, qgOnC, cbSubdiv + 2, cqtDepth, mttDepth + 1, depthOffset, 2, treeType, modeType)	
} else { /* SPLIT_TT_HOR */	
y1 = y0 + (cbHeight / 4)	
y2 = y0 + (3 * cbHeight / 4)	
qgOnY = qgOnY && (cbSubdiv + 2 <= cu_qp_delta_subdiv)	
qgOnC = qgOnC && (cbSubdiv + 2 <= cu_chroma_qp_offset_subdiv)	
coding_tree(x0, y0, cbWidth, cbHeight / 4, qgOnY, qgOnC, cbSubdiv + 2, cqtDepth, mttDepth + 1, depthOffset, 0, treeType, modeType)	
coding_tree(x0, y1, cbWidth, cbHeight / 2, qgOnY, qgOnC, cbSubdiv + 1, cqtDepth, mttDepth + 1, depthOffset, 1, treeType, modeType)	
coding_tree(x0, y2, cbWidth, cbHeight / 4, qgOnY, qgOnC, cbSubdiv + 2, cqtDepth, mttDepth + 1, depthOffset, 2, treeType, modeType)	
}	
} else {	
x1 = x0 + (cbWidth / 2)	
y1 = y0 + (cbHeight / 2)	
coding_tree(x0, y0, cbWidth / 2, cbHeight / 2, qgOnY, qgOnC, cbSubdiv + 2, cqtDepth + 1, 0, 0, 0, treeType, modeType)	
if(x1 < pic_width_in_luma_samples)	
coding_tree(x1, y0, cbWidth / 2, cbHeight / 2, qgOnY, qgOnC, cbSubdiv + 2, cqtDepth + 1, 0, 0, 1, treeType, modeType)	
if(y1 < pic_height_in_luma_samples)	
coding_tree(x0, y1, cbWidth / 2, cbHeight / 2, qgOnY, qgOnC, cbSubdiv + 2, cqtDepth + 1, 0, 0, 2, treeType, modeType)	
if(y1 < pic_height_in_luma_samples && x1 < pic_width_in_luma_samples)	
coding_tree(x1, y1, cbWidth / 2, cbHeight / 2, qgOnY, qgOnC, cbSubdiv + 2, cqtDepth + 1, 0, 0, 3, treeType, modeType)	

[0298]

}	
if(modeTypeCur = = MODE_TYPE_ALL && modeType = = MODE_TYPE_INTRA) {	
coding_tree(x0, y0, cbWidth, cbHeight, qgOnY, qgOnC, cbSubdiv, cqtDepth, mttDepth, 0, 0 DUAL_TREE_CHROMA , modeType)	
}	
else	
coding_unit(x0, y0, cbWidth, cbHeight, cqtDepth, treeTypeCurr , modeTypeCurr)	
}	
}	

[0299] 变量modeTypeCondition推导如下:

[0300] -如果以下条件之一为真,将modeTypeCondition设置为等于0。

[0301] -slice_type＝＝I并且qtbtt_dual_tree_intra_flag等于1,

[0302] -modeTypeCurr不等于MODE_TYPE_ALL。

[0303] -否则,如果以下条件之一为真,将modeTypeCondition设置为等于1。

[0304] -cbWidth*cbHeight等于64,并且split_qt_flag等于1,

[0305] -cbWidth*cbHeight等于64,并且MttSplitMode[x0][y0][mttDepth]等于SPLIT_TT_HOR或SPLIT_TT_VER,

[0306] -cbWidth*cbHeight等于32,并且MttSplitMode[x0][y0][mttDepth]等于SPLIT_BT_HOR or SPLIT_BT_VER。

[0307] -否则,如果以下条件之一为真时,将modeTypeCondition设置为等于1+(slice_type!＝I?1:0).

[0308] -cbWidth*cbHeight等于64,并且MttSplitMode[x0][y0][mttDepth]等于SPLIT_BT_HOR或SPLIT_BT_VER,

[0309] -cbWidth*cbHeight等于128,并且MttSplitMode[x0][y0][mttDepth]等于SPLIT_TT_HOR或SPLIT_TT_VER。

[0310] -否则,将modeTypeCondition设置为等于0。

[0311] 当变量mode_constraint_flag等于0,指定当前编码树节点内的编码单元只能使用帧间预测编码方式。当变量mode_constraint_flag等于1,指定当前编码树节点内的编码单元不能使用帧间预测编码方式。

[0312] XIII.不同的颜色格式

[0313] 图14A至图14D示出了不同YUV格式的一些示例。对于YUV 4:4:4格式,U图片和V图片的宽度和高度均与Y图片的宽度和高度相同。对于YUV 4:2:2格式,U图片和V图片的高度与Y图片的高度相同,但是U图片和V图片的宽度是Y图片的宽度的一半。对于YUV 4:1:1格式,U图片和V图片的高度与Y图片的高度相同,但是U图片和V图片的宽度是Y图片的宽度的四分之一。对于YUV 4:2:0格式,U图片和V图片的宽度和高度均是Y图片的宽度和高度的一

半。

[0314] XIV.多参考行帧内预测

[0315] 图15示出了根据本公开实施例的示例性多参考行帧内预测。多行帧内预测可以将更多的参考行用于帧内预测,并且编码器可以决定并发信号通知哪个参考行被用于生成帧内预测值。在示例中,可以使用参考行0、1和/或3,并且可以排除参考行2。可以在帧内预测模式之前发信号通知参考行索引,如果可以发信号通知非零参考行索引,仅允许最可能的模式。在图15中,描述了4个参考行的示例,其中每个参考行由六个段,即,A段至F段,以及左上参考样本组成。另外,A段和F段分别用来自B段和E段的最接近的样本填充。

[0316] XV.本地双树的改进

[0317] 本公开的实施例可以改进本地双树的使用。在一些相关示例中,SCIPU在单个树中避免2x2、2x4和4x2小色度块,但在单个树中采用本地双树。然而,本地双树可能难以实现。因此,SCIPU的修改可能不能证明其益处。

[0318] 根据本公开的各方面,一个SCIPU区域内的树深度可以小于或等于阈值T,其中T是整数,比如,0、1或2。T的值指示SCIPU区域可以被进一步分割的次数。例如,当T等于0时,这意味着SCIPU不能被进一步分割。当T等于1时,这意味着SCIPU仅可以被进一步分割一次。

[0319] 在一个实施例中,帧间编码的SCIPU区域和非帧间编码的SCIPU区域可以具有不同的阈值。在对当前SCIPU区域中的所有CU进行帧间编码时,当前SCIPU区域内的树深度可以小于或等于阈值T。若当前SCIPU区域中的所有CU是非帧间编码(例如,帧内编码或IBC编码)时,当前SCIPU区域内的树深度可以大于阈值T。即,非帧间编码的SCIPU的树深度不受阈值T的约束。注意,在该实施例中,T是正整数,比如,1或2。

[0320] XVI.本地双树的改进流程图

[0321] 图16示出了根据本公开实施例的概述示例过程(1600)的流程图。在各种实施例中,过程(1600)由处理电路执行,比如,终端设备(210),终端设备(220),终端设备(230),终端设备(240)中的处理电路,执行视频编码器(303)功能的处理电路,执行视频解码器(310)功能的处理电路,执行视频解码器(410)功能的处理电路,执行帧间预测模块(452)功能的处理电路,执行视频编码器(503)功能的处理电路,执行预测器(535)功能的处理电路,执行视频编码器(622)功能的处理电路,执行视频解码器(772)功能的处理电路,等等。在一些实施例中,过程(1600)在软件指令中实现,于是当处理电路执行软件指令时,处理电路执行过程(1600)。

[0322] 过程(1600)通常可以从步骤(S1610)开始,其中,步骤(1600)解码当前图片中当前块的预测信息,该当前图片是已编码视频序列的一部分。预测信息可以指示当前块的单个分区树结构和块大小。然后,过程(1600)进行到步骤(S1620)。

[0323] 在步骤(S1620),过程(1600)基于当前块的单个分区树结构和块大小,确定当前块是否是本地分区树结构的父节点。本地分区树结构的树深度小于或等于阈值。若确定当前块是本地分区树结构的父节点,过程(1600)进行到步骤(S1630)。

[0324] 在步骤(S1630),若当前块是本地分区树结构的父节点,过程(1600)根据本地分区树结构对当前块进行分区。在一个实施例中,过程(1600)响应于当前块为本地分区树结构的父节点并且树深度大于一个值(比如,0),根据本地分区树结构对当前块进行分区。然后,过程(1600)进行到步骤(S1640)。

[0325] 在步骤(S1640),过程(1600)基于当前块的预测模式,重建当前块。

[0326] 在重建当前块之后,过程(1600)终止。

[0327] 在一个实施例中,当满足三个条件之一时,将当前块的预测模式确定为非帧间预测。三个条件分别为:(i)块大小是64个样本,本地分区树结构为四叉树;(ii)块大小是64个样本,本地分区树结构为三叉树;(iii)块大小是32个样本,本地分区树结构是二叉树。

[0328] 在一个实施例中,当满足两个条件之一时,基于预测信息中包含的发信号通知的标志确定当前块的预测模式。两个条件包括:(i)块大小是64个样本,本地分区树结构是二叉树;(ii)块大小是128个样本,本地分区树结构是三叉树。

[0329] 在一个实施例中,阈值是当前块被帧间编码时的第一阈值和基于当前块被非帧间编码时的第二阈值。第一阈值可以不同于第二阈值。

[0330] 在一个实施例中,当确定当前块为本地分区树结构的父节点时,过程(1600)确定当前块为SCIPU。

[0331] 在一个实施例中,过程(1600)基于是否已经确定当前块的预测模式,对当前块进行分区。例如,通过向表8中所示的示例性规范添加条件,可以将阈值设置为1。添加的条件可以为“modeTypeCurr==Mode_TYPE_ALL”,如表9所示。对于初始分区,未确定当前块的预测模式,因此可以满足增加的条件“modeTypeCurr==Mode_TYPE_ALL”。如果还满足其他条件,可以进行初始分区。在初始分区之后,确定当前块的预测模式,于是不满足条件“modeTypeCurr==Mode_TYPE_ALL”.尽管仍然满足其他条件,不能进行第二次分区。因此,添加的条件“modeTypeCurr==Mode_TYPE_ALL”可能导致本地分区树结构的树深度受到阈值的约束,阈值设置为1。

