



(10) 授权公告号 CN 113412623 B

(45) 授权公告日 2025. 03. 25

(21) 申请号 202080011162.9

(22) 申请日 2020.01.31

(65) 同一申请的已公布的文献号
申请公布号 CN 113412623 A

(43) 申请公布日 2021.09.17

(66) 本国优先权数据
PCT/CN2019/074216 2019.01.31 CN
PCT/CN2019/074433 2019.02.01 CN
PCT/CN2019/079962 2019.03.27 CN

(85) PCT国际申请进入国家阶段日
2021.07.27

(86) PCT国际申请的申请数据
PCT/CN2020/074122 2020.01.31

(87) PCT国际申请的公布数据
W02020/156516 EN 2020.08.06

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

(72) 发明人 刘鸿彬 张莉 张凯 王悦

(74) 专利代理机构 北京市柳沈律师事务所
11105

专利代理师 张亮

(51) Int.Cl.
H04N 19/513 (2006.01)
H04N 19/176 (2006.01)

(56) 对比文件
US 2016337649 A1, 2016.11.17
Hongbin Liu.CE2: Adaptive Motion
Vector Resolution for Affine Inter Mode
(Test 2.1.2).Joint Video Experts Team
(JVET) of ITU-T SG 16 WP 3 and ISO/IEC
JTC 1/SC 29/WG 11 13th Meeting:
Marrakech, MA, 9-18 Jan. 2019, JVET-
M0246.2019, 第1-3节.
Benjamin Bross.Versatile Video Coding
(Draft 3).Joint Video Experts Team (JVET)
of ITU-T SG 16 WP 3 and ISO/IEC JTC 1/SC
29/WG 11 12th Meeting: Macao, CN, 3-12
Oct. 2018, JVET-L1001.2018, 第7.3.4.6、
9.5.4.2.2节.

审查员 李颖

权利要求书3页 说明书62页 附图33页

(54) 发明名称

记录仿射模式自适应运动矢量分辨率的上
下文

(57) 摘要

提供了一种视频处理方法。该方法包括：确定视频的当前视频块与当前视频块的编解码表示之间的转换基于非仿射帧间AMVR模式；以及基于该确定来进行转换，其中，当前视频块的编解码表示基于基于上下文的编解码，并且其中，在转换期间不使用相邻块的仿射AMVR模式信息的情况下对用于编解码当前视频块的上下文进行建模。



1. 一种视频处理方法,包括:
确定当前视频块的编解码模式;
对于所述当前视频块的运动信息,从包括多个运动精度的运动精度集合中确定运动精度;以及
基于所确定的编解码模式和所确定的运动精度,在所述当前视频块和包括所述当前视频块的视频的比特流之间执行转换;
其中,在所述比特流中选择性地存在多个语法元素以指示所述运动精度集合中的所述运动精度,其中所述多个语法元素包括第一语法元素和第二语法元素,其中在所述第一语法元素以特定值存在于所述比特流中的情况下,所述第二语法元素存在于所述比特流中;
其中,所述第二语法元素指示从1/16像素或1像素精度中针对第一模式的选择;
其中,基于所述当前视频块的编解码模式是否为第一模式而为所述第一语法元素从多个上下文中选择第一上下文并且为所述第二语法元素从多个上下文中选择第二上下文,
其中,在所述第一模式中,导出并进一步使用控制点运动矢量从所述当前视频块中导出子区域划分的运动矢量;
其中,在不使用所述当前视频块的相邻视频块的编解码信息的情况下选择所述第一上下文和所述第二上下文。
2. 根据权利要求1所述的方法,其中,所述相邻视频块的编解码信息包括所述相邻视频块的编解码模式。
3. 根据权利要求1所述的方法,其中,变量控制所述多个上下文中的每个上下文的两个概率更新速度。
4. 根据权利要求3所述的方法,其中,所述两个概率更新速度包括由 $(\text{shiftIdx} \gg 2) + 2$ 定义的第一更新速度,shiftIdx指示所述变量。
5. 根据权利要求3所述的方法,其中,所述两个概率更新速度包括由 $(\text{shiftIdx} \& 3) + 3 + \text{shift0}$ 定义的第二更新速度,shift0指示第一更新速度并且shiftIdx指示所述变量。
6. 根据权利要求5所述的方法,其中,所述第二更新速度在2和5之间,含端点。
7. 根据权利要求1所述的方法,其中,所述第二语法元素指示对于正常帧间模式从1像素或4像素精度中的选择。
8. 根据权利要求1所述的方法,其中,对于第一模式,所述第二语法元素等于0,并且所述运动精度是1/6像素精度。
9. 根据权利要求1所述的方法,其中,对于第一模式,所述第二语法元素等于1,并且所述运动精度是1像素精度。
10. 根据权利要求1所述的方法,其中,所述第一语法元素用所述第一上下文编解码,并且所述第二语法元素用所述第二上下文编解码。
11. 根据权利要求1所述的方法,其中,所述转换的执行包括基于所述当前视频块生成所述比特流。
12. 根据权利要求1所述的方法,其中,所述转换的执行包括从所述比特流生成所述当前视频块。
13. 一种用于处理视频数据的装置,包括处理器和其上具有指令的非暂时性存储器,其中所述指令在被所述处理器执行时使所述处理器:

确定当前视频块的编解码模式；

对于所述当前视频块的运动信息,从包括多个运动精度的运动精度集合中确定运动精度;以及

基于所确定的编解码模式和所确定的运动精度,在所述当前视频块和包括所述当前视频块的视频的比特流之间执行转换;

其中,在所述比特流中选择性地存在多个语法元素以指示所述运动精度集合中的所述运动精度,其中所述多个语法元素包括第一语法元素和第二语法元素,其中在所述第一语法元素以特定值存在于所述比特流中的情况下,所述第二语法元素存在于所述比特流中;

其中,所述第二语法元素指示从1/16像素或1像素精度中针对第一模式的选择;

其中,基于所述当前视频块的编解码模式是否为第一模式而为所述第一语法元素从多个上下文中选择第一上下文并且为所述第二语法元素从多个上下文中选择第二上下文,

其中,在所述第一模式中,导出并进一步使用控制点运动矢量从所述当前视频块中导出子区域划分的运动矢量;

其中,在不使用所述当前视频块的相邻视频块的编解码信息的情况下选择所述第一上下文和所述第二上下文。

14.一种存储指令的非暂时性计算机可读存储介质,所述指令使处理器:

确定当前视频块的编解码模式;

对于所述当前视频块的运动信息,从包括多个运动精度的运动精度集合中确定运动精度;以及

基于所确定的编解码模式和所确定的运动精度,在所述当前视频块和包括所述当前视频块的视频的比特流之间执行转换;

其中,在所述比特流中选择性地存在多个语法元素以指示所述运动精度集合中的所述运动精度,其中所述多个语法元素包括第一语法元素和第二语法元素,其中在所述第一语法元素以特定值存在于所述比特流中的情况下,所述第二语法元素存在于所述比特流中;

其中,所述第二语法元素指示从1/16像素或1像素精度中针对第一模式的选择;

其中,基于所述当前视频块的编解码模式是否为第一模式而为所述第一语法元素从多个上下文中选择第一上下文并且为所述第二语法元素从多个上下文中选择第二上下文,

其中,在所述第一模式中,导出并进一步使用控制点运动矢量从所述当前视频块中导出子区域划分的运动矢量;

其中,在不使用所述当前视频块的相邻视频块的编解码信息的情况下选择所述第一上下文和所述第二上下文。

15.一种用于存储视频的比特流的方法,包括:

确定当前视频块的编解码模式;

对于所述当前视频块的运动信息,从包括多个运动精度的运动精度集合中确定运动精度;以及

基于所确定的编解码模式和所确定的运动精度,生成所述比特流;

将所述比特流存储在非暂时性计算机可读记录介质中,

其中,在所述比特流中选择性地存在多个语法元素以指示所述运动精度集合中的所述运动精度,其中所述多个语法元素包括第一语法元素和第二语法元素,其中在所述第一语

法元素以特定值存在于所述比特流中的情况下,所述第二语法元素存在于所述比特流中;

其中,所述第二语法元素指示从1/16像素或1像素精度中针对第一模式的选择;

其中,基于所述当前视频块的编解码模式是否为第一模式而为所述第一语法元素从多个上下文中选择第一上下文并且为所述第二语法元素从多个上下文中选择第二上下文,

其中,在所述第一模式中,导出并进一步使用控制点运动矢量从所述当前视频块中导出子区域划分的运动矢量;

其中,在不使用所述当前视频块的相邻视频块的编解码信息的情况下选择所述第一上下文和所述第二上下文。

16.一种视频系统中的装置,包括处理器和其上具有指令的非暂时性存储器,其中,所述指令在由所述处理器执行时使所述处理器实现权利要求2-12中的任一项所述的方法。

17.一种存储在非暂时性计算机可读介质上的计算机程序产品,所述计算机程序产品包括用于实施权利要求2-12中的任一项所述的方法的程序代码。

记录仿射模式自适应运动矢量分辨率的上下文

[0001] 相关申请的交叉引用

[0002] 根据适用的专利法和/或依据巴黎公约的规则,本申请要求于2019年1月31日提交的国际专利申请第PCT/CN2019/074216号和于2019年2月1日提交的国际专利申请第PCT/CN2019/074433号以及于2019年3月27日提交的国际专利申请第PCT/CN2019/079962号的优先权和权益。出于所有目的,上述申请的所有公开内容通过引用并入作为本申请的公开内容的一部分。

技术领域

[0003] 本专利文件涉及视频处理技术、设备和系统。

背景技术

[0004] 尽管视频压缩有所进步,但数字视频仍占因特网和其它数字通信网络上最大的带宽使用。随着能够接收和显示视频的所连接的用户设备的数量增加,预计数字视频使用的带宽需求将继续增长。

发明内容

[0005] 描述了与数字视频编解码有关的设备、系统和方法,并且具体地,描述了具有自适应运动矢量分辨率 (AMVR) 的仿射模式的运动矢量预测 (MVP) 推导和信令。所描述的方法可以应用于现有视频编解码标准 (例如,高效视频编解码 (HEVC)) 和未来视频编解码标准或视频编解码器。

[0006] 在一个代表性方面,所公开的技术可以用于提供视频处理方法。该方法包括:确定视频的当前视频块与当前视频块的编解码表示之间的转换基于非仿射帧间AMVR模式;以及基于确定来进行转换,并且其中当前视频块的编解码表示基于基于上下文的编解码,并且其中在转换期间不使用相邻块的仿射AMVR模式信息的情况下来对用于编解码当前视频块的上下文进行建模。

[0007] 在另一个代表性方面,所公开的技术可以用于提供视频处理方法。该方法包括:确定视频的当前视频块与当前视频块的编解码表示之间的转换基于仿射自适应运动矢量分辨率 (仿射AMVR) 模式;以及基于确定来进行转换,并且其中当前视频块的编解码表示基于基于上下文的编解码,并且其中变量控制上下文的两个概率更新速度。

[0008] 在另一个代表性方面,所公开的技术可以用于提供视频处理方法。该方法包括:确定视频的当前视频块与当前视频块的编解码表示之间的转换基于仿射AMVR模式;以及基于确定来进行转换,并且其中当前视频块的编解码表示基于基于上下文的编解码,并且其中使用相邻块的编解码信息来对用于编解码当前视频块的上下文进行建模,相邻块的编解码信息在转换期间使用仿射帧间模式和正常帧间模式两者的AMVR模式。

[0009] 在另一个代表性方面,所公开的技术可以用于提供视频处理方法。该方法包括:对于视频的当前视频块与当前视频块的编解码表示之间的转换,确定用于转换的多个上下文

的使用;以及基于确定来进行转换,并且其中利用多个上下文来对指示粗运动精度的语法元素进行编解码。

[0010] 在另一个代表性方面,所公开的技术可以用于提供视频处理方法。该方法包括:对于视频的当前视频块与当前视频块的编解码表示之间的转换,基于用于转换的当前选择的最佳模式来确定是否使用对称运动矢量差 (SMVD) 模式;以及基于确定来进行转换。

[0011] 在另一个代表性方面,所公开的技术可以用于提供视频处理方法。该方法包括:对于视频的当前视频块与当前视频块的编解码表示之间的转换,基于用于转换的当前选择的最佳模式来确定是否使用仿射SMVD模式;以及基于确定来进行转换。

[0012] 在另一个代表性方面,上述方法以处理器可执行代码的形式实施并存储在计算机可读程序介质中。

[0013] 在又一个代表性方面,公开了一种被配置或可操作以执行上述方法的设备。该设备可以包括被编程为实现该方法的处理器。

[0014] 在又一个代表性方面,视频解码器装置可以实现如本文中所描述的方法。

[0015] 在附图、说明书和权利要求中更详细地描述了所公开技术的上述方面和特征以及其它方面和特征。

附图说明

[0016] 图1示出了构建Merge候选列表的示例。

[0017] 图2示出了空域候选的位置的示例。

[0018] 图3示出了经受空域Merge候选的冗余校验的候选对的示例。

[0019] 图4A和图4B示出了基于当前块的尺寸和形状的第二预测单元PU的位置的示例。

[0020] 图5示出了用于时域Merge候选的运动矢量缩放的示例。

[0021] 图6示出了用于时域Merge候选的候选位置的示例。

[0022] 图7示出了生成组合的双向预测Merge候选的示例。

[0023] 图8示出了构建运动矢量预测候选的示例。

[0024] 图9示出了用于空域运动矢量候选的运动矢量缩放的示例。

[0025] 图10示出了使用用于编解码单元 (CU) 的替代时域运动矢量预测 (ATMVP) 算法的运动预测的示例。

[0026] 图11示出了具有由空域-时域运动矢量预测 (STMVP) 算法使用的子块和相邻块的编解码单元 (CU) 的示例。

[0027] 图12A和图12B示出了当使用重叠块的运动补偿 (OBMC) 算法时子块的示例快照。

[0028] 图13示出了用于导出局部亮度补偿 (LIC) 算法的参数的相邻样本的示例。

[0029] 图14示出了简化的仿射运动模型的示例。

[0030] 图15示出了每个子块的仿射运动矢量场 (MVF) 的示例。

[0031] 图16示出了用于AF_INTER仿射运动模式的运动矢量预测 (MVP) 的示例。

[0032] 图17A和17B分别示出了4参数仿射模型和6参数仿射模型的示例。

[0033] 图18A和18B示出了用于AF_MERGE仿射运动模式的示例候选。

[0034] 图19示出了在模式匹配运动矢量推导 (PMMVD) 模式中的双边匹配的示例,模式匹配运动矢量推导 (PMMVD) 模式基于帧速率上转换 (FRUC) 算法的特殊Merge模式。

- [0035] 图20示出了在FRUC算法中的模板匹配的示例。
- [0036] 图21示出了在FRUC算法中的单向运动估计的示例。
- [0037] 图22示出了由双向光流 (BIO) 算法使用的光流轨迹的示例。
- [0038] 图23A和23B示出了使用不具有块扩展的双向光流 (BIO) 算法的示例快照。
- [0039] 图24示出了基于双边模板匹配的解码器侧运动矢量细化 (DMVR) 算法的示例。
- [0040] 图25A-25F示出了基于所公开的技术的实现方式的用于视频处理的示例方法的流程图。
- [0041] 图26是用于实现本文件中描述的视觉媒体解码或视觉媒体编码技术的硬件平台的示例的框图。
- [0042] 图27示出了对称模式的示例。
- [0043] 图28示出了用于实现本文件中描述的视频处理系统的硬件平台的示例的另一个框图。

具体实施方式

[0044] 由于对更高分辨率视频的需求的增加,视频编解码方法和技术在现代技术中普遍存在。视频编解码器通常包括压缩或解压缩数字视频的电子电路或软件,并且被不断改进以提供更高的编解码效率。视频编解码器将未压缩视频转换为压缩格式,反之亦然。视频质量、用于表示视频的数据量(由比特率确定)、编码和解码算法的复杂度、对数据丢失和错误的敏感性、编辑的简易性,随机访问和端到端延迟(时延)之间存在复杂的关系。压缩格式通常符合标准视频压缩规范,例如,高效视频编解码 (HEVC) 标准(也称为H.265或MPEG-H第2部分) [1]、待最终确定的通用视频编解码标准、或其它当前和/或未来的视频编解码标准。

[0045] 所公开的技术的实施例可以应用于现有视频编解码标准(例如,HEVC,H.265)和未来的标准以改进压缩性能。在本文件中使用节标题以改进描述的可读性,并且不以任何方式将讨论或实施例(和/或实现方式)限制于仅相应的节。

[0046] 1. 在HEVC/H.265中的帧间预测的示例

[0047] 多年来,视频编解码标准已经显着改进,并且现在部分地提供高编解码效率和对更高分辨率的支持。诸如HEVC和H.265的最新标准基于混合视频编解码结构,其中使用时域预测加变换编解码。

[0048] 1.1 预测模式的示例

[0049] 每个帧间预测的预测单元(PU)具有对于一个或两个参考图片列表的运动参数。在一些实施例中,运动参数包括运动矢量和参考图片索引。在其它实施例中,也可以使用inter_pred_idc来信令通知对两个参考图片列表中一个的使用。在另外的其它实施例中,运动矢量可以明确地被编解码为相对于预测器的增量。

[0050] 当使用跳过(skip)模式来编解码编解码单元(CU)时,一个PU与该CU相关联,并且不存在显著的残差系数,不存在编解码的运动矢量增量或参考图片索引。指定Merge模式,由此从相邻PU——包括空域和时域候选——获得用于当前PU的运动参数。Merge模式可以应用于任何帧间预测的PU,而不仅适用于跳过模式。Merge模式的替代方案是运动参数的显式传输(explicit transmission),其中运动矢量、每个参考图片列表的对应参考图片索引、参考图片列表使用对每个PU被显式信令通知。

[0051] 当信令指示要使用两个参考图片列表中的一个时,PU从一个样本块产生。这被称为“单向预测”。单向预测可用于P条带和B条带[2]。

[0052] 当信令指示要使用两个参考图片列表时,PU从两个样本块产生。这被称为“双向预测”。双向预测仅可用于B条带。

[0053] 1.1.1构建Merge模式的候选的实施例

[0054] 当使用Merge模式预测PU时,从比特流解析出指向Merge候选列表(merge candidates list)中的条目的索引,并且该索引被用于检索运动信息。该列表的构建可以根据以下步骤顺序进行总结:

[0055] 步骤1:初始候选推导

[0056] 步骤1.1:空域候选推导

[0057] 步骤1.2:空域候选的冗余校验

[0058] 步骤1.3:时域候选推导

[0059] 步骤2:附加候选插入

[0060] 步骤2.1:创建双向预测候选

[0061] 步骤2.2:插入零运动候选

[0062] 图1示出了基于上面总结的步骤顺序构建Merge候选列表的示例。对于空域Merge候选推导,在位于五个不同位置的候选中选择最多四个Merge候选。对于时域Merge候选推导,在两个候选中选择最多一个Merge候选。由于在解码器处假设每个PU的候选的数量为常数,因此当候选的数量未达到在条带报头中信令通知的最大Merge候选数量(MaxNumMergeCand)时,生成附加的候选。由于候选的数量是恒定的,因此使用二进制一元截断(TU)来编码最佳Merge候选的索引。如果CU的尺寸等于8,则当前CU的所有PU共享单个Merge候选列表,该单个Merge候选列表与 $2N \times 2N$ 预测单元的Merge候选列表相同。

[0063] 1.1.2构建空域Merge候选

[0064] 在空域Merge候选的推导中,在位于图2中描绘的位置中的候选中选择最多四个Merge候选。推导的顺序是 A_1 、 B_1 、 B_0 、 A_0 和 B_2 。仅当位置 A_1 、 B_1 、 B_0 、 A_0 的任何PU不可用时(例如,因为该PU属于另一个条带(slice)或片(tile))或者是帧内编解码时,才考虑位置 B_2 。在添加位置 A_1 处的候选之后,对剩余候选的添加进行冗余校验,该冗余校验确保具有相同运动信息的候选被排除在列表之外,从而改进编解码效率。

[0065] 为了降低计算复杂度,在所提到的冗余校验中并未考虑所有可能的候选对。相反,仅考虑与图3中的箭头链接的对,并且仅当如果用于冗余校验的对应候选具有不同的运动信息时,则该候选才被添加到列表中。重复的运动信息的另一个来源是与不同于 $2N \times 2N$ 的分割相关联的“第二PU”。作为示例,图4A和4B分别描绘了针对 $N \times 2N$ 和 $2N \times N$ 的情况的第二PU。当当前PU被分割为 $N \times 2N$ 时,位置 A_1 处的候选不被考虑用于列表构建。在一些实施例中,添加该候选可以导致具有相同运动信息的两个预测单元,这对于在编解码单元中仅具有一个PU是冗余的。类似地,当当前PU被分割为 $2N \times N$ 时,不考虑位置 B_1 。

[0066] 1.1.3构建时域Merge候选

[0067] 在此步骤中,只有一个候选被添加到列表中。特别地,在该时域Merge候选的推导中,基于共位(co-located)的PU来导出缩放的运动矢量,该共位的PU属于相对于给定参考图片列表内的当前图片具有最小图片顺序计数(POC)差的图片。在条带报头中明确地信令

通知用于共位的PU的推导的参考图片列表。

[0068] 图5示出了用于时域Merge候选(如虚线)的缩放的运动矢量的推导的示例,该用于时域Merge候选(如虚线)的缩放的运动矢量使用POC距离tb和td而从共位的PU的运动矢量被缩放,其中tb被定义为当前图片的参考图片与该当前图片之间的POC差,并且td被定义为共位图片的参考图片与该共位图片之间的POC差。时域Merge候选的参考图片索引被设置为等于零。对于B条带,获得两个运动矢量并将其组合以产生双向预测Merge候选,该两个运动矢量中的一个用于参考图片列表0(list 0)而另一个用于参考图片列表1(list 1)。

[0069] 如图6所示,在属于参考帧的共位的PU(Y)中,在候选 C_0 和 C_1 之间选择用于时域候选的位置。如果位置 C_0 处的PU不可用、被帧内编解码的或在当前CTU之外,则使用位置 C_1 。否则,在时域Merge候选的推导中使用位置 C_0 。

[0070] 1.1.4构建附加类型的Merge候选

[0071] 除了空域-时域Merge候选之外,还存在两种附加类型的Merge候选:组合的双向预测Merge候选和零Merge候选。通过利用空域-时域Merge候选来生成组合的双向预测Merge候选。组合的双向预测Merge候选仅用于B条带。通过将初始候选的第一参考图片列表运动参数与另一个候选的第二参考图片列表运动参数组合来生成组合的双向预测候选。如果这两个元组(tuple)提供不同的运动假设,则它们将形成一个新的双向预测候选。

[0072] 图7示出了该过程的示例,在该过程中原始列表(710,在左方)中具有mvL0和refIdxL0或mvL1和refIdxL1的两个候选被用于创建组合的双向预测Merge候选,该组合的双向预测Merge候选被添加到最终列表(720,在右方)。

[0073] 零运动候选被插入以填充Merge候选列表中的剩余条目,并且因此达到MaxNumMergeCand容量。这些候选具有零空域位移和参考图片索引,该参考图片索引从零开始并且每当新的零运动候选被添加到列表时则增加。这些候选使用的参考帧的数量是分别用于单向和双向预测的1和2。在一些实施例中,不对这些候选执行冗余校验。

[0074] 1.1.5用于并行处理的运动估计区域的示例

[0075] 为了加速编码过程,可以并行执行运动估计,由此同时导出给定区域内的所有预测单元的运动矢量。来自空域相邻的Merge候选的推导可能干扰并行处理,因为一个预测单元直到其相关联的运动估计完成才能从相邻PU导出运动参数。为了减轻编解码效率和处理等待时域之间的折衷,可以定义运动估计区域(MER)。MER的尺寸可以在图片参数集(PPS)中使用“log2_parallel_merge_level_minus2”语法元素来信令通知。当MER被定义时,落入同一区域的Merge候选被标记为不可用,并且因此在列表构建中也不被考虑。

[0076] 1.2高级运动矢量预测(AMVP)的实施例

[0077] AMVP利用运动矢量与相邻PU的空域-时域相关性,该空域-时域相关性用于运动参数的显式传输。通过下述操作来构建运动矢量候选列表:首先校验左方、上方在时域上相邻PU位置的可用性,移除冗余候选,并添加零矢量,以使候选列表为恒定长度。然后,编码器可以从候选列表中选择最佳预测器,并传输指示所选候选的对应索引。与Merge索引信令类似,使用一元截断来编码最佳运动矢量候选的索引。在这种情况下要编码的最大值是2(参见图8)。在以下的节中,提供了关于运动矢量预测候选的推导过程的细节。

[0078] 1.2.1构建运动矢量预测候选的示例

[0079] 图8总结了用于运动矢量预测候选的推导过程,并且可以针对具有refidx作为输

入的每个参考图片列表来实现。

[0080] 在运动矢量预测中,考虑两种类型的运动矢量候选:空域运动矢量候选和时域运动矢量候选。如图2先前示出的,对于空域运动矢量候选推导,最终基于位于五个不同位置的每个PU的运动矢量来导出两个运动矢量候选。

[0081] 对于时域运动矢量候选推导,从基于两个不同的共位位置导出的两个候选中选择一个运动矢量候选。在制作空域-时域候选的第一列表之后,移除列表中的重复的运动矢量候选。如果潜在候选的数量大于2,则从列表中移除其在相关联的参考图片列表内的参考图片索引大于1的运动矢量候选。如果空域-时域运动矢量候选的数量小于2,则将附加的零运动矢量候选添加到列表中。

[0082] 1.2.2构建空域运动矢量候选

[0083] 在空域运动矢量候选的推导中,在五个潜在候选中考虑最多两个候选,该五个潜在候选来自位于如图2先前示出的位置的PU,这些位置与运动Merge的那些位置相同。当前PU的左侧的推导顺序被定义为 A_0 、 A_1 以及缩放的 A_0 、缩放的 A_1 。当前PU的上侧的推导顺序被定义为 B_0 、 B_1 、 B_2 、缩放的 B_0 、缩放的 B_1 、缩放的 B_2 。因此,对于每一侧,存在四种可用作运动矢量候选的情况,其中两种情况不需要使用空域缩放,并且两种情况使用空域缩放。四种不同的情况总结如下。

[0084] • 无空域缩放

[0085] (1) 相同的参考图片列表,以及相同的参考图片索引(相同的POC)

[0086] (2) 不同的参考图片列表,但是相同的参考图片索引(相同的POC)

[0087] • 空域缩放

[0088] (3) 相同的参考图片列表,但是不同的参考图片索引(不同的POC)

[0089] (4) 不同的参考图片列表,以及不同的参考图片索引(不同的POC)

[0090] 首先校验无空域缩放情况,接下来校验允许空域缩放的情况。不管参考图片列表如何,当POC在相邻PU的参考图片与当前PU的参考图片之间是不同的时,考虑空域缩放。如果左方候选的所有PU都不可用或者是被帧内编解码的,则允许对上方运动矢量进行缩放,以帮助左方和上方MV候选的并行推导。否则,对上侧运动矢量不允许空域缩放。

[0091] 如图9中的示例所示,对于空域缩放情况,以与时域缩放类似的方式缩放相邻PU的运动矢量。一个差在于当前PU的参考图片列表和索引被给定为作为输入;实际缩放过程与时域缩放过程相同。

