(19)国家知识产权局

(12)发明专利

(10)授权公告号 CN 113545091 B
(45)授权公告日 2023.09.22

(21)申请号 202080018550.X

(22)申请日 2020.03.02

(65)同一申请的已公布的文献号
申请公布号 CN 113545091 A

(43)申请公布日 2021.10.22

(30)优先权数据
62/813,665 2019.03.04 US
16/804,547 2020.02.28 US

(85)PCT国际申请进入国家阶段日
2021.09.03

(86)PCT国际申请的申请数据
PCT/US2020/020607 2020.03.02

(87)PCT国际申请的公布数据
WO2020/180769 EN 2020.09.10

(73)专利权人 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
专利代理师 陈美娥 王琦

(51)Int.Cl.
H04N 19/70(2006.01)
H04N 19/44(2006.01)
H04N 19/96(2006.01)
H04N 19/119(2006.01)
H04N 19/174(2006.01)

(56)对比文件
US 2014219336 A1,2014.08.07
Y. Zhao, H. Yang, J. Chen.《CE6: Sub-block transform for inter blocks (CE6.1.2)》.《Joint Video Experts Team (JVET)of ITU-T SG 16 WP 3 and ISO/IEC JTC 1/SC 29/WG 11: JVET-L0358》.2018,全文.
Y. Zhao, H. Gao, H. Yang, J. Chen.《CE6: Sub-block transform for inter blocks (Test 6.4.1)》.《Joint Video Experts Team (JVET)of ITU-T SG 16 WP 3 and ISO/IEC JTC 1/SC 29/WG 11:JVET-M0140 》.2019,全文.

审查员 师佳彬

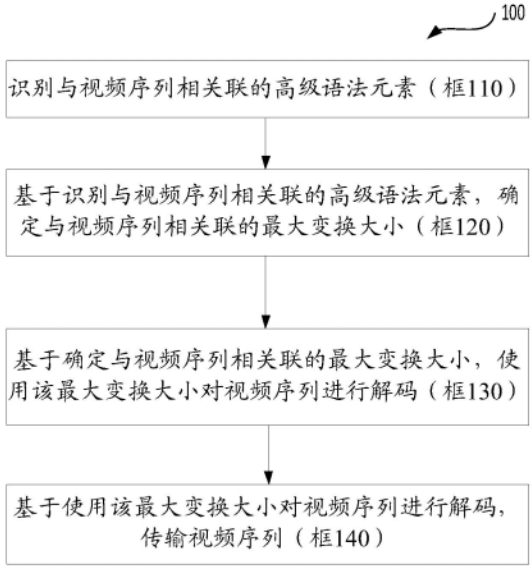
权利要求书1页 说明书22页 附图5页

(54)发明名称

对视频序列执行最大变换大小控制的解码方法和装置

(57)摘要

本公开实施例涉及对视频序列执行最大变换大小控制的方法和装置。一种对视频序列的解码执行最大变换大小控制的方法和装置,包括:解码器识别与所述视频序列相关联的高级语法元素。基于识别与所述视频序列相关联的所述高级语法元素,确定与所述视频序列相关联的最大变换大小。基于确定与所述视频序列相关联的所述最大变换大小,使用所述最大变换大小对所述视频序列进行解码。基于使用所述最大变换大小对所述视频序列进行解码,传输所述视频序列。



CN 113545091 B

1.一种对视频序列执行最大变换大小控制的解码方法,其特征在于,所述方法包括:

识别与所述视频序列相关联的高级语法元素;

基于识别与所述视频序列相关联的所述高级语法元素,确定与所述视频序列相关联的最大变换大小;

基于确定与所述视频序列相关联的所述最大变换大小,使用所述最大变换大小对所述视频序列进行解码;

当启用子块变换SBT模式时,允许所述SBT的实际最大编码单元CU宽度和高度导出为最大变换大小与高级语法中发信号通知的允许SBT的最大编码单元CU宽度和高度之间的最小值,其中高级语法中使用第一语法标识发信号通知允许SBT的最大CU宽度和高度,当所述第一语法标识等于0时,指示允许子块变换的最大CU宽度和高度为32,当所述第一语法标识等于1时,指示允许子块变换的最大CU宽度和高度为64。

2.根据权利要求1所述的方法,其特征在于,所述高级语法元素是视频参数集VPS、序列参数集SPS、图片参数集PPS、切片头、图块头、图块组头或编码树单元CTU头。

3.根据权利要求1所述的方法,其特征在于,所述最大变换大小对应于最大变换宽度和高度。

4.如权利要求1所述的方法,其特征在于,所述基于识别与所述视频序列相关联的所述高级语法元素,确定与所述视频序列相关联的最大变换大小包括:发信号通知最大变换单元区域大小。

5.一种对视频序列执行最大变换大小控制的设备,其特征在于,包括:

至少一个存储器,被配置为存储程序代码;

至少一个处理器,被配置为读取所述程序代码,并按照所述程序代码的指示执行如权利要求1-4任一项所述的方法。

6.一种视频解码装置,其特征在于,包括:

识别模块,被配置为识别与视频序列相关联的高级语法元素;

确定模块,被配置为基于识别与所述视频序列相关联的所述高级语法元素,确定与所述视频序列相关联的最大变换大小;

解码模块,被配置为基于确定与所述视频序列相关联的所述最大变换大小,使用所述最大变换大小对所述视频序列进行解码;

处理模块,被配置为当启用子块变换SBT模式时,允许所述SBT的实际最大编码单元CU宽度和高度导出为最大变换大小与高级语法中发信号通知的允许SBT的最大编码单元CU宽度和高度之间的最小值,其中高级语法中使用第一语法标识发信号通知允许SBT的最大CU宽度和高度,当所述第一语法标识等于0时,指示允许子块变换的最大CU宽度和高度为32,当所述第一语法标识等于1时,指示允许子块变换的最大CU宽度和高度为64。

7.一种非易失性计算机可读介质,其特征在于,用于存储指令,所述指令包括一个或多个指令,当所述一个或多个指令由对视频序列的解码执行最大变换大小控制的设备的一个或多个处理器执行时,使得所述一个或多个处理器执行权利要求1至4任一项所述的方法。

对视频序列执行最大变换大小控制的解码方法和装置

[0001] 相关申请的交叉引用

[0002] 本申请要求于2020年2月28日提交的、申请号为16/804,547、名称为“最大变换大小控制”的美国专利申请的优先权,其要求于2019年3月4日提交的、申请号为62/813,665、名称为“最大变换大小控制”的美国申请的优先权。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] 本公开是针对超越HEVC(高效视频编解码)的下一代视频编解码技术(例如通用视频编解码(VVC))而提出的。更具体地,提出了一种用于控制最大变换大小的方案,此外,还讨论了最大变换大小和变换分区方案之间的交互(例如,子块变换(SBT)和帧内子分区(ISP))。

背景技术

[0004] ITU-T VCEG(国际电信联盟(ITU)电信标准化部门(ITU-T)的视频编解码专家组(VCEG))(Q6/16)和ISO/IEC MPEG(JTC 1/SC 29/WG 11)(国际标准化组织(ISO)和国际电工委员会(IEC)联合技术委员会ISO/IEC JTC 1的标准化小组委员会)于2013年(第1版)、2014年(第2版)、2015年(第3版)和2016年(第4版)发布了H.265/HEVC(高效视频编解码)标准。从那时起,他们一直在研究对于压缩能力明显超过HEVC标准(包括其扩展)的未来视频编解码技术标准化的潜在需求。2017年10月,他们发布了针对能力超出HEVC(CfP)的视频压缩提案的联合征集。截至2018年2月15日,分别提交了针对标准动态范围(SDR)的共22个CfP回复、针对高动态范围(HDR)的12个CfP回复和针对360种视频类别的12个CfP回复。2018年4月,在122运动图象专家组(MPEG)/第十次联合视频开发组-联合视频专家组(JVET)会议上,对所有收到的CfP回复进行了评估。通过仔细的评估,JVET正式启动了超越HEVC的下一代视频编解码的标准化,即所谓的通用视频编解码(VVC)。VTM的当前版本(VVC测试模型),即VTM4。

发明内容

[0005] 根据本公开的一个方面,一种对视频序列的解码执行最大变换大小控制的方法包括:解码器识别与所述视频序列相关联的高级语法元素;基于识别与所述视频序列相关联的所述高级语法元素,所述解码器确定与所述视频序列相关联的最大变换大小;基于确定与所述视频序列相关联的所述最大变换大小,所述解码器使用所述最大变换大小对所述视频序列进行解码;以及基于使用所述最大变换大小对所述视频序列进行解码,所述解码器传输所述视频序列。

[0006] 根据本公开的一个方面,一种对视频序列的解码执行最大变换大小控制的设备包括:至少一个存储器,被配置为存储程序代码;至少一个处理器,被配置为读取所述程序代码,并按照所述程序代码的指示进行操作,所述程序代码包括:识别代码,被配置为使所述至少一个处理器识别与所述视频序列相关联的高级语法元素;确定代码,被配置为使所述

至少一个处理器基于识别与所述视频序列相关联的所述高级语法元素，确定与所述视频序列相关联的最大变换大小；解码代码，被配置为使所述至少一个处理器基于确定与所述视频序列相关联的所述最大变换大小，使用所述最大变换大小对所述视频序列进行解码；以及传输代码，被配置为使所述至少一个处理器基于使用所述最大变换大小对所述视频序列进行解码，传输所述视频序列。