[0332] 在一个实施例中,过程(1600)基于当前块是否被帧间编码对当前块进行分区。例如,可以通过向表8中所示的示例性规范添加另一个条件,将阈值设置为1。添加的条件可以为“modeTypeCurr!=Mode_TYPE_INTER”,如表10所示。对于初始分区,没有确定当前块的预测模式,因此可以满足增加的条件“modeTypeCurr!=Mode_TYPE_INTER”。如果还满足其他条件,则可以进行初始分区。在初始分区之后,确定当前块的预测模式。若确定当前块的预测模式为帧间预测,则不满足条件“modeTypeCurr!=Mode_TYPE_INTER”。虽然还可以满足其他条件,不能进行第二次分区。因此,若当前块的预测模式为帧间预测时,添加的条件“modeTypeCurr!=Mode_TYPE_INTER”可以导致本地分区树结构的树深度受到阈值的约束,阈值设置为1。

[0333] 表9

[0334]

coding_tree(x0, y0, cbWidth, cbHeight, qgOnY, qgOnC, cbSubdiv, cqtDepth, mttDepth, dept hOffset, partIdx, treeTypeCurr, modeTypeCurr) {	描 述 符 (Descriptor)
if((allowSplitBtVer | | allowSplitBtHor | | allowSplitTtVer | | allowSplitTtHor | | allowSplitQT) &&(x0 + cbWidth <= pic_width_in_luma_samples) && (y0 + cbHeight <= pic_height_in_luma_samples)&& modeTypeCurr = =MODE_TYPE_ALL)	
split_cu_flag	ae(v)
if(cu_qp_delta_enabled_flag && qgOnY && cbSubdiv <= cu_qp_delta_subdiv) {	
IsCuQpDeltaCoded = 0	
CuQpDeltaVal = 0	
CuQgTopLeftX = x0	
CuQgTopLeftY = y0	
}	
if(cu_chroma_qp_offset_enabled_flag && qgOnC && cbSubdiv <= cu_chroma_qp_offset_subdiv)	
IsCuChromaQpOffsetCoded = 0	

if(split_cu_flag) {	
if((allowSplitBtVer | | allowSplitBtHor | | allowSplitTtVer | | allowSplitTtHor) && allowSplitQT)	
split_qt_flag	ae(v)
if(!split_qt_flag) {	
if((allowSplitBtHor | | allowSplitTtHor) && (allowSplitBtVer | | allowSplitTtVer))	
mtt_split_cu_vertical_flag	ae(v)
if((allowSplitBtVer && allowSplitTtVer && mtt_split_cu_vertical_flag) | | (allowSplitBtHor && allowSplitTtHor && !mtt_split_cu_vertical_flag))	
mtt_split_cu_binary_flag	ae(v)
}	
if(modeTypeCondition = = 1)	
modeType = MODE_TYPE_INTRA	
else if(modeTypeCondition = = 2) {	
mode_constraint_flag	ae(v)
modeType = mode_constraint_flag ? MODE_TYPE_INTRA : MODE_TYPE_INTER	
} else {	
modeType = modeTypeCurr	
}	
treeType = (modeType = = MODE_TYPE_INTRA) ? DUAL_TREE_LUMA : treeTypeCurr	
if(!split_qt_flag) {	
if(MttSplitMode[x0][y0][mttDepth] = = SPLIT_BT_VER) {	
depthOffset += (x0 + cbWidth > pic_width_in_luma_samples) ? 1 : 0	
x1 = x0 + (cbWidth / 2)	
coding_tree(x0, y0, cbWidth / 2, cbHeight, qgOnY, qgOnC, cbSubdiv + 1, cqtDepth, mttDepth + 1, depthOffset, 0, treeType, modeType)	
if(x1 < pic_width_in_luma_samples)	
coding_tree(x1, y0, cbWidth / 2, cbHeightY, qgOnY, qgOnC, cbSubdiv + 1, cqtDepth, mttDepth + 1, depthOffset, 1, treeType,	

[0335]

[0336]

modeType)	
} else if(MttSplitMode[x0][y0][mttDepth] = = SPLIT_BT_HOR) {	
depthOffset += (y0 + cbHeight > pic_height_in_luma_samples) ? 1 : 0	
y1 = y0 + (cbHeight / 2)	
coding_tree(x0, y0, cbWidth, cbHeight / 2, qgOnY, qgOnC, cbSubdiv + 1, cqtDepth, mttDepth + 1, depthOffset, 0, treeType, modeType)	
if(y1 < pic_height_in_luma_samples)	
coding_tree(x0, y1, cbWidth, cbHeight / 2, qgOnY, qgOnC, cbSubdiv + 1, cqtDepth, mttDepth + 1, depthOffset, 1, treeType, modeType)	
} else if(MttSplitMode[x0][y0][mttDepth] = = SPLIT_TT_VER) {	
x1 = x0 + (cbWidth / 4)	
x2 = x0 + (3 * cbWidth / 4)	
qgOnY = qgOnY && (cbSubdiv + 2 <= cu_qp_delta_subdiv)	
qgOnC = qgOnC && (cbSubdiv + 2 <= cu_chroma_qp_offset_subdiv)	
coding_tree(x0, y0, cbWidth / 4, cbHeight, qgOnY, qgOnC, cbSubdiv + 2, cqtDepth, mttDepth + 1, depthOffset, 0, treeType, modeType)	
coding_tree(x1, y0, cbWidth / 2, cbHeight, qgOnY, qgOnC, cbSubdiv + 1, cqtDepth, mttDepth + 1, depthOffset, 1, treeType, modeType)	
coding_tree(x2, y0, cbWidth / 4, cbHeight, qgOnY, qgOnC, cbSubdiv + 2, cqtDepth, mttDepth + 1, depthOffset, 2, treeType, modeType)	
} else { /* SPLIT_TT_HOR */	
y1 = y0 + (cbHeight / 4)	
y2 = y0 + (3 * cbHeight / 4)	
qgOnY = qgOnY && (cbSubdiv + 2 <= cu_qp_delta_subdiv)	
qgOnC = qgOnC && (cbSubdiv + 2 <= cu_chroma_qp_offset_subdiv)	
coding_tree(x0, y0, cbWidth, cbHeight / 4, qgOnY, qgOnC,	

[0337]

cbSubdiv + 2, cqtDepth, mttDepth + 1, depthOffset, 0, treeType, modeType)	
coding_tree(x0, y1, cbWidth, cbHeight / 2, qgOnY, qgOnC, cbSubdiv + 1, cqtDepth, mttDepth + 1, depthOffset, 1, treeType, modeType)	
coding_tree(x0, y2, cbWidth, cbHeight / 4, qgOnY, qgOnC, cbSubdiv + 2, cqtDepth, mttDepth + 1, depthOffset, 2, treeType, modeType)	
}	
} else {	
x1 = x0 + (cbWidth / 2)	
y1 = y0 + (cbHeight / 2)	
coding_tree(x0, y0, cbWidth / 2, cbHeight / 2, qgOnY, qgOnC, cbSubdiv + 2, cqtDepth + 1, 0, 0, 0, treeType, modeType)	
if(x1 < pic_width_in_luma_samples)	
coding_tree(x1, y0, cbWidth / 2, cbHeight / 2, qgOnY, qgOnC, cbSubdiv + 2, cqtDepth + 1, 0, 0, 1, treeType, modeType)	
if(y1 < pic_height_in_luma_samples)	
coding_tree(x0, y1, cbWidth / 2, cbHeight / 2, qgOnY, qgOnC, cbSubdiv + 2, cqtDepth + 1, 0, 0, 2, treeType, modeType)	
if(y1 < pic_height_in_luma_samples && x1 < pic_width_in_luma_samples)	
coding_tree(x1, y1, cbWidth / 2, cbHeight / 2, qgOnY, qgOnC, cbSubdiv + 2, cqtDepth + 1, 0, 0, 3, treeType, modeType)	
}	
if(modeTypeCur = = MODE_TYPE_ALL && modeType = = MODE_TYPE_INTRA) {	
coding_tree(x0, y0, cbWidth, cbHeight, qgOnY, qgOnC, cbSubdiv, cqtDepth, mttDepth, 0, 0 DUAL_TREE_CHROMA , modeType)	
}	
else	
coding_unit(x0, y0, cbWidth, cbHeight, cqtDepth, treeTypeCurr , modeTypeCurr)	
}	
}	

[0338] 表10

[0339]

coding_tree(x0, y0, cbWidth, cbHeight, qgOnY, qgOnC, cbSubdiv, cqtDepth, mttDepth, dept hOffset, partIdx, treeTypeCurr, modeTypeCurr) {	描 述 符 (Descriptor)
if((allowSplitBtVer | | allowSplitBtHor | | allowSplitTtVer | | allowSplitTtHor | | allowSplitQT) &&(x0 + cbWidth <= pic_width_in_luma_samples) && (y0 + cbHeight <= pic_height_in_luma_samples)&& modeTypeCurr != MODE_TYPE_INTER)	
split_cu_flag	ae(v)
if(cu_qp_delta_enabled_flag && qgOnY && cbSubdiv <= cu_qp_delta_subdiv) {	
IsCuQpDeltaCoded = 0	
CuQpDeltaVal = 0	
CuQgTopLeftX = x0	
CuQgTopLeftY = y0	
}	
if(cu_chroma_qp_offset_enabled_flag && qgOnC && cbSubdiv <= cu_chroma_qp_offset_subdiv)	
IsCuChromaQpOffsetCoded = 0	
if(split_cu_flag) {	
if((allowSplitBtVer | | allowSplitBtHor | | allowSplitTtVer | | allowSplitTtHor) && allowSplitQT)	
split_qt_flag	ae(v)
if(!split_qt_flag) {	
if((allowSplitBtHor | | allowSplitTtHor) && (allowSplitBtVer | | allowSplitTtVer))	
mtt_split_cu_vertical_flag	ae(v)
if((allowSplitBtVer && allowSplitTtVer && mtt_split_cu_vertical_flag) | | (allowSplitBtHor && allowSplitTtHor && !mtt_split_cu_vertical_flag))	
mtt_split_cu_binary_flag	ae(v)
}	