[0092] 1.2.3构建时域运动矢量候选

[0093] 除了参考图片索引推导之外,用于时域Merge候选的推导的所有过程与用于空域运动矢量候选的推导的过程相同(如图6中的示例所示)。在一些实施例中,将参考图片索引信令通知给解码器。

[0094] 2.在联合探索模型(JEM)中的帧间预测方法的示例

[0095] 在一些实施例中,使用称为联合探索模型(JEM) [3] [4]的参考软件来探索未来的视频编解码技术。在JEM中,在数个编解码工具中采用基于子块的预测,诸如仿射预测、可选时域运动矢量预测(ATMVP)、空域-时域运动矢量预测(STMVP)、双向光流(BIO)、帧速率上转换(FRUC)、局部自适应运动矢量分辨率(LAMVR)、重叠块的运动补偿(OBMC)、局部亮度补偿(LIC)和解码器侧运动矢量细化(DMVR)。

[0096] 2.1基于子CU的运动矢量预测的示例

[0097] 在具有四叉树加二叉树(QTBT)的JEM中,每个CU可以具有针对每个预测方向的至多一个运动参数的集合。在一些实施例中,通过将大CU划分为子CU并且导出大CU的所有子CU的运动信息,在编码器中考虑两个子CU级别运动矢量预测方法。可选时域运动矢量预测(ATMVP)方法允许每个CU从比共位参考图片中的当前CU小的多个块中获取运动信息的多个集合。在空域-时域运动矢量预测(STMVP)方法中,通过使用时域运动矢量预测和空域相邻运动矢量来递归地导出子CU的运动矢量。在一些实施例中,并且为了保留用于子CU运动预测的更准确的运动场,可以禁用参考帧的运动压缩。

[0098] 2.1.1可选时域运动矢量预测(ATMVP)的示例

[0099] 在ATMVP方法中,通过从小于当前CU的块中取回运动信息(包括运动矢量和参考索引)的多个集合来修改时域运动矢量预测(TMVP)方法。

[0100] 图10示出了CU 1000的ATMVP运动预测过程的示例。ATMVP方法以两个步骤预测CU 1000内的子CU 1001的运动矢量。第一步是使用时域矢量识别参考图片1050中的对应块1051。参考图片1050也称为运动源图片。第二步是将当前CU 1000划分为子CU 1001,并从对应于每个子CU的块中获得运动矢量以及每个子CU的参考索引。

[0101] 在第一步骤中,参考图片1050和对应块由当前CU 1000的空域相邻块的运动信息确定。为了避免相邻块的重复扫描过程,使用当前CU 1000的Merge候选列表中的第一Merge候选。第一可用运动矢量及其相关联的参考索引被设置为时域矢量和运动源图片的索引。这样,与TMVP相比,可以更准确地识别对应块,其中对应块(有时称为共位块)总是相对于当前CU处于右下或中心位置。

[0102] 在第二步骤中,通过向当前CU的坐标添加时域矢量,由运动源图片1050中的时域矢量识别子CU 1051的对应块。对于每个子CU,其对应块的运动信息(例如,覆盖中心样本的最小运动网格)用于导出子CU的运动信息。在识别出对应的 $N \times N$ 块的运动信息之后,以与HEVC的TMVP相同的方式将其转换为当前子CU的运动矢量和参考索引,在该方式中应用运动缩放和其它过程。例如,解码器校验是否满足低延迟条件(例如,当前图片的所有参考图片的POC小于当前图片的POC)并且可能使用运动矢量 MV_x (例如,对应于参考图片列表X的运动矢量)用于预测每个子CU的运动矢量 MV_y (例如,其中X等于0或1并且Y等于1-X)。

[0103] 2.1.2空域-时域运动矢量预测(STMVP)的示例

[0104] 在STMVP方法中,按照光栅扫描顺序递归地导出子CU的运动矢量。图11示出了具有四个子块和相邻块的一个CU的示例。考虑包括四个 4×4 子CU A(1101)、B(1102)、C(1103)和D(1104)的 8×8 CU 1100。当前帧中的相邻 4×4 块标记为(1111)、b(1112)、c(1113)和d(1114)。

[0105] 子CU A的运动推导通过识别其两个空域邻域(neighbor)开始。第一邻域是子CU A 1101上方的 $N \times N$ 块(块c 1113)。如果该块c(1113)不可用或者是帧内编解码,则校验(从左到右,从块c 1113开始)子CU A(1101)上方的其它 $N \times N$ 块。第二邻域是子CU A 1101左侧的块(块b 1112)。如果块b(1112)不可用或者是帧内编解码,则校验子CU A 1101左侧的其它块(从上到下,从块b 1112开始)。对每个列表从相邻块获得的运动信息被缩放到给定列表的第一参考帧。接下来,通过遵循与HEVC中指定的TMVP推导相同的过程来导出子块A 1101的时域运动矢量预测(TMVP)。在块D 1104处的共位块的运动信息被相应地取回和缩放。最

后,在提取和缩放运动信息之后,对每个参考列表分开平均所有可用运动矢量。平均运动矢量被指定为当前子CU的运动矢量。

[0106] 2.1.3子CU运动预测模式信令通知的示例

[0107] 在一些实施例中,子CU模式被启用作为附加的Merge候选,并且不需要附加的语法元素来信令通知模式。添加两个附加的Merge候选以合并每个CU的候选列表以表示ATMVP模式和STMVP模式。在其它实施例中,如果序列参数集指示启用了ATMVP和STMVP,则可以使用多达七个Merge候选。附加Merge候选的编码逻辑与HM中的Merge候选的编码逻辑相同,这意味着,对于P或B条带中的每个CU,两个附加Merge候选可能需要再进行两次RD校验。在一些实施例中,例如JEM,Merge索引的所有二进制位由CABAC(基于上下文的自适应二进制算术编解码)进行上下文编解码。在其它实施例中,例如,HEVC,仅对第一个二进制数(bin)进行上下文编解码,并且对剩余的二进制数进行上下文旁路编解码。

[0108] 2.2自适应运动矢量差异分辨率的示例

[0109] 在一些实施例中,当条带报头中的use_integer_mv_flag等于0时,以四分之一亮度样本为单位,信令通知(PU的运动矢量与预测的运动矢量之间的)运动矢量差(MVD)。在JEM中,引入了局部自适应运动矢量分辨率(LAMVR)。在JEM中,MVD可以以四分之一亮度样本、整数亮度样本或四个亮度样本为单位进行编解码。在编解码单元(CU)级别控制MVD分辨率,并且向具有至少一个非零MVD分量的每个CU,有条件地信令通知MVD分辨率标志。

[0110] 对于具有至少一个非零MVD分量的CU,信令通知第一标志,以指示是否在CU中使用四分之一亮度样本MV精度。当第一标志(等于1)指示未使用四分之一亮度样本MV精度时,信令通知另一个标志,以指示是否使用整数亮度样本MV精度或四个亮度样本MV精度。

[0111] 当CU的第一MVD分辨率标志为零,或未针对CU编解码(意味着CU中的所有MVD均为零)时,四分之一亮度样本MV分辨率被用于该CU。当CU使用整数亮度样本MV精度或四个亮度样本MV精度时,该CU的AMVP候选列表中的MVP被取整到对应的精度。

[0112] 在编码器中,使用CU级别RD校验来确定将哪个MVD分辨率将用于CU。换言之,对于每个MVD分辨率,执行CU级RD校验三次。为了加快编码器速度,在JEM中应用以下编码方案。

[0113] • 在具有正常四分之一亮度样本MVD分辨率的CU的RD校验期间,存储该当前CU的运动信息(整数亮度样本精度)。对于具有整数亮度样本和4个亮度样本MVD分辨率的相同CU,存储的运动信息(取整后)被用作RD校验期间进一步小范围运动矢量细化的起点,使得耗时的运动估计过程不重复三次。

[0114] • 有条件地调用具有4个亮度样本MVD分辨率的CU的RD校验。对于CU,当整数亮度样本MVD分辨率的RD成本远大于四分之一亮度样本MVD分辨率的RD成本时,跳过该CU的4个亮度样本MVD分辨率的RD校验。

[0115] 2.3较高的运动矢量存储精度的示例

[0116] 在HEVC中,运动矢量精度是四分之一像素(pe1)(用于4:2:0视频的四分之一亮度样本和八分之一色度样本)。在JEM中,内部运动矢量存储和Merge候选的精度增加到1/16像素。较高的运动矢量精度(1/16像素)用于以跳过模式/Merge模式编解码的CU的运动补偿帧间预测。对于使用正常AMVP模式编解码的CU,使用整数像素或四分之一像素运动。

[0117] 具有与HEVC运动补偿插值滤波器相同的滤波器长度和归一化因子的SHVC上采样插值滤波器,被用作附加分数像素位置的运动补偿插值滤波器。在JEM中色度分量运动矢量

精度是1/32样本,通过使用两个相邻1/16像素分数位置的滤波器的平均值,来导出1/32像素分数位置的附加插值滤波器。

[0118] 2.4重叠块的运动补偿(OBMC)的示例

[0119] 在JEM中,OBMC可以在CU级别使用语法元素进行开关。当JEM中使用OBMC时,对除去CU的右侧边界和底部边界的所有运动补偿(MC)块边界执行OBMC。此外,它应用于亮度和色度分量。JEM中,MC块对应于编解码块。当使用子CU模式(包括子CU Merge、仿射以及FRUC模式)对CU进行编解码时,CU的每个子块是MC块。为了统一处理CU的边界,在子块的尺寸设置为4x4的情况下,对所有MC块边界以子块级别执行OBMC,如图12A和图12B所示。

[0120] 图12A示出了CU/PU边界处的子块,并且阴影子块是应用OBMC的位置。类似地,图12B示出了ATMVP模式中的子PU。

[0121] 当OBMC应用于当前子块时,除了当前MV之外,四个相邻子块的矢量(如果可用且与当前运动矢量不完全相同)也会被用于导出当前子块的预测块。组合这些基于多个运动矢量的多个预测块以生成当前子块的最终预测信号。

[0122] 基于相邻子块的运动矢量的预测块被表示为PN,其中N指示相邻上方、下方、左侧和右侧的子块的索引,并且基于当前子块的运动矢量的预测块被表示为PC。当PN基于相邻子块的运动信息且该运动信息与当前子块的运动信息相同时,不从PN执行OBMC。否则,将PN的每个样本添加到PC中的相同样本中,即将四行/列PN添加到PC。加权因子{1/4,1/8,1/16,1/32}用于PN,并且加权因子{3/4,7/8,15/16,31/32}用于PC。例外是对于小MC块(即,当编解码块的高度或宽度等于4或CU使用子CU模式编解码时)仅将PN的两行/列添加到PC。在这种情况下,加权因子{1/4,1/8}用于PN,并且加权因子{3/4,7/8}用于PC。对于基于垂直(水平)相邻子块的运动矢量生成的PN,将PN的相同行(列)中的样本添加到具有相同加权因子的PC。

[0123] 在JEM中,对于尺寸小于或等于256个亮度样本的CU,信令通知CU级别标志以指示是否对当前CU应用OBMC。对于尺寸大于256个亮度样本或未使用AMVP模式编解码的CU,默认情况下应用OBMC。在编码器处,当OBMC应用于CU时,在运动估计阶段期间考虑其影响。由OBMC使用顶部相邻块和左相邻块的运动信息形成的预测信号用于补偿当前CU的原始信号的顶部边界和左侧边界,并且然后应用正常运动估计过程。

[0124] 2.5局部亮度补偿(LIC)的示例

[0125] LIC基于用于亮度变化的线性模型,使用缩放因子a和偏移b。并且,针对每个帧间模式编解码的编解码单元(CU)自适应地启用或禁用LIC。

[0126] 当LIC应用于CU时,采用最小平方误差方法,通过使用当前CU的相邻样本及其对应的参考样本来导出参数a和b。图13示出了用于导出IC算法的参数的相邻样本的示例。具体地,并且如图13所示,使用了该CU的子采样(2:1子采样)的相邻样本和参考图片中的对应样本(其由当前CU或子CU的运动信息识别)。IC参数被导出并被分别应用于每个预测方向。

[0127] 当使用Merge模式对CU进行编解码时,以类似于Merge模式中的运动信息复制的方式从相邻块复制LIC标志;否则,向该CU信令通知LIC标志以指示LIC是否适用。

[0128] 当对图片启用LIC时,需要附加的CU级别RD校验以确定是否对CU应用LIC。当对CU启用LIC时,分别对整数像素运动搜索和分数像素运动搜索,使用均值移除的绝对差和(mean-removed sum of absolute difference, MR-SAD)以及均值移除的绝对哈达玛变

换差和(mean-removed sum of absolute Hadamard-transformed difference, MR-SATD), 而不是SAD和SATD。

[0129] 为了降低编码复杂度,在JEM中应用以下编码方案。

[0130] 当当前图片与其参考图片之间不存在明显的亮度变化时,对整个图片禁用LIC。为了识别这种情况,在编码器处,计算当前图片与该当前图片的每个参考图片的直方图。如果当前图片与该当前图片的每个参考图片之间的直方图差小于给定阈值,则对当前图片禁用LIC;否则,对当前图片启用LIC。

[0131] 2.6仿射运动补偿预测的示例

[0132] 在HEVC中,仅将平移运动模型应用于运动补偿预测(MCP)。然而,相机和对象可以具有多种运动,例如放大/缩小、旋转、透视运动和其它不规则的运动。在另一方面,JEM应用简化的仿射变换运动补偿预测。图14示出了由两个控制点运动矢量 v_0 和 v_1 描述的块1400的仿射运动场的示例。块1400的运动矢量场(MVF)可以由以下等式描述:

$$[0133] \quad \begin{cases} v_x = \frac{(v_{1x} - v_{0x})}{w}x - \frac{(v_{1y} - v_{0y})}{w}y + v_{0x} \\ v_y = \frac{(v_{1y} - v_{0y})}{w}x + \frac{(v_{1x} - v_{0x})}{w}y + v_{0y} \end{cases} \quad (1)$$

[0134] 如图14所示, (v_{0x}, v_{0y}) 是左顶角控制点的运动矢量,并且 (v_{1x}, v_{1y}) 是右顶角控制点的运动矢量。为了简化运动补偿预测,可以应用基于子块的仿射变换预测。子块尺寸 $M \times N$ 如下导出:

$$[0135] \quad \begin{cases} M = \text{clip3}(4, w, \frac{w \times MvPre}{\max(\text{abs}(v_{1x} - v_{0x}), \text{abs}(v_{1y} - v_{0y}))}) \\ N = \text{clip3}(4, h, \frac{h \times MvPre}{\max(\text{abs}(v_{2x} - v_{0x}), \text{abs}(v_{2y} - v_{0y}))}) \end{cases} \quad (2)$$

[0136] 其中MvPre是运动矢量分数精度(例如,在JEM中为1/16)。 (v_{2x}, v_{2y}) 是左下控制点的运动矢量,根据等式(1)计算。如果需要,可以向下调整M和N,以使其分别为w和h的除数。

[0137] 图15示出了块1500的每个子块的仿射MVF的示例。为了导出每个 $M \times N$ 子块的运动矢量,可以根据等式(1)计算每个子块的中心样本的运动矢量,并将其取整到运动矢量分数精度(例如,在JEM中为1/16)。然后,可以应用运动补偿插值滤波器,以利用导出的运动矢量生成每个子块的预测。在MCP之后,每个子块的高精度运动矢量以与正常运动矢量相同的精度被取整并保存。

[0138] 2.6.1AF_INTER模式的实施例

[0139] 在JEM中,存在两种仿射运动模式:AF_INTER模式和AF_MERGE模式。对于宽度和高度均大于8的CU,可以应用AF_INTER模式。在比特流中信令通知CU级别中的仿射标志,以指示是否使用AF_INTER模式。在AF_INTER模式中,使用相邻块构建具有运动矢量对 $\{(v_0, v_1) \mid v_0 = \{v_A, v_B, v_C\}, v_1 = \{v_D, v_E\}\}$ 的候选列表。

[0140] 图16示出了用于在AF_INTER模式中的块1600的运动矢量预测(MVP)的示例。如图16所示,从子块A、B或C的运动矢量中选择 v_0 。来自相邻块的运动矢量可以根据参考列表来缩放。运动矢量还可以根据相邻块的参考的图片顺序计数(POC)、当前CU的参考的POC和当

前CU的POC之间的关系来缩放。从相邻子块D和E中选择 v_1 的方法是类似的。如果候选列表的数量小于2,则由通过重复每个AMVP候选而构建的运动矢量对来填充该列表。当候选列表大于2时,首先可以根据相邻运动矢量(例如,基于候选对中的两个运动矢量的相似性)对候选进行分类。在一些实现方式中,保留前两个候选。在一些实现方式中,使用速率失真(RD)成本校验来确定选择哪个运动矢量对候选作为当前CU的控制点运动矢量预测(CPMVP)。可以在比特流中信令通知指示候选列表中的CPMVP的位置的索引。在确定当前仿射CU的CPMVP之后,应用仿射运动估计,并找到控制点运动矢量(CPMV)。然后在比特流中信令通知CPMV与CPMVP的差。

[0141] 在AF_INTER模式中,当使用4/6参数仿射模式时,需要2/3控制点,并且因此需要为这些控制点编解码2/3MVD,如图17所示。在现有实现方式中[5],MV可以如下导出,例如,它从mvd0预测mvd1和mvd2。

$$[0142] \quad mv_0 = \overline{mv}_0 + mvd_0$$

$$[0143] \quad mv_1 = \overline{mv}_1 + mvd_1 + mvd_0$$

$$[0144] \quad mv_2 = \overline{mv}_2 + mvd_2 + mvd_0$$

[0145] 在本文中, $\overline{mv}_i$ 、 mvd_i 和 mv_i 分别是左顶像素($i=0$)、右顶像素($i=1$)或左底像素($i=2$)的预测运动矢量、运动矢量差和运动矢量,如图18B所示。在一些实施例中,两个运动矢量(例如, $mvA(xA, yA)$ 和 $mvB(xB, yB)$)的相加等于单独地两个分量的求和。例如, $newMV = mvA + mvB$ 意味着 $newMV$ 的两个分量分别设置为 $(xA + xB)$ 和 $(yA + yB)$ 。

[0146] 2.6.2AF_INTER模式中的快速仿射ME算法的示例

[0147] 在仿射模式的一些实施例中,需要联合确定2个或3个控制点的MV。直接联合搜索多个MV在计算上是复杂的。在示例中,快速仿射ME算法[6]被提出并被采用到VTM/BMS中。

[0148] 例如,针对4参数仿射模型描述快速仿射ME算法,并且该想法可以扩展到6参数仿射模型:

$$[0149] \quad \begin{cases} x' = ax + by + c \\ y' = -bx + ay + d \end{cases} \quad (3)$$

$$[0150] \quad \begin{cases} mv_{(x,y)}^h = x' - x = (a-1)x + by + c \\ mv_{(x,y)}^v = y' - y = -bx + (a-1)y + d \end{cases} \quad (4)$$

[0151] 使用 a' 替换 $(a-1)$ 使得运动矢量能够被重写为:

$$[0152] \quad \begin{cases} mv_{(x,y)}^h = x' - x = a'x + by + c \\ mv_{(x,y)}^v = y' - y = -bx + a'y + d \end{cases} \quad (5)$$

[0153] 如果假设两个控制点 $(0,0)$ 和 $(0,w)$ 的运动矢量是已知的,则根据等式(5),仿射参数可以被导出为:

$$[0154] \quad \begin{cases} c = mv_{(0,0)}^h \\ d = mv_{(0,0)}^v \end{cases} \quad (6)$$

[0155] 可以以矢量形式将运动矢量重写为:

[0156] $MV(p) = A(P) * MV_C^T$ (7)

[0157] 在本文中, $P = (x, y)$ 是像素位置,

[0158] $A(P) = \begin{bmatrix} 1 & x & 0 & y \\ 0 & y & 1 & -x \end{bmatrix}$ (8)

[0159] 并且

[0160] $MV_C = [mv_{(0,0)}^h \ a \ mv_{(0,0)}^v \ b]$ (9)

[0161] 在一些实施例中, 并且在编码器处, 可以迭代地导出AF_INTER的MVD。将 $MV^i(P)$ 表示为在位置P的第i次迭代中导出的MV, 并且将 dMV_C^i 表示为在第i次迭代中针对MVC更新的增量。然后在第(i+1)次迭代中,

[0162]
$$\begin{aligned} MV^{i+1}(P) &= A(P) * ((MV_C^i)^T + (dMV_C^i)^T) \\ &= A(P) * (MV_C^i)^T + A(P) * (dMV_C^i)^T \\ &= MV^i(P) + A(P) * (dMV_C^i)^T \end{aligned}$$
 (10)

[0163] 将 Pic_{ref} 表示为参考图片并将 Pic_{cur} 表示为当前图片并表示 $Q = P + MV^i(P)$ 。如果MSE用作匹配准则, 则需要最小化的函数可以写为:

[0164]
$$\begin{aligned} \min_P \sum_P (Pic_{cur}(P) - Pic_{ref}(P + MV^{i+1}(P)))^2 \\ = \min_P \sum_P (Pic_{cur}(P) - Pic_{ref}(Q + A(P) * (dMV_C^i)^T))^2 \end{aligned}$$
 (11)

[0165] 如果假设 $(dMV_C^i)^T$ 足够小, 则可以基于1阶泰勒展开将 $Pic_{ref}(Q + A(P) * (dMV_C^i)^T)$ 重写为近似值, 如:

$$Pic_{ref}(Q + A(P) * (dMV_C^i)^T) \approx Pic_{ref}(Q) + Pic_{ref}'(Q) * A(P) * (dMV_C^i)^T \quad (12)$$

[0166] 在本文中, $Pic_{ref}'(Q) = \left[\frac{dPic_{ref}(Q)}{dx} \ \frac{dPic_{ref}(Q)}{dy} \right]$ 。如果采用记号法 $E^{i+1}(P) = Pic_{cur}(P) - Pic_{ref}(Q)$, 则

[0167]
$$\begin{aligned} \min_P \sum_P (Pic_{cur}(P) - Pic_{ref}(Q) - Pic_{ref}'(Q) * A(P) * (dMV_C^i)^T)^2 \\ = \min_P \sum_P (E^{i+1}(P) - Pic_{ref}'(Q) * A(P) * (dMV_C^i)^T)^2 \end{aligned}$$
 (13)

[0168] 术语 dMV_C^i 可以通过将误差函数的导数设置为零, 并且然后根据 $A(P) * (dMV_C^i)^T$ 计算控制点(0,0)和(0,w)的增量MV而导出, 如下:

[0169] $dMV_{(0,0)}^h = dMV_C^i[0]$ (14)

[0170] $dMV_{(0,w)}^h = dMV_C^i[1] * w + dMV_C^i[2]$ (15)

$$[0171] \quad dMV_{(0,0)}^v = dMV_C^i[2] \quad (16)$$

$$[0172] \quad dMV_{(0,w)}^v = -dMV_C^i[3] * w + dMV_C^i[2] \quad (17)$$

[0173] 在一些实施例中,该MVD推导过程可以迭代n次,并且最终MVD可以如下计算:

$$[0174] \quad fdMV_{(0,0)}^h = \sum_{i=0}^{n-1} dMV_C^i[0] \quad (18)$$

$$[0175] \quad fdMV_{(0,w)}^h = \sum_{i=0}^{n-1} dMV_C^i[1] * w + \sum_{i=0}^{n-1} dMV_C^i[0] \quad (19)$$

$$[0176] \quad fdMV_{(0,0)}^v = \sum_{i=0}^{n-1} dMV_C^i[2] \quad (20)$$

$$[0177] \quad fdMV_{(0,w)}^v = \sum_{i=0}^{n-1} -dMV_C^i[3] * w + \sum_{i=0}^{n-1} dMV_C^i[2] \quad (21)$$

[0178] 在上述实现方式[5]中,从由mvd₀表示的控制点(0,0)的增量MV预测由mvd₁表示的预测控制点(0,w)的增量MV导致 $(\sum_{i=0}^{n-1} dMV_C^i[1] * w, -\sum_{i=0}^{n-1} -dMV_C^i[3] * w)$ 仅针对mvd₁而被编码。

[0179] 2.6.3AF_MERGE模式的实施例

[0180] 当在AF_MERGE模式中应用CU时,它从有效的相邻重建块获得使用仿射模式编解码的第一块。图18A示出了当前CU 1800的候选块的选择顺序的示例。如图18A所示,选择顺序可以是当前CU 1800的左方(1801)、上方(1802)、右上方(1803)、左下方(1804)到左上方(1805)。图18B示出了在AF_MERGE模式中的当前CU 1800的候选块的另一示例。如图18B所示,如果相邻左下块1801以仿射模式编解码,则导出包含子块1801的CU的左顶角、右上角和左底角的运动矢量v₂、v₃和v₄。基于v₂、v₃和v₄来计算当前CU 1800的左顶角的运动矢量v₀。可以相应地计算当前CU的右上方的运动矢量v₁。

[0181] 在根据等式(1)中的仿射运动模型计算当前CU v₀和v₁的CPMV之后,可以生成该当前CU的MVF。为了识别当前CU是否使用AF_MERGE模式编解码,当存在至少一个相邻块以仿射模式编解码时,可以在比特流中信令通知仿射标志。

[0182] 2.7模式匹配的运动矢量推导(PMMVD)的示例

[0183] PMMVD模式是一种基于帧速率上转换(FRUC)方法的特殊Merge模式。使用该模式,块的运动信息不被信令通知,而是在解码器侧导出。

[0184] 当CU的Merge标志为真时,可以向该CU信令通知FRUC标志。当FRUC标志为假时,可以信令通知Merge索引,并使用常规Merge模式。当FRUC标志为真时,可以信令通知附加的FRUC模式标志以指示将使用哪种方法(例如,双边匹配或模板匹配)来导出该块的运动信息。

[0185] 在编码器侧,关于是否对CU使用FRUC Merge模式的决定基于如对正常Merge候选那样所做的RD成本选择。例如,通过使用RD成本选择来校验CU的多种匹配模式(例如,双边匹配和模板匹配)。导致最小成本的匹配模式与其它CU模式进一步比较。如果FRUC匹配模式是最有效的模式,则对于CU将FRUC标志设置为真,并且使用有关匹配模式。

[0186] 通常,FRUC Merge模式中的运动推导过程有两个步骤。首先执行CU级别运动搜索,接下来执行子CU级别运动细化。在CU级别,基于双边匹配或模板匹配为整个CU导出初始运动矢量。首先,生成MV候选列表,并且选择导致最小匹配成本的候选作为进一步CU级别细化

的起点。然后,围绕起始点执行基于双边匹配或模板匹配的局部搜索。将导致最小匹配成本的MV作为整个CU的MV。随后,运动信息在子CU级别进一步细化,其中导出的CU运动矢量作为起点。

[0187] 例如,针对 $W \times H$ CU运动信息推导执行以下推导处理。在第一阶段,导出整体 $W \times H$ CU的MV。在第二阶段,CU进一步划分为 $M \times M$ 子CU。如等式(3)中计算M的值,D是预定义的划分深度,其在JEM中默认设置为3。然后导出每个子CU的MV。

$$[0188] \quad M = \max \{4, \min \left\{ \frac{M}{2^D}, \frac{N}{2^D} \right\} \} \quad (3)$$