[0007] 根据本公开的一个方面，一种非易失性计算机可读介质存储指令，所述指令包括一个或多个指令，当所述一个或多个指令由对视频序列的解码执行最大变换大小控制的设备的一个或多个处理器执行时，使得所述一个或多个处理器：识别与所述视频序列相关联的高级语法元素；基于识别与所述视频序列相关联的所述高级语法元素，确定与所述视频序列相关联的最大变换大小；基于确定与所述视频序列相关联的所述最大变换大小，使用所述最大变换大小对所述视频序列进行解码；以及基于使用所述最大变换大小对所述视频序列进行解码，传输所述视频序列。

附图说明

[0008] 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：

[0009] 图1是对视频序列的编码或解码执行最大变换大小控制的示例过程的流程图。

[0010] 图2是根据本公开实施例的通信系统的简化框图。

[0011] 图3是视频编码器和解码器在流式传输环境中的放置的示意图。

[0012] 图4是根据本公开实施例的视频解码器的功能框图。

[0013] 图5是根据本公开实施例的视频编码器的功能框图。

[0014] 图6是根据实施例的计算机系统的示意图。

[0015] 要解决的问题

[0016] 在最新的VVC草案中，最大TU大小是固定数字64，这意味着没有能力对最大TU大小执行控制。然而，由于最大TU大小对编码器实现的硬件复杂度有影响(例如，流水线(pipeline)中间缓冲器大小、乘法器的数量等)，因此可能需要控制VVC中的最大TU大小。

[0017] 在最新的VVC草案中，已经包括了SBT和ISP，并且需要处理SBT、ISP和最大TU大小之间的交互。例如，在SBT中，发信号通知SPS标志sps_sbt_max_size_64_flag，以指示最大SBT大小是32长度还是64长度。当sps_sbt_max_size_64_flag为真且最大TU大小为32点时，当前VVC草案无法处理，并且可能触发编码器崩溃。

[0018] 当前，对于所有CU大小都允许使用ISP模式，然而，当最大变换大小设置为小于64时，执行隐式变换分割还是使用具有信令的ISP执行显式变换分割就产生了冲突。例如，当最大变换大小为16时，对于64×16TU，在没有ISP的情况下，它应该被隐式地分割成四个16×16TU，然而，在有ISP的情况下，它可以使用垂直ISP分割来进行分区，这会导致相同的四个16×16TU，但是使用信令。

具体实施方式

[0019] 在HEVC中，使用表示为编码树的四叉树结构将编码树单元(CTU)分割成多个编码单元(CU)，以适应各种局部特征。在CU级别决定是使用帧间图片(时间)预测还是使用帧内

图片(空间)预测来编码图片区域。根据预测单元(PU)分割类型,每个CU可以进一步被分割成一个、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将相关信息发送到解码器。通过应用基于PU分割类型的预测过程而获得残差块之后,可以根据与CU的编码树类似的另一种四叉树结构,将CU划分为多个变换单元(TU)。HEVC结构的关键特征之一是它具有多个分区概念,包括CU、PU和TU。

[0020] 四叉树加二叉树(QTBT)结构消除了多个分区类型的概念,即,它消除了CU、PU和TU概念的分离,并且支持更灵活的CU分区形状。在QTBT块结构中,CU可以具有正方形或矩形形状。如图1所示,首先采用四叉树结构对编码树单元(CTU)进行分区。然后,进一步采用二叉树结构对四叉树叶节点进行分区。在二叉树分割中,存在对称水平分割和对称垂直分割两种分割类型。二叉树叶节点被称为编码单元(CU),并且该分割用于预测和变换处理而无需任何进一步的分区。这意味着,CU、PU和TU在QTBT编码块结构中具有相同的块大小。在JEM中,CU有时候由不同颜色分量的编码块(CB)组成,例如,在4:2:0色度格式的P切片和B切片的情况下,一个CU包含一个亮度CB和两个色度CB,并且CU有时候包含单个分量的CB,例如,在I切片的情况下,一个CU仅包含一个亮度CB或仅包含两个色度CB。

[0021] 为QTBT分区方案定义了以下参数:

[0022] CTU大小:四叉树的根节点大小,与HEVC中的概念相同;

[0023] MinQTSize:允许的最小四叉树叶节点大小;

[0024] MaxBTSize:允许的最大二叉树根节点大小;

[0025] MaxBTDepth:允许的最大二叉树深度;

[0026] MinBTSize:允许的最小二叉树叶节点大小。

[0027] 在QTBT分区结构的一个示例中,将CTU大小设置为具有两个相应的64×64色度样本块的128×128亮度样本。将MinQTSize设置为16×16,将MaxBTSize设置为64×64,将MinBTSize(对于宽度和高度)设置为4×4,将MaxBTDepth设置为4。四叉树分区首先应用于CTU,以生成四叉树叶节点。四叉树叶节点可以具有从16×16(即MinQTSize)到128×128(即CTU大小)的大小。如果四叉树叶节点是128×128,则由于其大小超过MaxBTSize(即64×64),因此将不会采用二叉树对其进一步分割。否则,四叉树叶节点会采用二叉树进一步分割。因此,四叉树叶节点也是二叉树的根节点,并且其二叉树深度为0。当二叉树深度达到MaxBTDepth(即4)时,不考虑进一步的分割。当二叉树节点的宽度等于MinBTSize(即4)时,不考虑进一步的水平分割。类似地,当二叉树节点的高度等于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二叉树的叶节点,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。

[0028] 在使用QTBT的块分区中,关于二叉树的分割(即,非叶)节点,发信号通知一个标志,以指示使用哪种分割类型(即,水平或垂直),其中0表示水平分割,1表示垂直分割。对于四叉树分割,不需要指示分割类型,因为四叉树分割始终水平和垂直地分割块,以产生具有相等大小的4个子块。

[0029] 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切片和B切片,一个CTU中的亮度和色度CTB共用相同的QTBT结构。然而,对于I切片,亮度CTB通过QTBT结构划分为多个CU,色度CTB通过另一个QTBT结构划分为多个色度CU。这意味着,I切片中的CU由亮度分量的编码块或两个色度分量的编码块组成,而P切片或B切片中的CU由所有

三个颜色分量的编码块组成。

[0030] 在HEVC中,对小块的帧间预测进行限制,以减少运动补偿的存储器访问,这使得4×8块和8×4块不支持双向预测,而4×4块不支持帧间预测。在JEM-7.0中实施的QTBT中,消除了这些限制。

[0031] 提出了一种多类型树(MTT)结构。MTT是比QTBT更灵活的树结构。在MTT中,除了四叉树和二叉树之外,还引入了水平和垂直中心-侧(center-side)三叉树。三叉树分区的主要优点是:补充四叉树分区和二叉树分区;三叉树分区能够捕获位于块中心的对象,而四叉树和二叉树始终沿着块中心进行分割。所提出的三叉树的分区的宽度和高度始终为2,因此不需要额外的变换。

[0032] 两级树的设计主要是为了降低复杂度。理论上,遍历树的复杂度为T^D,其中,T表示分割类型的数量,D是树的深度。

[0033] 在HEVC中,主变换是4点、8点、16点和32点DCT-2,并且变换核矩阵用8位整数来表示,即8位变换核。较小的DCT-2的变换核矩阵是较大的DCT-2的一部分。

[0034] DCT-2核显示出对称/反对称特性,因此支持所谓的“部分蝶形”实施方式,以减少运算计数(乘法、加法/减法、移位)的数量,并且使用部分蝶形可获得相同的矩阵乘法结果。

[0035] 在当前的VVC中,除了与HEVC相同的4点、8点、16点和32点DCT-2变换之外,还包括附加的2点和64点DCT-2。

[0036] 除了已经在HEVC中使用的DCT-2和4×4DST-7之外,自适应多变换(AMT,或称为增强多变换(EMT),或称为多变换选择(MTS))方案已经在VVC中用于帧间和帧内编码块的残差编码。它使用从DCT/DST族中选定的、除HEVC中的当前变换之外的多个变换。新引入的变换矩阵是DST-7、DCT-8。

[0037] VVC中的所有主变换矩阵都使用8位表示。AMT应用于宽度和高度都小于或等于32的CU,并且是否应用AMT由名为mts_flag的标志控制。当mts_flag等于0时,仅采用DCT-2对残差进行编码。当mts_flag等于1时,使用2个二进制数(bin)进一步发信号通知索引mts_idx,以指定要使用的水平和垂直变换。

[0038] 帧内子分区(ISP)编码模式根据块大小的维度将亮度帧内预测块垂直或水平划分为2或4个子分区。

[0039] 对于这些子分区中的每一个,通过对编码器发送的系数进行熵解码然后对其进行逆量化和逆变换,生成残差信号。然后,对子分区进行帧内预测,最后通过将残差信号添加到预测信号得到相应的重建样本。因此,每个子分区的重建值将可用于生成下一个子分区的预测,这将重复该过程等。所有子分区共享相同的帧内模式。

[0040] ISP算法将仅使用作为MPM列表的一部分的帧内模式进行测试。因此,如果块使用ISP,则MPM标志将被推断为1。此外,如果ISP用于某个块,则MPM列表将被修改,以排除DC模式,并对用于ISP水平分割的水平帧内模式和用于垂直分割的垂直帧内模式进行优先级排序。