[0340]

if(modeTypeCondition = = 1)	
modeType = MODE_TYPE_INTRA	
else if(modeTypeCondition = = 2) {	
mode_constraint_flag	ae(v)
modeType = mode_constraint_flag ? MODE_TYPE_INTRA : MODE_TYPE_INTER	
} else {	
modeType = modeTypeCurr	
}	
treeType = (modeType = = MODE_TYPE_INTRA) ? DUAL_TREE_LUMA : treeTypeCurr	
if(!split_qt_flag) {	
if(MttSplitMode[x0][y0][mttDepth] = = SPLIT_BT_VER) {	
depthOffset += (x0 + cbWidth > pic_width_in_luma_samples) ? 1 : 0	
x1 = x0 + (cbWidth / 2)	
coding_tree(x0, y0, cbWidth / 2, cbHeight, qgOnY, qgOnC, cbSubdiv + 1, cqtDepth, mttDepth + 1, depthOffset, 0, treeType, modeType)	
if(x1 < pic_width_in_luma_samples)	
coding_tree(x1, y0, cbWidth / 2, cbHeightY, qgOnY, qgOnC, cbSubdiv + 1, cqtDepth, mttDepth + 1, depthOffset, 1, treeType, modeType)	
} else if(MttSplitMode[x0][y0][mttDepth] = = SPLIT_BT_HOR) {	
depthOffset += (y0 + cbHeight > pic_height_in_luma_samples) ? 1 : 0	
y1 = y0 + (cbHeight / 2)	
coding_tree(x0, y0, cbWidth, cbHeight / 2, qgOnY, qgOnC, cbSubdiv + 1, cqtDepth, mttDepth + 1, depthOffset, 0, treeType, modeType)	
if(y1 < pic_height_in_luma_samples)	
coding_tree(x0, y1, cbWidth, cbHeight / 2, qgOnY, qgOnC, cbSubdiv + 1, cqtDepth, mttDepth + 1, depthOffset, 1, treeType, modeType)	
} else if(MttSplitMode[x0][y0][mttDepth] = = SPLIT_TT_VER) {	

[0341]

x1 = x0 + (cbWidth / 4)	
x2 = x0 + (3 * cbWidth / 4)	
qgOnY = qgOnY && (cbSubdiv + 2 <= cu_qp_delta_subdiv)	
qgOnC = qgOnC && (cbSubdiv + 2 <= cu_chroma_qp_offset_subdiv)	
coding_tree(x0, y0, cbWidth / 4, cbHeight, qgOnY, qgOnC, cbSubdiv + 2, cqtDepth, mttDepth + 1, depthOffset, 0, treeType, modeType)	
coding_tree(x1, y0, cbWidth / 2, cbHeight, qgOnY, qgOnC, cbSubdiv + 1, cqtDepth, mttDepth + 1, depthOffset, 1, treeType, modeType)	
coding_tree(x2, y0, cbWidth / 4, cbHeight, qgOnY, qgOnC, cbSubdiv + 2, cqtDepth, mttDepth + 1, depthOffset, 2, treeType, modeType)	
} else { /* SPLIT_TT_HOR */	
y1 = y0 + (cbHeight / 4)	
y2 = y0 + (3 * cbHeight / 4)	
qgOnY = qgOnY && (cbSubdiv + 2 <= cu_qp_delta_subdiv)	
qgOnC = qgOnC && (cbSubdiv + 2 <= cu_chroma_qp_offset_subdiv)	
coding_tree(x0, y0, cbWidth, cbHeight / 4, qgOnY, qgOnC, cbSubdiv + 2, cqtDepth, mttDepth + 1, depthOffset, 0, treeType, modeType)	
coding_tree(x0, y1, cbWidth, cbHeight / 2, qgOnY, qgOnC, cbSubdiv + 1, cqtDepth, mttDepth + 1, depthOffset, 1, treeType, modeType)	
coding_tree(x0, y2, cbWidth, cbHeight / 4, qgOnY, qgOnC, cbSubdiv + 2, cqtDepth, mttDepth + 1, depthOffset, 2, treeType, modeType)	
}	
} else {	
x1 = x0 + (cbWidth / 2)	
y1 = y0 + (cbHeight / 2)	
coding_tree(x0, y0, cbWidth / 2, cbHeight / 2, qgOnY, qgOnC, cbSubdiv + 2, cqtDepth + 1, 0, 0, 0, treeType, modeType)	
if(x1 < pic_width_in_luma_samples)	
coding_tree(x1, y0, cbWidth / 2, cbHeight / 2, qgOnY,	

[0342]

qgOnC, cbSubdiv + 2, cqtDepth + 1, 0, 0, 1, treeType, modeType)	
if(y1 < pic_height_in_luma_samples)	
coding_tree(x0, y1, cbWidth / 2, cbHeight / 2, qgOnY, qgOnC, cbSubdiv + 2, cqtDepth + 1, 0, 0, 2, treeType, modeType)	
if(y1 < pic_height_in_luma_samples && x1 < pic_width_in_luma_samples)	
coding_tree(x1, y1, cbWidth / 2, cbHeight / 2, qgOnY, qgOnC, cbSubdiv + 2, cqtDepth + 1, 0, 0, 3, treeType, modeType)	
}	
if(modeTypeCur = = MODE_TYPE_ALL && modeType = = MODE_TYPE_INTRA) {	
coding_tree(x0, y0, cbWidth, cbHeight, qgOnY, qgOnC, cbSubdiv, cqtDepth, mttDepth, 0, 0 DUAL_TREE_CHROMA , modeType)	
}	
else	
coding_unit(x0, y0, cbWidth, cbHeight, cqtDepth, treeTypeCurr , modeTypeCurr)	
}	
}	

[0343] XVII.计算机系统

[0344] 提出的方法可以单独使用或以任何顺序组合使用。此外,每个实施例中,编码器和解码器可以由处理电路(例如,至少一个处理器,或至少一个集成电路)实现。在示例中,至少一个处理器执行存储在非易失性计算机可读介质中的程序。

[0345] 上述技术可以通过计算机可读指令实现为计算机软件,并且物理地存储在至少一个计算机可读介质中。例如,图17示出了计算机系统(1700),其适于实现所公开主题的某些实施例。

[0346] 所述计算机软件可通过任何合适的机器代码、或计算机语言进行编码,通过汇编、编译、链接等机制创建包括指令的代码,所述指令可由至少一个计算机中央处理单元(CPU),图形处理单元(GPU)等直接执行或通过译码、微代码等方式执行。

[0347] 所述指令可以在各种类型的计算机或其组件上执行,包括例如个人计算机、平板电脑、服务器、智能手机、游戏设备、物联网设备等。

[0348] 图17所示的用于计算机系统(1700)的组件本质上是示例性的,并不用于对实现本申请实施例的计算机软件的使用范围或功能进行任何限制。也不应将组件的配置解释为与计算机系统(1700)的示例性实施例中所示的任一组件或其组合具有任何依赖性或要求。

[0349] 计算机系统(1700)可以包括某些人机界面输入设备。这种人机界面输入设备可以通过触觉输入(如:键盘输入、滑动、数据手套移动)、音频输入(如:声音、掌声)、视觉输入(如:手势)、嗅觉输入(未示出),对至少一个人类用户的输入做出响应。所述人机界面设备还可用于捕获某些媒体,气与人类有意识的输入不必直接相关,如音频(例如:语音、音乐、

环境声音)、图像(例如:扫描图像、从静止影像相机获得的摄影图像)、视频(例如二维视频、包括立体视频的三维视频)。

[0350] 人机界面输入设备可包括以下中的至少一个(仅绘出其中一个):键盘(1701)、鼠标(1702)、触控板(1703)、触摸屏(1710)、数据手套(未示出)、操纵杆(1705)、麦克风(1706)、扫描仪(1707)、照相机(1708)。

[0351] 计算机系统(1700)还可以包括某些人机界面输出设备。这种人机界面输出设备可以通过例如触觉输出、声音、光和嗅觉/味觉来刺激至少一个人类用户的感觉。这样的人机界面输出设备可包括触觉输出设备(例如通过触摸屏(1710)、数据手套(未示出)或操纵杆(1705)的触觉反馈,但也可以有不用作输入设备的触觉反馈设备)、音频输出设备(例如,扬声器(1709)、耳机(未示出))、视觉输出设备(例如,包括阴极射线管屏幕、液晶屏幕、等离子屏幕、有机发光二极管屏的屏幕(1710),其中每一个都具有或没有触摸屏输入功能、每一个都具有或没有触觉反馈功能——其中一些可通过诸如立体画面输出的手段输出二维视觉输出或三维以上的输出;虚拟现实眼镜(未示出)、全息显示器和放烟箱(未示出))以及打印机(未示出)。这些视觉输出设备(例如屏幕(1710))可以通过图形适配器(1750)连接到系统总线(1748)。

[0352] 计算机系统(1700)还可以包括人可访问的存储设备及其相关介质,如包括具有CD/DVD的高密度只读/可重写式光盘(CD/DVD ROM/RW)(1720)或类似介质(1721)的光学介质、拇指驱动器(1722)、可移动硬盘驱动器或固体状态驱动器(1723),诸如磁带和软盘(未示出)的传统磁介质,诸如安全软件保护器(未示出)等的基于ROM/ASIC/PLD的专用设备,等等。

[0353] 本领域技术人员还应当理解,结合所公开的主题使用的术语“计算机可读介质”不包括传输介质、载波或其它瞬时信号。

[0354] 计算机系统(1700)还可以包括通往至少一个通信网络(1755)的网络接口(1754)。例如,所述至少一个通信网络(1755)可以是无线的、有线的、光学的。所述至少一个通信网络(1755)还可为局域网、广域网、城域网、车载网络和工业网络、实时网络、延迟容忍网络等等。所述至少一个通信网络(1755)的示例还包括以太网、无线局域网、蜂窝网络(GSM、3G、4G、5G、LTE等)等局域网、电视有线或无线广域数字网络(包括有线电视、卫星电视、和地面广播电视)、车载和工业网络(包括CANBus)等等。某些网络通常需要外部网络接口适配器,用于连接到某些通用数据端口或外围总线(1749)(例如,计算机系统(1700)的USB端口);其它系统通常通过连接到如下所述的系统总线集成到计算机系统(1700)的核心(例如,以太网接口集成到PC计算机系统或蜂窝网络接口集成到智能电话计算机系统)。通过使用这些网络中的任何一个,计算机系统(1700)可以与其它实体进行通信。所述通信可以是单向的,仅用于接收(例如,无线电视),单向的仅用于发送(例如CAN总线到某些CAN总线设备),或双向的,例如通过局域或广域数字网络到其它计算机系统。上述的每个网络和网络接口可使用某些协议和协议栈。

[0355] 上述的人机界面设备、人可访问的存储设备以及网络接口可以连接到计算机系统(1700)的核心(1740)。

[0356] 核心(1740)可包括至少一个中央处理单元(CPU)(1741)、图形处理单元(GPU)(1742)、以现场可编程门阵列(FPGA)(1743)形式的专用可编程处理单元、用于特定任务的

硬件加速器(1744)等。这些设备以及只读存储器(ROM)(1745)、随机存取存储器(1746)、内部大容量存储器(例如内部非用户可存取硬盘驱动器、固态硬盘等)(1747)等可通过系统总线(1748)进行连接。在某些计算机系统中,可以以至少一个物理插头的形式访问系统总线(1748),以便可通过额外的中央处理单元、图形处理单元等进行扩展。外围装置可直接附接到核心的系统总线(1748),或通过外围总线(1749)进行连接。外围总线的体系结构包括外部控制器接口PCI、通用串行总线USB等。