[0189] 图19示出了帧速率上转换(FRUC)方法中使用的双边匹配的示例。双边匹配用于通过在两个不同参考图片(1910,1911)中沿当前CU(1900)的运动轨迹找到两个块之间的最接近匹配,来导出当前CU的运动信息。在连续运动轨迹的假设下,指向两个参考块的运动矢量MV0(1901)和MV1(1902)与在当前图片和两个参考图片之间的时域距离——例如TD0(1903)和TD1(1904)——成比例。在一些实施例中,当当前图片(1900)在时域上在两个参考图片(1910,1911)之间并且从当前图片到两个参考图片的时域距离相同时,双边匹配变为基于镜像的双向MV。

[0190] 图20示出了帧速率上转换(FRUC)方法中使用的模板匹配的示例。模板匹配可以用于通过找到在当前图片中的模板(例如,当前CU的顶部相邻块和/或左方相邻块)与参考图片2010中的块(例如,具有与模板相同的尺寸)之间的最接近匹配,来导出当前CU 2000的运动信息。除了上述FRUC Merge模式之外,模板匹配也可以适用于AMVP模式。在JEM和HEVC两者中,AMVP有两个候选。使用模板匹配方法,可以导出新的候选。如果由模板匹配的新导出的候选与第一现有AMVP候选不同,则将其插入AMVP候选列表的最开始,并且然后将列表尺寸设置为2(例如,通过移除第二现有AMVP候选)。当应用于AMVP模式时,仅应用CU级别搜索。

[0191] CU级别的MV候选集合可以包括:(1)如果当前CU处于AMVP模式,则为原始AMVP候选,(2)所有Merge候选,(3)插值MV场(稍后描述)中的数个MV,以及顶部和左方相邻的运动矢量。

[0192] 当使用双边匹配时,Merge候选的每个有效MV可以被用作输入,以在假设双边匹配的情况下生成MV对。例如,Merge候选的一个有效MV是在参考列表A中的 (MV_a, ref_a) 。然后,在其它参考列表B中找到其配对双边MV的参考图片 ref_b ,使得 ref_a 和 ref_b 在时域上位于当前图片的不同侧。如果参考列表B中这样的 ref_b 不可用,则 ref_b 被确定为与 ref_a 不同的参考,并且 ref_b 到当前图片的时域距离是列表B中的最小的一个。在确定 ref_b 之后,基于当前图片与 ref_a 、 ref_b 之间的时域距离,通过缩放 MV_a 来导出 MV_b 。

[0193] 在一些实现方式中,来自插值MV场的四个MV也可以被添加到CU级别候选列表。更具体地,添加当前CU的位置(0,0)、(W/2,0)、(0,H/2)和(W/2,H/2)处的插值MV。当FRUC应用于AMVP模式时,原始AMVP候选也被添加到CU级别MV候选集合。在一些实现方式中,在CU级别,用于AMVP CU的15个MV、用于Merge CU的13个MV可以被添加到候选列表。

[0194] 子CU级别的MV候选集合包括:(1)从CU级别搜索确定的MV,(2)顶部、左方、左顶和右顶的相邻MV,(3)来自参考图片的并列MV的缩放版本,(4)一个或多个ATMVP候选(例如,最多四个),以及(5)一个或多个STMVP候选(例如,最多四个)。来自参考图片的缩放MV如下导出。遍历两个列表中的参考图片。参考图片中的子CU的并列位置处的MV被缩放到起始CU级

别MV的参考。ATMVP和STMVP候选可以是前四个候选。在子CU级别,一个或多个MV(例如,最多十七个)被添加到候选列表中。

[0195] 插值MV场的生成。在对帧进行编解码之前,基于单边ME为整个图片生成插值运动场。然后,运动场可以稍后用作CU级别或子CU级别MV候选。

[0196] 在一些实施例中,两个参考列表中的每个参考图片的运动场以 4×4 块级别遍历。图21示出了在FRUC方法中的单向运动估计(ME)2100的示例。对于每个 4×4 块,如果与块相关联的运动通过当前图片中的 4×4 块并且该块尚未被分配任何插值运动,则参考块的运动根据时域距离TD0和TD1(与HEVC中的TMVP的MV缩放的方式相同的方式)缩放到当前图片,并且将缩放的运动分配给当前帧中的块。如果无缩放的MV被分配到 4×4 块,则在插值的运动场中将块的运动标记为不可用。

[0197] 插值和匹配成本。当运动矢量指向分数样本位置时,需要运动补偿插值。为了降低复杂度,双边匹配和模板匹配两者可以使用双线性插值而不是常规的8抽头HEVC插值。

[0198] 匹配成本的计算在不同的步骤略有不同。当从CU级别的候选集合中选择候选时,匹配成本可以是双边匹配或模板匹配的绝对差之和(SAD)。在确定起始MV之后,如下计算子CU级别搜索的双边匹配的匹配成本C:

$$[0199] \quad C = SAD + w \cdot (|MV_x - MV_x^s| + |MV_y - MV_y^s|) \quad (4)$$

[0200] 其中w是一个加权因子。在一些实施例中,w根据经验设置为4,MV和 MV^s 分别指示当前MV和起始MV。SAD可以仍用作子CU级别搜索的模板匹配的匹配成本。

[0201] 在FRUC模式中,仅通过使用亮度样本来导出MV。导出的运动将用于MC帧间预测的亮度和色度两者。在确定MV之后,使用用于亮度的8抽头插值滤波器和用于色度的4抽头插值滤波器来执行最终MC。

[0202] MV细化是以双边匹配成本或模板匹配成本为准则的基于模式的MV搜索。在JEM中,支持两种搜索模式——分别用于CU级别和子CU级别的MV细化的无限制的中心偏置菱形搜索(unrestricted center-biased diamond search,UCBDS)和自适应交叉搜索(adaptive cross search)。对于CU级别和子CU级别MV细化,以四分之一亮度样本MV精度直接搜索MV,并且接下来以八分之一亮度样本MV细化。对于CU步骤和子CU步骤的MV细化的搜索范围被设置为等于8个亮度样本。

[0203] 在双边匹配Merge模式中,应用双向预测,因为基于在两个不同参考图片中沿当前CU的运动轨迹的两个块之间的最接近匹配来导出CU的运动信息。在模板匹配Merge模式中,编码器可以在针对CU的来自列表0的单向预测、来自列表1的单向预测或者双向预测之中进行选择。选择可以基于模板匹配成本,如下:

[0204] 如果 $\text{costBi} \leq \text{factor} * \min(\text{cost0}, \text{cost1})$

[0205] 使用双向预测;

[0206] 否则,如果 $\text{cost0} \leq \text{cost1}$

[0207] 使用来自列表0的单向预测;

[0208] 否则,

[0209] 使用来自列表1的单向预测;

[0210] 其中cost0是列表0模板匹配的SAD,cost1是列表1模板匹配的SAD,并且costBi是

双向预测模板匹配的SAD。例如,当factor的值等于1.25时,这意味着选择过程偏向于双向预测。帧间预测方向选择可以应用于CU级别模板匹配过程。

[0211] 2.8双向光流 (BIO) 的示例

[0212] 双向光流 (BIO) 方法是在用于双向预测的逐块运动补偿之上执行的逐样本运动细化。在一些实现方式中,样本级别运动细化不使用信令通知。

[0213] 让 $I^{(k)}$ 是在块运动补偿之后的参考 k ($k = 0, 1$) 的亮度值,并且将 $\partial I^{(k)} / \partial x$ 、 $\partial I^{(k)} / \partial y$ 分别表示为 $I^{(k)}$ 梯度的水平分量和垂直分量。假设光流有效,则运动矢量场 (v_x , v_y) 由以下等式给定。

$$[0214] \quad \partial I^{(k)} / \partial t + v_x \partial I^{(k)} / \partial x + v_y \partial I^{(k)} / \partial y = 0 \quad (5)$$

[0215] 将此光流等式与用于每个样本的运动轨迹的埃尔米特 (Hermite) 插值组合,其结果是在末端匹配函数值 $I^{(k)}$ 和导数 $\partial I^{(k)} / \partial x$ 、 $\partial I^{(k)} / \partial y$ 的唯一的三阶多项式。在 $t = 0$ 时此多项式的值是BIO预测:

$$[0216] \quad pred_{BIO} = 1/2 \cdot \left(I^{(0)} + I^{(1)} + v_x / 2 \cdot \left(\tau_1 \partial I^{(1)} / \partial x - \tau_0 \partial I^{(0)} / \partial x \right) + v_y / 2 \cdot \left(\tau_1 \partial I^{(1)} / \partial y - \tau_0 \partial I^{(0)} / \partial y \right) \right) \quad (6)$$

[0217] 图22示出了双向光流 (BIO) 方法中的光流轨迹的示例。其中, τ_0 和 τ_1 表示到参考帧的距离。距离 τ_0 和 τ_1 基于 Ref_0 和 Ref_1 的POC来计算: $\tau_0 = POC(\text{当前}) - POC(Ref_0)$, $\tau_1 = POC(Ref_1) - POC(\text{当前})$ 。如果两个预测都来自相同的时域方向(既可以是来自过去的,也可以是来自未来的),则符号是不同的(例如, $\tau_0 \cdot \tau_1 < 0$)。在这种情况下,如果预测不是来自相同的时刻(例如, $\tau_0 \neq \tau_1$),则应用BIO。两个参考区域都具有非零运动(例如, $MVx_0, MVy_0, MVx_1, MVy_1 \neq 0$) 并且块运动矢量与时域距离(例如, $MVx_0 / MVx_1 = MVy_0 / MVy_1 = -\tau_0 / \tau_1$) 成比例。

[0218] 通过最小化点A和点B中的值之间的差 Δ 来确定运动矢量场 (v_x, v_y)。图23A和23B示出了使用不具有块扩展的双向光流 (BIO) 算法的示例快照。模型仅使用对于 Δ 的局部泰勒展开的第一线性项:

$$[0219] \quad \Delta = \left(I^{(0)} - I^{(1)}_0 + v_x \left(\tau_1 \partial I^{(1)} / \partial x + \tau_0 \partial I^{(0)} / \partial x \right) + v_y \left(\tau_1 \partial I^{(1)} / \partial y + \tau_0 \partial I^{(0)} / \partial y \right) \right) \quad (7)$$

[0220] 上述等式中的所有值都取决于表示为 (i', j') 的样本位置。假设运动在局部周围区域是一致的,可以在以当前预测点 (i, j) 为中心的 $(2M+1) \times (2M+1)$ 方形窗口 Ω 内最小化 Δ , 其中 M 等于2:

$$[0221] \quad (v_x, v_y) = \arg \min_{v_x, v_y} \sum_{[i', j'] \in \Omega} \Delta^2 [i', j'] \quad (8)$$

[0222] 对于此优化问题, JEM使用简化方法,首先在垂直方向上进行最小化,并且然后在水平方向上进行最小化。这导致

$$[0223] \quad v_x = (s_1 + r) > m? clip3 \left(-thBIO, thBIO, -\frac{s_3}{(s_1 + r)} \right) : 0 \quad (9)$$

$$[0224] \quad v_y = (s_5 + r) > m? \text{clip3} \left(-thBIO, thBIO, -\frac{s_6 - v_x s_2 / 2}{(s_5 + r)} \right) : 0 \quad (10)$$

[0225] 其中,

$$[0226] \quad \begin{aligned} s_1 &= \sum_{[i',j] \in \Omega} \left(\tau_1 \partial I^{(1)} / \partial x + \tau_0 \partial I^{(0)} / \partial x \right)^2; s_3 = \sum_{[i',j] \in \Omega} \left(I^{(1)} - I^{(0)} \right) \left(\tau_1 \partial I^{(1)} / \partial x + \tau_0 \partial I^{(0)} / \partial x \right); \\ s_2 &= \sum_{[i',j] \in \Omega} \left(\tau_1 \partial I^{(1)} / \partial x + \tau_0 \partial I^{(0)} / \partial x \right) \left(\tau_1 \partial I^{(1)} / \partial y + \tau_0 \partial I^{(0)} / \partial y \right); \\ s_5 &= \sum_{[i',j] \in \Omega} \left(\tau_1 \partial I^{(1)} / \partial y + \tau_0 \partial I^{(0)} / \partial y \right)^2; s_6 = \sum_{[i',j] \in \Omega} \left(I^{(1)} - I^{(0)} \right) \left(\tau_1 \partial I^{(1)} / \partial y + \tau_0 \partial I^{(0)} / \partial y \right) \end{aligned} \quad (11)$$

[0227] 为了避免除以零或非常小的值,在等式(28)和等式(29)中引入正则化参数r和m。

$$[0228] \quad r = 500 \cdot 4^{d-8} \quad (12)$$

$$[0229] \quad m = 700 \cdot 4^{d-8} \quad (13) \text{ 其中 } d \text{ 是视频样本的位深度。}$$

[0230] 为了保持BIO的存储器访问与常规双向预测运动补偿相同,针对当前块内的位置计算所有预测和梯度值 $I^{(k)}$ 、 $\partial I^{(k)} / \partial x$ 、 $\partial I^{(k)} / \partial y$ 。图23A示出了块2300外部的访问位置的示例。如图23A所示,在等式(9)中,以预测块的边界上的当前预测点为中心的 $(2M+1) \times (2M+1)$ 方形窗口 Ω 需要访问块外的位置。在JEM中,块外的 $I^{(k)}$ 、 $\partial I^{(k)} / \partial x$ 、 $\partial I^{(k)} / \partial y$ 的值被设置为等于块内最近的可用值。例如,这可以实现为填充区域2301,如图23B所示。

[0231] 使用BIO,可以针对每个样品来细化运动场是可能的。为了降低计算复杂度,可以在JEM中使用基于块的BIO设计。可以基于 4×4 块来计算运动细化。在基于块的BIO中,可以聚合 4×4 块中的所有样本在等式(9)中的 s_n 值,并且然后将聚合的 s_n 值用于 4×4 块的导出的BIO运动矢量偏移。更具体地,以下公式可以用于基于块的BIO推导:

$$[0232] \quad \begin{aligned} s_{1,b_k} &= \sum_{(x,y) \in b_k} \sum_{[i',j] \in \Omega(x,y)} \left(\tau_1 \partial I^{(1)} / \partial x + \tau_0 \partial I^{(0)} / \partial x \right)^2; s_{3,b_k} = \sum_{(x,y) \in b_k} \sum_{[i',j] \in \Omega} \left(I^{(1)} - I^{(0)} \right) \left(\tau_1 \partial I^{(1)} / \partial x + \tau_0 \partial I^{(0)} / \partial x \right); \\ s_{2,b_k} &= \sum_{(x,y) \in b_k} \sum_{[i',j] \in \Omega} \left(\tau_1 \partial I^{(1)} / \partial x + \tau_0 \partial I^{(0)} / \partial x \right) \left(\tau_1 \partial I^{(1)} / \partial y + \tau_0 \partial I^{(0)} / \partial y \right); \\ s_{5,b_k} &= \sum_{(x,y) \in b_k} \sum_{[i',j] \in \Omega} \left(\tau_1 \partial I^{(1)} / \partial y + \tau_0 \partial I^{(0)} / \partial y \right)^2; s_{6,b_k} = \sum_{(x,y) \in b_k} \sum_{[i',j] \in \Omega} \left(I^{(1)} - I^{(0)} \right) \left(\tau_1 \partial I^{(1)} / \partial y + \tau_0 \partial I^{(0)} / \partial y \right) \end{aligned} \quad (14)$$

[0233] 其中 b_k 表示属于预测块的第k个 4×4 块的样品的集合。由 $((s_{n,b_k}) \gg 4)$ 替换等式(28)和等式(29)中的 s_n 以导出相关联的运动矢量偏移。

[0234] 在一些情况下,BIO的MV团(regiment)可能由于噪声或不规则运动而不可靠。因此,在BIO中,MV团的量级被修剪为阈值。基于当前图片的参考图片是否都来自一个方向来确定阈值。例如,如果当前图片的所有参考图片都来自一个方向,则阈值被设置为 $12 \times 2^{14-d}$;否则,它被设置为 $12 \times 2^{13-d}$ 。

[0235] 可以同时计算BIO的梯度与运动补偿插值,该运动补偿插值使用与HEVC运动补偿过程一致的操作(例如,2D可分离有限脉冲响应(FIR))。在一些实施例中,2D可分离FIR的输入是与根据块运动矢量的分数部分的分数位置(fracX, fracY)和运动补偿过程相同的参考帧样本。对于水平梯度 $\partial I / \partial x$,首先对应于具有去缩放偏移d-8的分数位置fracY,使用BIOfilterS对信号进行垂直地插值。然后对应于具有去缩放偏移18-d的分数位置fracX,在

水平方向上应用梯度滤波器BIOfilterG。对于垂直梯度 $\partial I / \partial y$,首先对应于具有去缩放偏移d-8的分数位置fracY,使用BIOfilterG垂直应用梯度滤波器。然后对应于具有去缩放偏移18-d的分数位置fracX,在水平方向上使用BIOfilterS来执行信号位移。用于梯度计算BIOfilterG和信号位移BIOfilterF的插值滤波器的长度可以较短(例如,6抽头)以维持合理的复杂度。表1示出了可以用于BI0中块运动矢量的不同分数位置的梯度计算的示例滤波器。表2示出了可以用于BI0中预测信号生成的示例插值滤波器。

[0236] 表1 BI0中梯度计算的示例性滤波器

[0237]	分数像素位置	梯度插值滤波器(BIOfilterG)
	0	{ 8, -39, -3, 46, -17, 5}
	1/16	{ 8, -32, -13, 50, -18, 5}
	1/8	{ 7, -27, -20, 54, -19, 5}
	3/16	{ 6, -21, -29, 57, -18, 5}
[0238]	1/4	{ 4, -17, -36, 60, -15, 4}
	5/16	{ 3, -9, -44, 61, -15, 4}
	3/8	{ 1, -4, -48, 61, -13, 3}
	7/16	{ 0, 1, -54, 60, -9, 2}
	1/2	{ -1, 4, -57, 57, -4, 1}

[0239] 表2 BI0中预测信号生成的示例性插值滤波器

[0240]	分数像素位置	预测信号的插值滤波器 (BIOfilterS)
	0	{0, 0, 64, 0, 0, 0}
	1/16	{1, -3, 64, 4, -2, 0}
	1/8	{1, -6, 62, 9, -3, 1}
	3/16	{2, -8, 60, 14, -5, 1}
	1/4	{2, -9, 57, 19, -7, 2}
	5/16	{3, -10, 53, 24, -8, 2}
	3/8	{3, -11, 50, 29, -9, 2}
	7/16	{3, -11, 44, 35, -10, 3}
	1/2	{3, -10, 35, 44, -11, 3}

[0241] 在JEM中,当两个预测来自不同的参考图片时,BI0可以应用于所有双向预测的块。当对CU启用局部亮度补偿(LIC)时,可以禁用BI0。

[0242] 在一些实施例中,在正常MC过程之后,对块应用OBMC。为了降低计算复杂度,在OBMC过程期间不可以应用BI0。这意味着当使用其自己的MV时BI0应用于块的MC过程,当在OBMC过程中使用相邻块的MV时BI0不应用于MC过程。

[0243] 2.9解码器侧运动矢量细化(DMVR)的示例

[0244] 在双向预测操作中,为了预测一个块区域,分别使用列表0的运动矢量(MV)和列表

1的MV形成的两个预测块被组合以形成单个预测信号。在解码器侧运动矢量细化(DMVR)方法中,双向预测的两个运动矢量通过双边模板匹配过程进一步细化。双边模板匹配应用于解码器中,以在双边模板和参考图片中的重建样本之间执行基于失真的搜索,以便在不传输附加的运动信息的情况下获得细化的MV。

[0245] 如图24所示,在DMVR中,双边模板分别从列表0的初始MV0和列表1的MV1生成成为两个预测块的加权组合(即平均)。模板匹配操作包括计算生成的模板与参考图片中的样本区域(在初始预测块周围)之间的成本度量。对于两个参考图片中的每一个,产生最小模板成本的MV被视为该列表的更新MV以替换原始模板。在JEM中,对于每个列表,搜索九个MV候选。该九个MV候选包括原始MV和8个周围MV,该周围的MV具有相对于原始MV的在水平方向上或垂直方向上或两者上的一个亮度样本的偏移。最后,如图24所示,两个新的MV,即MV0'和MV1',被用于生成最终的双向预测结果。绝对差之和(SAD)用作成本度量。

[0246] DMVR被应用于双向预测的Merge模式,在不传输附加的语法元素的情况下使用来自过去的参考图片中的一个MV和来自将来的参考图片中的另一个MV。在JEM中,当为CU启用LIC、仿射运动、FRUC或子CU Merge候选时,不应用DMVR。

[0247] 2.2.9对称运动矢量差异的示例

[0248] 在[8]中,提出对称运动矢量差(SMVD)以将MVD更有效地编码。

[0249] 首先,在条带级别中,变量BiDirPredFlag、RefIdxSymL0和RefIdxSymL1如下导出:

[0250] 搜索参考图片列表0中最接近当前图片的前向参考图片。如果找到,则将RefIdxSymL0设置为等于前向图片的参考索引。

[0251] 搜索参考图片列表1中最接近当前图片的后向参考图片。如果找到,则将RefIdxSymL1设置为等于后向图片的参考索引。

[0252] 如果前向和后向图片两者均被找到,则将BiDirPredFlag设置为等于1。

[0253] 否则,以下适用:

[0254] 搜索参考图片列表0中最接近当前参考图片的后向参考图片。如果找到,则将RefIdxSymL0设置为等于后向图片的参考索引。

[0255] 搜索参考图片列表1中最接近当前参考图片的前向参考图片。如果找到,则将RefIdxSymL1设置为等于前向图片的参考索引。

[0256] 如果前向和后向图片两者均被找到,则将BiDirPredFlag设置为等于1。否则,将BiDirPredFlag设置为等于0。

[0257] 其次,在CU级别,如果CU的预测方向是双向预测并且BiDirPredFlag等于1,则明确地信令通知对称模式标志,该对称模式标志指示是否使用对称模式。

[0258] 当该标志为真时,仅明确地信令通知mvp_l0_flag、mvp_l1_flag和MVD0。对于列表0和列表1,将参考索引分别设置为等于RefIdxSymL0、RefIdxSymL1。将MVD1设置为等于-MVD0。最终的运动矢量如下式所示。

$$[0259] \quad \begin{cases} (mvx_0, mvy_0) = (mvp_x_0 + mvd_x_0, mvp_y_0 + mvd_y_0) \\ (mvx_1, mvy_1) = (mvp_x_1 - mvd_x_0, mvp_y_1 - mvd_y_0) \end{cases}$$

[0260] 图27示出了对称模式的示例。

[0261] 编解码单元语法的修改在表3中示出。

[0262] 表3:在编解码单元语法的修改

[0263]

coding_unit(x0, y0, cbWidth, cbHeight, treeType) {	描述符
...	
if(slice_type == B)	
inter_pred_idc [x0][y0]	ae(v)
if(sps_affine_enabled_flag && cbWidth >= 16 && cbHeight >= 16) {	
inter_affine_flag [x0][y0]	ae(v)
if(sps_affine_type_flag && inter_affine_flag[x0][y0])	
cu_affine_type_flag [x0][y0]	ae(v)
}	
if(inter_pred_idc[x0][y0] == PRED_BI && BiDirPredFlag && inter_affine_flag[x0][y0] == 0)	
symmetric_mvd_flag [x0][y0]	ae(v)
if(inter_pred_idc[x0][y0] != PRED_L1) {	
if(num_ref_idx_l0_active_minus1 > 0 && !symmetric_mvd_flag[x0][y0])	
ref_idx_l0 [x0][y0]	ae(v)
mvd_coding(x0, y0, 0, 0)	
if(MotionModelIdc[x0][y0] > 0)	
mvd_coding(x0, y0, 0, 1)	
if(MotionModelIdc[x0][y0] > 1)	

[0264]

mvd_coding(x0, y0, 0, 2)	
mvp_l0_flag [x0][y0]	ae(v)
} else {	
MvdL0[x0][y0][0] = 0	
MvdL0[x0][y0][1] = 0	
}	
if(inter_pred_idc[x0][y0] != PRED_L0) {	
if(num_ref_idx_l1_active_minus1 > 0 && !symmetric_mvd_flag[x0][y0])	
ref_idx_l1 [x0][y0]	ae(v)
if(mvd_l1_zero_flag && inter_pred_idc[x0][y0] == PRED_BI) {	
...	
} else {	
if(!symmetric_mvd_flag[x0][y0]) {	
mvd_coding(x0, y0, 1, 0)	
if(MotionModelIdc[x0][y0] > 0)	
mvd_coding(x0, y0, 1, 1)	
if(MotionModelIdc[x0][y0] > 1)	
mvd_coding(x0, y0, 1, 2)	
}	
mvp_l1_flag [x0][y0]	ae(v)
} else {	
MvdL1[x0][y0][0] = 0	
MvdL1[x0][y0][1] = 0	
}	
...	
}	
}	
...	
}	

[0265] 2.10.1仿射双向预测编解码的对称MVD

[0266] 提议了仿射模式的SMVD,将对称MVD模式扩展到了仿射双向预测。当对仿射双向预测编解码应用对称MVD模式时,控制点的MVD不是信令发送,而是推导的。基于线性运动的假设,从List0中推导出List1的左上控制点的MVD。List0的其他控制点的MVD设置为0

[0267] 2.11基于上下文的自适应二进制算术编解码(CABAC)

[0268] 2.11.1HEVC中的CABAC设计

[0269] 2.11.1.1HEVC中的上下文表示和初始化过程

[0270] 在HEVC中,对于每个上下文变量,初始化两个变量pStateIdx和valMps。

[0271] 从8比特表条目initValue,两个4比特变量slopeIdx和offsetIdx如下导出:

$$\begin{aligned} \text{slopeIdx} &= \text{initValue} \gg 4 \\ \text{offsetIdx} &= \text{initValue} \& 15 \end{aligned} \quad (34)$$

[0273] 在上下文变量的初始化中使用的变量m和n从slopeIdx和offsetIdx如下导出:

$$\begin{aligned} \text{m} &= \text{slopeIdx} * 5 - 45 \\ \text{n} &= (\text{offsetIdx} \ll 3) - 16 \end{aligned} \quad (35)$$

[0276] 分配给pStateIdx和valMps用于初始化的两个值从由SliceQpY表示的条带的亮度的量化参数导出。给定变量m和n,初始化指定如下:

$$\begin{aligned} \text{preCtxState} &= \text{Clip3}(1, 126, ((\text{m} * \text{Clip3}(0, 51, \text{SliceQpY})) \gg 4) + \text{n}) \\ \text{valMps} &= (\text{preCtxState} \leq 63) ? 0 : 1 \\ \text{pStateIdx} &= \text{valMps} ? (\text{preCtxState} - 64) : (63 - \text{preCtxState}) \end{aligned} \quad (36)$$