[0041] 在ISP中,因为对每个子分区单独执行变换和重建,所以每个子分区都可以视为一个子TU。

[0042]

coding_unit(x0, y0, cbWidth, cbHeight, treeType) {	描述符
if(tile_group_type != I || sps_ibc_enabled_flag) {	
if(treeType != DUAL_TREE_CHROMA)	
cu_skip_flag[x0][y0]	ae(v)
if(cu_skip_flag[x0][y0] = = 0 && tile_group_type != I)	
pred_mode_flag	ae(v)
if(((tile_group_type = = I && cu_skip_flag[x0][y0] = =0) || (tile_group_type != I && CuPredMode[x0][y0] != MODE_INTRA)) && sps_ibc_enabled_flag)	
pred_mode_ibc_flag	ae(v)
}	
if(CuPredMode[x0][y0] = = MODE_INTRA) {	
if(sps_pcm_enabled_flag && cbWidth >= MinIpcmCbSizeY && cbWidth <= MaxIpcmCbSizeY && cbHeight >= MinIpcmCbSizeY && cbHeight <= MaxIpcmCbSizeY)	
pcm_flag[x0][y0]	ae(v)
if(pcm_flag[x0][y0]) {	
while(!byte_aligned( ))	
pcm_alignment_zero_bit	f(1)
pcm_sample(cbWidth, cbHeight, treeType)	
} else {	
if(treeType = = SINGLE_TREE || treeType = = DUAL_TREE_LUMA) {	
if((y0 % CtbSizeY) > 0)	
intra_luma_ref_idx[x0][y0]	ae(v)
if (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 && cbWidth <= MaxTbSizeY && cbHeight <= MaxTbSizeY)	
intra_subpartitions_split_flag[x0][y0]	ae(v)
if(intra_luma_ref_idx[x0][y0] = = 0 && intra_subpartitions_mode_flag[x0][y0] = = 0)	
intra_luma_mpm_flag[x0][y0]	ae(v)
if(intra_luma_mpm_flag[x0][y0])	
intra_luma_mpm_idx[x0][y0]	ae(v)
else	
intra_luma_mpm_remainder[x0][y0]	ae(v)
}	
if(treeType = = SINGLE_TREE || treeType = = DUAL_TREE_CHROMA)	
intra_chroma_pred_mode[x0][y0]	ae(v)
}	
} else if(treeType != DUAL_TREE_CHROMA) { /* MODE_INTER or MODE_IBC */	
……	

[0043]

}	

[0044] 在JVET-J0024、JVET-K0139和JVET-L0358中，提出了一种空间变化变换(SVT)方案。使用SVT，对于帧间预测残差，编码块中只有残差块，但是残差块小于编码块，因此SVT中的变换大小小于编码块大小。对于未被残差块或变换覆盖的区域，假设残差为零。

[0045] 在VVC顶部使用建议的SBT如下所示,添加的文本以灰色突出显示。可以看出,SBT方法要求发信号通知额外的开销位(cu_sbt_flag、cu_sbt_quad_flag、cu_sbt_horizontal_flag和cu_sbt_pos_flag),以指示子块类型(水平或垂直)、大小(一半或四分之一)和位置(左或右、上或下)。

[0046]

seq_parameter_set_rbsp( ) {	描述符
sps_seq_parameter_set_id	ue(v)
…	
sps_mts_intra_enabled_flag	u(1)
sps_mts_inter_enabled_flag	u(1)
sps_sbt_enable_flag	u(1)

[0047]

rbsp_trailing_bits( )	
}	
slice_header( ) {	Descriptor
slice_pic_parameter_set_id	ue(v)
slice_address	u(v)
slice_type	ue(v)
if (slice_type != I) {	
log2_diff_ctu_max_bt_size	ue(v)
if(sps_sbtmvp_enabled_flag) {	
sbtmvp_size_override_flag	u(1)
if(sbtmvp_size_override_flag)	
log2_sbtmvp_active_size_minus2	u(3)
}	
if(sps_temporal_mvp_enabled_flag)	
slice_temporal_mvp_enabled_flag	u(1)
if(slice_type == B)	
mvd_l1_zero_flag	u(1)
if(slice_temporal_mvp_enabled_flag) {	
if(slice_type == B)	
collocated_from_l0_flag	u(1)
}	
six_minus_max_num_merge_cand	ue(v)
if(sps_sbt_enable_flag)	
slice_max_sbt_size_64_flag	u(1)
}	
if (sps_alf_enabled_flag) {	
slice_alf_enabled_flag	u(1)
if(slice_alf_enabled_flag)	
alf_data( )	
}	
dep_quant_enabled_flag	u(1)
if(!dep_quant_enabled_flag)	
sign_data_hiding_enabled_flag	u(1)
byte_alignment( )	
}	
coding_unit(x0, y0, cbWidth, cbHeight, treeType) {	Descriptor
…	
if(CuPredMode[x0][y0] != MODE_INTRA && cu_skip_flag[x0][y0] == 0)	
cu_cbf	ae(v)
if(cu_cbf) {	
if(CuPredMode[x0][y0] != MODE_INTRA && sps_sbt_enable_flag) {	
if(cbWidth <= maxSbtSize && cbHeight <= maxSbtSize) {	
allowSbtVerHalf = cbWidth >= 8	
allowSbtVerQuad = cbWidth >= 16	

allowSbtHorHalf = cbHeight >= 8	
allowSbtHorQuad = cbHeight >= 16	
if(allowSbtVerHalf || allowSbtHorHalf || allowSbtVerQuad || allowSbtHorQuad)	
cu_sbt_flag[x0][y0]	ae(v)
}	
if(cu_sbt_flag[x0][y0]) {	
if((allowSbtVerHalf || allowSbtHorHalf) && (allowSbtVerQuad || allowSbtHorQuad))	
cu_sbt_quad_flag[x0][y0]	ae(v)
if((cu_sbt_quad_flag[x0][y0] && allowSbtVerQuad && allowSbtHorQuad) || (!cu_sbt_quad_flag[x0][y0] && allowSbtVerHalf && allowSbtHorHalf))	
cu_sbt_horizontal_flag[x0][y0]	ae(v)
cu_sbt_pos_flag[x0][y0]	ae(v)
}	
}	
transform_tree(x0, y0, cbWidth, cbHeight, treeType)	
}	
}	
transform_tree(x0, y0, tbWidth, tbHeight , treeType) {	Descriptor
if(tbWidth > MaxTbSizeY || tbHeight > MaxTbSizeY) {	
trafoWidth = (tbWidth > MaxTbSizeY) ? (tbWidth / 2) : tbWidth	
trafoHeight = (tbHeight > MaxTbSizeY) ? (tbHeight / 2) : tbHeight	
transform_tree(x0, y0, trafoWidth, trafoHeight)	
if(tbWidth > MaxTbSizeY)	
transform_tree(x0 + trafoWidth, y0, trafoWidth, trafoHeight, treeType)	
if(tbHeight > MaxTbSizeY)	
transform_tree(x0, y0 + trafoHeight, trafoWidth, trafoHeight, treeType)	
if(tbWidth > MaxTbSizeY && tbHeight > MaxTbSizeY)	
transform_tree(x0 + trafoWidth, y0 + trafoHeight, trafoWidth, trafoHeight, treeType)	
} else if(cu_sbt_flag[x0][y0])	
factorTb0 = cu_sbt_quad_flag[x0][y0] ? 1 : 2	
factorTb0 = cu_sbt_pos_flag[x0][y0] ? (4 - factorTb0) : factorTb0	
noResiTb0 = cu_sbt_pos_flag[x0][y0] ? 1 : 0	
if(!cu_sbt_horizontal_flag[x0][y0]) {	
trafoWidth = tbWidth * factorTb0 / 4	
transform_tree(x0, y0, trafoWidth, tbHeight, treeType , noResiTb0)	
transform_tree(x0 + trafoWidth, y0, tbWidth − trafoWidth, tbHeight, treeType , !noResiTb0)	
}	
else {	
trafoHeight = tbHeight * factorTb0 / 4	
transform_tree(x0, y0, tbWidth, trafoHeight, treeType , noResiTb0)	

[0048]

transform_tree(x0, y0 + trafoHeight, tbWidth, tbHeight − trafoHeight, treeType , !noResiTb0)	
}	
} else {	
transform_unit(x0, y0, tbWidth, tbHeight, treeType , 0)	
}	
}	

[0049]

transform_unit(x0, y0, tbWidth, tbHeight, treeType , noResi) {	描述符
if((treeType = = SINGLE_TREE || treeType = = DUAL_TREE_LUMA) && !noResi)	
tu_cbf_luma[x0][y0]	ae(v)
if((treeType = = SINGLE_TREE || treeType = = DUAL_TREE_CHROMA) && !noResi) {	
tu_cbf_cb[x0][y0]	ae(v)
tu_cbf_cr[x0][y0]	ae(v)
}	
if((((CuPredMode[x0][y0] = = MODE_INTRA) && sps_mts_intra_enabled_flag) || ((CuPredMode[x0][y0] = = MODE_INTER) && sps_mts_inter_enabled_flag)) && tu_cbf_luma[x0][y0] && treeType != DUAL_TREE_CHROMA && (tbWidth <= 32) && (tbHeight <= 32) && !cu_sbt_flag[x0][y0])	
cu_mts_flag[x0][y0]	ae(v)
if(tu_cbf_luma[x0][y0])	
residual_coding(x0, y0, log2(tbWidth), log2(tbHeight), 0)	
if(tu_cbf_cb[x0][y0])	
residual_coding(x0, y0, log2(tbWidth / 2), log2(tbHeight / 2), 1)	
if(tu_cbf_cr[x0][y0])	
residual_coding(x0, y0, log2(tbWidth / 2), log2(tbHeight / 2), 2)	
}	

[0050] “sps_sbt_enabled_flag”等于0指定对于帧间预测CU禁用子块变换。sps_sbt_enabled_flag等于1指定对于帧间预测CU启用子块变换。

[0051] slice_sbt_max_size_64_flag等于0指定允许子块变换的最大CU宽度和高度为32。slice_sbt_max_size_64_flag等于1指定允许子块变换的最大CU宽度和高度为64。