[0357] CPU(1741)、GPU(1742)、FPGA(1743)和加速器(1744)可以执行某些指令,这些指令组合起来可以构成上述计算机代码。该计算机代码可以存储在ROM(1745)或RAM(1746)中。过渡数据也可以存储在RAM(1746)中,而永久数据可以存储在例如内部大容量存储器(1747)中。通过使用高速缓冲存储器可实现对任何存储器设备的快速存储和检索,高速缓冲存储器可与至少一个CPU(1741)、GPU(1742)、大容量存储器(1747)、ROM(1745)、RAM(1746)等紧密关联。

[0358] 所述计算机可读介质上可具有计算机代码,用于执行各种计算机实现的操作。介质和计算机代码可以是为本申请的目的而特别设计和构造的,也可以是计算机软件领域的技术人员所熟知和可用的介质和代码。

[0359] 作为实施例而非限制,具有体系结构(1700)的计算机系统,特别是核心(1740),可以作为处理器(包括CPU、GPU、FPGA、加速器等)提供执行包含在至少一个有形的计算机可读介质中的软件的功能。这种计算机可读介质可以是与上述的用户可访问的大容量存储器相关联的介质,以及具有非易失性的核心(1740)的特定存储器,例如核心内部大容量存储器(1747)或ROM(1745)。实现本申请的各种实施例的软件可以存储在这种设备中并且由核心(1740)执行。根据特定需要,计算机可读介质可包括一个或一个以上存储设备或芯片。该软件可以使得核心(1740)特别是其中的处理器(包括CPU、GPU、FPGA等)执行本文所述的特定过程或特定过程的特定部分,包括定义存储在RAM(1746)中的数据结构以及根据软件定义的过程来修改这种数据结构。另外或作为替代,计算机系统可以提供逻辑硬连线或以其它方式包含在电路(例如,加速器(1744))中的功能,该电路可以代替软件或与软件一起运行以执行本文所述的特定过程或特定过程的特定部分。在适当的情况下,对软件的引用可以包括逻辑,反之亦然。在适当的情况下,对计算机可读介质的引用可包括存储执行软件的电路(如集成电路(IC)),包含执行逻辑的电路,或两者兼备。本申请包括任何合适的硬件和软件组合。

[0360] 虽然本申请已对多个示例性实施例进行了描述,但实施例的各种变更、排列和各种等同替换均属于本申请的范围内。因此应理解,本领域技术人员能够设计多种系统和方法,所述系统和方法虽然未在本文中明确示出或描述,但其体现了本申请的原则,因此属于本申请的精神和范围之内。

[0361] 附录A:首字母缩略词

[0362] AMVP:高级运动矢量预测(Advanced Motion Vector Prediction)

[0363] ASIC:专用集成电路(Application-Specific Integrated Circuit)

[0364] ATMVP:可选/高级时间运动矢量预测(Alternative/Advanced Temporal Motion Vector Prediction)

[0365] BDOF:双向光流(Bi-directional Optical Flow)

[0366] BDPCM(or RDPCM):基于块的差分脉冲编码调制(Block based DPCM,或者残差脉冲编码调制(Residual Difference Pulse Coded Modulation))

[0367] BIO:双向光流(Bi-directional Optical Flow)

[0368] BMS:基准集合(Benchmark Set)

[0369] BT:二叉树(Binary Tree)

[0370] BV:块矢量(Block Vector)

[0371] CANBus:控制器局域网络总线(Controller Area Network Bus)

[0372] CB:编码块(Coding Block)

[0373] CBF:编码块标志(Coded Block Flag)

[0374] CCLM:跨组件线性模式/模型(Cross-Component Linear Mode/Model)

[0375] CD:光盘(Compact Disc)

[0376] CPR:当前图片参考(Current Picture Referencing)

[0377] CPU:中央处理单元(Central Processing Unit)

[0378] CRT:阴极射线管(Cathode Ray Tube)

[0379] CTB:编码树块(Coding Tree Block)

[0380] CTU:编码树单元(Coding Tree Unit)

[0381] CU:编码单元(Coding Unit)

[0382] DM:衍生模式(Derived Mode)

[0383] DPB:解码器图片缓冲器(Decoder Picture Buffer)

[0384] DVD:数字化视频光盘(Digital Video Disc)

[0385] FPGA:现场可编程门阵列(Field Programmable Gate Areas)

[0386] GOP:图片群组(Group of Picture)

[0387] GPU:图形处理单元(Graphics Processing Unit)

[0388] GSM:全球移动通信系统(Global System for Mobile communications)HDR:高动态范围图像(High Dynamic Range)

[0389] HEVC:高效视频编码(High Efficiency Video Coding)

[0390] HRD:假想参考解码器(Hypothetical Reference Decoder)

[0391] IBC:帧内块复制(Intra Block Copy)

[0392] IC:集成电路(Integrated Circuit)

[0393] ISP:帧内子分区(Intra Sub-Partitions)

[0394] JEM:联合开发模型(Joint Exploration Model)

[0395] JVET:联合视频开发组(Joint Video Exploration Team)

[0396] LAN:局域网(Local Area Network)

[0397] LCD:液晶显示器(Liquid-Crystal Display)

[0398] LFNST:低频不可分离的转换,

[0399] LTE:长期演进(Long-Term Evolution)

[0400] L_CCLM:左交叉分量线性模式/模型(Left-Cross-Component Linear Mode/Model)

[0401] LT_CCLM:左上交叉分量线性模式/模型(Left and Top Cross-Component Linear

Mode/Model)

[0402] MIP:基于矩阵的帧内预测(Matrix based Intra Prediction)

[0403] MPM:最可能模式(Most Probable Mode)

[0404] MRLP(or MRL):多参考线预测(Multiple Reference Line Prediction)

[0405] MTS:多变换选择(Multiple Transform Selection)

[0406] MV:运动矢量(Motion Vector)

[0407] OLED:有机发光二极管(Organic Light-Emitting Diode)

[0408] PBs:预测块(Prediction Blocks)

[0409] PCI:外围设备互连(Peripheral Component Interconnect)

[0410] PDPC:位置决定的联合预测(Position Dependent Prediction Combination)

[0411] PLD:可编程逻辑设备(Programmable Logic Device)

[0412] PPR:并行处理区域(Parallel-Processable Region)

[0413] PU:预测单元(Prediction Unit)

[0414] QT:四叉树(Quad-Tree)

[0415] RAM:随机存储器(Random Access Memory)

[0416] ROM:只读存储器(Read-Only Memory)

[0417] SBT:子块变换(Sub-block Transform)

[0418] SCC:屏幕内容编码(Screen Content Coding)

[0419] SCIPU:小色度帧内预测单元(Small Chroma Intra Prediction Unit)

[0420] SDR:标准动态范围(Standard Dynamic Range)

[0421] SEI:辅助增强信息(Supplementary Enhancement Information)

[0422] SNR:信噪比(Signal Noise Ratio)

[0423] SSD:固态驱动器(Solid-state Drive)

[0424] TT:三叉树(Ternary Tree)

[0425] TU:变换单元(Transform Unit)

[0426] T_CCLM:顶部跨组件线性模式/模型(Top Cross-Component Linear Mode/Model)

[0427] USB:通用串行总线(Universal Serial Bus)

[0428] VPDU:可视化过程数据单元(Visual Process Data Unit)

[0429] VUI:视频可用性信息(Video Usability Information)

[0430] VVC:通用视频编码(Versatile Video Coding)

[0431] WAIP:宽角度帧内预测(Wide-Angle Intra Prediction)

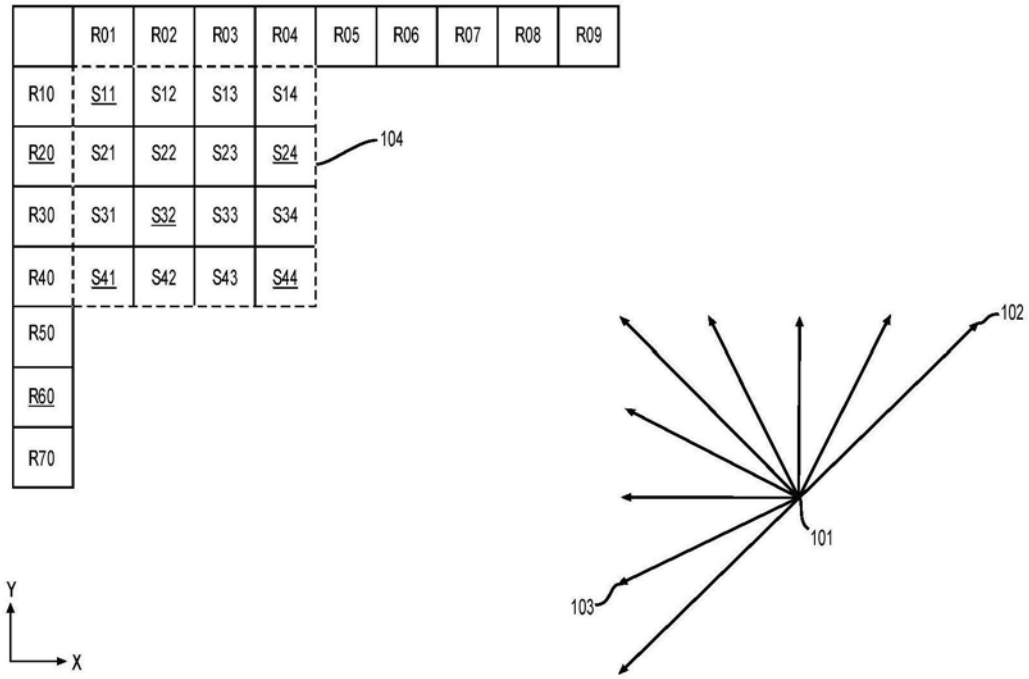

图1A(现有技术)