[0278] 2.11.1.2HEVC中的状态转变过程

[0279] 该过程的输入是与ctxTable和ctxIdx相关联的上下文变量的当前pStateIdx、解码值binVal和valMps值。

[0280] 该过程的输出是与ctxIdx相关联的上下文变量的更新pStateIdx和更新valMps。

[0281] 取决于解码值binVal,与ctxIdx相关联的两个变量pStateIdx和valMps的更新如在(37)中导出:

$$\begin{aligned} &\text{if} (\text{binVal} == \text{valMps}) \\ &\quad \text{pStateIdx} = \text{transIdxMps} (\text{pStateIdx}) \\ &\text{else}\{ \\ &\quad \text{if} (\text{pStateIdx} == 0) \text{valMps} = 1 - \text{valMps} \\ &\quad \text{pStateIdx} = \text{transIdxLps} (\text{pStateIdx}) \\ &\quad \} \end{aligned} \quad (37)$$

[0283] 2.11.2VVC中的CABAC设计

[0284] VVC中的基于上下文的自适应二进制算术编解码器(BAC)已经在VVC中改变,其在上下文更新过程和算术编解码器方面与HEVC中的不同。

[0285] 以下是最近采用的提议(JVET-M0473,CE测试5.1.13)的概述。

[0286] 表4:VVC中CABAC修改的概述

[0287]	状态表示	10+14 比特线性, 缩小的范围
	rLPS 计算	5x4 比特乘法器
	初始化	将类似 HEVC 的状态映射到线性表示的 128x16 比特, 重新训练的初始化值
	速率估算	256x2x19 比特表
	窗口尺寸 (控制概率更新速度)	变量, 根据上下文定义 $a = 2..5$ $b = a + 3..6$ 换言之, 每个上下文都具有用于记录相关联概率的两个变量, 每个概率都以其自身的速度更新 (基于变量 a 的较快速度和基于变量 b 的较低速度)
	其它	保证 $rMPS \geq 128$

[0288] 2.11.2.1VVC中的上下文初始化过程

[0289] 在VVC中, 分配给pStateIdx0和pStateIdx1用于初始化的两个值从SliceQpY导出。给定变量m和n, 初始化指定如下:

[0290] $preCtxState = Clip3(0, 127, ((m * Clip3(0, 51, SliceQpY)) \gg 4) + n)$

[0291] $pStateIdx0 = initStateIdxToState[preCtxState] \gg 4$

[0292] $pStateIdx1 = initStateIdxToState[preCtxState] \quad (38)$

[0293] 2.11.2.2VVC中的状态转变过程

[0294] 该过程的输入是当前pStateIdx0和当前pStateIdx1以及解码值binVal。

[0295] 该过程的输出是与的ctxIdx相关联的上下文变量的更新pStateIdx0和更新pStateIdx1。

[0296] 变量shift0 (对应于VVC表4中的CABAC修改的概述中的变量a) 和shift1 (对应于VVC表4中的CABAC修改的概述中的变量b) 是从与ctxTable和ctxInc相关联的shiftIdx值导出的。

[0297]
$$\begin{aligned} shift0 &= (shiftIdx \gg 2) + 2 \\ shift1 &= (shiftIdx \& 3) + 3 + shift0 \end{aligned} \quad (39)$$

[0298] 取决于解码值binVal, 与ctxIdx相关联的两个变量pStateIdx0和pStateIdx1的更新如下导出:

[0299]
$$\begin{aligned} pStateIdx0 &= pStateIdx0 - (pStateIdx0 \gg shift0) + (1023 * binVal \gg shift0) \\ pStateIdx1 &= pStateIdx1 - (pStateIdx1 \gg shift1) + (16383 * binVal \gg shift1) \end{aligned} \quad (40)$$

[0300] 3. 现有实现方式的缺点

[0301] 在一些现有的实现方式中, 当可以从用于仿射编解码块的多个MV/MVD精度的集合中选择MV/MV差 (MVD) 时, 仍然无法确定如何获得更准确的运动矢量。

[0302] 在其它现有的实现方式中,MV/MVD精度信息在确定应用于仿射模式的AMVR的整体编解码增益方面也起着重要作用,但实现该目标仍然无法确定。

[0303] 4.具有AMVR的仿射模式的MV预测(MVP)的示例方法

[0304] 当前公开的技术的实施例克服了现有实现方式的缺点,从而提供具有更高编解码效率的视频编解码。基于所公开的技术,用于具有自适应运动矢量分辨率(AMVR)的仿射模式的运动矢量预测的推导和信令可以增强现有和未来的视频编解码标准,在以下针对各种实现方式所描述的示例中阐明。以下提供的所公开技术的示例解释了一般概念,并不意味着被解释为限制性的。在示例中,除非明确地相反指示,否则可以组合这些示例中描述的各种特征。

[0305] 在一些实施例中,当应用AMVR时,以下示例可以应用于仿射模式或正常模式。这些示例假设精度Prec(即,MV具有 $1/(2^{\text{Prec}})$ 精度)用于在AF_INTER模式中编码MVD或用于在正常帧间模式中编码MVD。运动矢量预测(例如,从相邻块MV继承)及其精度分别由MVPred(MVPred_X,MVPred_Y)和PredPrec表示。

[0306] 在以下讨论中,将SatShift(x,n)定义为

$$[0307] \quad \text{SatShift}(x,n) = \begin{cases} (x + \text{offset0}) \gg n & \text{if } x \geq 0 \\ -((-x + \text{offset1}) \gg n) & \text{if } x < 0 \end{cases} \quad (41)$$

[0308] Shift(x,n)定义为Shift(x,n) = (x+offset0) >> n。在一个示例中,将offset0和/或offset1设置为 $(1 \ll n) \gg 1$ 或 $(1 \ll (n-1))$ 。在另一示例中,offset0和/或offset1设置为0。在另一示例中,offset0=offset1 = $((1 \ll n) \gg 1) - 1$ 或 $((1 \ll (n-1))) - 1$ 。

[0309] 在下面的讨论中,两个运动矢量之间的操作意味着该操作将被应用于运动矢量的两个分量。例如,MV3=MV1+MV2等价于MV3_x=MV1_x+MV2_x和MV3_y=MV1_y+MV2_y。替代地,该操作可以仅应用于两个运动矢量的水平或垂直分量。

[0310] 示例1.提议最终的MV精度可以保持不变,即,与要存储的运动矢量的精度相同。

[0311] (a) 在一个示例中,最终的MV精度可以设置为1/16像素或1/8像素。

[0312] (b) 在一个示例中,可以首先缩放信令通知的MVD,并然后将其添加到MVP以形成一个块的最终的MV。

[0313] 示例2.可以首先修改直接从相邻块(例如,空域或时域)或默认MVP导出的MVP,并然后将其添加到信令通知的MVD中以形成(当前)块的最终的MV。

[0314] (a) 替代地,对于不同的Prec值,是否应用以及如何应用MVP的修改可能是不同的。

[0315] (b) 在一个示例中,如果Prec大于1(即,MVD具有分数精度),则相邻MV的精度不变,并且不进行缩放。

[0316] (c) 在一个示例中,如果Prec等于1(即,MVD具有1像素精度),则MV预测器(即,相邻块的MV)需要缩放,例如,根据PCT申请PCT/CN2018/104723的示例4(b)。

[0317] (d) 在一个示例中,如果Prec小于1(即,MVD具有4像素精度),则MV预测器(即,相邻块的MV)需要缩放,例如,根据PCT申请PCT/CN2018/104723的示例4(b)。

[0318] 示例3.在一个示例中,如果信令通知的MVD的精度与存储的MV的精度相同,则在仿射MV被重建之后不需要缩放,否则,以使用信令通知的MVD的精度来重建MV并然后将其缩放到存储的MV的精度。

[0319] 示例4.在一个示例中,正常帧间模式和AF_INTER模式可以基于上述不同示例来选

择实现方式。

[0320] 示例5. 在一个示例中,可以使用以下语义来信令通知指示用于仿射模式的MV/MVD精度的语法元素:

[0321] (a) 在一个示例中,等于0、1和2的语法元素分别指示1/4像素、1/16像素和1像素MV精度。

[0322] (b) 替代地,在仿射模式中,等于0、1和2的语法元素分别指示1/4像素、1像素和1/16像素MV精度。

[0323] (c) 替代地,在仿射模式中,等于0、1和2的语法元素分别指示1/16像素、1/4像素和1像素MV精度。

[0324] 示例6. 在一个示例中,可以在SPS、PPS、VPS、序列/图片/条带报头/片等中信令通知是否为仿射模式启用或禁用AMVR。

[0325] 示例7. 在一个示例中,可以在SPS、PPS、VPS、序列/图片/条带报头/片等中信令通知允许的MV/MVD精度的指示。

[0326] (a) 可以针对每个编解码树单元(CTU)和/或每个区域信令通知选择的MVD精度的指示。

[0327] (b) 允许的MV/MVD精度的集合可以取决于当前块的编解码模式(例如,仿射或非仿射)。

[0328] (c) 允许的MV/MVD精度的集合可以取决于条带类型/时域层索引/低延迟校验标志。

[0329] (d) 允许的MV/MVD精度的集合可以取决于当前块或相邻块的块尺寸和/或块形状。

[0330] (e) 允许的MV/MVD精度的集合可以取决于要存储在解码图片缓冲器中的MV的精度。

[0331] (i) 在一个示例中,如果存储的MV在X像素中,则允许的MV/MVD精度集合可以至少具有X像素。

[0332] 支持AMVR的仿射模式的改进

[0333] 示例8. 允许的MVD精度的集合可能随图片而不同、随条带而不同或随块而不同。

[0334] a. 在一个示例中,该允许的MVD精度的集合可以取决于编解码信息,例如块尺寸、块形状等。

[0335] b. 可以预定义允许的MV精度的集合,诸如{1/16,1/4,1}。

[0336] c. 可以在SPS/PPS/VPS/序列报头/图片报头/条带报头/CTU组等中信令通知允许的MV精度的指示。

[0337] d. 来自允许的MV精度的集合的所选MV精度的信令进一步取决于块的允许MV精度的数量。

[0338] 示例9. 向解码器信令通知语法元素以指示仿射帧间模式中使用的MVD精度。

[0339] a. 在一个示例中,仅使用一个单个语法元素来指示应用于仿射模式和AMVR模式的MVD精度。

[0340] i. 在一个示例中,使用相同的语义,即,对于AMVR和仿射模式,将语法元素的相同值映射到相同MVD精度。

[0341] ii. 替代地,单个语法元素的语义对于AMVR模式和仿射模式是不同的。换言之,对

于AMVR和仿射模式,可以将语法元素的相同值映射到不同MVD精度。

[0342] b. 在一个示例中,当仿射模式使用与AMVR相同的MVD精度的集合(例如,MVD精度设置为{1,1/4,4}像素)时,在仿射模式中重用AMVR中的MVD精度语法元素,即,仅使用一个单个语法元素。

[0343] i. 替代地,此外,当在CABAC编码器/解码器中对该语法元素进行编码/解码时,相同或不同的上下文模型可以用于AMVR和仿射模式。

[0344] ii. 替代地,此外,该语法元素在AMVR和仿射模式中可以具有不同的语义。例如,等于0、1和2的语法元素在AMVR中分别指示1/4像素、1像素和4像素MV精度,而在仿射模式中,等于0、1和2的语法元素分别指示1/4像素、1/16像素和1像素MV精度。

[0345] c. 在一个示例中,当仿射模式使用与AMVR相同的数目的MVD精度但是不同的MVD精度的集合(例如,对于AMVR的MVD精度设置为{1,1/4,4}像素,而对于仿射的MVD精度设置为{1/16,1/4,1}像素)时,在仿射模式中重用AMVR中的MVD精度语法元素,即,仅使用一个单个语法元素。

[0346] i. 替代地,此外,当在CABAC编码器/解码器中对该语法元素进行编码/解码时,相同或不同的上下文模型可以用于AMVR和仿射模式。

[0347] ii. 替代地,此外,该语法元素在AMVR和仿射模式中可以具有不同的语义。

[0348] d. 在一个示例中,仿射模式使用比AMVR更少的MVD精度,在仿射模式中重用AMVR中的MVD精度语法元素。然而,仅语法元素值的子集对仿射模式有效。

[0349] i. 替代地,此外,当在CABAC编码器/解码器中对该语法元素进行编码/解码时,相同或不同的上下文模型可以用于AMVR和仿射模式。

[0350] ii. 替代地,此外,该语法元素在AMVR和仿射模式中可以具有不同的语义。

[0351] e. 在一个示例中,仿射模式使用比AMVR更多的MVD精度,在仿射模式中重用AMVR中的MVD精度语法元素。然而,扩展这样的语法元素以允许仿射模式中的更多值。

[0352] i. 替代地,此外,当在CABAC编码器/解码器中对该语法元素进行编码/解码时,相同或不同的上下文模型可以用于AMVR和仿射模式。

[0353] ii. 替代地,此外,该语法元素在AMVR和仿射模式中可以具有不同的语义。

[0354] f. 在一个示例中,新语法元素用于编解码仿射模式的MVD精度,即,两个不同的语法元素用于编解码AMVR和仿射模式的MVD精度。

[0355] g. 用于指示仿射模式的MVD精度的语法可以在以下一个或所有条件下信令通知:

[0356] i. 所有控制点的MVD均非零。

[0357] ii. 至少一个控制点的MVD非零。

[0358] iii. 一个控制点(例如,第一CPMV)的MVD非零

[0359] 在这种情况下,当上述条件中的任何一个或它们所有都失败时,不需要信令通知MVD精度。

[0360] h. 用于指示仿射模式或AMVR模式的MVD精度的语法元素可以使用上下文编解码,并且上下文取决于编解码信息。

[0361] i. 在一个示例中,当仅存在一个单个语法元素时,上下文可以取决于当前块是否使用仿射模式编解码。

[0362] i. 在一个示例中,上下文可以取决于块尺寸/块形状/相邻块的MVD精度/时域层索

引/预测方向等。

[0363] j.可以在SPS/PPS/VPS/序列报头/图片报头/条带报头/CTU组等中信令通知是否启用或禁用仿射模式的多个MVD精度的使用。

[0364] i.在一个示例中,是否信令通知启用或禁用仿射模式的多个MVD精度的使用的信息可以取决于其它语法元素。例如,当启用仿射模式时,信令通知启用或禁用仿射模式的多个MV和/或MVP和/或MVD精度的使用的信息;当禁用仿射模式时,不信令通知启用或禁用仿射模式的多个MV和/或MVP和/或MVD精度的使用的信息并将其推断为0。

[0365] k.替代地,可以信令通知多个语法元素以指示在仿射帧间模式中使用的MV和/或MVP和/或MVD精度(在下面的讨论中,它们都被称为“MVD精度”)。

[0366] i.在一个示例中,用于指示在仿射帧间模式和正常帧间模式中使用的MVD精度的语法元素可以是不同的。

[0367] 1.指示在仿射帧间模式和正常帧间模式中使用的MVD精度的语法元素的数目可以是不同的。

[0368] 2.指示在仿射帧间模式和正常帧间模式中使用的MVD精度的语法元素的语义可以是不同的。

[0369] 3.在算术编解码中用于编解码一个语法元素以指示在仿射帧间模式和正常帧间模式中使用的MVD精度的上下文模型可以是不同的。

[0370] 4.在算术编解码中导出上下文模型以编解码一个语法元素以指示在仿射帧间模式和正常帧间模式中使用的MVD精度的方法可以是不同的。

[0371] ii.在一个实例中,可以信令通知第一语法元素(例如,amvr_flag)以指示是否在仿射编解码块中应用AMVR。

[0372] 1.有条件地信令通知第一语法元素。

[0373] a.在一个示例中,当使用某一模式(例如,CPR/IBC模式)编解码当前块时,跳过第一语法元素(amvr_flag)的信令通知。

[0374] b.在一个示例中,当所有CPMV的MVD(包括水平和垂直分量)都为零时,跳过第一语法元素(amvr_flag)的信令通知。

[0375] c.在一个示例中,当一个所选择的CPMV的MVD(包括水平和垂直分量)都为零时,跳过第一语法元素(amvr_flag)的信令通知。

[0376] i.在一个示例中,所选择的CPMV的MVD是要被编解码/解码的第一CPMV的MVD。

[0377] d.在一个示例中,当启用仿射编解码块的多个MVD精度的使用为假时,跳过第一语法元素(amvr_flag)的信令通知。

[0378] e.在一个示例中,可以在以下条件下信令通知第一语法元素:

[0379] i.启用仿射编解码块的多个MVD精度的使用为真,并且当前块使用仿射模式编解码;

[0380] ii.替代地,启用仿射编解码块的多个MVD精度的使用为真,当前块使用仿射模式编解码,并且CPMV的MVD的至少一个分量不等于0。

[0381] iii.替代地,启用仿射编解码块的多个MVD精度的使用为真,当前块使用仿射模式编解码,并且所选择的CPMV的MVD的至少一个分量不等于0。

[0382] 1.在一个示例中,所选择的CPMV的MVD是要被编解码/解码的第一CPMV的MVD。

[0383] 2. 当AMVR未应用于仿射编解码块或者不存在第一语法元素时,采用默认MV和/或MVD精度。

[0384] a. 在一个示例中,默认精度是1/4像素(1/4-pel)。

[0385] b. 替代地,默认精度设置为仿射编解码块的运动补偿中使用的精度。

[0386] 3. 例如,如果amvr_flag等于0,则仿射模式的MVD精度为1/4像素;否则仿射模式的MVD精度可以是其它值。

[0387] a. 替代地,此外,可以经由第二语法元素进一步信令通知附加MVD精度。

[0388] iii. 在一个实例中,可以信令通知第二语法元素(例如amvr_coarse_precision_flag)以指示仿射模式的MVD精度。

[0389] 1. 在一个示例中,是否信令通知第二语法元素可以取决于第一语法元素。例如,仅当第一语法元素是1时信令通知第二语法元素。

[0390] 2. 在一个示例中,如果第二语法元素是0,则仿射模式的MVD精度是1像素;否则,仿射模式的MVD精度是1/16像素。

[0391] 3. 在一个示例中,如果第二语法元素是0,则仿射模式的MVD精度是1/16像素;否则,仿射模式的MVD精度为全像素。

[0392] iv. 在一个示例中,用于指示在仿射帧间模式中使用的MVD精度的语法元素与具有相同名称但用于指示在正常帧间模式中使用的MVD精度的语法元素共享相同的上下文模型。

[0393] 1. 替代地,用于指示在仿射帧间模式中使用的MVD精度的语法元素与具有相同名称但用于指示在正常帧间模式中使用的MVD精度的语法元素使用不同的上下文模型。

[0394] 示例10.在仿射编解码块上是否应用或如何应用AMVR可以取决于当前块的参考图片。

[0395] a. 在一个示例中,如果参考图片是当前图片,则不应用AMVR,即,

[0396] 在当前块中应用帧内块复制。

[0397] 编码器仿射模式中的AVMR的快速算法

[0398] 将仿射模式和AMVP模式的RD成本(实际RD成本,或SATD/SSE/SAD成本加粗比特成本)表示为 $IMV=i$ 的 $affineCost_i$ 和 $amvpCost_i$,其中 $i=0,1$ 或 2 。在本文中, $IMV=0$ 意味着1/4像素MV,并且 $IMV=1$ 意味着AMVP模式的整数MV和仿射模式的1/16像素MV,并且 $IMV=2$ 意味着AMVP模式的4像素MV和仿射模式的整数MV。将Merge模式的RD成本表示为 $mergeCost$ 。

[0399] 示例11.如果其父代CU的最佳模式不是AF_INTER模式或AF_MERGE模式,则提议对当前CU的仿射模式禁用AMVR。

[0400] 替代地,如果其父代CU的最佳模式不是AF_INTER模式,则对当前CU的仿射模式禁用AMVR。

[0401] 示例12.如果 $affineCost0 > th1 * amvpCost0$,其中 $th1$ 是正阈值,则提议对仿射模式禁用AMVR。

[0402] a. 替代地,此外,如果 $\min(affineCost0, amvpCost0) > th2 * mergeCost$,其中 $th2$ 是正阈值,则对仿射模式禁用AMVR。

[0403] b. 替代地,此外,如果 $affineCost0 > th3 * affineCost1$,其中 $th3$ 是正阈值,则对仿射模式禁用整数MV。

[0404] 示例12.如果 $\text{amvpCost0} > \text{th4} * \text{affineCost0}$,其中th4正阈值,则建议对AMVP模式禁用AMVR。

[0405] a.替代地,如果 $\min(\text{affineCost0}, \text{amvpCost0}) > \text{th5} * \text{mergeCost}$,其中th5是正阈值,则对AMVP模式禁用AMVR。

[0406] 示例13.提议在一个MV精度中获得的4/6参数仿射模型可以用作其它MV精度的候选开始搜索点。

[0407] a.在一个示例中,在1/16MV中获得的4/6参数仿射模型可以用作其它MV精度的候选开始搜索点。

[0408] b.在一个示例中,在1/4MV中获得的4/6参数仿射模型可以用作其它MV精度的候选开始搜索点。

[0409] 示例14.如果当前块的父代块未选择仿射模式,则对当前块不在编码器处校验用于仿射模式的AMVR。

[0410] 示例15.可以利用先前编解码的帧/条带/片/CTU行中的仿射编解码块的不同MV精度的使用的统计来提前终止当前条带/片/CTU行中的仿射编解码块的MV精度的速率失真计算。

[0411] a.在一个示例中,记录具有某一MV精度的仿射编解码块的百分比。如果百分比太低,则跳过校验对应的MV精度。

[0412] b.在一个示例中,利用具有相同时域层的先前编解码帧来决定是否跳过某一MV精度。

[0413] 对仿射AMVR进行编解码的上下文

[0414] 示例16.对于用于对仿射AMVR代码进行编解码的每个上下文,提议设置一个变量(由shiftIdx表示)以控制与此上下文相关联的两个概率更新速度。

[0415] a.在一个示例中,较快的更新速度由 $(\text{shiftIdx} \gg 2) + 2$ 定义。

[0416] b.在一个示例中,较慢的更新速度由 $(\text{shiftIdx} \& 3) + 3 + \text{shift0}$ 定义。

[0417] c.在一个示例中,一致性比特流应遵循以下规则:导出的较快的更新速度应在[2, 5]范围内,含端点。

[0418] d.在一个示例中,一致性比特流应遵循以下规则:导出的较快的更新速度应在[3, 6]范围内,含端点。

[0419] 示例17.在对一个块的AMVR模式进行编解码时,提议不允许将相邻块的仿射AMVR模式信息用于上下文建模。

[0420] a.在一个示例中,可以利用相邻块的AMVR模式索引,并且排除相邻块的仿射AMVR模式信息。表5(包括表5-1和5-2)中示出了示例,其中 (x_{NbL}, y_{NbL}) 和 (x_{NbA}, y_{NbA}) 表示左侧和上方相邻块。在一个示例中,上下文索引偏移 $\text{ctxInc} = (\text{condL} \& \& \text{availableL}) + (\text{condA} \& \& \text{availableA}) + \text{ctxSetIdx} * 3$ 。

[0421] 表5-1使用左方和上方语法元素的ctxInc规范

	语法元素	condL	condA	ctxSet Idx
[0422]	amvr_mode[x0][y0] (inter_affine_flag[x0][y0] is equal to 0)	amvr_mode[xNbL][yNbL] && !inter_affine_flag[xNbL][yNbL]	amvr_mode[xNbA][yNbA] && !inter_affine_flag[xNbL][yNbL]	0

[0423] 表5-2使用左方和上方语法元素的ctxInc规范

	语法元素	condL	condA	ctxSet Idx
[0424]	amvr_flag[x0][y0] (inter_affine_flag[x0][y0] is equal to 0)	amvr_flag[xNbL][yNbL] && !inter_affine_flag[xNbL][yNbL]	amvr_flag[xNbA][yNbA] && !inter_affine_flag[xNbL][yNbL]	0

[0425] b. 替代地,相邻块的仿射AMVR模式信息可以被进一步利用,但是使用函数而不是直接使用。在一个示例中,当仿射编解码的相邻块的amvr_mode[xNbL][yNbL]指示某一MV精度(例如1/4像素MV精度)时,表6-1中所述的函数func可以返回真。在一个示例中,当仿射编解码的相邻块的amvr_flag[xNbL][yNbL]指示某一MV精度(例如1/4像素MV精度)时,表6-2中所述的函数func可以返回真。

[0426] 表6-1使用左方和上方语法元素的ctxInc规范

	语法元素	condL	condA	ctxSet Idx
[0427]	amvr_mode[x0][y0] (inter_affine_flag[x0][y0] is equal to 0)	(amvr_mode[xNbL][yNbL] && !inter_affine_flag[xNbL][yNbL]) (inter_affine_flag[xNbL][yNbL] && func(amvr_mode[xNbL][yNbL]))	amvr_mode[xNbA][yNbA] && !inter_affine_flag[xNbL][yNbL] (inter_affine_flag[xNbL][yNbL] && func(amvr_mode[xNbL][yNbL]))	0

[0428] 表6-2使用左方和上方语法元素的ctxInc规范

	语法元素	condL	condA	ctxSet Idx
[0429]	amvr_flag[x0][y0] (inter_affine_flag[x0][y0] is equal to 0)	amvr_flag[xNbL][yNbL] && !inter_affine_flag[xNbL][yNbL]	amvr_flag[xNbA][yNbA] && !inter_affine_flag[xNbL][yNbL]	0

[0430] c. 替代地,相邻块的仿射AMVR模式信息可以进一步用于编解码AMVR模式(应用于正常帧间模式)的第一语法元素(例如,amvr_flag)。表6-3和6-4给出了一些示例。

[0431] 表6-3使用左方和上方语法元素的ctxInc规范

	语法元素	condL	condA	ctxSet Idx
[0432]	amvr_flag[x0][y0] (inter_affine_flag[x0][y0] is equal to 0)	amvr_flag[xNbL][yNbL]	amvr_flag[xNbA][yNbA]	0

[0433] 表6-4使用左方和上方语法元素的ctxInc规范

	语法元素	condL	condA	ctxSetIdx
[0434]	amvr_flag[x0][y0]	amvr_flag[xNbL][yNbL] && ! inter_affine_flag[x0][y0]	amvr_flag[xNbA][yNbA] && ! inter_affine_flag[x0][y0]	0

[0435] d. 当AMVR模式信息由多个语法元素(例如,第一和第二语法元素,由amvr_flag、amvr_coarse_precision_flag表示)表示时,以上语法amvr_mode可以由多个语法元素中的任何一个替换,并且仍然可以应用上述方法。

[0436] 示例18. 在对仿射AMVR模式进行编解码时,提议可以将相邻块的AMVR模式信息用于上下文编解码。

[0437] a. 在一个示例中,直接使用了相邻块的AMVR模式信息。表7中示出了一个示例。替代地,此外,上下文索引偏移ctxInc = (condL&&availableL) + (condA&&availableA) + ctxSetIdx*3。

[0438] 表7使用左方和上方语法元素的ctxInc规范

	语法元素	condL	condA	ctxSetIdx
[0439]	amvr_mode[x0][y0] (inter_affine_flag[x0][y0] is equal to 1)	amvr_mode[xNbL][yNbL]	amvr_mode[xNbA][yNbA]	0

[0440] b. 替代地,不允许将相邻块的AMVR模式信息用于上下文建模。表8中示出了示例。

[0441] 表8使用左方和上方语法元素的ctxInc规范

	语法元素	condL	condA	ctxSetIdx
[0442]	amvr_mode[x0][y0] (inter_affine_flag[x0][y0] is equal to 1)	amvr_mode[xNbL][yNbL] && inter_affine_flag[xNbL][yNbL]	amvr_mode[xNbA][yNbA] && inter_affine_flag[xNbA][yNbA]	0