[0052] maxSbtSize＝slice_sbt_max_size_64_flag?64:32

[0053] cu_sbt_flag[x0][y0]等于1指定,对于当前编码单元,使用子块变换。cu_sbt_flag[x0][y0]等于0指定,对于当前编码单元,不使用子块变换。

[0054] 当cu_sbt_flag[x0][y0]不存在时,推断其值等于0。

[0055] 当使用子块变换时,将编码单元平铺成两个变换单元,一个变换单元具有残差,另一个变换单元不具有残差。

[0056] cu_sbt_quad_flag[x0][y0]等于1指定,对于当前编码单元,子块变换包括当前编码单元的1/4大小的变换单元。cu_sbt_quad_flag[x0][y0]等于0指定,对于当前编码单元,子块变换包括当前编码单元的1/2大小的变换单元。

[0057] 当cu_sbt_quad_flag[x0][y0]不存在时,推断其值等于0。

[0058] cu_sbt_horizontal_flag[x0][y0]等于1指定通过水平分割将当前编码单元平铺

成2个变换单元。cu_sbt_horizontal_flag[x0][y0]等于0指定通过垂直分割将当前编码单元平铺成2个变换单元。

[0059] 当cu_sbt_horizontal_flag[x0][y0]不存在时,其值导出如下:

[0060] 如果cu_sbt_quad_flag[x0][y0]等于1,则将cu_sbt_horizontal_flag[x0][y0]设置为等于allowSbtHoriQuad。

[0061] 否则(cu_sbt_quad_flag[x0][y0]等于0),将cu_sbt_horizontal_flag[x0][y0]设置为等于allowSbtHoriHalf。

[0062] cu_sbt_pos_flag[x0][y0]等于1指定,当前编码单元中的第一变换单元的tu_cbf_luma、tu_cbf_cb和tu_cbf_cr不存在于比特流中。cu_sbt_pos_flag[x0][y0]等于0指定,当前编码单元中的第二变换单元的tu_cbf_luma、tu_cbf_cb和tu_cbf_cr不存在于比特流中。

[0063] 该过程的输入为:

[0064] 亮度位置(xTbY,yTbY),其指定相对于当前图片的左上亮度样本的当前亮度变换块的左上样本,

[0065] 指定当前变换块的宽度的变量nTbW,

[0066] 指定当前变换块的高度的变量nTbH,

[0067] 指定当前块的颜色分量的变量cIdx,

[0068] 缩放的变换系数的(nTbW)x(nTbH)数组d[x][y],其中x=0..nTbW-1,y=0..nTbH-1。

[0069] 该过程的输出是残差样本的(nTbW)x(nTbH)数组r[x][y],其中x=0..nTbW-1,y=0..nTbH-1。

[0070] 如果cu_sbt_flag[xTbY][yTbY]等于1,则根据cu_sbt_horizontal_flag[xTbY][yTbY]和cu_sbt_pos_flag[xTbY][yTbY],在表8-X中导出指定水平变换核的变量trTypeHor和指定垂直变换核的变量trTypeVer。

[0071] 否则(cu_sbt_flag[xTbY][yTbY]等于0),则根据mts_idx[xTbY][yTbY]和CuPredMode[xTbY][yTbY],在表8-9中导出指定水平变换核的变量trTypeHor和指定垂直变换核的变量trTypeVer。

[0072] 残差样本的(nTbW)x(nTbH)数组r导出如下:

[0073] 通过为每列x=0..nTbW-1(其具有变换块的高度nTbH)调用一维变换过程,将缩放的变换系数d[x][y](x=0..nTbW-1,y=0..nTbH-1)的每一(垂直)列变换为e[x][y](x=0..nTbW-1,y=0..nTbH-1)。y=0..nTbH-1的列表d[x][y]和设置为等于trTypeVer的变换类型变量trType作为输入,输出是y=0..nTbH-1的列表e[x][y]。

[0074] x=0..nTbW-1,y=0..nTbH-1的中间样本值g[x][y]导出如下:

[0075] g[x][y]=Clip3(CoeffMin,CoeffMax,(e[x][y]+256)>>9)

[0076] 通过为每行y=0..nTbH-1(其具有变换块的宽度nTbW)调用一维变换过程,将所得数组g[x][y](x=0..nTbW-1,y=0..nTbH-1)的每一(水平)行变换为r[x][y](x=0..nTbW-1,y=0..nTbH-1)。x=0..nTbW-1的列表g[x][y]和设置为等于trTypeHor的变换类型变量trType作为输入,输出是x=0..nTbW-1的列表r[x][y]。

[0077]

	cu_sbt_horizontal_flag [xTbY][yTbY] = = 0		cu_sbt_horizontal_flag [xTbY][yTbY] = = 1	
	cu_sbt_pos_flag [xTbY][yTbY] = = 0	cu_sbt_pos_flag [xTbY][yTbY] = = 1	cu_sbt_pos_flag [xTbY][yTbY] = = 0	cu_sbt_pos_flag [xTbY][yTbY] = = 1
trTypeHor	2	1	nTbW > 32 ? 0 : 1	nTbW > 32 ? 0 : 1
trTypeVer	nTbH > 32 ? 0 : 1	nTbH > 32 ? 0 : 1	2	1

mts_idx[xTbY][yTbY]	CuPredMode[xTbY][yTbY] = = MODE_INTRA		CuPredMode[xTbY][xTbY] = = MODE_INTER	
	trTypeHor	trTypeVer	trTypeHor	trTypeVer
−1 (inferred)	0	0	0	0
0 (00)	1	1	2	2
1 (01)	2	1	1	2
2 (10)	1	2	2	1
3 (11)	2	2	1	1

[0078] 存在不同的YUV格式。对于4:2:0格式，LM预测应用六抽头插值滤波器，以获得对应于图6所示的色度样本的下采样亮度样本。以公式的方式，根据重建亮度样本计算下采样亮度样本Rec'L[x,y]：

[0079] $Rec'_L[x,y] = (2 \times Rec_L[2x,2y] + 2 \times Rec_L[2x,2y+1] +$

[0080] $Rec_L[2x-1,2y] + Rec_L[2x+1,2y] +$

[0081] $Rec_L[2x-1,2y+1] + Rec_L[2x+1,2y+1] + 4) >> 3$

[0082] 图1是对视频序列的编码或解码执行最大变换大小控制的方法的示例过程100的流程图。在一些实施方式中，图1的一个或多个过程框可以由解码器执行。在一些实施方式中，图1的一个或多个过程框可以由与解码器分离或包括解码器的另一设备或一组设备来执行，例如编码器。

[0083] 如图1所示，过程100可以包括：解码器识别与视频序列相关联的高级语法元素(框110)。

[0084] 如图1中进一步所示，过程100可以包括：解码器基于识别与视频序列相关联的高级语法元素，确定与视频序列相关联的最大变换大小(框120)。

[0085] 如图1中进一步所示，过程100可以包括：解码器基于确定与视频序列相关联的最大变换大小，使用该最大变换大小对视频序列进行解码(框130)。

[0086] 如图1中进一步所示，过程100可以包括：解码器基于使用该最大变换大小对视频序列进行解码，传输视频序列(框140)。

[0087] 如本文所使用的，高级语法元素可以指视频参数集(VPS)、序列参数集(SPS)、图片参数集(PPS)、切片头、图块头、图块组头等中的任何一个。此外，CTU(编码树单元，其是最大的CU大小)头可以指为每个CTU发信号通知的语法元素，例如作为头信息。此外，“变换大小”可以指最大变换宽度和/或高度，或最大变换单元区域大小。

[0088] 根据实施例，在高级语法元素或CTU头中发信号通知最大变换大小。最小变换大小不发信号通知，而是设置为默认值。最小变换大小的示例值包括4长度、8长度和16长度。在

一个实施例中，限制最大变换大小必须是几个预定义值中的一个值。预定义值的示例包括16长度、32长度和64长度。

[0089] 在一个实施例中，发信号通知最大变换大小减去常数的对数值。例如，将支持的最小的最大变换大小设置为16，将最大变换大小发信号通知为log2_max_transform_size_minus_4，即，如果最大变换大小是64，那么发信号通知值2，并且对于最大变换大小32，发信号通知值1。在另一示例中，将可能的最小的最大变换大小设置为32，对于最大变换大小32和64，分别发信号通知值0和1。

[0090] 根据实施例，在高级语法元素或CTU头中仅发信号通知最大变换单元区域大小。最小变换单元区域大小不发信号通知，而是设置为默认值，示例值包括16个样本、32个样本、64个样本。在实施例中，限制最大变换单元区域大小至少应为默认值。默认值的示例包括64个样本、128个样本、256个样本、512个样本、1024个样本、2048个样本或4096个样本。在实施例中，发信号通知最大变换单元区域大小除以可能的最小的最大变换单元区域大小的对数值。例如，将可能的最小的最大变换单元区域大小设置为256，发信号通知最大变换大小除以256的对数值，即，如果最大变换单元区域大小分别是256、512、1024、2048和4096，则发信号通知0、1、2、3、4。

[0091] 根据实施例，根据发信号通知的最大变换大小或变换单元大小来约束最大SBT大小。

[0092]

if(sps_sbt_enable_flag && sps_max_transform_size == 64)	
slice_max_sbt_size_64_flag	u(1)

[0093] 在一个实施例中，当最大变换大小小于64时，不发信号通知sps_sbt_max_size_64_flag，而是导出为默认值，指示允许SBT的最大CU宽度和高度是32个亮度样本。