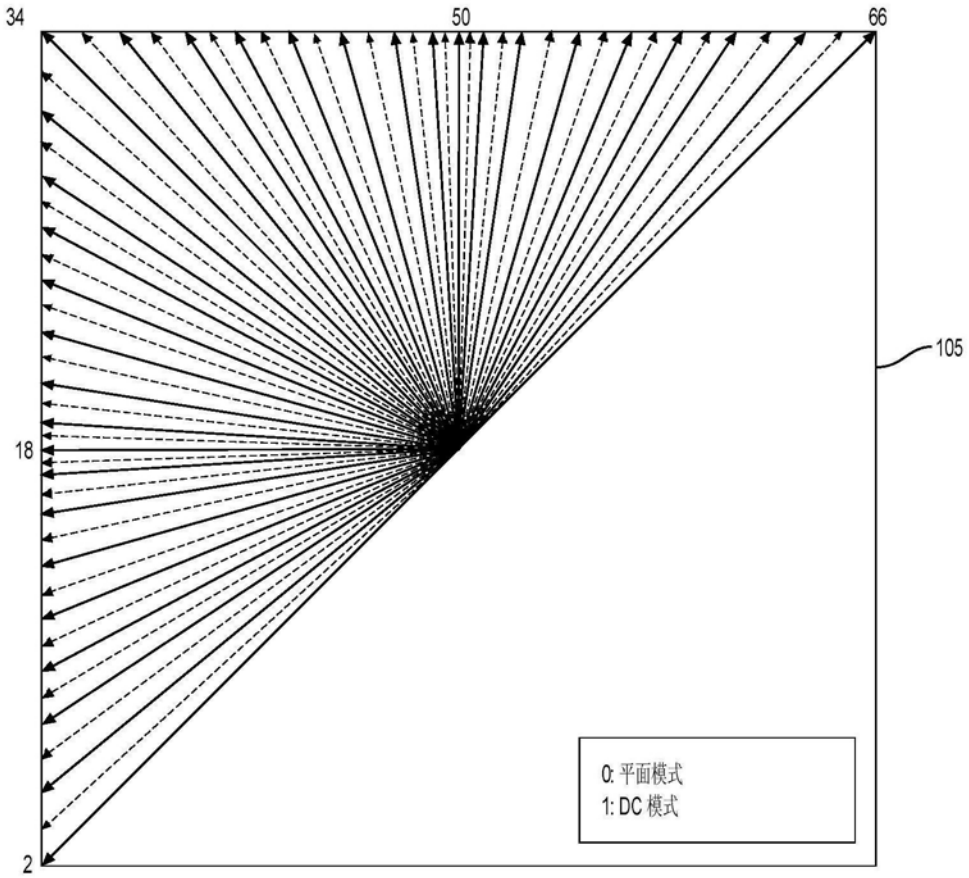

图1B(现有技术)

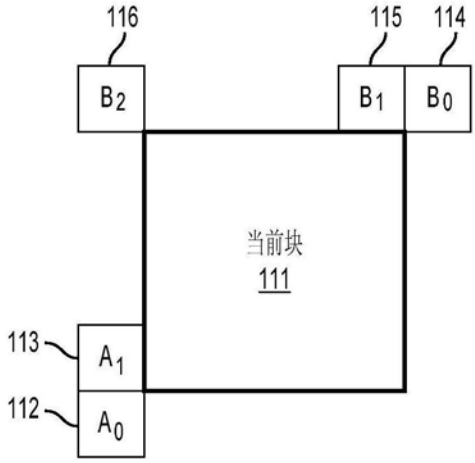
116
115
114
B2
B1
B0
当前块
111
113
A1
112
A0

图1C(现有技术)

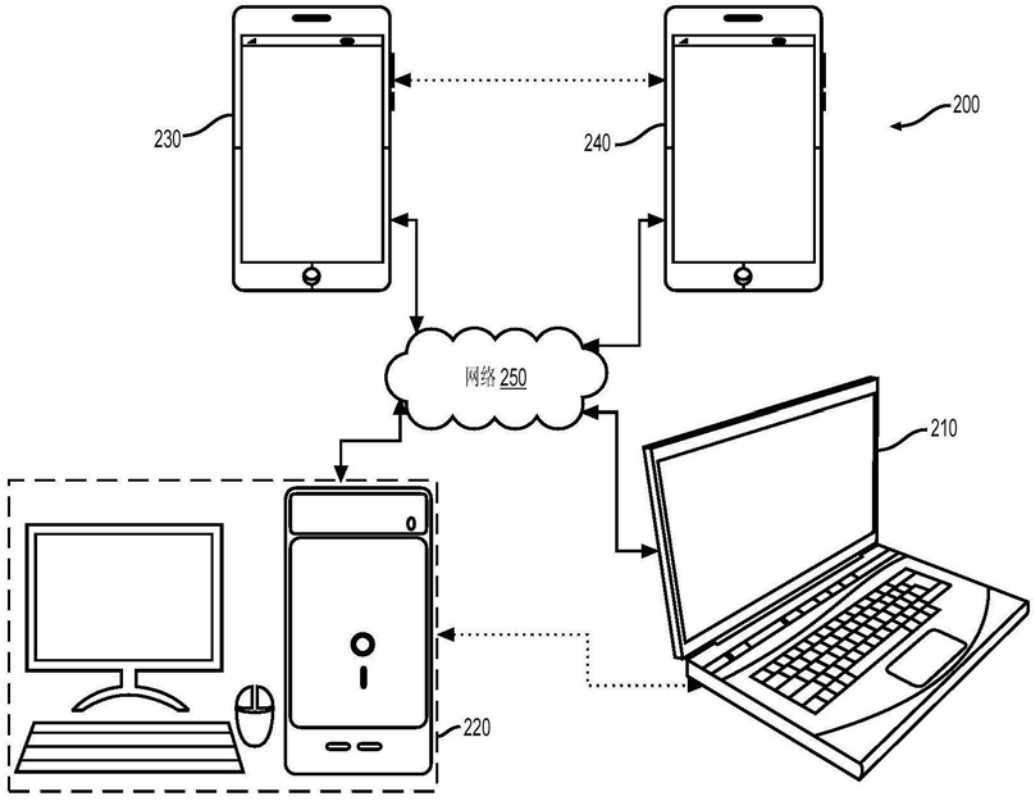
230
240
200
网络 250
210
220

图2

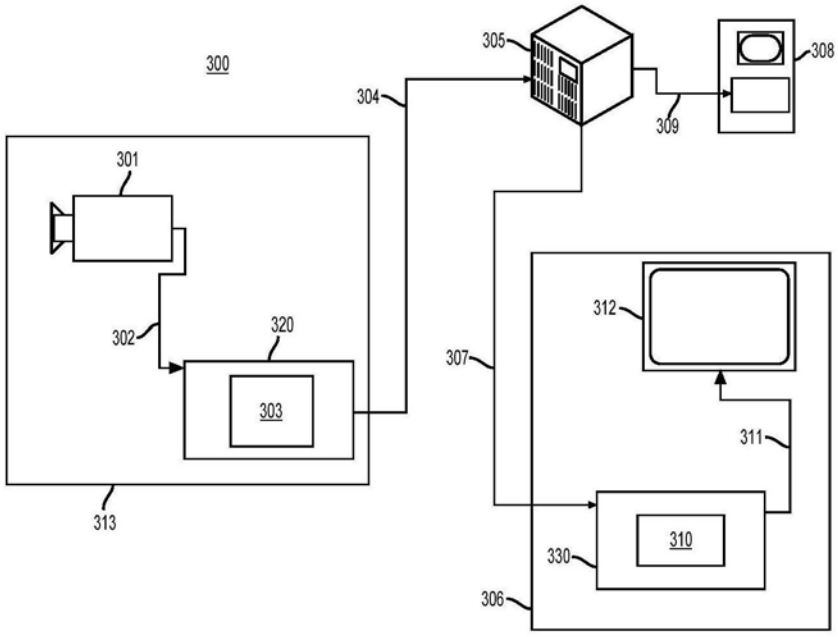
300
301
302
303
304
305
306
307
308
309
310
311
312
313
320
330

图3

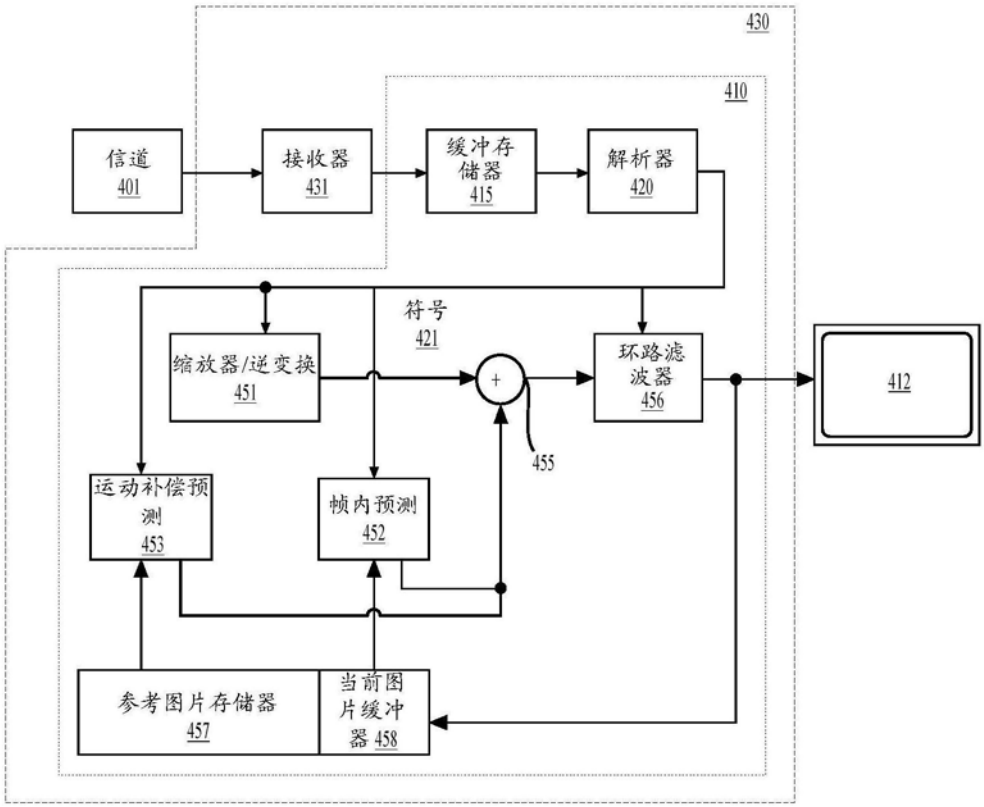
430
410
信道
401
接收器
431
缓冲存储器
415
解析器
420
符号
421
缩放器/逆变换
451
环路滤波器
456
412
455
运动补偿预测
453
帧内预测
452
参考图片存储器
457
当前图片缓冲器 458