[0443] c. 替代地,可以进一步利用相邻块的AMVR模式信息,但是使用函数而不是直接使用。在一个示例中,当非仿射编解码的相邻块的amvr_mode[xNbL][yNbL]指示某一MV精度(诸如1/4像素MV精度)时,表9中所述的函数func可以返回真。

[0444] 表9使用左方和上方语法元素的ctxInc规范

	语法元素	condL	condA	ctxSetIdx
[0445]	amvr_mode[x0][y0]	(amvr_mode[xNbL][yNbL] && inter_affine_flag[xNbL][yNbL])	amvr_mode[xNbA][yNbA] && inter_affine_flag[xNbA][yNbA]	0

[0446]	(inter_affine_flag[x0][y0] is equal to 1)	(!inter_affine_flag[xNbL][yNbL] && func(amvr_mode[xNbL][yNbL]))	(!inter_affine_flag[xNbL][yNbL] && func(amvr_mode[xNbL][yNbL]))
--------	---	--	--

[0447] d. 当仿射AMVR模式信息由多个语法元素(例如,第一和第二语法元素,由amvr_flag、amvr_coarse_precision_flag表示)表示时,以上语法amvr_mode可以由多个语法元素中的任何一个替换,并且以上方法仍然可以应用。

[0448] SMVD和仿射SMVD的快速算法

[0449] 在校验SMVD模式时,假设当前选择的最佳模式为CurBestMode,并且AMVR中AMVR的MVD精度为MvdPrec或仿射AMVR的MVD精度为MvdPrecAff。

[0450] 示例19. 取决于当前选择的最佳模式(即CurBestMode)、AMVR中的MVD精度,可以跳过SMVD模式。

[0451] a. 在一个示例中,如果CurBestMode是Merge模式或/和UMVE模式,则可以不校验SMVD模式。

[0452] b. 在一个示例中,如果未使用SMVD模式对CurBestMode进行编解码,则可以不校验SMVD模式。

[0453] c. 在一个示例中,如果CurBestMode是仿射模式,则可以不校验SMVD模式。

[0454] d. 在一个示例中,如果CurBestMode是子块Merge模式,则可以不校验SMVD模式。

[0455] e. 在一个示例中,如果CurBestMode是仿射SMVD模式,则可以不校验SMVD模式。

[0456] f. 在一个示例中,如果CurBestMode是仿射Merge模式,则可以不校验SMVD模式。

[0457] g. 在一个示例中,以上快速方法,即项目符号13.a-13.f,可以仅对一些MVD精度应用。

[0458] i. 在一个示例中,仅当MVD精度大于或等于精度(例如,整数像素精度)时,才可以应用以上快速方法。

[0459] ii. 在一个示例中,仅当MVD精度大于精度(例如,整数像素精度)时,才可以应用以上快速方法。

[0460] iii. 在一个示例中,仅当MVD精度小于或等于精度(例如,整数像素精度)时,才可以应用以上快速方法。

[0461] iv. 在一个示例中,仅当MVD精度小于精度(例如,整数像素精度)时,才可以应用以上快速方法。

[0462] 示例20. 取决于当前选择的最佳模式(即CurBestMode)、仿射AMVR中的MVD精度,可以跳过仿射SMVD模式。

[0463] a. 在一个示例中,如果CurBestMode是Merge模式或/和UMVE模式,则可以不校验仿射SMVD模式。

[0464] b. 在一个示例中,如果未使用仿射SMVD模式对CurBestMode进行编解码,则可以不校验仿射SMVD模式。

[0465] c. 在一个示例中,如果CurBestMode是子块Merge模式,则可以不校验仿射SMVD模式。

[0466] d. 在一个示例中,如果CurBestMode是SMVD模式,则可以不校验仿射SMVD模式。

[0467] e. 在一个示例中,如果CurBestMode是仿射Merge模式,则可以不校验仿射SMVD模式。

[0468] f. 在一个示例中,以上快速方法,即,项目符号20.a至20.e,可以仅用于一些MVD精度。

[0469] i. 在一个示例中,仅当仿射MVD精度大于或等于精度(例如,整数像素精度)时,才可以应用以上快速方法。

[0470] ii. 在一个示例中,仅当仿射MVD精度大于精度(例如,整数像素精度)时,才可以应用以上快速方法。

[0471] iii. 在一个示例中,仅当仿射MVD精度小于或等于精度(例如,整数像素精度)时,才可以应用以上快速方法。

[0472] iv. 在一个示例中,仅当仿射MVD精度小于精度(例如,整数像素精度)时,才可以应用以上快速方法。

[0473] 示例21. 可以在某些条件下应用上面提议的方法,诸如块尺寸、条带/图片/片类型或运动信息。

[0474] a. 在一个示例中,当块尺寸包含小于M*H个样本(例如,16或32或64个亮度样本)时,不允许提议的方法。

[0475] b. 替代地,当块的宽度或/和高度最小尺寸小于或不大于X时,不允许提议的方法。在一个示例中,X设置为8。

[0476] c. 替代地,当块的宽度或/和高度最小尺寸不小于X时,不允许提议的方法。在一个示例中,X设置为8。

[0477] d. 替代地,当块的宽度 $>th1$ 或 $\geq th1$ 和/或块的高度 $>th2$ 或 $\geq th2$ 时,不允许提议的方法。在一个示例中,th1和/或th2设置为8。

[0478] e. 替代地,当块的宽度 $<th1$ 或 $\leq th1$ 和/或块的高度 $<th2$ 或 $\leq th2$ 时,不允许提议的方法。在一个示例中,th1和/或th2设置为8。

[0479] f. 替代地,是否启用或禁用上述方法和/或应用哪种方法可以取决于块尺寸、视频处理数据单元(VPDU)、图片类型、低延迟校验标志、当前块的编解码信息(诸如参考图片、单向或双向预测)或先前编解码的块。

[0480] 示例22. 当应用或不应用帧内块复制(IBC,也称为当前图片参考(CPR))时,可以以不同方式执行用于仿射模式的AMVR方法。

[0481] a. 在一个示例中,如果块由IBC编解码,则不可以使用用于仿射模式的AMVR。

[0482] b. 在一个示例中,如果块由IBC编解码,则可以使用用于仿射模式的AMVR,但是候选MV/MVD/MVP精度可以与用于非IBC编解码的仿射编解码块的精度不同。

[0483] 示例23. 文件中的所有术语“条带”可以由“片组”或“片”代替。

[0484] 示例24. 在VPS/SPS/PPS/条带报头/片组报头中,等于1的语法元素(例如,no_amvr_constraint_flag)指定对比特流一致性的要求是,指示是否启用AMVR的语法元素(例如,sps_amvr_enabled_flag)和指示是否启用仿射AMVR的语法元素(例如,sps_affine_avmr_enabled_flag)两者应该等于0。语法元素(例如,no_amvr_constraint_flag)等于0不施加约束。

[0485] 示例25. 在VPS/SPS/PPS/条带报头/片组报头或其它视频数据单元中,可以信令通

知语法元素(例如,no_affine_amvr_constraint_flag)。

[0486] a.在一个示例中,no_affine_amvr_constraint_flag等于1指定对比特流一致性的要求是,指示是否启用仿射AMVR的语法元素(例如,sps_affine_avmr_enabled_flag)应当等于0。语法元素(例如,no_affine_amvr_constraint_flag)等于0不施加约束。

[0487] 示例26.可以利用多个上下文来对第二语法元素进行编解码,该第二语法元素指示粗运动精度(例如amvr_coarse_precision_flag)。

[0488] a.在一个示例中,可以利用两个上下文。

[0489] b.在一个示例中,上下文的选择可以取决于当前块是否被仿射编解码。

[0490] c.在一个示例中,对于第一语法,可以仅使用一个上下文来对其进行编解码,并且对于第二语法,也可以仅使用一个上下文来对其进行编解码。

[0491] d.在一个示例中,对于第一语法,可以仅使用一个上下文对其进行编解码,并且对于第二语法,也可以对其进行旁路编解码。

[0492] e.在一个示例中,对于第一语法,可以对其进行旁路编解码,并且对于第二语法,也可以对其进行旁路编解码。

[0493] f.在一个示例中,对于与运动矢量精度有关的所有语法元素,可以对它们进行旁路编解码。

[0494] 示例27.例如,仅语法元素amvr_mode的第一个二进制数使用(一个或多个)算术编解码上下文来编解码。以下所有amvr_mode的二进制数均被编解码为旁路编解码。

[0495] a.以上公开的方法也可以应用于其他语法元素。

[0496] b.例如,仅语法元素SE的第一二进制数使用(一个或多个)算术编解码上下文来编解码。以下所有SE的二进制数均被编解码为旁路编解码。SE可以为

[0497] 1)alf_ctb_flag

[0498] 2)sao_merge_left_flag

[0499] 3)sao_merge_up_flag

[0500] 4)sao_type_idx_luma

[0501] 5)sao_type_idx_chroma

[0502] 6)split_cu_flag

[0503] 7)split_qt_flag

[0504] 8)mtt_split_cu_vertical_flag

[0505] 9)mtt_split_cu_binary_flag

[0506] 10)cu_skip_flag

[0507] 11)pred_mode_ibc_flag

[0508] 12)pred_mode_flag

[0509] 13)intra_luma_ref_idx

[0510] 14)intra_subpartitions_mode_flag

[0511] 15)intra_subpartition_split_flag

[0512] 16)intra_luma_mpm_flag

[0513] 17)intra_chroma_pred_mode

[0514] 18)merge_flag

[0515] 19) inter_pred_idc
[0516] 20) inter_affine_flag
[0517] 21) cu_affine_type_flag
[0518] 22) ref_idx_l0
[0519] 23) mvp_l0_flag
[0520] 24) ref_idx_l1
[0521] 25) mvp_l1_flag
[0522] 26) avmr_flag
[0523] 27) amvr_precision_flag
[0524] 28) gbi_idx
[0525] 29) cu_cbf
[0526] 30) cu_sbt_flag
[0527] 31) cu_sbt_quad_flag
[0528] 32) cu_sbt_horizontal_flag
[0529] 33) cu_sbt_pos_flag
[0530] 34) mmvd_flag
[0531] 35) mmvd_merge_flag
[0532] 36) mmvd_distance_idx
[0533] 37) ciip_flag
[0534] 38) ciip_luma_mpm_flag
[0535] 39) merge_subblock_flag
[0536] 40) merge_subblock_idx
[0537] 41) merge_triangle_flag
[0538] 42) merge_triangle_idx0
[0539] 43) merge_triangle_idx1
[0540] 44) merge_idx
[0541] 45) abs_mvd_greater0_flag
[0542] 46) abs_mvd_greater1_flag
[0543] 47) tu_cbf_luma
[0544] 48) tu_cbf_cb
[0545] 49) tu_cbf_cr
[0546] 50) cu_qp_delta_abs
[0547] 51) transform_skip_flag
[0548] 52) tu_mts_idx
[0549] 53) last_sig_coeff_x_prefix
[0550] 54) last_sig_coeff_y_prefix
[0551] 55) coded_sub_block_flag
[0552] 56) sig_coeff_flag
[0553] 57) par_level_flag

- [0554] 58)abs_level_gt1_flag
- [0555] 59)abs_level_gt3_flag
- [0556] c.替代地,此外,如果语法元素SE是二进制值(即,它可以仅等于0或1),则可以进行上下文编解码。
- [0557] i.替代地,此外,如果语法元素SE是二进制值(即,它可以仅等于0或1),则可以进行旁路编解码。
- [0558] d.替代地,此外,仅一个上下文可以用于对第一二进制数进行编解码。
- [0559] 示例28.运动矢量预测(MVP)或运动矢量差(MVD)或重建的运动矢量(MV)的精度,可以取决于可以信令通知的运动精度而改变。
- [0560] a.在一个示例中,如果MVP的原始预测低于(或不高于)目标精度,则 $MVP=MVP\ll s$ 。s是整数,可以取决于原始精度和目标精度之间的差。
- [0561] ii.替代地,如果MVD的原始精度低于(或不高于)目标精度,则 $MVD=MVD\ll s$ 。s是整数,可以取决于原始精度和目标精度之间的差。
- [0562] iii.替代地,如果MV的原始精度低于(或不高于)目标精度,则 $MV=MV\ll s$ 。s是整数,可以取决于原始精度和目标精度之间的差。
- [0563] b.在一个示例中,如果MVP的原始预测高于(或不低于)目标精度,则 $MVP=Shift(MVP,s)$ 。s是整数,可以取决于原始精度和目标精度之间的差。
- [0564] i.替代地,如果MVD的原始精度高于(或不低于)目标精度,则 $MVD=Shift(MVD,s)$ 。s是整数,可以取决于原始精度和目标精度之间的差。
- [0565] ii.替代地,如果MV的原始精度高于(或不低于)目标精度,则 $MV=Shift(MV,s)$ 。s是整数,可以取决于原始精度和目标精度之间的差。
- [0566] c.在一个示例中,如果MVP的原始预测高于(或不低于)目标精度,则 $MVP=SatShift(MVP,s)$ 。s是整数,可以取决于原始精度和目标精度之间的差。
- [0567] i.替代地,如果MVD的原始精度高于(或不低于)目标精度,则 $MVD=SatShift(MVD,s)$ 。s是整数,可以取决于原始精度和目标精度之间的差。
- [0568] ii.替代地,如果MV的原始精度高于(或不低于)目标精度,则 $MV=SatShift(MV,s)$ 。s是整数,可以取决于原始精度和目标精度之间的差。
- [0569] d.在当前块未使用仿射模式编解码时,可以应用所公开的以上方法。
- [0570] e.在当前块使用仿射模式编解码时,可以应用所公开的以上方法。
- [0571] 5.实施例
- [0572] 高亮部分显示修改后的规格。
- [0573] 5.1实施例1:仿射AMVR模式的使用的指示
- [0574] 可以在SPS/PPS/VPS/APS/序列报头/图片报头/片组报头等中信令通知该指示。本节介绍SPS中的信令。
- [0575] 5.1.1SPS语法表

[0576]	seq_parameter_set_rbsp() {	描述符
	sps_seq_parameter_set_id	ue(v)

[0577]

...	
sps_amvr_enabled_flag	u(1)
sps_bdof_enabled_flag	u(1)
sps_affine_amvr_enabled_flag	u(1)
sps_cclm_enabled_flag	u(1)
sps_mts_intra_enabled_flag	u(1)
sps_mts_inter_enabled_flag	u(1)
sps_affine_enabled_flag	u(1)
if(sps_affine_enabled_flag)	
sps_affine_type_flag	u(1)
sps_gbi_enabled_flag	u(1)
sps_cpr_enabled_flag	u(1)
...	
rsbsp_trailing_bits()	
}	

[0578] 给定替代SPS语法表如下：

[0579]

seq_parameter_set_rbsp() {	描述符
sps_seq_parameter_set_id	ue(v)
sps_amvr_enabled_flag	u(1)
sps_bdof_enabled_flag	u(1)
sps_cclm_enabled_flag	u(1)
sps_mts_intra_enabled_flag	u(1)
sps_mts_inter_enabled_flag	u(1)
sps_affine_enabled_flag	u(1)
if(sps_affine_enabled_flag){	
sps_affine_type_flag	u(1)
sps_affine_amvr_enabled_flag	u(1)
}	
sps_gbi_enabled_flag	u(1)
sps_cpr_enabled_flag	u(1)

[0580]

sps_ciip_enabled_flag	u(1)
sps_triangle_enabled_flag	u(1)
sps_ladf_enabled_flag	u(1)
...	
rsbsp_trailing_bits()	
}	

[0581] 语义:

[0582] sps_affine_amvr_enabled_flag等于1指定在仿射帧间模式的运动矢量编解码中使用自适应运动矢量差分辨率。amvr_enabled_flag等于0指定在仿射帧间模式的运动矢量编解码中不使用自适应运动矢量差分辨率。

[0583] 5.2AMVR模式信息的解析过程

[0584] 仿射AMVR模式信息的语法可以将其重用于AMVR模式信息(应用于正常帧间模式)。替代地,可以使用不同的语法元素。

[0585] 可以有条件地信令通知仿射AMVR模式信息。下面的不同实施例示出了条件的一些示例。

[0586] 5.2.1实施例#1:CU语法表

[0587]

coding_unit(x0, y0, cbWidth, cbHeight, treeType) {	描述符
if(tile_group_type != I) {	
if(treeType != DUAL_TREE_CHROMA)	
cu_skip_flag[x0][y0]	ac(v)
if(cu_skip_flag[x0][y0] == 0)	
pred_mode_flag	ac(v)
}	
if(CuPredMode[x0][y0] == MODE_INTRA) {	
...	
}	
} else if(treeType != DUAL_TREE_CHROMA) { /* MODE_INTER */	
if(cu_skip_flag[x0][y0] == 0)	
merge_flag[x0][y0]	ac(v)
if(merge_flag[x0][y0]) {	
merge_data(x0, y0, cbWidth, cbHeight)	
} else {	
if(tile_group_type == B)	
inter_pred_idc[x0][y0]	ac(v)
if(sps_affine_enabled_flag && cbWidth >= 16 && cbHeight >= 16) {	
inter_affine_flag[x0][y0]	ac(v)
if(sps_affine_type_flag && inter_affine_flag[x0][y0])	
cu_affine_type_flag[x0][y0]	ac(v)
}	

[0588]

if(inter_pred_idc[x0][y0] != PRED_L1) {	
if(num_ref_idx_l0_active_minus1 > 0)	
ref_idx_l0[x0][y0]	ae(v)
mvd_coding(x0, y0, 0, 0)	
if(MotionModelIdc[x0][y0] > 0)	
mvd_coding(x0, y0, 0, 1)	
if(MotionModelIdc[x0][y0] > 1)	
mvd_coding(x0, y0, 0, 2)	
mvp_l0_flag[x0][y0]	ae(v)
} else {	
MvdL0[x0][y0][0] = 0	
MvdL0[x0][y0][1] = 0	
}	
if(inter_pred_idc[x0][y0] != PRED_L0) {	
if(num_ref_idx_l1_active_minus1 > 0)	
ref_idx_l1[x0][y0]	ae(v)
if(mvd_l1_zero_flag && inter_pred_idc[x0][y0] ==	
PRED_BI) {	
MvdL1[x0][y0][0] = 0	
MvdL1[x0][y0][1] = 0	
MvdCpL1[x0][y0][0][0] = 0	
MvdCpL1[x0][y0][0][1] = 0	
MvdCpL1[x0][y0][1][0] = 0	
MvdCpL1[x0][y0][1][1] = 0	
MvdCpL1[x0][y0][2][0] = 0	
MvdCpL1[x0][y0][2][1] = 0	
} else {	
mvd_coding(x0, y0, 1, 0)	
if(MotionModelIdc[x0][y0] > 0)	
mvd_coding(x0, y0, 1, 1)	
if(MotionModelIdc[x0][y0] > 1)	
mvd_coding(x0, y0, 1, 2)	
mvp_l1_flag[x0][y0]	ae(v)
} else {	
MvdL1[x0][y0][0] = 0	
MvdL1[x0][y0][1] = 0	
}	

[0589]	<pre> if((sps_amvr_enabled_flag && inter_affine_flag == 0 && (MvdL0[x0][y0][0] != 0 MvdL0[x0][y0][1] != 0 MvdL1[x0][y0][0] != 0 MvdL1[x0][y0][1] != 0)) (sps_affine_amvr_enabled_flag && inter_affine_flag == 1 && (MvdCpL0[x0][y0][0][0] != 0 MvdCpL0[x0][y0][0][1] != 0 MvdCpL1[x0][y0][0][0] != 0 MvdCpL1[x0][y0][0][1] != 0 MvdCpL0[x0][y0][1][0] != 0 MvdCpL0[x0][y0][1][1] != 0 MvdCpL1[x0][y0][1][0] != 0 MvdCpL1[x0][y0][1][1] != 0 MvdCpL0[x0][y0][2][0] != 0 MvdCpL0[x0][y0][2][1] != 0 MvdCpL1[x0][y0][2][0] != 0 MvdCpL1[x0][y0][2][1] != 0))) { </pre>		
	<pre> if(!sps_cpr_enabled_flag !(inter_pred_idc[x0][y0] == PRED_L0 && ref_idx_l0[x0][y0] == num_ref_idx_l0_active_minus1)) </pre>		
	<pre> amvr_flag[x0][y0] </pre>	ac(v)	
	<pre> if(amvr_flag[x0][y0]) </pre>		
	<pre> amvr_coarse_precision_flag[x0][y0] </pre>	ac(v)	
	<pre> } </pre>		
	<pre> if(sps_gbi_enabled_flag && inter_pred_idc[x0][y0] == PRED_BI && cbWidth * cbHeight >= 256) </pre>		
	<pre> gbi_idx[x0][y0] </pre>	ac(v)	
	<pre> } </pre>		
	<pre> } </pre>		
	<pre> if(!pcm_flag[x0][y0]) { </pre>		
	<pre> if(CuPredMode[x0][y0] != MODE_INTRA && cu_skip_flag[x0][y0] == 0) </pre>		
	<pre> cu_cbf </pre>	ac(v)	
	<pre> if(cu_cbf) </pre>		
	<pre> transform_tree(x0, y0, cbWidth, cbHeight, treeType) </pre>		
	<pre> } </pre>		
	<pre> } </pre>		

[0590] 5.2.2 实施例2: 替代的CU语法表设计

[0591]	coding_unit(x0, y0, cbWidth, cbHeight, treeType) {		描述符
	if(tile_group_type != I) {		
	if(treeType != DUAL_TREE_CHROMA)		
	cu_skip_flag[x0][y0]	ac(v)	
	if(cu_skip_flag[x0][y0] == 0)		
	pred_mode_flag	ac(v)	
	}		

[0592]

if(CuPredMode[x0][y0] == MODE_INTRA) {	
...	
}	
} else if(treeType != DUAL_TREE_CHROMA) { /* MODE_INTER */	
if(cu_skip_flag[x0][y0] == 0)	
merge_flag[x0][y0]	ae(v)
if(merge_flag[x0][y0]) {	
merge_data(x0, y0, cbWidth, cbHeight)	
} else {	
if(tile_group_type == B)	
inter_pred_idc[x0][y0]	ae(v)
if(sps_affine_enabled_flag && cbWidth >= 16 && cbHeight >= 16) {	
inter_affine_flag[x0][y0]	ae(v)
if(sps_affine_type_flag && inter_affine_flag[x0][y0])	
cu_affine_type_flag[x0][y0]	ae(v)
}	
if(inter_pred_idc[x0][y0] != PRED_L1) {	
if(num_ref_idx_l0_active_minus1 > 0)	
ref_idx_l0[x0][y0]	ae(v)
mvd_coding(x0, y0, 0, 0)	
if(MotionModelIdc[x0][y0] > 0)	
mvd_coding(x0, y0, 0, 1)	
if(MotionModelIdc[x0][y0] > 1)	
mvd_coding(x0, y0, 0, 2)	
mvp_l0_flag[x0][y0]	ae(v)
} else {	
MvdL0[x0][y0][0] = 0	
MvdL0[x0][y0][1] = 0	
}	
if(inter_pred_idc[x0][y0] != PRED_L0) {	
if(num_ref_idx_l1_active_minus1 > 0)	
ref_idx_l1[x0][y0]	ae(v)
if(mvd_l1_zero_flag && inter_pred_idc[x0][y0] ==	
PRED_BI) {	
MvdL1[x0][y0][0] = 0	
MvdL1[x0][y0][1] = 0	
MvdCpL1[x0][y0][0][0] = 0	
MvdCpL1[x0][y0][0][1] = 0	
MvdCpL1[x0][y0][1][0] = 0	
MvdCpL1[x0][y0][1][1] = 0	
MvdCpL1[x0][y0][2][0] = 0	
MvdCpL1[x0][y0][2][1] = 0	
} else {	
mvd_coding(x0, y0, 1, 0)	
if(MotionModelIdc[x0][y0] > 0)	
mvd_coding(x0, y0, 1, 1)	
if(MotionModelIdc[x0][y0] > 1)	
mvd_coding(x0, y0, 1, 2)	
mvp_l1_flag[x0][y0]	ae(v)

[0593]

} else {	
MvdL1[x0][y0][0] = 0	
MvdL1[x0][y0][1] = 0	
}	
if((sps_amvr_enabled_flag && inter_affine_flag == 0	
&&	
(MvdL0[x0][y0][0] != 0 MvdL0[x0][y0][1] != 0	
MvdL1[x0][y0][0] != 0 MvdL1[x0][y0][1] !=	
0))	
(sps_affine_amvr_enabled_flag && inter_affine_flag	
== 1 &&	
(MvdCpL0[x0][y0][0][0] != 0	
MvdCpL0[x0][y0][0][1] != 0	
MvdCpL1[x0][y0][0][0] != 0	
MvdCpL1[x0][y0][0][1] != 0))) {	
if(!sps_cpr_enabled_flag !(inter_pred_idc[x0][y0] == PRED_L0	
&&	
ref_idx_l0[x0][y0] == num_ref_idx_l0_active_minus1))	
amvr_flag[x0][y0]	ae(v)
if(amvr_flag[x0][y0])	
amvr_coarse_precision_flag[x0][y0]	ae(v)
}	
if(sps_gbi_enabled_flag && inter_pred_idc[x0][y0] ==	
PRED_BI &&	
cbWidth * cbHeight >= 256)	
gbi_idx[x0][y0]	ae(v)
}	
}	
if(!pcm_flag[x0][y0]) {	
if(CuPredMode[x0][y0] != MODE_INTRA &&	
cu_skip_flag[x0][y0] == 0)	
cu_cbf	ae(v)
if(cu_cbf)	
transform_tree(x0, y0, cbWidth, cbHeight, treeType)	
}	
}	

[0594] 5.2.3 实施例3: 第三CU语法表设计

[0595]

coding_unit(x0, y0, cbWidth, cbHeight, treeType) {		描述符
if(tile_group_type != 1) {		
if(treeType != DUAL_TREE_CHROMA)		
cu_skip_flag[x0][y0]		ae(v)
if(cu_skip_flag[x0][y0] == 0)		
pred_mode_flag		ae(v)
}		
if(CuPredMode[x0][y0] == MODE_INTRA) {		
...		
}		

[0596]