[0094] 在另一实施例中，允许SBT的实际最大CU宽度和高度由指示最大变换大小的高级语法元素进一步调整，并且允许SBT的实际最大CU宽度和高度导出为最大变换大小与高级语法中发信号通知的允许SBT的最大CU宽度和高度之间的最小值。例如，如果最大变换大小为16，则不论slice_max_sbt_size_64_flag发信号通知为0(允许SBT的最大CU宽度和高度为32)还是1(允许SBT的最大CU宽度和高度为64)，允许SBT的最大CU宽度和高度都是16。VVC规范文本的建议变更描述如下：

[0095] slice_sbt_max_size_64_flag等于0指定允许子块变换的最大CU宽度和高度为32。slice_sbt_max_size_64_flag等于1指定允许子块变换的最大CU宽度和高度为64。

[0096] maxSbtSize＝min(max_transform_size,slice_sbt_max_size_64_flag?64:32)，其中max_transform_size定义最大变换大小。

[0097] 建议仅当CU宽度和高度都不大于最大TU大小时才应用SBT。

[0098] 不是像在SBT中那样指示哪个变换单元具有非零系数，而是建议首先发信号通知编码单元的哪个子部分(例如使用SBT分区，即左半部分、右半部分、上半部分、下半部分、左四分之一部分、右四分之一部分、上四分之一部分或下四分之一部分)具有至少一个非零系数，而编码单元的每个部分可以包括一个或多个TU。

[0099] 在一个实施例中，如果子部分大小大于最大TU大小，则将其分割为多个TU，并且每个TU不大于最大TU大小。例如，如果最大TU大小为16长度并且确定64×32CU只有右半(32×32)子部分与非零系数相关联，而左半(32×32)子部分只与零系数相关联，在这种情况下，

将右半部分进一步分割为四个16×16TU,并且每个16×16TU可以具有非零系数。64×32CU仅包括与非零系数相关联的右半部分(纹理子部分),并且右半32×32子部分被进一步分割为四个16×16TU。

[0100] 在一个实施例中,当子部分大小大于最大TU大小时,将其分割为多个TU,每个TU不大于最大TU大小,并且如果除了编码顺序中的最后TU之外的子部分的所有TU都用零CBF进行编码,则不发信号通知该子部分的最后TU的CBF,而是使用非零CBF值导出该子部分的最后TU的CBF。

[0101] 根据实施例,不采用SBT中使用的四分之一分区(其将CU水平或垂直划分为3:1或1:3子部分),建议当CU宽度(W)大于高度(H)时,不是使用垂直分割的3:1或1:3,而是将CU划分为一个左HxH和一个右(W-H)xH子部分,或一个右HxH和一个左(W-H)xH子部分。当W大于4*H时,建议的分区不同于3:1和1:3分区。类似地,当CU宽度(W)小于高度(H)时,则不是使用水平分割的3:1或1:3,而是将CU划分为一个上HxH和一个下(W-H)xH子部分,或一个下HxH和一个上(W-H)xH子部分。当H大于4*W时,建议的分区不同于3:1和1:3分区。

[0102] 在一个示例中,对于128×16CU,不是划分为一个96×16零TU和一个32×16非零TU,而是建议划分为一个112×16零TU和一个16×16非零TU。在另一示例中,对于128×8CU,不是划分为一个96×8零TU和一个32×8非零TU,而是建议划分为一个120×8零TU和一个8×8非零TU。

[0103] 在一个实施例中,如果CU大小为W×H,并且采用水平1:3或3:1分区,如果0.25W大于最大TU大小(max_transform_size),则不是将CU分割为一个0.25W×H TU和一个0.75W×H TU,而是将CU分割为一个max_transform_size×H TU和一个(W-max_transform_size)×H TU。

[0104] 在一个示例中,对于128×8CU,如果最大TU大小为16,则不是划分为一个96×8零TU和一个32×8非零TU,而是建议划分为一个112×8零TU和一个16×8非零TU。

[0105] 在一个实施例中,如果CU大小为W×H,并且采用垂直1:3或3:1分区,如果0.25H大于最大TU大小(max_transform_size),则不是将CU分割为一个Wx0.25H TU和一个Wx0.75H TU,而是将CU分割为一个max_transform_size×H TU和一个(W-max_transform_size)×H TU。

[0106] 不像当前SBT设计中的高级语法slice_max_sbt_size_64_flag那样发信号通知允许SBT的最大CU宽度和高度,不发信号通知SBT允许的最大变换大小。

[0107] 在一个实施例中,即使当当前CU宽度或高度大于允许SBT的最大CU宽度和高度时,只要所得到的非零子TU的宽度和高度不大于最大TU宽度和高度,就允许SBT分区并且可以发信号通知。例如,如果最大TU大小(宽度和高度)是32点,当前CU是64×32,则允许将当前CU垂直划分为两个32×32TU,或一个非零16×32TU和一个零48×32TU,但是,不允许水平分割当前CU,因为所得到的变换宽度是64,超过了最大变换大小。

[0108] SBT分区和方向的可用性可以基于所得到的非零子TU是否满足最大TU的约束。如果不可用,则不发信号通知相关标志而是推断该相关标志。在一个示例中,CU是64×16,并且最大变换大小是16。在这种情况下,半SBT分割不可用,因此不发信号通知cu_sbt_quad_flag而是推断cu_sbt_quad_flag为真。此外,水平分割SBT也不可用,因此不发信号通知cu_sbt_horizontal_flag而是推断cu_sbt_horizontal_flag为假。

[0109] 建议不允许并且不发信号通知任何导致高度或宽度大于最大TU大小的非零TU的SBT分区。

[0110] 当前，对于所有CU大小都允许使用ISP模式，但是，当最大变换大小设置为小于64时，进行隐式变换分割还是使用具有信令的ISP进行显式变换分割来解决该问题就产生了冲突。建议限制允许ISP的最大CU大小，以便ISP仅应用于没有隐式变换分割的CU。

[0111] 在一个实施例中，语法表的修改变化如下所述。

[0112]

if (intra_luma_ref_idx[x0][y0] = = 0 && (cbWidth <= MaxTbSizeY 卄 && cbHeight <= MaxTbSizeY) && (cbWidth * cbHeight > MinTbSizeY * MinTbSizeY))	
intra_subpartitions_mode_flag[x0][y0]	ae(v)
if(intra_subpartitions_mode_flag[x0][y0] = = 1 && cbWidth <= MaxTbSizeY && cbHeight <= MaxTbSizeY)	
intra_subpartitions_split_flag[x0][y0]	ae(v)

[0113] 当CU大小大于最大TU大小时，在使用隐式变换分割将CU分割成多个TU之后，可以对每个TU(大小为w×h)进一步应用ISP，以指示是否将TU进一步分割成多个0.5w×h、或w×0.5h、或0.25w×h、或w×0.25h的较小TU。

[0114] 在一个示例中，如果CU大小是64×16，并且最大TU大小是16，则首先将CU隐式地分割成四个16×16TU，然后对于每个16×16TU，应用ISP，以发信号通知是否将其进一步分割成四个4×16或四个16×4的较小TU。

[0115] 建议针对不同颜色分量分别发信号通知最大变换大小。

[0116] 在一个实施例中，针对亮度分量发信号通知一个最大变换大小，并且针对色度分量发信号通知一个最大变换大小。

[0117] 建议仅发信号通知一个颜色分量的最大变换大小，其它颜色分量的最大变换大小隐式地导出。

[0118] 在一个实施例中，针对第一分量发信号通知一个最大变换大小，针对另一分量，根据关于第一分量的下采样率，相应地调整最大水平变换和/或垂直变换大小。

[0119] 在一个示例中，如果当前颜色分量的样本在水平(和/或垂直)方向上相对于第一颜色分量以N下采样，即，第一颜色分量沿水平(和/或垂直方向)轴相对于当前样本具有N倍的样本，则应用于当前颜色分量的最大水平(和/或垂直)变换大小为最大变换大小除以N。

[0120] 在一个实施例中，针对亮度分量发信号通知一个最大变换大小，如果它是YUV 444格式，则应用于色度分量的最大变换大小设置为与应用于亮度分量的最大变换大小相同。

[0121] 在一个实施例中，针对亮度分量发信号通知一个最大变换大小，如果它是YUV 422格式，则应用于色度分量的最大水平变换大小设置为应用于亮度分量的最大变换大小的一半，并且应用于色度分量的最大垂直变换大小设置为与应用于亮度分量的最大变换大小相同。

[0122] 建议针对不同颜色分量应用不同的变换归零(zero out)方案。

[0123] 建议仅当不同颜色分量之间的下采样率不同时，才针对不同颜色分量应用不同的变换归零方案。

[0124] 尽管图1示出了过程100的示例框，但是在一些实施方式中，过程100可以包括比图1中所描绘的更多的框、更少的框、不同的框或不同排列的框。另外或可替代地，过程100的

两个或更多个框可以并行执行。

[0125] 图2示出了根据本公开实施例的通信系统(200)的简化框图。通信系统(200)可以包括通过网络(250)互联的至少两个终端装置(210,220)。对于单向数据传输,第一终端装置(210)可在本地位置对视频数据进行编码,以通过网络(250)传输到另一终端装置(220)。第二终端装置(220)可从网络(250)接收另一终端装置的已编码视频数据,对已编码视频数据进行解码并显示恢复的视频数据。单向数据传输在媒体服务等应用中是较常见的。

[0126] 图2示出了支持已编码视频的双向传输的第二对终端装置(230,240),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,每个终端装置(230,240)可对在本地位置采集的视频数据进行编码,以通过网络(250)传输到另一终端装置。每个终端装置(230,240)还可接收由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并在本地显示设备上显示恢复的视频数据。