图4

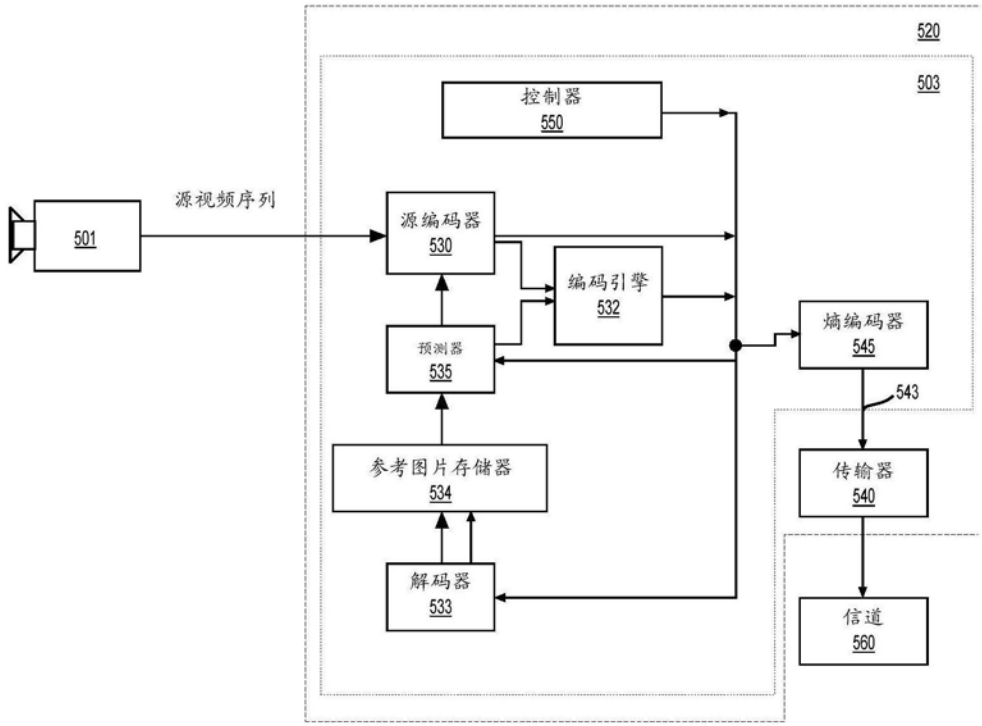
520
503
控制器
550
源视频序列
501
源编码器
530
编码引擎
532
熵编码器
545
预测器
535
543
参考图片存储器
534
传输器
540
解码器
533
信道
560

图5

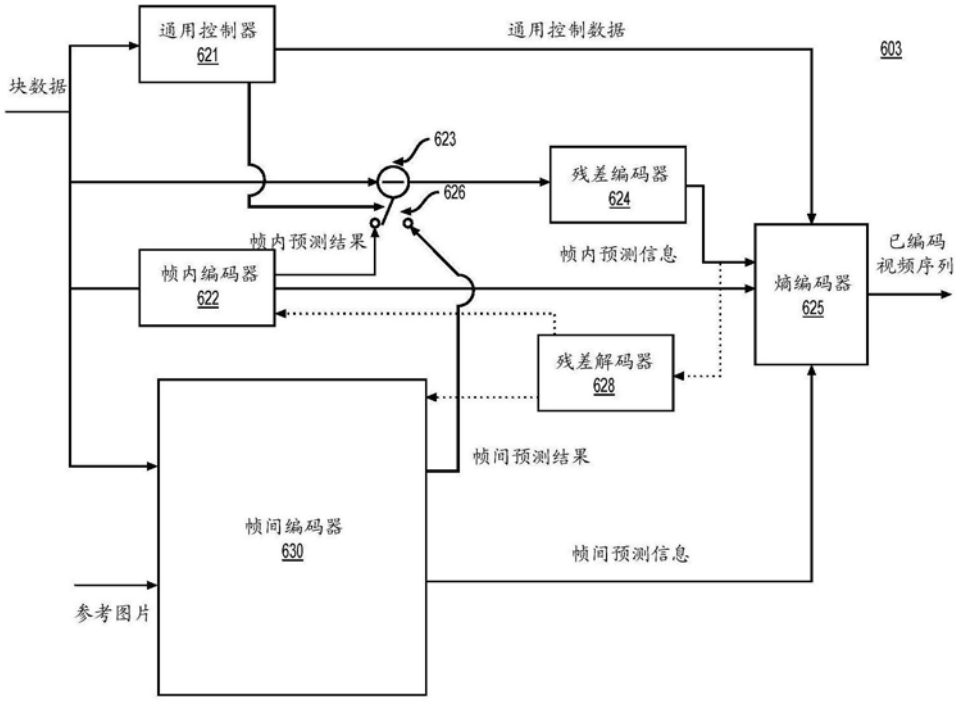
通用控制器
621
通用控制数据
603
块数据
623
残差编码器
624
626
帧内预测结果
帧内预测信息
已编码
视频序列
帧内编码器
622
熵编码器
625
残差解码器
628
帧间预测结果
帧间编码器
630
帧间预测信息
参考图片

图6

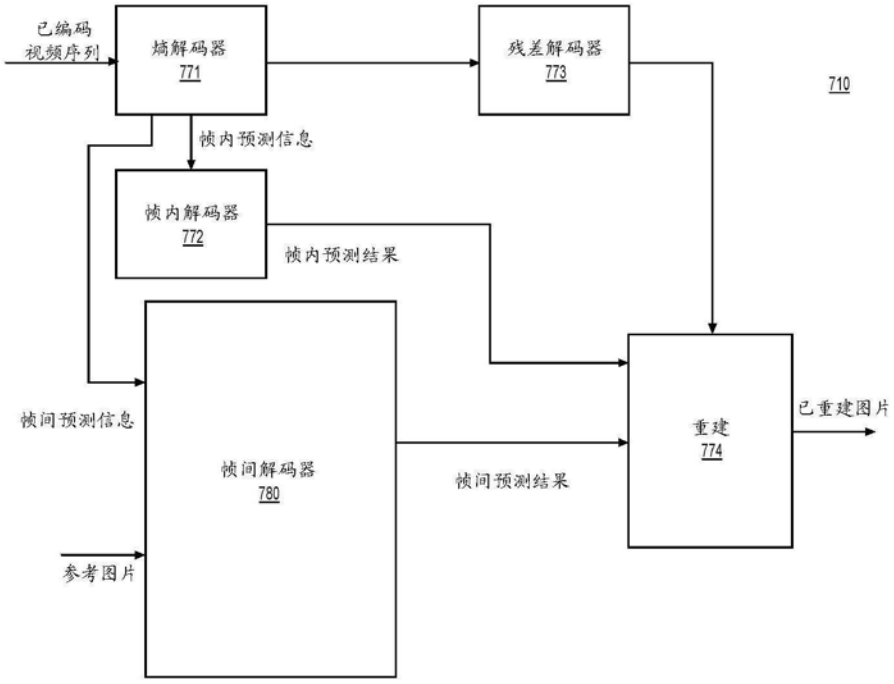
已编码
视频序列
熵解码器
771
残差解码器
773
710
帧内预测信息
帧内解码器
772
帧内预测结果
帧间预测信息
帧间解码器
780
帧间预测结果
重建
774
已重建图片
参考图片

图7

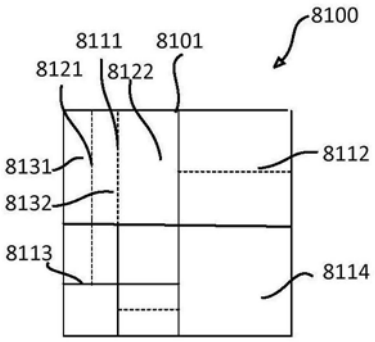
8100
8111
8101
8121
8122
8131
8112
8132
8113
8114

图8A

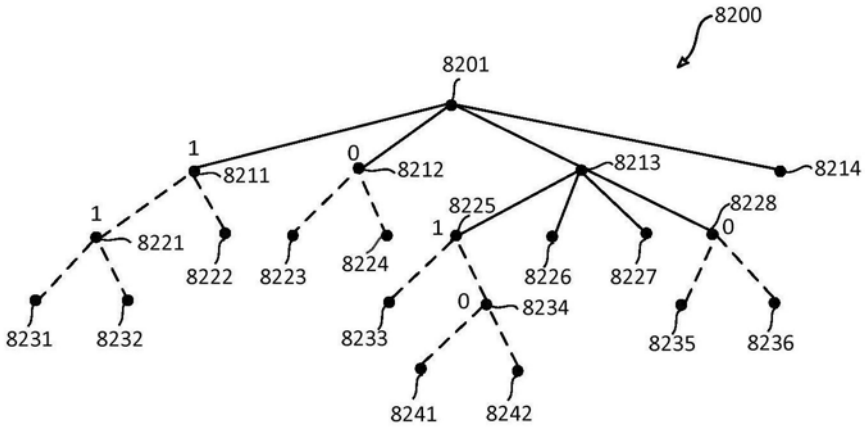
8200
8201
1
8211
0
8212
8213
8214
1
8221
8222
8223
8224
8225
1
8226
8227
8228
0
8231
8232
8233
0
8234
8235
8236
8241
8242