} else if(treeType != DUAL_TREE_CHROMA) { /* MODE_INTER */	
if(cu_skip_flag[x0][y0] == 0)	
merge_flag[x0][y0]	ae(v)
if(merge_flag[x0][y0]) {	
merge_data(x0, y0, cbWidth, cbHeight)	
} else {	
if(tile_group_type == B)	
inter_pred_idc[x0][y0]	ae(v)
if(sps_affine_enabled_flag && cbWidth >= 16 &&	
cbHeight >= 16) {	
inter_affine_flag[x0][y0]	ae(v)
if(sps_affine_type_flag && inter_affine_flag[x0][y0])	
cu_affine_type_flag[x0][y0]	ae(v)
}	
if(inter_pred_idc[x0][y0] != PRED_L1) {	
if(num_ref_idx_l0_active_minus1 > 0)	
ref_idx_l0[x0][y0]	ae(v)
mvd_coding(x0, y0, 0, 0)	
if(MotionModelIdc[x0][y0] > 0)	
mvd_coding(x0, y0, 0, 1)	
if(MotionModelIdc[x0][y0] > 1)	
mvd_coding(x0, y0, 0, 2)	
mvp_l0_flag[x0][y0]	ae(v)
} else {	
MvdL0[x0][y0][0] = 0	
MvdL0[x0][y0][1] = 0	
}	
if(inter_pred_idc[x0][y0] != PRED_L0) {	
if(num_ref_idx_l1_active_minus1 > 0)	
ref_idx_l1[x0][y0]	ae(v)
if(mvd_l1_zero_flag && inter_pred_idc[x0][y0] ==	
PRED_BI) {	
MvdL1[x0][y0][0] = 0	
MvdL1[x0][y0][1] = 0	
MvdCpL1[x0][y0][0][0] = 0	
MvdCpL1[x0][y0][0][1] = 0	
MvdCpL1[x0][y0][1][0] = 0	
MvdCpL1[x0][y0][1][1] = 0	
MvdCpL1[x0][y0][2][0] = 0	
MvdCpL1[x0][y0][2][1] = 0	
} else {	
mvd_coding(x0, y0, 1, 0)	
if(MotionModelIdc[x0][y0] > 0)	
mvd_coding(x0, y0, 1, 1)	
if(MotionModelIdc[x0][y0] > 1)	
mvd_coding(x0, y0, 1, 2)	
mvp_l1_flag[x0][y0]	ae(v)
} else {	
MvdL1[x0][y0][0] = 0	
MvdL1[x0][y0][1] = 0	

	<pre> } if((sps_amvr_enabled_flag && inter_affine_flag == 0 && (MvdL0[x0][y0][0] != 0 MvdL0[x0][y0][1] != 0 MvdL1[x0][y0][0] != 0 MvdL1[x0][y0][1] != 0)) (sps_affine_amvr_enabled_flag && inter_affine_flag == 1)) { </pre>	
	<pre> if(!sps_cpr_enabled_flag !(inter_pred_idc[x0][y0] == PRED_L0 && ref_idx_l0[x0][y0] == num_ref_idx_l0_active_minus1)) </pre>	
	<pre> amvr_flag[x0][y0] </pre>	ac(v)
	<pre> if(amvr_flag[x0][y0]) </pre>	
[0597]	<pre> amvr_coarse_precision_flag[x0][y0] </pre>	ac(v)
	<pre> } if(sps_gbi_enabled_flag && inter_pred_idc[x0][y0] == PRED_BI && cbWidth * cbHeight >= 256) </pre>	
	<pre> gbi_idx[x0][y0] </pre>	ac(v)
	<pre> } } if(!pcm_flag[x0][y0]) { </pre>	
	<pre> if(CuPredMode[x0][y0] != MODE_INTRA && cu_skip_flag[x0][y0] == 0) </pre>	
	<pre> cu_cbf </pre>	ac(v)
	<pre> if(cu_cbf) </pre>	
	<pre> transform_tree(x0, y0, cbWidth, cbHeight, treeType) </pre>	
	<pre> } } } </pre>	

[0598] 5.2.4 实施例4: 具有AMVR和仿射AMVR模式的不同语法的语法表设计

	<pre> coding_unit(x0, y0, cbWidth, cbHeight, treeType) { if(tile_group_type != I) { if(treeType != DUAL_TREE_CHROMA) cu_skip_flag[x0][y0] if(cu_skip_flag[x0][y0] == 0) pred_mode_flag } </pre>	
	<pre> if(CuPredMode[x0][y0] == MODE_INTRA) { </pre>	
[0599]	<pre> ... } } else if(treeType != DUAL_TREE_CHROMA) { /* MODE_INTER */ if(cu_skip_flag[x0][y0] == 0) merge_flag[x0][y0] if(merge_flag[x0][y0]) { merge_data(x0, y0, cbWidth, cbHeight) } else { </pre>	

[0600]

```

    if( tile_group_type == B )
        inter_pred_idc[ x0 ][ y0 ]
        if( sps_affine_enabled_flag && cbWidth >= 16 &&
cbHeight >= 16 ) {
            inter_affine_flag[ x0 ][ y0 ]
            if( sps_affine_type_flag && inter_affine_flag[ x0 ][ y0 ] )
                cu_affine_type_flag[ x0 ][ y0 ]
        }
        if( inter_pred_idc[ x0 ][ y0 ] != PRED_L1 ) {
            if( num_ref_idx_l0_active_minus1 > 0 )
                ref_idx_l0[ x0 ][ y0 ]
            mvd_coding( x0, y0, 0, 0 )
            if( MotionModelIdc[ x0 ][ y0 ] > 0 )
                mvd_coding( x0, y0, 0, 1 )
            if( MotionModelIdc[ x0 ][ y0 ] > 1 )
                mvd_coding( x0, y0, 0, 2 )
           .mvp_l0_flag[ x0 ][ y0 ]
        } else {
            MvdL0[ x0 ][ y0 ][ 0 ] = 0
            MvdL0[ x0 ][ y0 ][ 1 ] = 0
        }
        if( inter_pred_idc[ x0 ][ y0 ] != PRED_L0 ) {
            if( num_ref_idx_l1_active_minus1 > 0 )
                ref_idx_l1[ x0 ][ y0 ]
            if( mvd_l1_zero_flag && inter_pred_idc[ x0 ][ y0 ] ==
PRED_BI ) {
                MvdL1[ x0 ][ y0 ][ 0 ] = 0
                MvdL1[ x0 ][ y0 ][ 1 ] = 0
                MvdCpL1[ x0 ][ y0 ][ 0 ][ 0 ] = 0
                MvdCpL1[ x0 ][ y0 ][ 0 ][ 1 ] = 0
                MvdCpL1[ x0 ][ y0 ][ 1 ][ 0 ] = 0
                MvdCpL1[ x0 ][ y0 ][ 1 ][ 1 ] = 0
                MvdCpL1[ x0 ][ y0 ][ 2 ][ 0 ] = 0
                MvdCpL1[ x0 ][ y0 ][ 2 ][ 1 ] = 0
            } else {
                mvd_coding( x0, y0, 1, 0 )
                if( MotionModelIdc[ x0 ][ y0 ] > 0 )
                    mvd_coding( x0, y0, 1, 1 )
                if( MotionModelIdc[ x0 ][ y0 ] > 1 )
                    mvd_coding( x0, y0, 1, 2 )
               .mvp_l1_flag[ x0 ][ y0 ]
            } else {
                MvdL1[ x0 ][ y0 ][ 0 ] = 0
                MvdL1[ x0 ][ y0 ][ 1 ] = 0
            }
        }
        if( sps_amvr_enabled_flag && inter_affine_flag == 0 &&
( MvdL0[ x0 ][ y0 ][ 0 ] != 0 || MvdL0[ x0 ][ y0 ][ 1 ] != 0 ||
MvdL1[ x0 ][ y0 ][ 0 ] != 0 || MvdL1[ x0 ][ y0 ][ 1 ] != 0 ) )
    {

```

[0601]

if(!sps_cpr_enabled_flag !(inter_pred_idc[x0][y0] == PRED_L0 &&
ref_idx_l0[x0][y0] == num_ref_idx_l0_active_minus1))
amvr_flag[x0][y0]
if(amvr_flag[x0][y0])
amvr_coarse_precision_flag[x0][y0]
} else if(conditionsA) {
if(conditionsB)
affine_amvr_flag[x0][y0]
if(amvr_flag[x0][y0])
affine_amvr_coarse_precision_flag[x0][y0]
}
if(sps_gbi_enabled_flag && inter_pred_idc[x0][y0] ==
PRED_BI &&
cbWidth * cbHeight >= 256)
gbi_idx[x0][y0]
}
}
if(!pcm_flag[x0][y0]) {
if(CuPredMode[x0][y0] != MODE_INTRA &&
cu_skip_flag[x0][y0] == 0)
cu_cbf
if(cu_cbf)
transform_tree(x0, y0, cbWidth, cbHeight, treeType)
}
}

[0602] • 在一个示例中, conditionsA定义如下:

[0603] (sps_affine_amvr_enabled_flag&&inter_affine_flag==1&&(MvdCpL0[x0][y0][0][0]!=0|MvdCpL0[x0][y0][0][1]!=0|MvdCpL1[x0][y0][0][0]!=0|MvdCpL1[x0][y0][0][1]!=0|MvdCpL0[x0][y0][1][0]!=0|MvdCpL0[x0][y0][1][1]!=0|MvdCpL1[x0][y0][1][0]!=0|MvdCpL1[x0][y0][1][1]!=0|MvdCpL0[x0][y0][2][0]!=0|MvdCpL0[x0][y0][2][1]!=0|MvdCpL1[x0][y0][2][0]!=0|MvdCpL1[x0][y0][2][1]!=0))

[0604] 替代地, conditionsA定义如下:

[0605] (sps_affine_amvr_enabled_flag&&inter_affine_flag==1&&(MvdCpL0[x0][y0][0][0]!=0|MvdCpL0[x0][y0][0][1]!=0|MvdCpL1[x0][y0][0][0]!=0|MvdCpL1[x0][y0][0][1]!=0))

[0606] 替代地, conditionsA定义如下:

[0607] (sps_affine_amvr_enabled_flag&&inter_affine_flag==1&&(MvdCpLX[x0][y0][0][0]!=0|MvdCpLX[x0][y0][0][1]!=0))

[0608] 其中X是0或1。

[0609] 替代地, conditionsA定义如下:

[0610] (sps_affine_amvr_enabled_flag&&inter_affine_flag==1)

[0611] • 在一个示例中, conditionsB定义如下:

[0612] $!sps_cpr_enabled_flag || (inter_pred_idc[x0][y0] == PRED_L0 \& \& ref_idx_l0[x0][y0] == num_ref_idx_l0_active_minus1)$

[0613] 替代地, conditionsB定义如下:

[0614] $!sps_cpr_enabled_flag || (pred_mode[x0][y0] == CPR)$

[0615] 替代地, conditionsB定义如下:

[0616] $!sps_ibc_enabled_flag || (pred_mode[x0][y0] == IBC)$

[0617] 当利用不同的语法元素来编解码AMVR或仿射AMVR时,可以相应地应用于5.5中的实施例的应用于仿射AMVR的上下文建模和/或上下文。

[0618] 5.2.5语义

[0619] $amvr_flag[x0][y0]$ 指定运动矢量差的分辨率。数组索引 $x0$ 、 $y0$ 指定所考虑的编解码块的左顶亮度样本相对于图片的左顶亮度样本的位置 $(x0, y0)$ 。 $amvr_flag[x0][y0]$ 等于0指定运动矢量差的分辨率是亮度样本的1/4。 $amvr_flag[x0][y0]$ 等于1指定运动矢量差的分辨率由 $amvr_coarse_precisoin_flag[x0][y0]$ 进一步指定。

[0620] 当 $amvr_flag[x0][y0]$ 不存在时,推断如下:

[0621] ——如果 $sps_cpr_enabled_flag$ 等于1,则推断 $amvr_flag[x0][y0]$ 等于1。

[0622] ——否则($sps_cpr_enabled_flag$ 等于0),则推断 $amvr_flag[x0][y0]$ 等于0。

[0623] $amvr_coarse_precisoin_flag[x0][y0]$ 等于1指定当 $inter_affine_flag$ 等于0时运动矢量差的分辨率是4个亮度样本,并且当 $inter_affine_flag$ 等于1时运动矢量差的分辨率是1个亮度样本。数组索引 $x0$ 、 $y0$ 指定所考虑的编解码块的左顶亮度样本相对于图片的左顶亮度样本的位置 $(x0, y0)$ 。

[0624] 当 $amvr_coarse_precisoin_flag[x0][y0]$ 不存在时,推断其等于0。

[0625] 如果 $inter_affine_flag[x0][y0]$ 等于0,则变量 $MvShift$ 设置为等于 $(amvr_flag[x0][y0] + amvr_coarse_precisoin_flag[x0][y0]) \ll 1$,并且变量 $MvdL0[x0][y0][0]$ 、 $MvdL0[x0][y0][1]$ 、 $MvdL1[x0][y0][0]$ 、 $MvdL1[x0][y0][1]$ 修改如下:

[0626] $MvdL0[x0][y0][0] = MvdL0[x0][y0][0] \ll (MvShift + 2)$ (7-70)

[0627] $MvdL0[x0][y0][1] = MvdL0[x0][y0][1] \ll (MvShift + 2)$ (7-71)

[0628] $MvdL1[x0][y0][0] = MvdL1[x0][y0][0] \ll (MvShift + 2)$ (7-72)

[0629] $MvdL1[x0][y0][1] = MvdL1[x0][y0][1] \ll (MvShift + 2)$ (7-73)

[0630] 如果 $inter_affine_flag[x0][y0]$ 等于1,变量 $MvShift$ 设置为等于 $(amvr_coarse_precisoin_flag ? (amvr_coarse_precisoin_flag \ll 1) : (- (amvr_flag \ll 1)))$,并且变量 $MvdCpL0[x0][y0][0][0]$ 、 $MvdCpL0[x0][y0][0][1]$ 、 $MvdCpL0[x0][y0][1][0]$ 、 $MvdCpL0[x0][y0][1][1]$ 、 $MvdCpL0[x0][y0][2][0]$ 、 $MvdCpL0[x0][y0][2][1]$ 修改如下:

[0631] $MvdCpL0[x0][y0][0][0] = MvdCpL0[x0][y0][0][0] \ll (MvShift + 2)$ (7-73)

[0632] $MvdCpL1[x0][y0][0][1] = MvdCpL1[x0][y0][0][1] \ll (MvShift + 2)$ (7-67)

[0633] $MvdCpL0[x0][y0][1][0] = MvdCpL0[x0][y0][1][0] \ll (MvShift + 2)$ (7-66)

[0634] $MvdCpL1[x0][y0][1][1] = MvdCpL1[x0][y0][1][1] \ll (MvShift + 2)$ (7-67)

[0635] $MvdCpL0[x0][y0][2][0] = MvdCpL0[x0][y0][2][0] \ll (MvShift + 2)$ (7-66)

[0636] $MvdCpL1[x0][y0][2][1] = MvdCpL1[x0][y0][2][1] \ll (MvShift + 2)$ (7-67)

[0637] 替代地,如果 $inter_affine_flag[x0][y0]$ 等于1,则变量 $MvShift$ 设为等于

(affine_amvr_coarse_precision_flag?(affine_amvr_coarse_precision_flag<<1):(-(affine_amvr_flag<<1)))。

[0638] 5.3运动矢量的取整过程

[0639] 修改取整过程,当给定的rightShift值等于0(发生于1/16像素精度)时,取整偏移设置为0而不是 $(1 \ll (\text{rightShift}-1))$ 。

[0640] 例如,MV的取整过程的子条款修改如下:

[0641] 该过程的输入是:

[0642] -运动矢量mvX,

[0643] -用于取整的右移位参数rightShift,

[0644] -用于分辨率增加的左移位参数leftShift。

[0645] 该过程的输出是取整的运动矢量mvX。

[0646] 对于mvX的取整,以下适用:

[0647] $\text{offset} = (\text{rightShift} == 0) ? 0 : (1 \ll (\text{rightShift}-1))$ (8-371)

[0648] $\text{mvX}[0] = (\text{mvX}[0] \geq 0 ? (\text{mvX}[0] + \text{offset}) \gg$

[0649] $\text{rightShift} : -((-\text{mvX}[0] + \text{offset}) \gg \text{rightShift})) \ll \text{leftShift}$ (8-372)

[0650] $\text{mvX}[1] = (\text{mvX}[1] \geq 0 ? (\text{mvX}[1] + \text{offset}) \gg$

[0651] $\text{rightShift} : -((-\text{mvX}[1] + \text{offset}) \gg \text{rightShift})) \ll \text{leftShift}$ (8-373)

[0652] 5.4解码过程

[0653] 在仿射运动矢量推导过程中调用的取整过程是使用(MvShift+2)而不是固定为2的输入来执行的。

[0654] 亮度仿射控制点运动矢量预测的推导过程

[0655] 该过程的输入是:

[0656] -当前亮度编解码块的左顶样本相对于当前图片的左顶亮度样本的亮度位置(xCb,yCb),

[0657] -指定当前亮度编解码块的宽度和高度的两个变量cbWidth和cbHeight,

[0658] -当前编解码单元refIdxLX的参考索引,其中X为0或1,

[0659] -控制点运动矢量的数目numCpMv。

[0660] 该过程的输出是亮度仿射控制点运动矢量预测mvpCpLX[cpIdx],其中X为0或1,并且cpIdx=0..numCpMv-1。

[0661] 为了推导控制点运动矢量预测候选列表、cpMvpListLX,其中X为0或1,应用以下有序步骤:

[0662] 列表numCpMvpCandLX中的控制点运动矢量预测候选的数目设置为等于0。

[0663] 变量availableFlagA和availableFlagB两者设置为FALSE。

[0664]

[0665] 调用8.4.2.14条款中指定的运动矢量的取整过程,其中mvX设置为等于cpMvpLX[cpIdx]、rightShift设置为等于(MvShift+2)、以及leftShift设置为等于(MvShift+2)作为输入,并且取整的cpMvpLX[cpIdx] (其中cpIdx=0..numCpMv-1)作为输出。

[0666]

[0667] 变量availableFlagA设置为等于TRUE

[0668] 调用8.4.4.5条款中指定的来自相邻块的亮度仿射控制点运动矢量的推导过程,其中亮度编解码块位置(xCb,yCb)、亮度编解码块宽度和高度(cbWidth,cbHeight)、相邻亮度编解码块位置(xNb,yNb)、相邻亮度编解码块宽度和高度(nbW,nbH)、以及控制点运动矢量的数目numCpMv作为输入,控制点运动矢量预测候选cpMvpLY[cpIdx] (其中cpIdx=0..numCpMv-1)作为输出。

[0669] 调用8.4.2.14条款中指定的运动矢量的取整过程,其中mvX设置为等于cpMvpLY[cpIdx]、rightShift设置为等于(MvShift+2)、以及leftShift设置为等于(MvShift+2)作为输入,并且取整的cpMvpLY[cpIdx] (其中cpIdx=0..numCpMv-1)作为输出。

[0670]

[0671] 调用8.4.4.5条款中规定的来自相邻块的亮度仿射控制点运动矢量的推导过程,其中亮度编解码块位置(xCb,yCb)、亮度编解码块宽度和高度(cbWidth,cbHeight)、相邻亮度编解码块位置(xNb,yNb)、相邻亮度编解码块宽度和高度(nbW,nbH)、以及控制点运动矢量的数目numCpMv作为输入,控制点运动矢量预测候选cpMvpLX[cpIdx] (其中cpIdx=0..numCpMv-1)作为输出。

[0672] 调用8.4.2.14条款中指定的运动矢量的取整过程,其中mvX设置为等于cpMvpLX[cpIdx]、rightShift设置为等于(MvShift+2)、以及leftShift设置为等于(MvShift+2)作为输入,并且取整的cpMvpLX[cpIdx] (其中cpIdx=0..numCpMv-1)作为输出。

[0673] 做出以下分配:

[0674] cpMvpListLX[numCpMvpCandLX][0]=cpMvpLX[0] (8-618)

[0675] cpMvpListLX[numCpMvpCandLX][1]=cpMvpLX[1] (8-619)

[0676] cpMvpListLX[numCpMvpCandLX][2]=cpMvpLX[2] (8-620)

[0677] numCpMvpCandLX=numCpMvpCandLX+1 (8-621)

[0678] 否则,如果PredFlagLY[xNbBk][yNbBk] (with Y=!X) 等于1且DiffPicOrderCnt(RefPicListY[RefIdxLY[xNbBk][yNbBk]],RefPicListX[refIdxLX]) 等于0,则应用以下:

[0679] 变量availableFlagB设置为TRUE

[0680] 调用8.4.4.5条款中指定的来自相邻块的亮度仿射控制点运动矢量的推导过程,其中亮度编解码块位置(xCb,yCb)、亮度编解码块宽度和高度(cbWidth,cbHeight)、相邻亮度编解码块位置(xNb,yNb)、相邻亮度编解码块宽度和高度(nbW,nbH)、以及控制点运动矢量的数目numCpMv作为输入,控制点运动矢量预测候选cpMvpLY[cpIdx] (其中cpIdx=0..numCpMv-1)作为输出。

[0681] 调用8.4.2.14条款中指定的运动矢量的取整过程,其中mvX设置为等于cpMvpLY[cpIdx]、rightShift设置为等于(MvShift+2)、以及leftShift设置为等于(MvShift+2)作为输入,并且取整的cpMvpLY[cpIdx] (其中cpIdx=0..numCpMv-1)作为输出。

[0682] 做出以下分配:

[0683] cpMvpListLX[numCpMvpCandLX][0]=cpMvpLY[0] (8-622)

[0684] cpMvpListLX[numCpMvpCandLX][1]=cpMvpLY[1] (8-623)

[0685] cpMvpListLX[numCpMvpCandLX][2]=cpMvpLY[2] (8-624)

[0686] numCpMvpCandLX=numCpMvpCandLX+1 (8-625)

[0687] 当numCpMvpCandLX小于2时,应用以下:

[0688] 调用8.4.4.8条款中指定的用于构建的仿射控制点运动矢量预测候选的推导过程,其中亮度编解码块位置(xCb,yCb)、亮度编解码块宽度cbWidth、亮度编解码块高度cbHeight、以及当前编解码单元的参考索引refIdxLX作为输入,并且可用性标志availableConsFlagLX和cpMvpLX[cpIdx] (其中cpIdx=0..numCpMv-1)作为输出。

[0689] 当availableConsFlagLX等于1且numCpMvpCandLX等于0时,做出以下分配:

[0690] $cpMvpListLX[numCpMvpCandLX][0] = cpMvpLX[0]$ (8-626)

[0691] $cpMvpListLX[numCpMvpCandLX][1] = cpMvpLX[1]$ (8-627)

[0692] $cpMvpListLX[numCpMvpCandLX][2] = cpMvpLX[2]$ (8-628)

[0693] $numCpMvpCandLX = numCpMvpCandLX + 1$ (8-629)

[0694] 对于cpIdx=0..numCpMv-1应用以下:

[0695] 当numCpMvpCandLX小于2且availableFlagLX[cpIdx]等于1时,做出以下分配:

[0696] $cpMvpListLX[numCpMvpCandLX][0] = cpMvpLX[cpIdx]$ (8-630)

[0697] $cpMvpListLX[numCpMvpCandLX][1] = cpMvpLX[cpIdx]$ (8-631)

[0698] $cpMvpListLX[numCpMvpCandLX][2] = cpMvpLX[cpIdx]$ (8-632)

[0699] $numCpMvpCandLX = numCpMvpCandLX + 1$ (8-633)

[0700] 当numCpMvpCandLX小于2,应用以下:

[0701] 调用8.4.2.11条款中指定的时域亮度运动矢量预测的推导过程,其中亮度编解码块位置(xCb,yCb)、亮度编解码块宽度cbWidth、亮度编解码块高度cbHeight以及refIdxLX作为输入,并且其中输出为可用性标志availableFlagLXCol和时域运动矢量预测mvLXCol。

[0702] 当availableFlagLXCol等于1时,应用以下:

[0703] 调用8.4.2.14条款中指定的运动矢量的取整过程,其中mvX设置为等于mvLXCol、rightShift设置为等于(MvShift+2)、以及leftShift设置为等于(MvShift+2)作为输入,并且取整的mvLXCol作为输出。

[0704] 做出以下分配:

[0705] $cpMvpListLX[numCpMvpCandLX][0] = mvLXCol$ (8-634)

[0706] $cpMvpListLX[numCpMvpCandLX][1] = mvLXCol$ (8-635)

[0707] $cpMvpListLX[numCpMvpCandLX][2] = mvLXCol$ (8-636)

[0708] $numCpMvpCandLX = numCpMvpCandLX + 1$ (8-637)

[0709] 当numCpMvpCandLX小于2时,重复以下步骤直到numCpMvpCandLX等于2,其中mvZero[0]和mvZero[1]两者等于0:

[0710] $cpMvpListLX[numCpMvpCandLX][0] = mvZero$ (8-638)

[0711] $cpMvpListLX[numCpMvpCandLX][1] = mvZero$ (8-639)

[0712] $cpMvpListLX[numCpMvpCandLX][2] = mvZero$ (8-640)

[0713] $numCpMvpCandLX = numCpMvpCandLX + 1$ (8-641) 仿射控制点运动矢量预测

cpMvpLX (其中X为0或1) 如下导出:

[0714] $cpMvpLX = cpMvpListLX[mvp_lX_flag[xCb][yCb]]$ (8-642)

[0715] 构建的仿射控制点运动矢量预测候选的推导过程

[0716] 该过程的输入是:

[0717] -指定相对于当前图片的左顶亮度样本的当前亮度编解码块的左顶样本的亮度位

置 (xCb, yCb) ,

[0718] -指定当前亮度编解码块的宽度和高度的两个变量cbWidth和cbHeight,

[0719] -当前预测单元分区的参考索引refIdxLX,其中X为0或1,

[0720] 该过程的输出是:

[0721] -构建的仿射控制点运动矢量预测候选的可用性标志availableConsFlagLX,其中X为0或1,

[0722] -可用性标志availableFlagLX[cpIdx],其中cpIdx=0..2并且X为0或1,

[0723] -构建的仿射控制点运动矢量预测候选cpMvLX[cpIdx],其中cpIdx=0..numCpMv-1并且X为0或1。

[0724] 第一(左顶)控制点运动矢量cpMvLX[0]和可用性标志availableFlagLX[0]在以下有序步骤中导出:

[0725] 分别将样本位置 (xNbB2, yNbB2)、(xNbB3, yNbB3) 和 (xNbA2, yNbA2) 设置为等于 (xCb-1, yCb-1)、(xCb, yCb-1) 和 (xCb-1, yCb)。

[0726] 将可用性标志availableFlagLX[0]设置为等于0,并且将cpMvLX[0]的两个分量都设置为等于0。

[0727] 对于 (xNbTL, yNbTL) 应用以下步骤,其中TL由B2、B3和A2替换:

[0728] 调用条款中指定的编解码块的可用性推导过程,其中亮度编解码块位置 (xCb, yCb)、亮度编解码块宽度cbWidth、亮度编解码块高度cbHeight、亮度位置 (xNbY, yNbY) 设置为等于 (xNbTL, yNbTL) 作为输入,并且输出被分配给编解码块可用性标志availableTL。

[0729] 当availableTL等于TRUE且availableFlagLX[0]等于0时,应用以下步骤:

[0730] 如果PredFlagLX[xNbTL][yNbTL]等于1,并且DiffPicOrderCnt (RefPicListX[RefIdxLX[xNbTL][yNbTL]], RefPicListX[refIdxLX]) 等于0,并且对应于RefIdxLX[xNbTL][yNbTL]的参考图片不是当前图片,则将availableFlagLX[0]设置为等于1并做出以下分配:

[0731]
$$\text{cpMvLX}[0] = \text{MvLX}[\text{xNbTL}][\text{yNbTL}] \quad (8-643)$$

[0732] 否则,当PredFlagLY[xNbTL][yNbTL] (with Y=!X) 等于1,并且DiffPicOrderCnt (RefPicListY[RefIdxLY[xNbTL][yNbTL]], RefPicListX[refIdxLX]) 等于0,并且对应于RefIdxLY[xNbTL][yNbTL]的参考图片不是当前图片时,则将availableFlagLX[0]设置为等于1并做出以下分配:

[0733]
$$\text{cpMvLX}[0] = \text{MvLY}[\text{xNbTL}][\text{yNbTL}] \quad (8-644)$$

[0734] 当availableFlagLX[0]等于1时,调用条款8.4.2.14中指定的运动矢量的取整过程,其中mvX设置为等于cpMvLX[0]、rightShift设置为等于(MvShift+2)、以及leftShift设置为等于(MvShift+2)作为输入,并且取整的cpMvLX[0]作为输出。

[0735] 第二(右顶)控制点运动矢量cpMvLX[1]和可用性标志availableFlagLX[1]在以下有序步骤中导出:

[0736] 分别将样本位置 (xNbB1, yNbB1) 和 (xNbB0, yNbB0) 设置为等于 (xCb+cbWidth-1, yCb-1) 和 (xCb+cbWidth, yCb-1)。

[0737] 将可用性标志availableFlagLX[1]设置为等于0,并且将cpMvLX[1]的两个分量都设置为等于0。

[0738] 对于(xNbTR,yNbTR)应用以下步骤,其中TR由B1和B0替换:

[0739] 调用条款6.4.X中指定的编解码块的可用性推导过程,其中亮度编解码块位置(xCb,yCb)、亮度编解码块宽度cbWidth、亮度编解码块高度cbHeight、亮度位置(xNbY,yNbY)设置为等于(xNbTR,yNbTR)作为输入,并且输出被分配给编解码块可用性标志availableTR。

[0740] 当availableTR等于TRUE且availableFlagLX[1]等于0时,应用以下步骤:

[0741] 如果PredFlagLX[xNbTR][yNbTR]等于1,并且DiffPicOrderCnt(RefPicListX[RefIdxLX[xNbTR][yNbTR]],RefPicListX[refIdxLX])等于0,并且对应于RefIdxLX[xNbTR][yNbTR]的参考图片不是当前图片,则将availableFlagLX[1]设置为等于1并做出以下分配:

[0742] $cpMvLX[1] = MvLX[xNbTR][yNbTR]$ (8-645)

[0743] 否则,当PredFlagLY[xNbTR][yNbTR] (with Y=!X) 等于1,并且DiffPicOrderCnt(RefPicListY[RefIdxLY[xNbTR][yNbTR]],RefPicListX[refIdxLX])等于0,并且对应于RefIdxLY[xNbTR][yNbTR]的参考图片不是当前图片时,则将availableFlagLX[1]设置为等于1并做出以下分配:

[0744] $cpMvLX[1] = MvLY[xNbTR][yNbTR]$ (8-646)

[0745] 当availableFlagLX[1]等于1时,调用条款8.4.2.14中指定的运动矢量的取整过程,其中mvX设置为等于cpMvLX[1]、rightShift设置为等于(MvShift+2)、以及leftShift设置为等于(MvShift+2)作为输入,并且取整的cpMvLX[1]作为输出。

[0746] 第三(左底)控制点运动矢量cpMvLX[2]和可用性标志availableFlagLX[2]在以下有序步骤中导出:

[0747] 分别将样本位置(xNbA1,yNbA1)和(xNbA0,yNbA0)设置为等于(xCb-1,yCb+cbHeight-1)和(xCb-1,yCb+cbHeight)。

[0748] 将可用性标志availableFlagLX[2]设置为等于0,并且将cpMvLX[2]的两个分量都设置为等于0。

[0749] 对于(xNbBL,yNbBL)应用以下步骤,其中BL由A1和A0替换:

[0750] 调用条款6.4.X中指定的编解码块的可用性推导过程,其中亮度编解码块位置(xCb,yCb)、亮度编解码块宽度cbWidth、亮度编解码块高度cbHeight、亮度位置(xNbY,yNbY)设置为等于(xNbBL,yNbBL)作为输入,并且输出被分配给编解码块可用性标志availableBL。

[0751] 当availableBL等于TRUE且availableFlagLX[2]等于0时,应用以下步骤:

[0752] 如果PredFlagLX[xNbBL][yNbBL]等于1,并且DiffPicOrderCnt(RefPicListX[RefIdxLX[xNbBL][yNbBL]],RefPicListX[refIdxLX])等于0,并且对应于RefIdxLY[xNbBL][yNbBL]的参考图片不是当前图片,则将availableFlagLX[2]设置为等于1并做出以下分配:

[0753] $cpMvLX[2] = MvLX[xNbBL][yNbBL]$ (8-647)

[0754] 否则,当PredFlagLY[xNbBL][yNbBL] (with Y=!X) 等于1,并且DiffPicOrderCnt(RefPicListY[RefIdxLY[xNbBL][yNbBL]],RefPicListX[refIdxLX])等于0,并且对应于RefIdxLY[xNbBL][yNbBL]的参考图片不是当前图片时,则将availableFlagLX[2]设置为等

于1并做出以下分配：

[0755] $cpMvLX[2] = MvLY[xNbL][yNbL]$ (8-648)

[0756] 当availableFlagLX[2]等于1时,调用条款8.4.2.14中指定的运动矢量的取整过程,其中mvX设置为等于cpMvLX[2]、rightShift设置为等于(MvShift+2)、以及leftShift设置为等于(MvShift+2)作为输入,并且取整的cpMvLX[2]作为输出。

[0757] 5.5上下文建模

[0758] 使用上下文编解码的二进制数将ctxInc分配给语法元素：

语法元素	binIdx					
	0	1	2	3	4	>= 5
[0759] amvr_flag[][]	0,1,2 (条款 9.5.4.2.2, 当 inter_affine_flag[][]等于 0 时)	na	na	na	na	na

语法元素	binIdx					
	0	1	2	3	4	>= 5
[0760] amvr_coarse_precision_flag[][]	0	na	na	na	na	na

[0761] 使用左方和上方的语法元素指定ctxInc：

[0762] 在一个示例中,上下文增加偏移 $ctxInc = (condL \&\&availableL) + (condA \&\&availableA) + ctxSetIdx * 3$ 。

[0763] 替代地, $ctxInc = ((condL \&\&availableL) \parallel (condA \&\&availableA)) + ctxSetIdx * 3$ 。

[0764] $ctxInc = (condL \&\&availableL) + M * (condA \&\&availableA) + ctxSetIdx * 3$. (例如, $M = 2$)

[0765] $ctxInc = M * (condL \&\&availableL) + (condA \&\&availableA) + ctxSetIdx * 3$. (例如, $M = 2$)

语法元素	condL	condA	ctxSetIdx
[0766] e	cu_skip_flag[xNbL][yNbL]	cu_skip_flag[xNbA][yNbA]	0
e	!inter_affine_flag[x0][y0] && amvr_flag[xNbL][yNbL]	!inter_affine_flag[x0][y0] && amvr_flag[xNbA][yNbA]	0

[0767] amvr_flag的ctxIdx的initValue值：

[0768] 当前块为仿射或非仿射时使用不同的上下文。

[0769]

初始化变量	当 inter_affine_flag 等于 0 时的 amvr_flag 的 ctxIdx			当 inter_affine_flag 等于 1 时的 amvr_flag 的 ctxIdx
	0	1	2	3
initValue	xx	xx	xx	xx

[0770] 替代地,

[0771]

初始化变量	当 inter_affine_flag 等于 0 时的 amvr_flag 的 ctxIdx	当 inter_affine_flag 等于 1 时的 amvr_flag 的 ctxIdx
	0	1
initValue	xx	xx

[0772] 替代地,当当前块为仿射或非仿射时可以使用相同的上下文

[0773]

初始化变量	当 inter_affine_flag 等于 0 时的 amvr_flag 的 ctxIdx	当 inter_affine_flag 等于 1 时的 amvr_flag 的 ctxIdx
	0	0
initValue	xx	xx

[0774] 替代地,amvr_flag被旁路编解码。

[0775] amvr_coarse_precision_flag的ctxIdx的initValue值:

[0776] 当当前块为仿射或非仿射时使用不同的上下文。

[0777]

初始化变量	当 inter_affine_flag 等于 0 时的 amvr_coarse_precision_flag 的 ctxIdx	当 inter_affine_flag 等于 1 时的 amvr_coarse_precision_flag 的 ctxIdx
	0	0
initValue	xxx	xx

[0778] 替代地,

[0779]

初始化变量	当 inter_affine_flag 等于 0 时的 amvr_coarse_precision_flag 的 ctxIdx	当 inter_affine_flag 等于 1 时的 amvr_coarse_precision_flag 的 ctxIdx
	0	1
initValue	xxx	xx

[0780] 替代地,当当前块为仿射或非仿射时可以使用相同的上下文。

[0781]	初始化变量	amvr_coarse_precision_flag 的 ctxIdx
		0
	initValue	xxx

[0782] 替代地,amvr_coarse_precision_flag被旁路编解码。

[0783] 上述示例可以并入在下面描述的方法的上下文中,例如,方法2510至2540,其可以在视频解码器或视频编码器处实现。

[0784] 图25A-25D示出了用于视频处理的示例性方法的流程图。图25A所示的方法2510包括,在步骤2512,确定视频的当前视频块与当前视频块的编解码表示之间的转换基于非仿射帧间AMVR模式。方法2510进一步包括,在步骤2514处,基于该确定来进行转换。在一些实现方式中,当前视频块的编解码表示基于基于上下文的编解码,并且其中在转换期间不使用相邻块的仿射AMVR模式信息的情况下来对用于编解码当前视频块的上下文进行建模。

[0785] 如图25B所示的方法2520包括,在步骤2522处,确定视频的当前视频块与当前视频块的编解码表示之间的转换基于仿射自适应运动矢量分辨率(仿射AMVR)模式。方法2520进一步包括,在步骤2524处,基于该确定来进行转换。在一些实现方式中,当前视频块的编解码表示基于基于上下文的编解码,并且其中变量控制上下文的两个概率更新速度。

[0786] 如图25C所示的方法2530包括,在步骤2532处,确定视频的当前视频块与当前视频块的编解码表示之间的转换基于仿射AMVR模式。方法2530进一步包括,在步骤2534处,基于该确定来进行转换。在一些实现方式中,当前视频块的编解码表示基于基于上下文的编解码,并且其中使用相邻块的编解码信息来对用于编解码当前视频块的上下文进行建模,该相邻块的编解码信息在转换期间使用仿射帧间模式和正常帧间模式两者的AMVR模式。

[0787] 如图25D所示的方法2540包括,在步骤2542处,对于视频的当前视频块与当前视频块的编解码表示之间的转换,确定用于转换的多个上下文的使用。方法2540进一步包括,在步骤2544处,基于该确定来进行转换。在一些实现方式中,多个上下文被用于对指示粗运动精度的语法元素进行编解码。

[0788] 如图25E中所示的方法2550包括,在步骤2552处,对于视频的当前视频块与当前视频块的编解码表示之间的转换,基于用于转换的当前选择的最佳模式,确定是否使用对称运动矢量差(SMVD)模式。方法2554进一步包括:基于该确定来进行转换。

[0789] 如图25F所示的方法2560包括,在步骤2562处,对于视频的当前视频块与当前视频块的编解码表示之间的转换,基于用于转换的当前选择的最佳模式,确定是否使用仿射SMVD模式。方法2564进一步包括,基于该确定来进行转换。

[0790] 6.所公开的技术的示例实现方式

[0791] 图26是视频处理装置2600的框图。装置2600可以用于实现本文描述的一种或多种方法。装置2600可以实施在智能手机、平板计算机、计算机、物联网(IoT)接收器等中。装置2600可以包括一个或多个处理器2602、一个或多个存储器2604和视频处理硬件2606。(一个或多个)处理器2602可以被配置为实现本文件中描述的一种或多种方法(包括但不限于方

法2500)。(一个或多个)存储器2604可以用于存储用于实现本文描述的方法和技术的数据和代码。视频处理硬件2606可以用于在硬件电路中实现本文件中描述的一些技术。

[0792] 图28是可以实现所公开的技术的视频处理系统的框图的另一个示例。图28是示出可以实现本文公开的各种技术的示例视频处理系统2800的框图。各种实现可以包括系统2800的一些或所有组件。系统2800可以包括用于接收视频内容的输入2802。可以以例如8或10比特多分量像素值的原始或未压缩格式接收视频内容,或者可以以压缩或编码格式接收视频内容。输入2802可以表示网络接口、外围总线接口或存储接口。网络接口的示例包括诸如以太网、无源光网络(PON)等的有线接口和诸如Wi-Fi或蜂窝接口的无线接口。

[0793] 系统2800可以包括编解码组件2804,其可以实现本文件中描述的各种编解码(coding)或编码(encoding)方法。编解码组件2804可以降低从输入2802到编解码组件2804的输出的视频的平均比特率以生成视频的编解码表示。因此,编解码技术有时被称为视频压缩或视频转码技术。编解码组件2804的输出可以被存储,或者经由连接的通信被传输,如组件2806所示。在输入2802处接收的视频的存储或传送的比特流(或编解码)表示可以由组件2808使用,以用于生成发送到显示接口2810的像素值或可显示视频。从比特流表示生成用户可视视频的过程有时被称为视频解压缩。此外,虽然某些视频处理操作被称为“编解码”操作或工具,但是应当理解,编解码工具或操作在编码器处使用,并且对应的解码工具或反转编解码结果的操作将由解码器执行。

[0794] 外围总线接口或显示接口的示例可以包括通用串行总线(USB)或高清晰度多媒体接口(HDMI)或Displayport等。存储接口的示例包括SATA(串行高级技术附件)、PCI、IDE接口等。本文件中描述的技术可以实施在各种电子设备中,诸如移动电话、膝上型计算机、智能手机或能够执行数字数据处理和/或视频显示的其它设备。

[0795] 在一些实施例中,视频编解码方法可以使用在如关于图26或28所描述的硬件平台上实现的装置来实现。

[0796] 可以使用以下基于条款的格式来描述各种技术和实施例。这些条款可以实现为一些实施例的优选特征。

[0797] 条款的第一集合使用先前节中描述的一些技术,包括例如先前节中的项目16-18、23和26。

[0798] 1.一种视频处理方法,包括:

[0799] 确定视频的当前视频块与所述当前视频块的编解码表示之间的转换基于非仿射帧间AMVR模式;以及

[0800] 基于所述确定来进行所述转换,并且

[0801] 其中,所述当前视频块的编解码表示基于基于上下文的编解码,并且其中在所述转换期间不使用相邻块的仿射AMVR模式信息的情况下来对用于编解码所述当前视频块的上下文进行建模。

[0802] 2.根据条款1所述的方法,其中,利用编解码的所述相邻块的非仿射帧间AMVR模式索引。

[0803] 3.根据条款2所述的方法,其中,用于对所述当前视频块的非仿射帧间AMVR模式索引进行编解码的上下文索引取决于以下中的至少一个:AMVR模式索引、仿射模式标志、或者左方和上方相邻块的可用性、指示某一MVD精度的所述非仿射帧间AMVR模式索引。

[0804] 4. 根据条款3所述的方法, 其中, 分别为所述左方相邻块和所述上方相邻块导出偏移值, 并且将所述上下文索引导出为所述左方相邻块和所述上方相邻块的偏移值之和。

[0805] 5. 根据条款4所述的方法, 其中, 在对应的相邻块可用且未以仿射模式编解码并且所述相邻块的AMVR模式索引不等于0的情况下, 将所述偏移值中的至少一个导出为1, 否则将所述偏移值中的至少一个导出为0。

[0806] 6. 根据条款1所述的方法, 其中, 所述相邻块的仿射AMVR模式信息不是直接使用的, 而是以所述仿射AMVR模式信息的函数来间接使用。

[0807] 7. 根据条款6所述的方法, 其中, 在所述相邻块覆盖 (xNbL, yNbL) 的amvr_mode[xNbL][yNbL]或amvr_flag[xNbL][yNbL]指示某一MVD精度的情况下, 所述函数返回真。

[0808] 8. 根据条款1所述的方法, 其中, 利用所述相邻块的仿射AMVR模式信息对所述当前视频块的非仿射帧间AMVR模式的第一语法元素进行编解码。

[0809] 9. 根据条款1所述的方法, 其中, 在所述当前视频块的AMVR模式由多个语法元素表示的情况下, 利用以非仿射帧间模式和仿射帧间模式两者编解码的所述相邻块的AMVR模式信息来对所述多个语法元素中的任何一个进行编解码。

[0810] 10. 根据条款1所述的方法, 其中, 在所述当前视频块的AMVR模式由多个语法元素表示的情况下, 不利用以仿射帧间模式编解码的所述相邻块的AMVR模式信息来对所述多个语法元素中的任何一个进行编解码。

[0811] 11. 根据条款1所述的方法, 其中, 在所述当前视频块的AMVR模式由多个语法元素表示的情况下, 不直接利用而是间接使用以仿射帧间模式编解码的所述相邻块的AMVR模式信息来对所述多个语法元素中的任何一个进行编解码。

[0812] 12. 根据条款9-11中的任一项所述的方法, 其中, 所述当前视频块的AMVR模式包括仿射间AMVR模式和非仿射间AMVR模式。

[0813] 13. 一种视频处理方法, 包括:

[0814] 确定视频的当前视频块与所述当前视频块的编解码表示之间的转换基于仿射自适应运动矢量分辨率(仿射AMVR)模式; 以及

[0815] 基于所述确定来进行所述转换, 并且

[0816] 其中, 所述当前视频块的编解码表示基于基于上下文的编解码, 并且其中变量控制上下文两个概率更新速度。

[0817] 14. 根据条款13所述的方法, 其中, 所述两个概率更新速度包括由 $(\text{shiftIdx} \gg 2) + 2$ 定义的较快的更新速度, shiftIdx指示所述变量。

[0818] 15. 根据条款13所述的方法, 其中, 所述两个概率更新速度包括由 $(\text{shiftIdx} \& 3) + 3 + \text{shift0}$ 定义的较慢的更新速度, shift0由 $(\text{shiftIdx} \gg 2) + 2$ 定义, 并且shiftIdx指示所述变量。

[0819] 16. 根据条款14所述的方法, 其中, 所述较快的更新速度在2到5之间。

[0820] 17. 一种视频处理方法, 包括:

[0821] 确定视频的当前视频块与所述当前视频块的编解码表示之间的转换基于仿射AMVR模式; 以及

[0822] 基于所述确定来进行所述转换, 并且

[0823] 其中, 所述当前视频块的编解码表示基于基于上下文的编解码, 并且其中使用相

邻块的编解码信息来对用于编解码所述当前视频块的上下文进行建模,所述相邻块的编解码信息在所述转换期间使用仿射帧间模式和正常帧间模式两者的AMVR模式。

[0824] 18.根据条款17所述的方法,其中,所述相邻块的AMVR模式信息被直接用于所述基于上下文的编解码。

[0825] 19.根据条款17所述的方法,其中,不允许将以正常帧间模式编解码的所述相邻块的AMVR模式信息用于所述基于上下文的编解码。

[0826] 20.根据条款17所述的方法,其中,以正常帧间模式编解码的所述相邻块的AMVR模式信息不是直接使用的,而是以所述AMVR模式信息的函数来间接使用。

[0827] 21.根据条款17所述的方法,其中,利用所述相邻块的AMVR模式信息来对所述当前视频块的仿射AMVR模式的第一语法元素进行编解码。

[0828] 22.根据条款17所述的方法,其中,在所述仿射AMVR模式由多个语法元素表示的情况下,利用以仿射帧间模式和正常帧间模式两者编解码的所述相邻块的AMVR模式信息来对所述多个语法元素中的任何一个进行编解码。

[0829] 23.根据条款17所述的方法,其中,在所述仿射AMVR模式由多个语法元素表示的情况下,不利用以正常帧间模式编解码的所述相邻块的AMVR模式信息来对所述多个语法元素中的任何一个进行编解码。

[0830] 24.根据条款17所述的方法,其中,在所述仿射AMVR模式由多个语法元素表示的情况下,不直接利用而是间接使用以正常帧间模式编解码的所述相邻块的AMVR模式信息来对所述多个语法元素中的任何一个进行编解码。

[0831] 25.一种视频处理方法,包括:

[0832] 对于视频的当前视频块与所述当前视频块的编解码表示之间的转换,确定用于所述转换的多个上下文的使用;以及

[0833] 基于所述确定来进行所述转换,并且

[0834] 其中,利用所述多个上下文来对指示粗运动精度的语法元素进行编解码。

[0835] 26.根据条款25所述的方法,其中,所述多个上下文恰好对应于两个上下文。

[0836] 27.根据条款26所述的方法,其中,基于所述当前视频块是否被仿射编解码来选择所述多个上下文。

[0837] 28.根据条款25所述的方法,其中,另一语法元素用于指示所述转换基于仿射AMVR模式。

[0838] 29.根据条款28所述的方法,其中,所述另一语法元素仅使用第一上下文进行编解码,并且所述语法元素仅使用第二上下文进行编解码。

[0839] 30.根据条款28所述的方法,其中,所述另一语法元素仅使用第一上下文进行编解码,并且所述语法元素被旁路编解码。

[0840] 31.根据条款28所述的方法,其中,所述另一语法元素被旁路编解码,并且所述语法元素被旁路编解码。

[0841] 32.根据条款25所述的方法,其中,与运动矢量精度有关的所有语法元素均被旁路编解码。

[0842] 33.根据条款25或26所述的方法,其中,对于正常帧间模式,所述语法元素指示从包括1像素或4像素精度的集合中进行选择。

[0843] 34. 根据条款25或26所述的方法,其中,对于仿射模式,所述语法元素指示从至少1/16像素或1像素精度之间进行选择。

[0844] 35. 根据条款34所述的方法,其中,所述语法元素等于0,则所述仿射模式为1/6像素精度。

[0845] 36. 根据条款34所述的方法,其中,所述语法元素等于1,则所述仿射模式为1像素精度。

[0846] 37. 根据条款1至36中任一项所述的方法,其中,进行所述转换包括从所述当前视频块生成所述编解码表示。

[0847] 38. 根据条款1至36中任一项所述的方法,其中,进行所述转换包括从所述编解码表示生成所述当前视频块。

[0848] 39. 一种视频系统中的装置,包括处理器和其上具有指令的非暂时性存储器,其中所述指令在由所述处理器执行时使所述处理器实现条款1至38中的一项或多项所述的方法。

[0849] 40. 一种存储在非暂时性计算机可读介质上的计算机程序产品,所述计算机程序产品包括用于实施条款1至38中的一项或多项所述的方法的程序代码。

[0850] 条款的第二集合使用先前节中描述的一些技术,包括例如先前节中的项目19-21和23。

[0851] 1. 一种视频处理方法,包括:

[0852] 对于视频的当前视频块与所述当前视频块的编解码表示之间的转换,基于用于所述转换的当前选择的最佳模式来确定是否使用对称运动矢量差 (SMVD) 模式;以及

[0853] 基于所述确定来进行所述转换。

[0854] 2. 根据条款1所述的方法,其中,在不显式信令通知参考列表的至少一个参考索引的情况下使用所述SMVD模式。

[0855] 3. 根据条款2所述的方法,其中,基于递归图片顺序计数 (POC) 计算来导出所述参考索引。

[0856] 4. 根据条款1所述的方法,其中,在当前选择的最佳模式是Merge模式或UMVE模式的情况下,所述确定禁用所述SMVD模式的使用。

[0857] 5. 根据条款4所述的方法,其中,所述UMVE模式应用运动矢量偏移以细化从Merge候选列表导出的运动候选。

[0858] 6. 根据条款1所述的方法,其中,在当前选择的最佳模式未使用所述SMVD模式编解码的情况下,所述确定禁用所述SMVD模式的使用。

[0859] 7. 根据条款1所述的方法,其中,在所述当前选择的最佳模式是仿射模式的情况下,所述确定禁用所述SMVD模式的使用。

[0860] 8. 根据条款1所述的方法,其中,在所述当前选择的最佳模式是子块Merge模式的情况下,所述确定禁用所述SMVD模式的使用。

[0861] 9. 根据条款1所述的方法,其中,在所述当前选择的最佳模式是仿射SMVD模式的情况下,所述确定禁用所述SMVD模式的使用。

[0862] 10. 根据条款1所述的方法,其中,在所述当前选择的最佳模式是仿射Merge模式的情况下,所述确定禁用所述SMVD模式的使用。

[0863] 11. 根据条款2至10中的任一项所述的方法, 其中, 仅当MVD(运动矢量差)精度大于或等于精度时才应用所述确定。

[0864] 12. 根据条款2至10中任一项所述的方法, 其中, 仅当MVD精度大于精度时才应用所述确定。

[0865] 13. 根据条款2至10中的任一项所述的方法, 其中, 仅当MVD精度小于或等于精度时才应用所述确定。

[0866] 14. 根据条款2至10中的任一项所述的方法, 其中, 仅当MVD精度小于精度时才应用所述确定。

[0867] 15. 一种视频处理方法, 包括:

[0868] 对于视频的当前视频块与所述当前视频块的编解码表示之间的转换, 基于用于所述转换的当前选择的最佳模式来确定是否使用仿射SMVD模式; 以及

[0869] 基于所述确定来进行所述转换。

[0870] 16. 根据条款15所述的方法, 其中, 在所述当前选择的最佳模式是Merge模式或UMVE模式的情况下, 所述确定禁用所述仿射SMVD模式的使用。

[0871] 17. 根据条款15所述的方法, 其中, 在所述当前选择的最佳模式未使用仿射SMVD模式编解码的情况下, 所述确定禁用所述仿射SMVD模式的使用。

[0872] 18. 根据条款15所述的方法, 其中, 在所述当前选择的最佳模式是子块Merge模式的情况下, 所述确定禁用所述仿射SMVD模式的使用。

[0873] 19. 根据条款15所述的方法, 其中, 在所述当前选择的最佳模式是SMVD模式的情况下, 所述确定禁用所述仿射SMVD模式的使用。

[0874] 20. 根据条款15所述的方法, 其中, 在所述当前选择的最佳模式是仿射Merge模式的情况下, 所述确定禁用所述仿射SMVD模式的使用。

[0875] 21. 根据条款16至20中的任一项所述的方法, 其中, 仅当仿射MVD精度大于或等于精度时才应用所述确定。

[0876] 22. 根据条款16至20中任一项所述的方法, 其中, 仅当仿射MVD精度大于精度时才应用所述确定。

[0877] 23. 根据条款16至20中的任一项所述的方法, 其中, 仅当仿射MVD精度小于或等于精度时才应用所述确定。

[0878] 24. 根据条款16至20中的任一项所述的方法, 其中, 仅当仿射MVD精度小于精度时才应用所述确定。

[0879] 25. 根据条款1或15所述的方法, 进一步包括:

[0880] 确定所述当前视频块的特性, 所述特性包括以下中一个或多个: 所述视频块的块尺寸、所述视频块的条带或图片或片的类型或与所述视频块有关的运动信息, 并且其中所述确定为也基于所述特性。

[0881] 26. 根据条款1至25中任一项所述的方法, 其中, 进行所述转换包括从所述当前块生成所述编解码表示。

[0882] 27. 根据条款1至25中任一项所述的方法, 其中, 进行所述转换包括从所述编解码表示生成所述当前块。

[0883] 28. 一种视频系统中的装置, 包括处理器和其上具有指令的非暂时性存储器, 其

中,所述指令在由所述处理器执行时使所述处理器实现条款1至27中的一项或多项所述的方法。

[0884] 29.一种存储在非暂时性计算机可读介质上的计算机程序产品,所述计算机程序产品包括用于实施条款1至27中的一项或多项所述的方法的程序代码。

[0885] 从前述内容可以理解,本文已经出于说明的目的描述了本公开技术的具体实施例,但是在不脱离本发明的范围的情况下可以进行各种修改。因此,除了所附权利要求之外,本发明所公开的技术不受限制。