[0127] 在图2中,终端装置(210-240)可为服务器、个人计算机和智能手机,但本公开的原理不限于此。本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)表示在终端装置(210-240)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。代表性网络包括电信网络、局域网、广域网和/或互联网。出于本讨论的目的,除非在下文中有所解释,否则网络(250)的架构和拓扑对于本公开的操作来说可能是无关紧要的。

[0128] 作为所公开主题的应用实施例,图3示出了视频解码器和编码器在流式传输环境中的放置方式。所公开的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。

[0129] 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诸如数码相机等视频源(301),所述视频源创建例如未压缩的视频样本流(302)。较于已编码的视频比特流,样本流(302)被描绘为粗线,以强调其为高数据量的视频样本流,样本流(302)可由耦接到相机(301)的编码器(303)处理。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。相较于样本流(302),已编码的视频比特流(304)被描绘为细线,以强调较低数据量的已编码的视频比特流,其可存储在流式传输服务器(305)上以供将来使用。一个或多个流式传输客户端(306,308)可访问流式传输服务器(305)以检索已编码的视频比特流(304)的副本(307,309)。客户端(306)可包括视频解码器(310)。视频解码器(310)对已编码的视频比特流的传入副本(307)进行解码,且产生可在显示器(312)或另一呈现装置(未示出)上呈现的输出视频样本流(311)。在一些流式传输系统中,可以根据某些视频编解码/压缩标准对视频比特流(304,307,309)进行编码。这些标准的示例包括ITU-T H.265建议书。正在开发的视频编解码标准被非正式地称为多功能视频编解码(VVC)。所公开的主题可以在VVC的上下文中使用。

[0130] 图4是根据本发明实施例的视频解码器(310)的功能框图。

[0131] 接收器(410)可接收将由视频解码器(310)解码的一个或多个已编码视频序列;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(412)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(410)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(410)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(410)与熵解码器/解析器(420)(此后称为“解析器”)之间。当接收器(410)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有利地具有自适应性大小。

[0132] 视频解码器(310)可包括解析器(420)以根据熵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(310)的操作的信息，以及用以控制诸如显示器312的显示装置的潜在信息，所述显示装置不是解码器的组成部分，但可耦接到解码器，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(Supplementary 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)的形式。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解码可根据视频编解码技术或标准进行，且可遵循本领域技术人员公知的原理，包括可变长度编解码、霍夫曼编解码(Huffman coding)、具有或不具有上下文灵敏度的算术编解码等等。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(tile)、切片(slice)、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。熵解码器/解析器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(QP)值、运动矢量等等。

[0133] 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)。解析器(420)可以接收已编码数据，并且选择性地解码特定符号(421)。此外，解析器(420)可以确定是否将特定符号(421)提供给运动补偿预测单元(453)、缩放器/逆变换单元(451)、帧内预测单元(452)或环路滤波器(456)。

[0134] 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。

[0135] 除已经提及的功能块以外，视频解码器(310)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中，这些单元中的许多单元彼此紧密交互并且可以至少部分地彼此集成。然而，出于描述所公开的主题的目的，概念上细分成下文的功能单元是适当的。

[0136] 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。

[0137] 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(部分重建的)当前图片(454)提取的周围已重建的信息生成大小和形状与正在重建的块相同的块。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。

[0138] 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。运动补偿预测单元从参考图片存储器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。

[0139] 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比特流中的参数,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,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。

[0140] 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(312)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。

[0141] 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。一旦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(454)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存储器。

[0142] 视频解码器(310)可根据例如ITU-T H.265标准中记录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在视频压缩技术文献或标准、特别是配置文件中所规定的视频压缩技术或标准的语法的意义上,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。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。

[0143] 在实施例中,接收器(410)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(310)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。

[0144] 图5是根据本公开实施例的视频编码器(303)的功能框图。

[0145] 视频编码器(303)可从视频源(301)(并非该编码器的一部分)接收视频样本,所述视频源可采集将由视频编码器(303)编码的视频图像。

[0146] 视频源(301)可提供将由视频编码器(303)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 CrCb4:4:4)。在媒体服务系统中,视频源(301)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(301)可以为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,每个像素可包括一个或多个样本。本领域技术人员可以容易地理解像素和样本之间的关系。下文侧重于描述样本。

[0147] 根据实施例,视频编码器(303)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)。施行适当的编速度是控制器(550)的一个功能。控制器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。由控制器设置的参数可包括速率控制相关参数(例如,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,最大运动矢量搜索范围等。本领域技术人员可以容易地识别控制器(550)的其他功能,因为它们可能属于针对特定系统设计而优化的视频编码器(303)。

[0148] 一些视频编码器以本领域技术人员容易识别为"编码环路"方式进行操作。作为简单的描述,编码环路可包括编码器(530)(下文称为"源编码器(source coder)",其负责基于待编码的输入图片和参考图片创建符号)的编码部分、和嵌入于视频编码器(303)中的(本地)解码器(533)。"本地"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(因为在所公开的主题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)是本领域技术人员公知的。

[0149] "本地"解码器(533)的操作可与已在上文结合图4详细描述的"远程"解码器(310)相同。然而,另外简要参考图5,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,视频解码器(310)的熵解码部分(包括信道(412)、接收器(410)、缓冲存储器(415)和解析器(420))可能无法完全在本地解码器(533)中实施。

[0150] 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。对编码器技术的描述可以简写,因为它们与全面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。

[0151] 作为操作的一部分,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"参考帧"的一个或多个先前已编码帧,所述运动补偿预测编码对输入帧进行预测性编码。以此方式,编码引擎(532)对输入帧的像素块与参考帧的像素块之间的差异进行编码,所述参考帧可被选作所述输入帧的预测参考。

[0152] 本地视频解码器(533)可基于源编码器(530)创建的符号,对可指定为参考帧的帧的已编码视频数据进行解码。编码引擎(532)的操作可以有利地为有损过程。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帧执行,且可使重建的参考帧存储在参考图片存储器(534)中。以此方式,编码器(303)可在本地存储重建的参考帧的副本,所述副本与将由远端视频解码器获得的重建参考帧具有共同内容(不存在传输误差)。

[0153] 预测器(535)可针对编码引擎(532)执行预测搜索。即,对于将要编码的新帧,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。

[0154] 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。

[0155] 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。熵编码器可根据例如霍夫曼编码、可变长度编码、算术编码等本领域技术人员公知的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。

[0156] 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)可将来自源编码器(530)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)。

[0157] 控制器(550)可管理视频编码器(303)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。

[0158] 帧内图片(I图片),其可以是不将序列中的任何其它帧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。本领域技术人员知道I图片的那些变体以及它们各自的应用和特征。

[0159] 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。

[0160] 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。

[0161] 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。

[0162] 视频编码器(303)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(303)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。

[0163] 在实施例中，传输器(540)可传输附加数据和已编码的视频。源编码器(30)可以包括诸如已编码视频序列的一部分的此类数据。附加数据可包括时间/空间/SNR增层、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。

[0164] 此外，所提出的方法可以由处理电路(例如，一个或多个处理器或一个或多个集成电路)来实现。在一个示例中，一个或多个处理器执行存储在非易失性计算机可读介质中的程序，以执行所提出的方法中的一个或多个。

[0165] 上述技术可以通过计算机可读指令实现为计算机软件，并且物理地存储在一个多个计算机可读介质中。例如，图6示出了计算机系统600，其适于实现所公开主题的某些实施例。

[0166] 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。

[0167] 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。

[0168] 图6所示的用于计算机系统600的组件本质上是示例性的，并不用于对实现本公开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600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。

[0169] 计算机系统600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)，对一个或多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)。

[0170] 人机界面输入设备可包括以下中的一个或多个(仅绘出其中一个):键盘601、鼠标602、触控板603、触摸屏610、数据手套、操纵杆605、麦克风606、扫描仪607、照相机608。

[0171] 计算机系统600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/味觉来刺激一个或多个人类用户的感觉。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610、数据手套或操纵杆605的触觉反馈，但也可以有不用作输入设备的触觉反馈设备)、音频输出备(例如，扬声器609、耳机(未示出))、视觉输出设备(例如，包括阴极射线管(CRT)屏幕、液晶显示(LCD)屏幕、等离子屏幕、有机发光二极管(OLED)屏幕的屏幕610，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)。

[0172] 计算机系统600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)620或类似介质621的光学介质、拇指驱动器622、可移动硬盘驱动器或固体状态驱动器623，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。

[0173] 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。

[0174] 计算机系统600还可以包括通往一个或多个通信网络的一个或多个接口。例如,网络可以是无线的、有线的、光学的。网络还可为局域网、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。网络还包括以太网、无线局域网、蜂窝网络(全球移动通信系统(GSM)、第三代(3G)、第四代(4G)、第五代(5G)、长期演进(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649(例如,计算机系统600的通用串行总线(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600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600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。

[0175] 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600的核心640。

[0176] 核心640可包括一个或多个中央处理单元(CPU)641、图形处理单元(GPU)642、以现场可编程门阵列(FPGA)643形式的专用可编程处理单元、用于特定任务的硬件加速器644等。这些设备以及只读存储器(ROM)645、随机存取存储器(RAM)646、内部大容量存储器(例如内部非用户可存取硬盘驱动器、固态驱动器(SSD)等)647等可通过系统总线648进行连接。在某些计算机系统中,可以以一个或多个物理插头的形式访问系统总线648,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648,或通过外围总线649进行连接。外围总线的体系结构包括外围组件互联(PCI)、USB等。

[0177] CPU 641、GPU 642、FPGA 643和加速器644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 645或RAM 646中。过渡数据也可以存储在RAM 646中,而永久数据可以存储在例如内部大容量存储器647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一个或多个CPU 641、GPU 642、大容量存储器647、ROM 645、RAM 646等紧密关联。