图8B

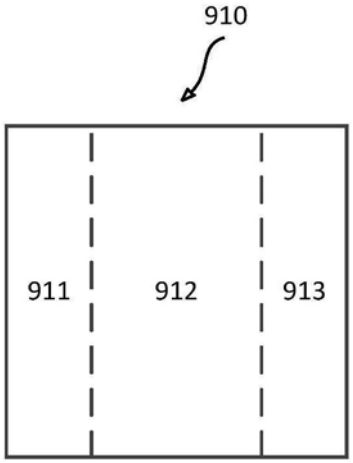
910
911
912
913

图9A

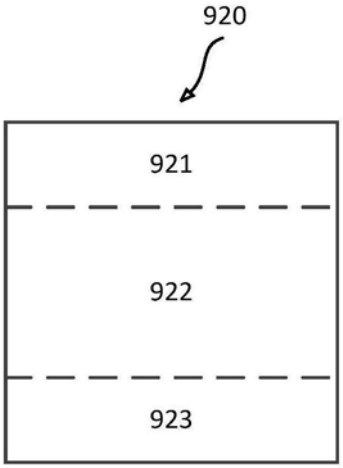
920
921
922
923

图9B

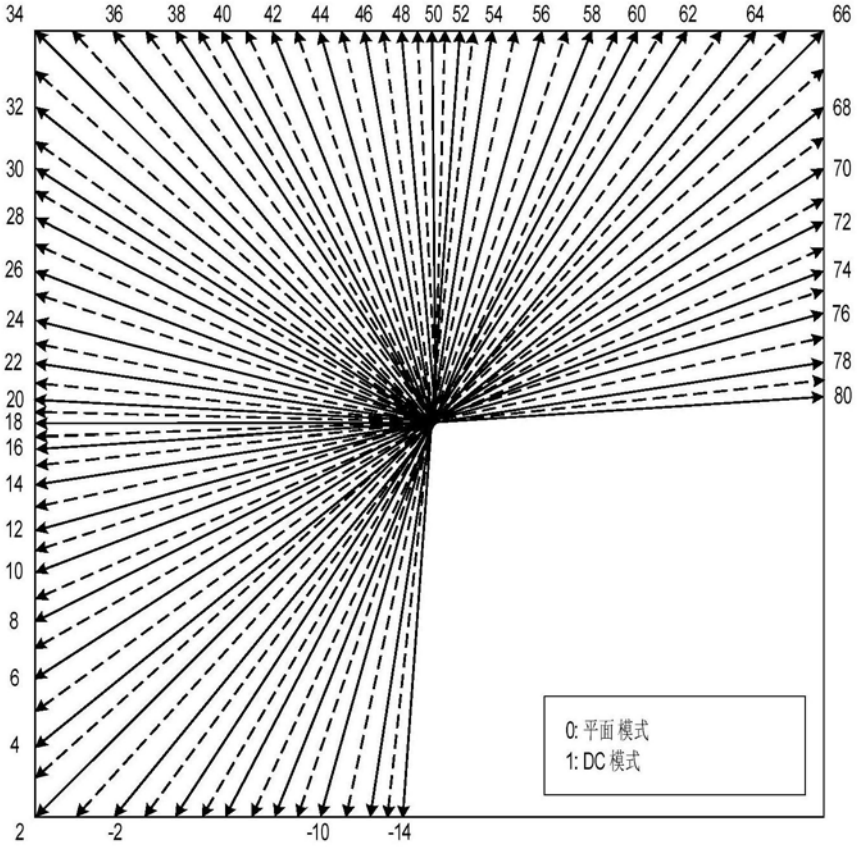

图10

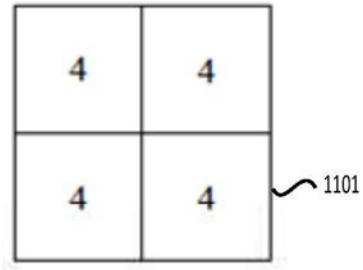

图11A

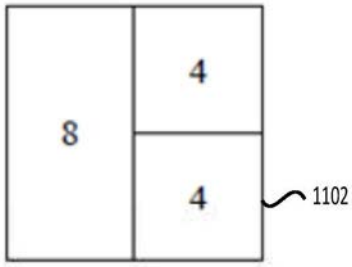

图11B

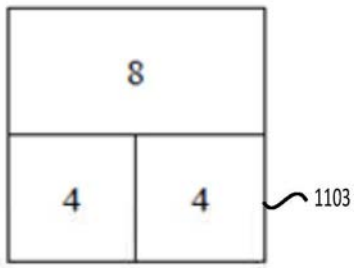

图11C

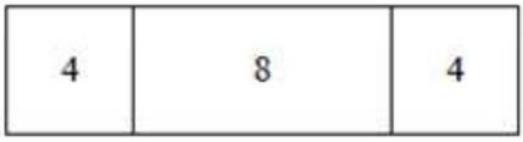

图11D

图11E

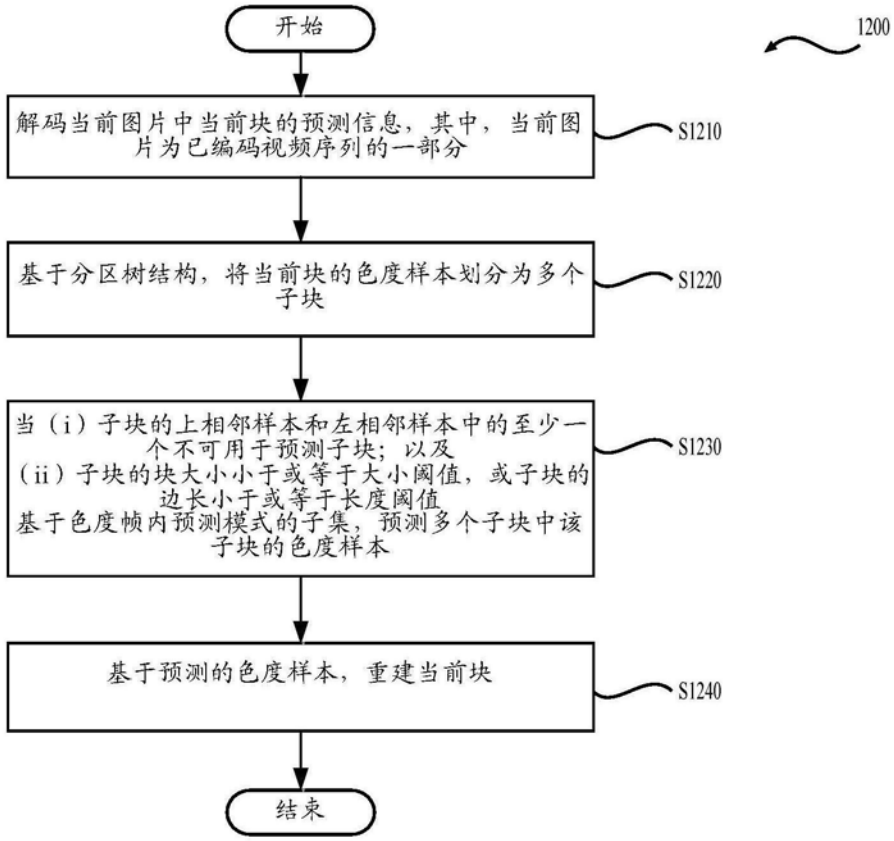
开始
1200
解码当前图片中当前块的预测信息，其中，当前图片为已编码视频序列的一部分
S1210
基于分区树结构，将当前块的色度样本划分为多个子块
S1220
当（i）子块的上相邻样本和左相邻样本中的至少一个不可用于预测子块；以及
（ii）子块的块大小小于或等于大小阈值，或子块的边长小于或等于长度阈值
基于色度帧内预测模式的子集，预测多个子块中该子块的色度样本
S1230
基于预测的色度样本，重建当前块
S1240
结束

图12

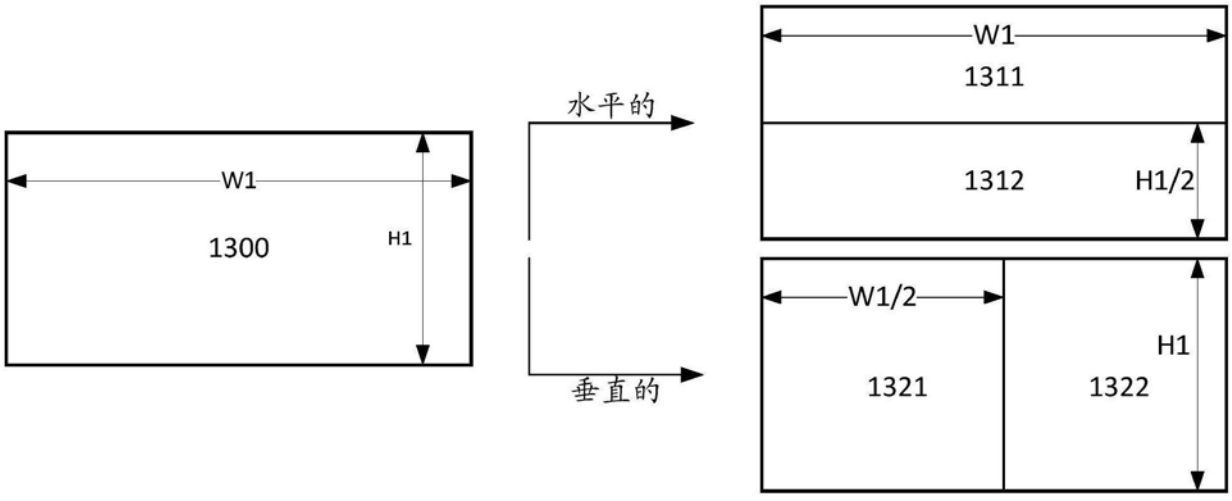
W1
1300
H1
水平的
垂直的
W1
1311
1312
H1/2
W1/2
1321
1322
H1