[0178] 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公开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。

[0179] 作为实施例而非限制,具有体系结构600的计算机系统,特别是核心640,可以作为处理器(包括CPU、GPU、FPGA、加速器等)提供执行包含在一个或多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640的特定存储器,例如核心内部大容量存储器647或ROM 645。实现本公开的各种实施例的软件可以存储在这种设备中并且由核心640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640

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 646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644)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。本公开包括任何合适的硬件和软件组合。

[0180] 虽然本公开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公开的范围内。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,但其体现了本公开的原则,因此属于本公开的精神和范围之内。

[0181] 首字母缩略词:

[0182] HEVC:高效视频编解码(High Efficiency Video Coding)

[0183] HDR:高动态范围(high dynamic range)

[0184] SDR:标准动态范围(standard dynamic range)

[0185] VVC:通用视频编解码(Versatile Video Coding)

[0186] JVET:联合视频开发组(Joint Video Exploration Team)

[0187] MPM:最可能模式(most probable mode)

[0188] WAIP:宽角度帧内预测(Wide-Angle Intra Prediction)

[0189] CU:编码单元(Coding Unit)

[0190] PU:预测单元(Prediction Unit)

[0191] ISP:帧内子分区(Intra Sub-Partitions)

[0192] SBT:子块变换(Sub-block transform)

[0193] CBF:已编码块标志(Coded block flag)

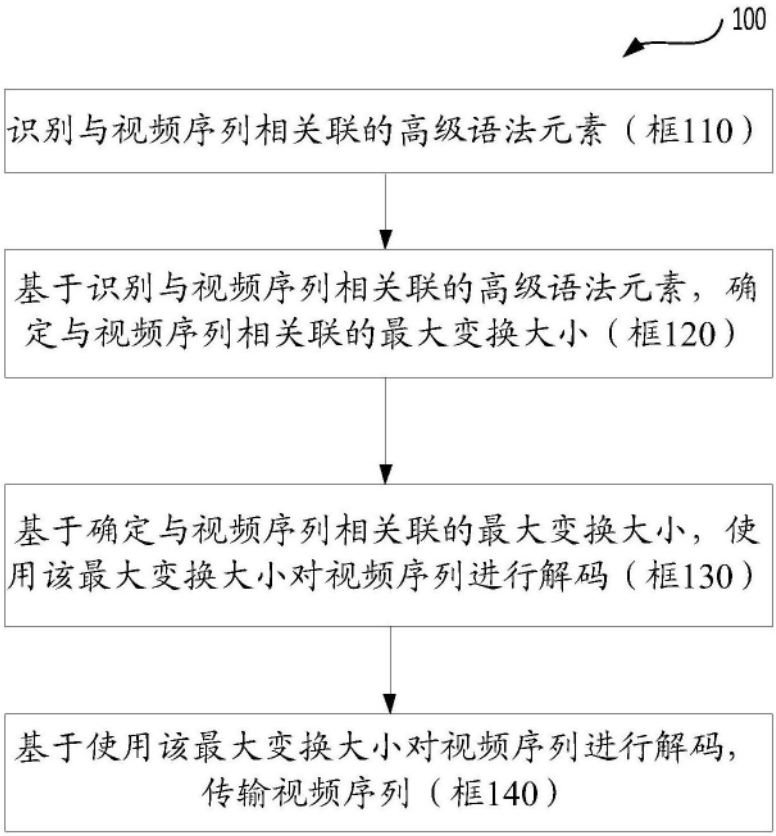

100
识别与视频序列相关联的高级语法元素（框110）
基于识别与视频序列相关联的高级语法元素，确定与视频序列相关联的最大变换大小（框120）
基于确定与视频序列相关联的最大变换大小，使用该最大变换大小对视频序列进行解码（框130）
基于使用该最大变换大小对视频序列进行解码，传输视频序列（框140）