图13A

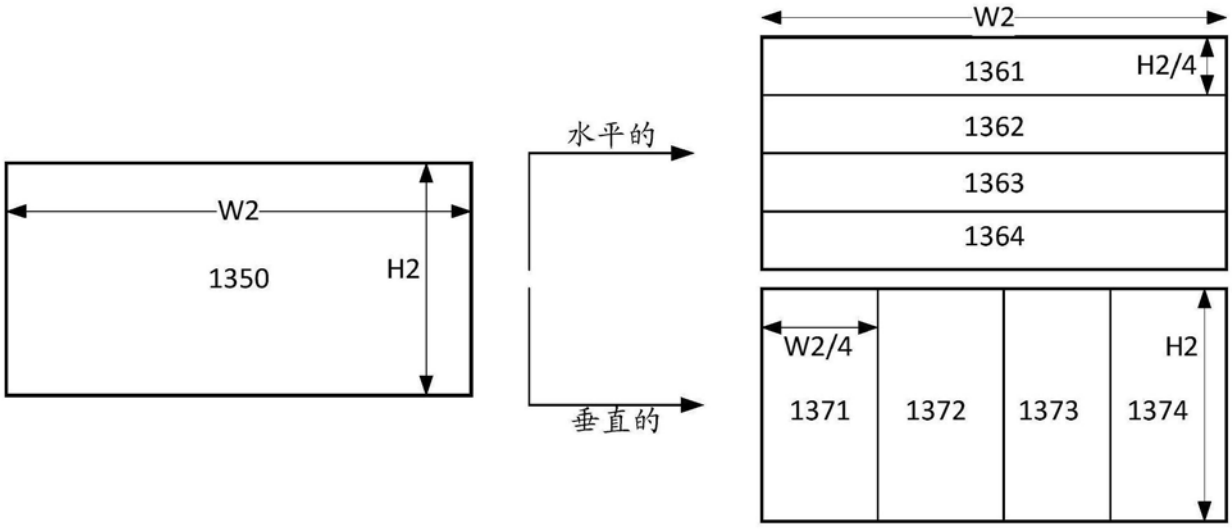

图13B

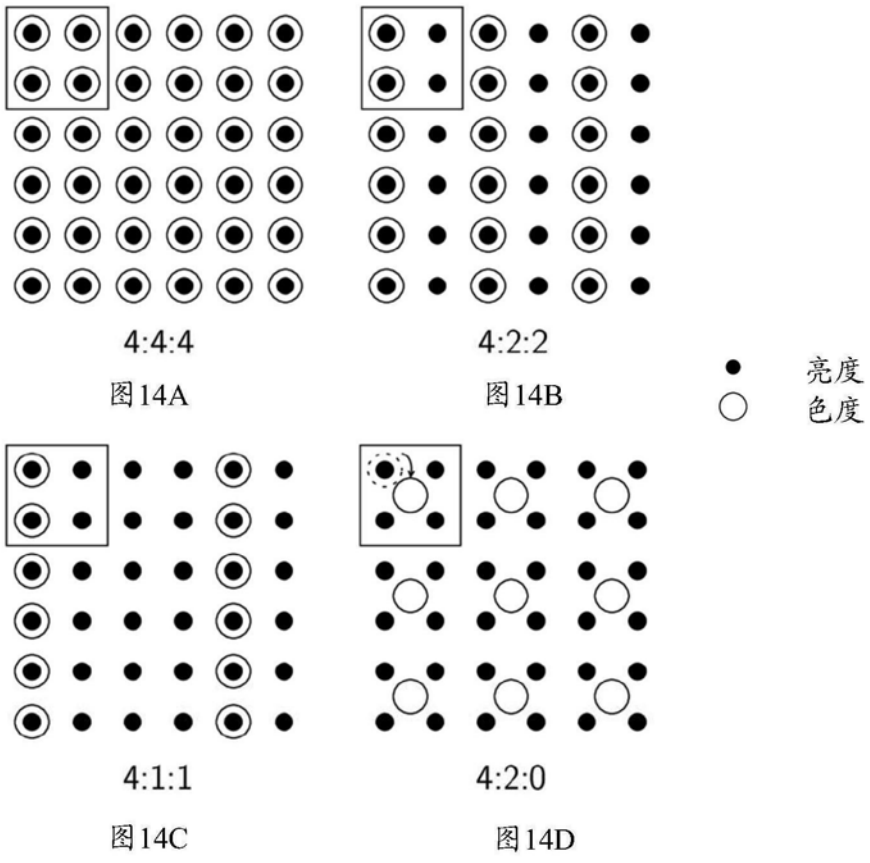

图14A 图14B 图14C 图14D

图14

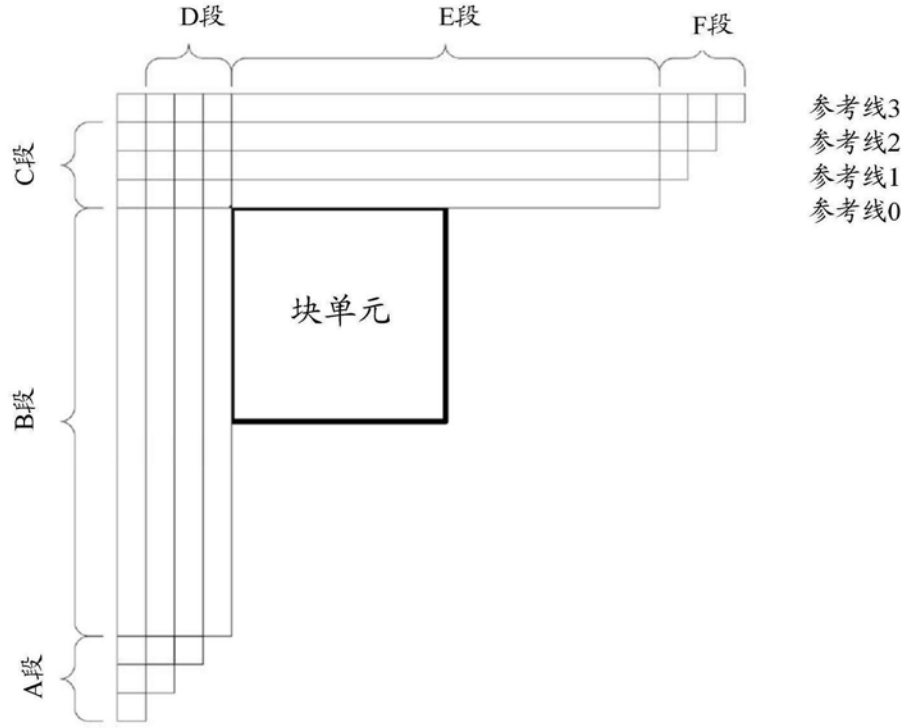
D段
E段
F段
C段
B段
A段
参考线3
参考线2
参考线1
参考线0
块单元

图15

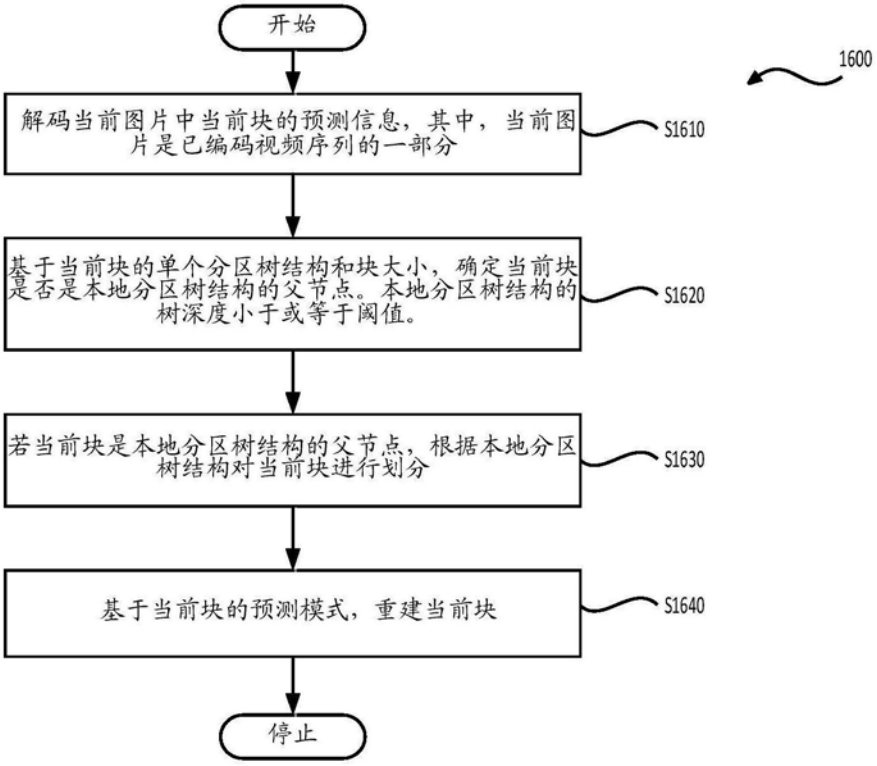
开始
1600
解码当前图片中当前块的预测信息，其中，当前图片是已编码视频序列的一部分
S1610
基于当前块的单个分区树结构和块大小，确定当前块是否是本地分区树结构的父节点。本地分区树结构的树深度小于或等于阈值。
S1620
若当前块是本地分区树结构的父节点，根据本地分区树结构对当前块进行划分
S1630
基于当前块的预测模式，重建当前块
S1640
停止

图16

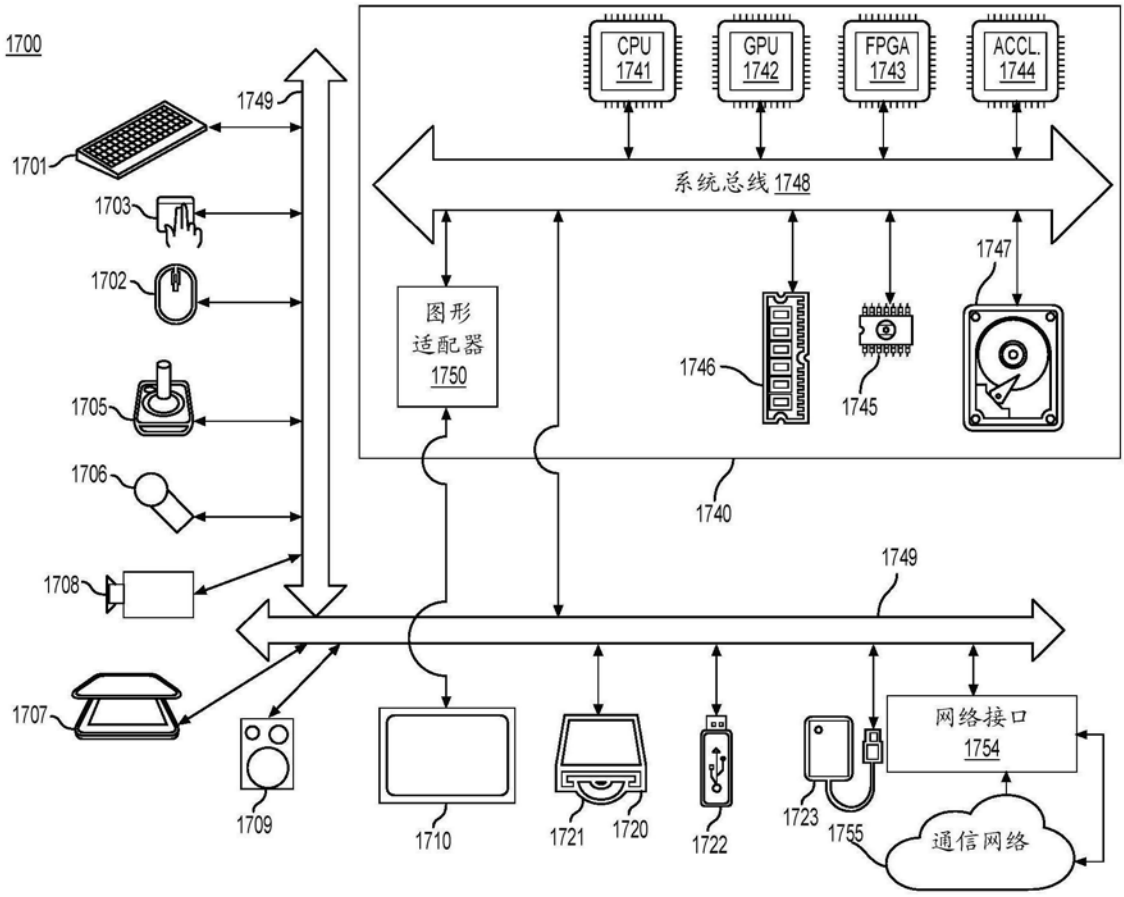
1700
1749
1701
1703
1702
1705
1706
1708
1707
1709
CPU 1741
GPU 1742
FPGA 1743
ACCL. 1744
系统总线 1748
图形适配器 1750
1747
1746
1745
1740
1749
1710
1721
1720
1722
1723
网络接口 1754
1755
通信网络

图17