图1

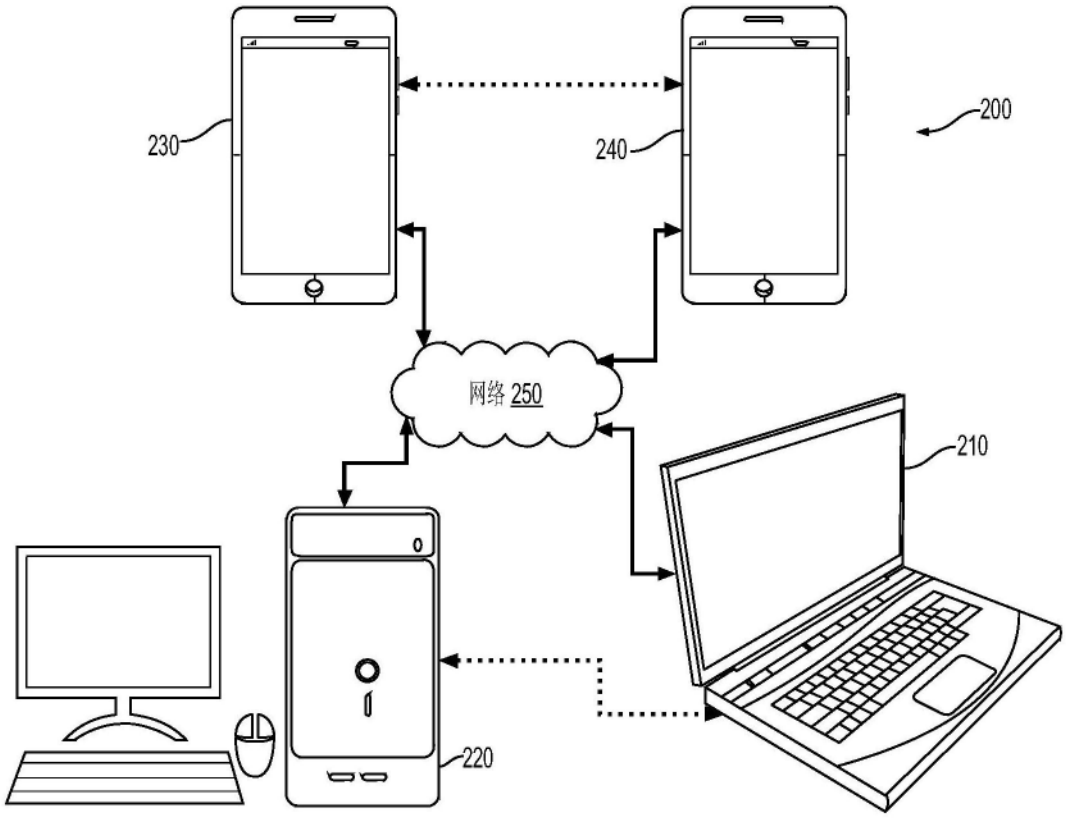


图2

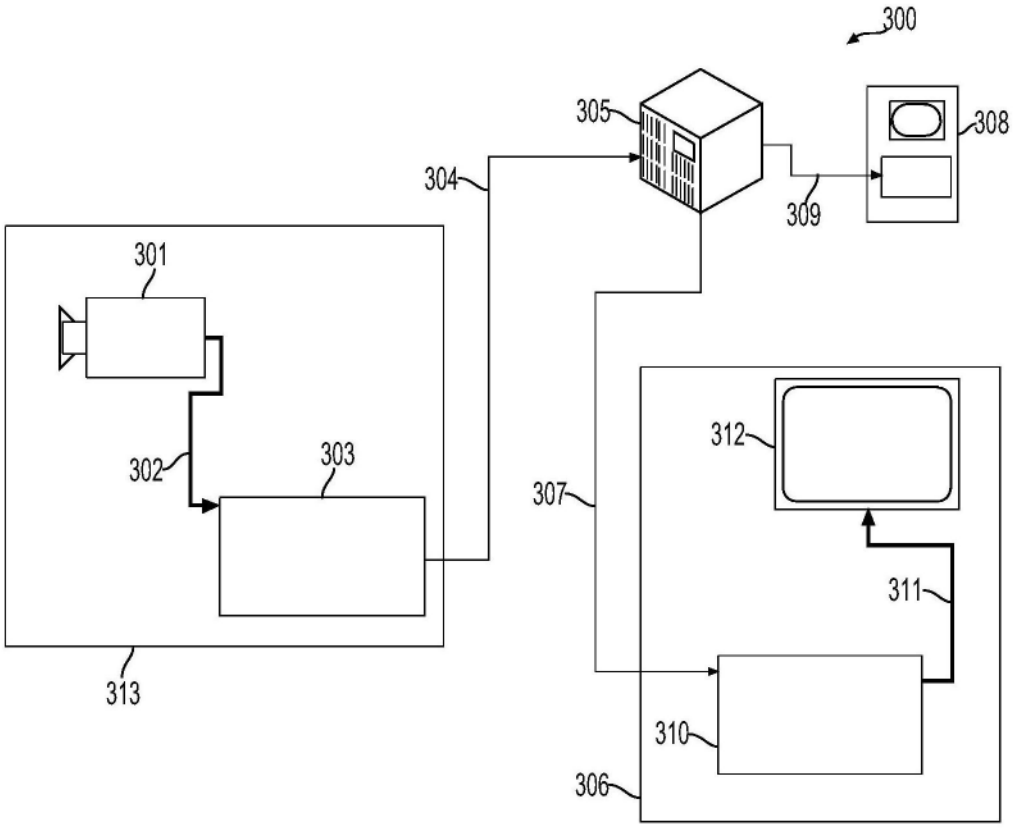


图3

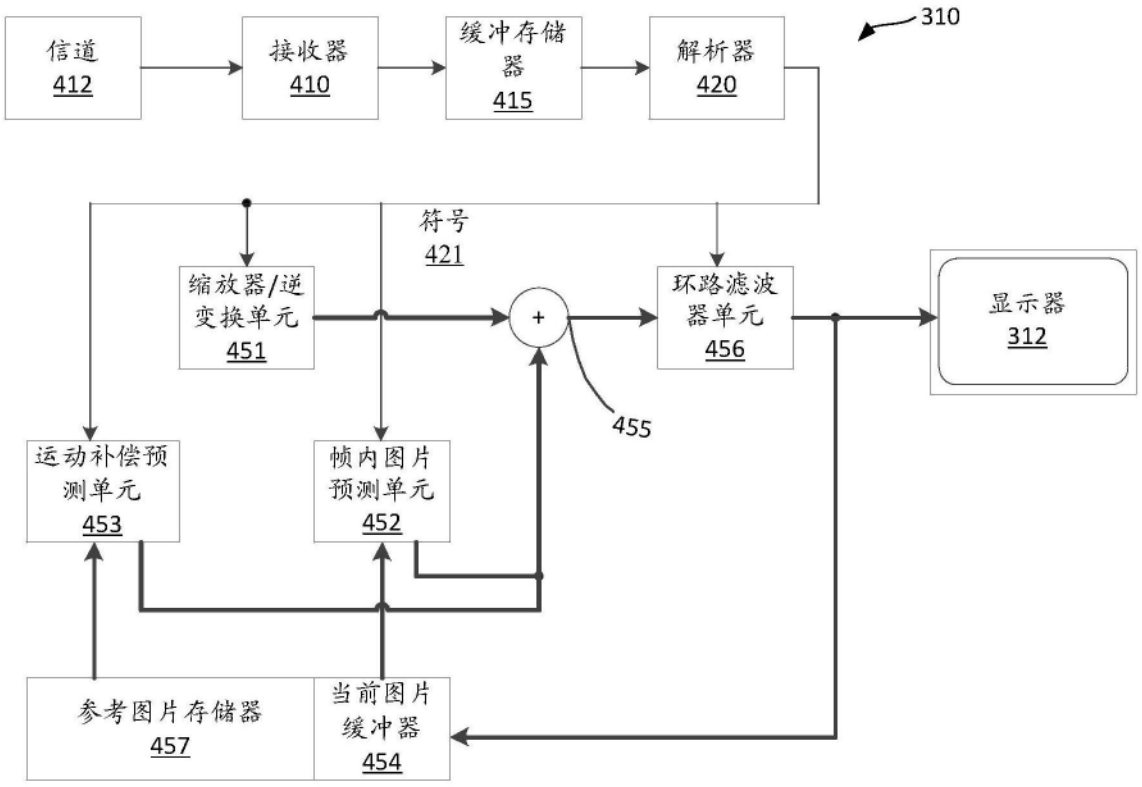

310
信道
412
接收器
410
缓冲存储器
415
解析器
420
符号
421
缩放器/逆变换单元
451
+
环路滤波器单元
456
显示器
312
455
运动补偿预测单元
453
帧内图片预测单元
452
参考图片存储器
457
当前图片缓冲器
454

图4

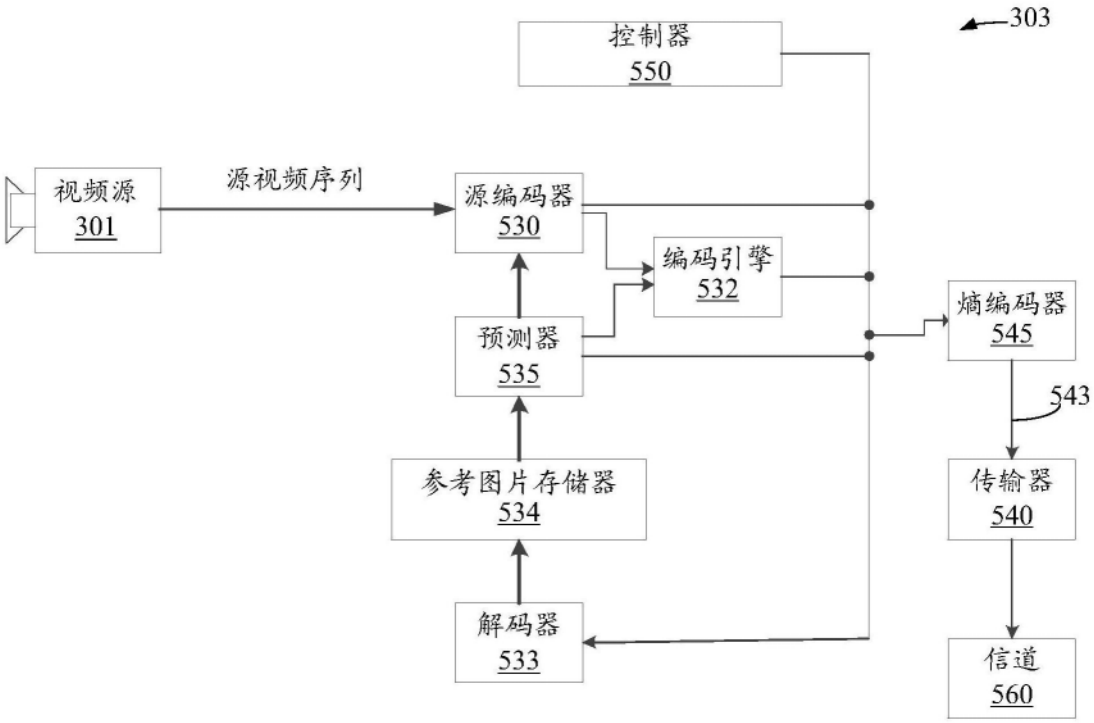

303
控制器
550
视频源
301
源视频序列
源编码器
530
编码引擎
532
熵编码器
545
预测器
535
543
参考图片存储器
534
传输器
540
解码器
533
信道
560

图5

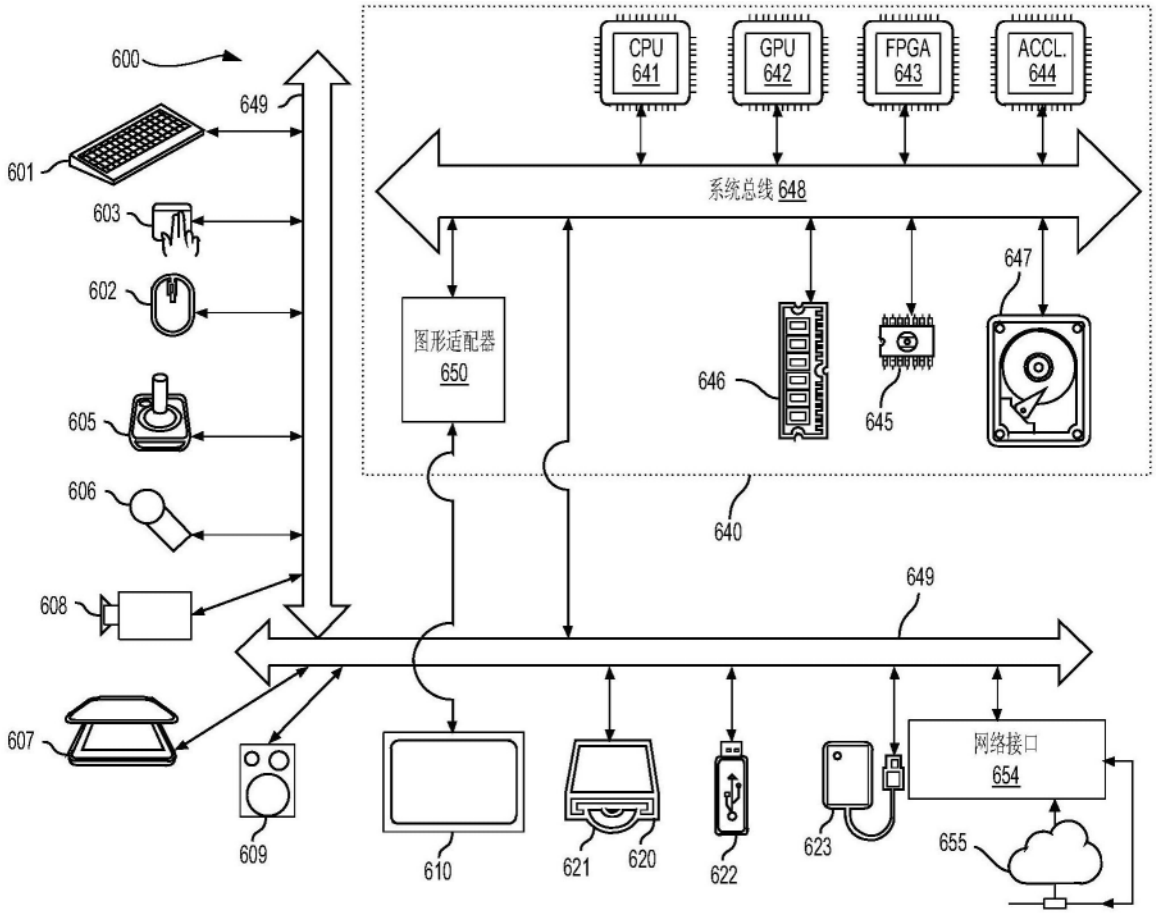

600
649
601
603
602
605
606
608
607
609
610
621
620
622
623
655
CPU 641
GPU 642
FPGA 643
ACCL. 644
系统总线 648
647
图形适配器 650
646
645
640
649
网络接口 654

图6