[0886] 本专利文件描述的主题和功能操作的实现方式可以以各种系统实现,以数字电子电路实现,或者以计算机软件、固件或硬件实现,包括本说明书中公开的结构及其结构等同物,或者以它们中的一个或多个的组合实现。本说明书所描述的主题的实现方式可以实现为一个或多个计算机程序产品,即,在有形和非暂时性计算机可读介质上编码的一个或多个计算机程序指令模块,用于由数据处理装置执行或控制数据处理装置的操作。计算机可读介质可以是机器可读存储设备、机器可读存储基板、存储器设备、影响机器可读传播信号的物质组合、或者它们中的一个或多个的组合。术语“数据处理单元”或“数据处理装置”涵盖用于处理数据的所有装置、设备和机器,包括例如可编程处理器、计算机或多个处理器或计算机。除了硬件之外,该装置还可以包括为所讨论的计算机程序创建执行环境的代码,例如,构成处理器固件、协议栈、数据库管理系统、操作系统、或者它们中的一个或多个的组合的代码。

[0887] 计算机程序(也称为程序、软件、软件应用、脚本或代码)可以以任何形式的编程语言编写,包括编译或解释语言,并且可以以任何形式来部署计算机程序,包括作为独立程序或作为适合在计算环境中使用的模块、组件、子例程或其它单元。计算机程序不一定对应于文件系统中的文件。程序可以存储在保存其它程序或数据的文件的一部分中(例如,存储在标记语言文件中的一个或多个脚本),存储在专用于所讨论的程序的单个文件中,或存储在多个协调文件中(例如,存储一个或多个模块、子程序或代码部分的文件)。可以部署计算机程序以在一个计算机上或在位于一个站点上或分布在多个站点上并由通信网络互连的多个计算机上执行。

[0888] 本说明书中描述的过程和逻辑流程可以由执行一个或多个计算机程序的一个或多个可编程处理器执行,以通过对输入数据进行操作并生成输出来执行功能。过程和逻辑流程也可以由专用逻辑电路执行,并且装置也可以实现为专用逻辑电路,例如FPGA(现场可编程门阵列)或ASIC(专用集成电路)。

[0889] 举例来说,适合于执行计算机程序的处理器包括通用和专用微处理器、以及任何种类的数字计算机的任何一个或多个处理器。通常,处理器将从只读存储器或随机存取存储器或两者接收指令和数据。计算机的基本元件是用于执行指令的处理器和用于存储指令和数据的一个或多个存储器设备。通常,计算机还将包括或可操作地耦合到用于存储数据的一个或多个大容量存储设备,例如磁盘、磁光盘或光盘,以从该一个或多个大容量存储设备接收数据,或将数据传递到该一个或多个大容量存储设备,或者既接收又传递数据。然而,计算机不需要具有这样的设备。适用于存储计算机程序指令和数据的计算机可读介质包括所有形式的非易失性存储器、介质和存储器设备,举例来说,包括半导体存储器设备,例如EPROM、EEPROM和闪存设备。处理器和存储器可以由专用逻辑电路补充或并入专用逻辑

电路中。

[0890] 旨在将说明书与附图一起仅视为示例性的,其中示例性意味着示例。如本文所使用的,除非上下文另有明确指示,否则“或”的使用旨在包括“和/或”。

[0891] 虽然本专利文件包含许多细节,但这些细节不应被解释为对任何发明或可要求保护的范围的限制,而是作为特定于特定发明的特定实施例的特征的描述。在本专利文件中,在分开的实施例的上下文中描述的某些特征也可以在单个实施例中组合实现。相反,在单个实施例的上下文中描述的各种特征也可以分开地或以任何合适的子组合在多个实施例中实现。此外,尽管上面的特征可以描述为以某些组合起作用并且甚至最初如此要求保护,但是在一些情况下,可以从所要求保护的组合中去除来自该组合的一个或多个特征,并且所要求保护的组合可以指向子组合或子组合的变型。

[0892] 类似地,虽然在附图中以特定顺序描绘了操作,但是这不应该被理解为要求以所示的特定顺序或按顺序执行这样的操作,或者执行所有示出的操作,以实现期望的结果。此外,在本专利文件中描述的实施例中的各种系统组件的分离不应被理解为在所有实施例中都要求这样的分离。

[0893] 仅描述了几个实现方式和示例,并且可以基于本专利文件中描述和示出的内容来做出其它实现方式、增强和变型。

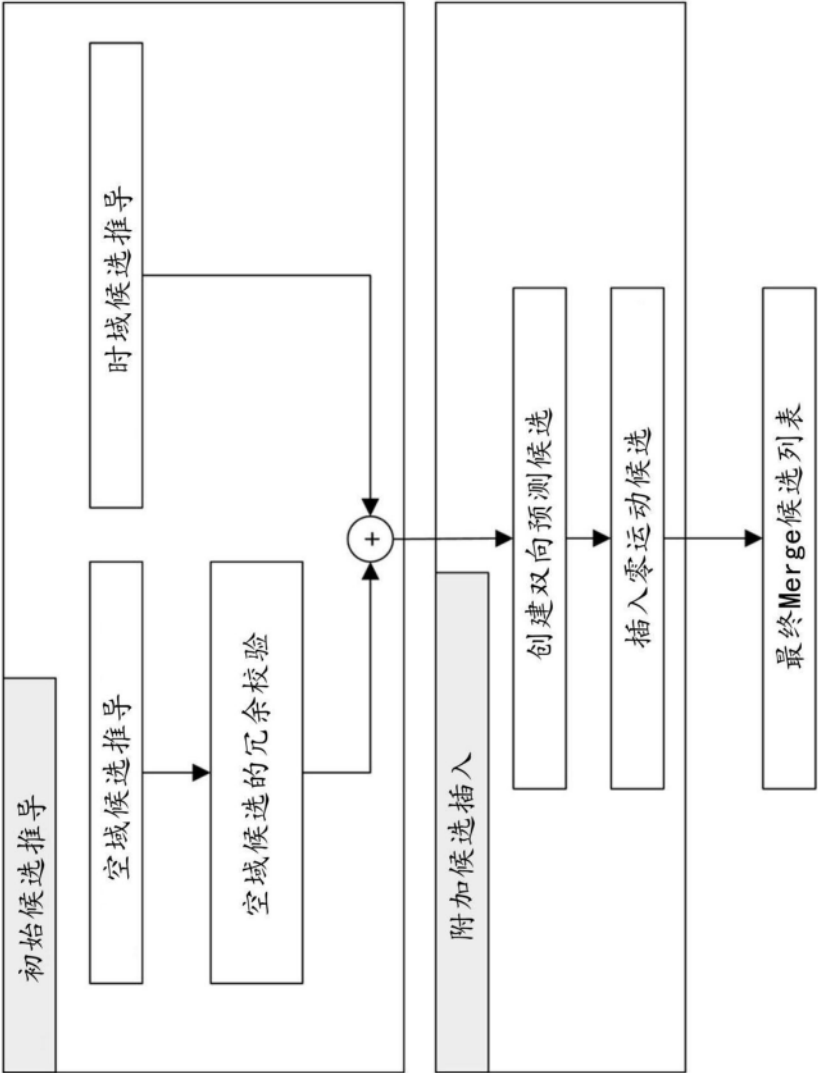


图1

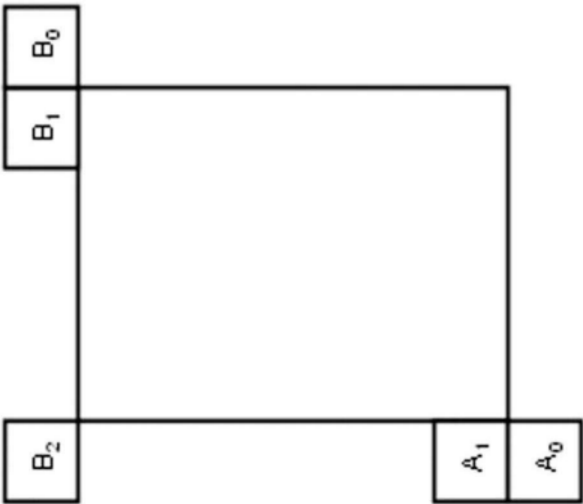


图2

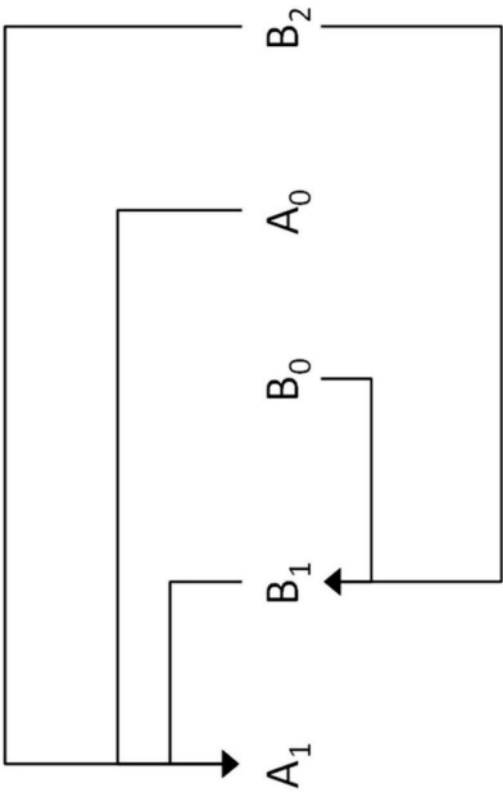


图3

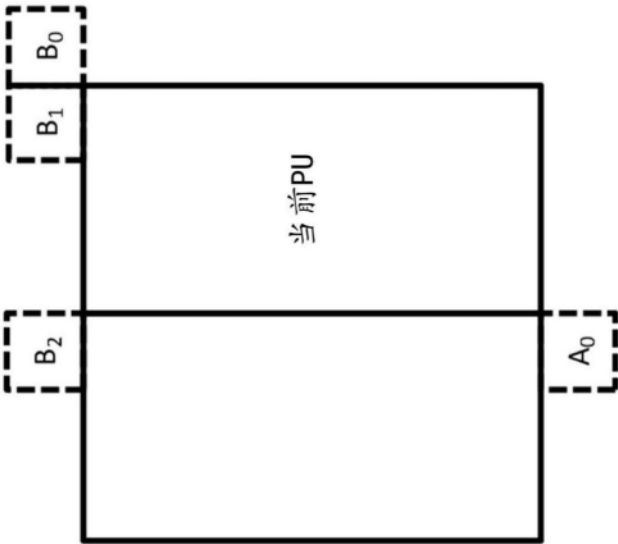


图4A

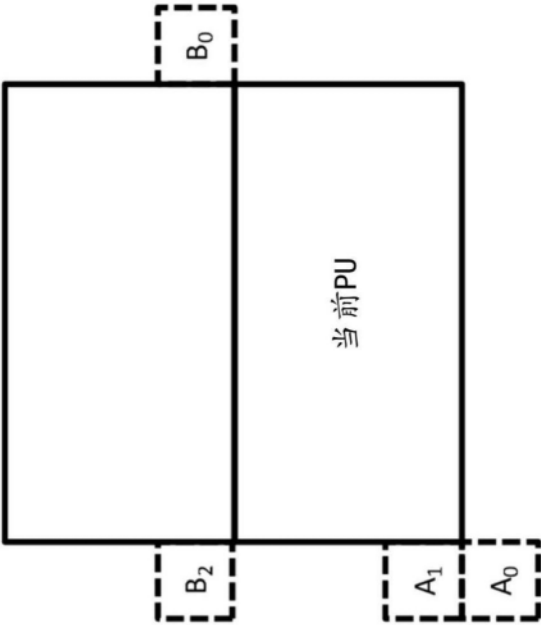


图4B

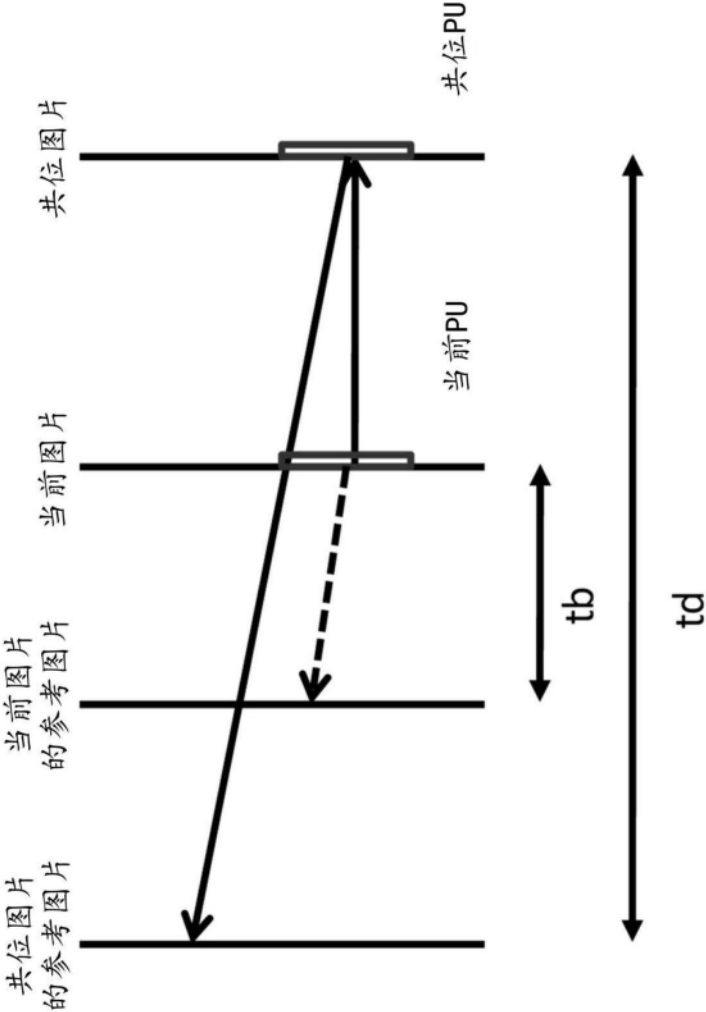


图5

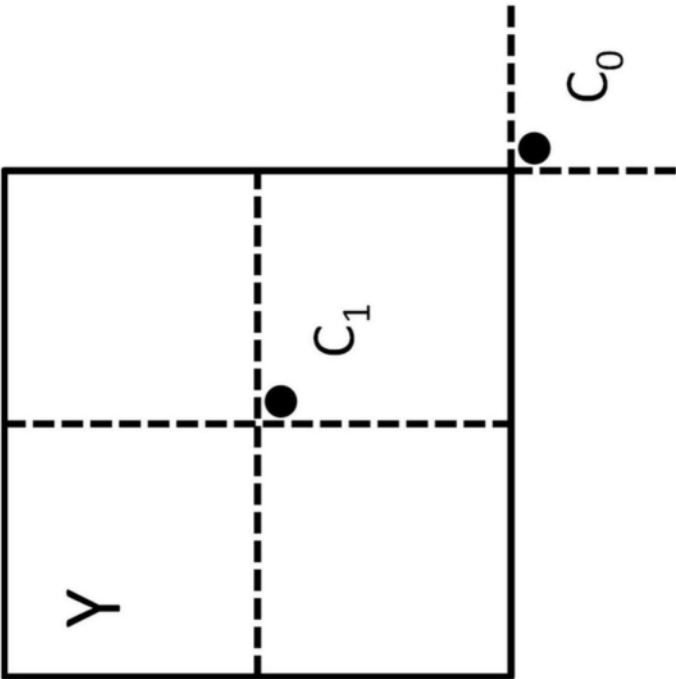


图6

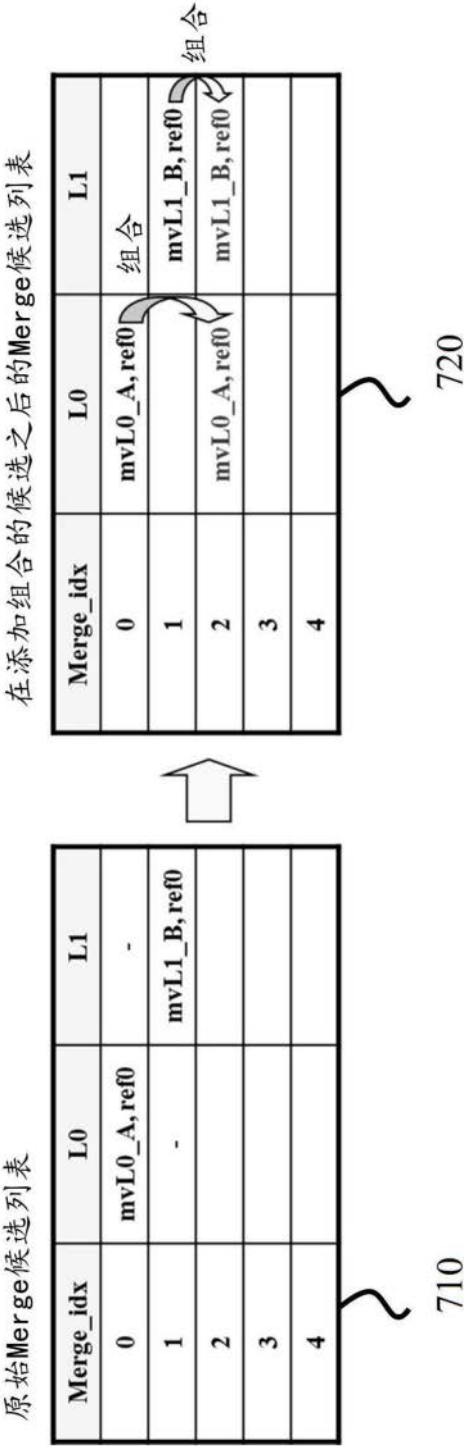


图7

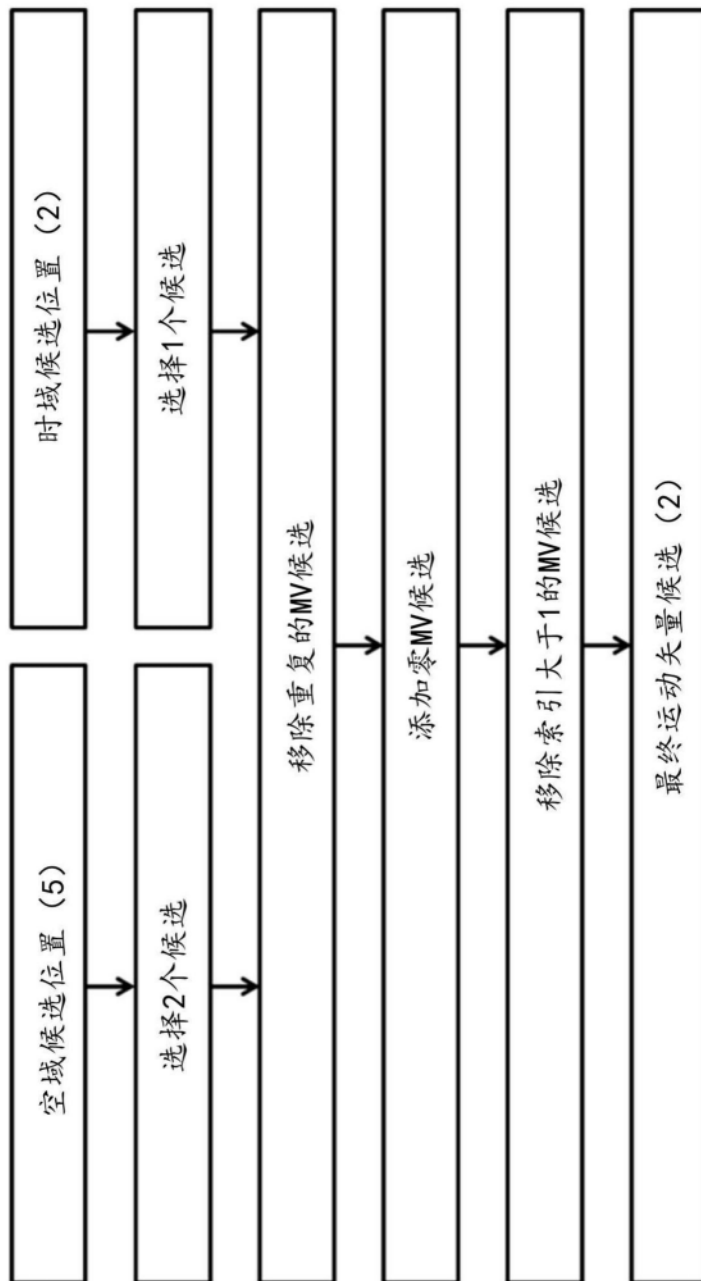


图8

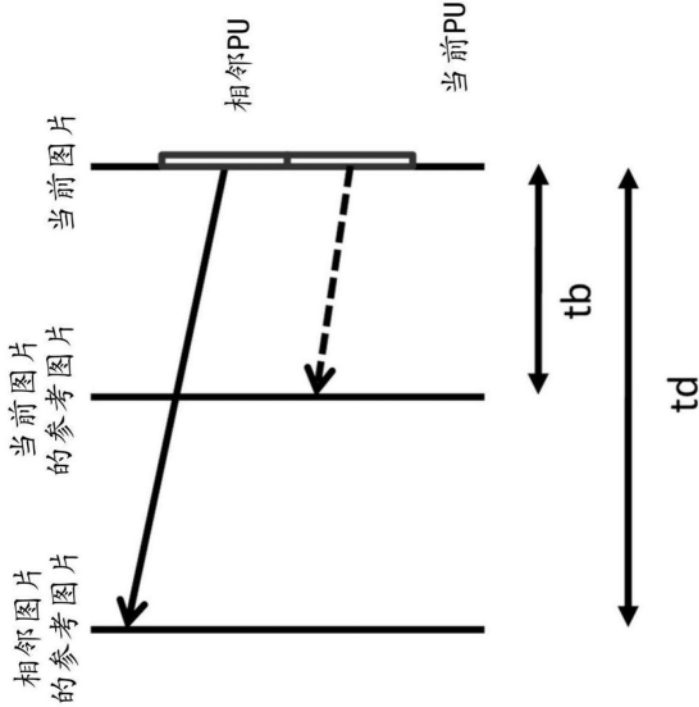


图9

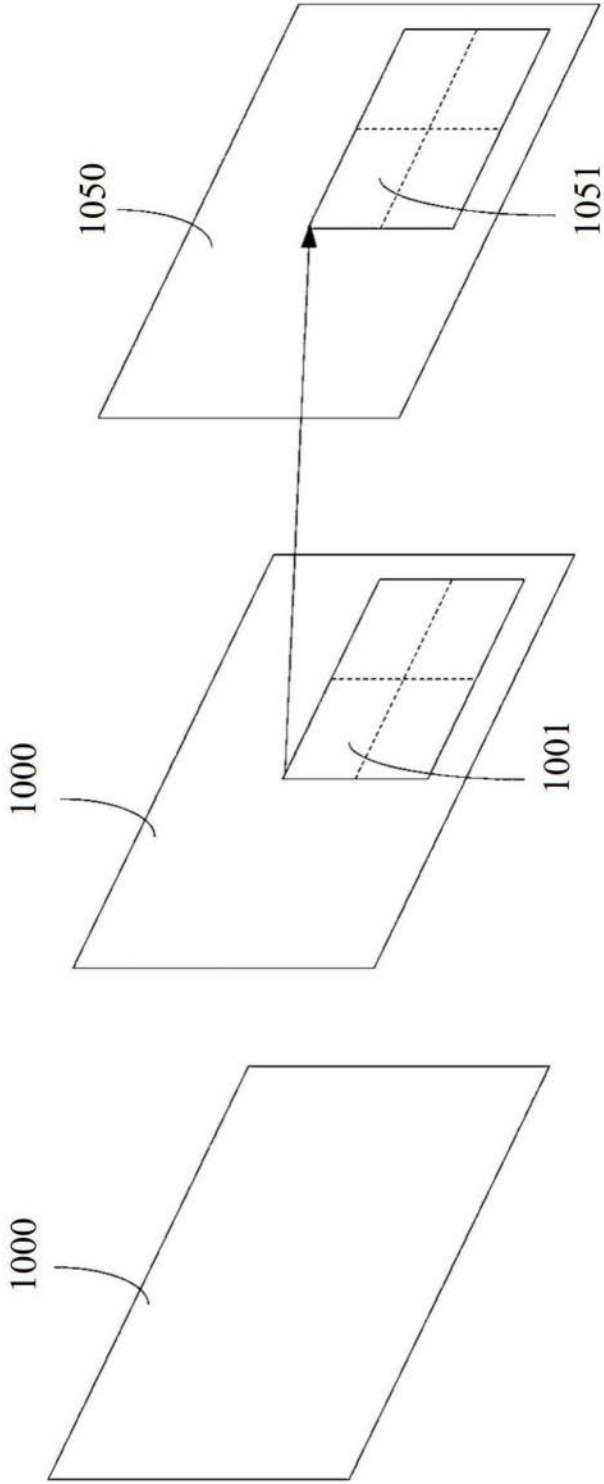


图10

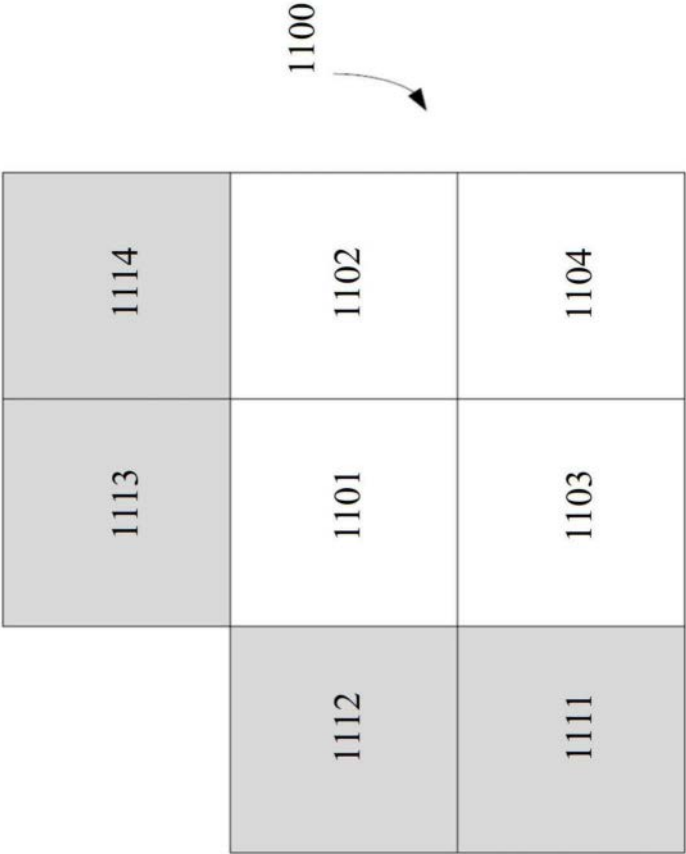


图11

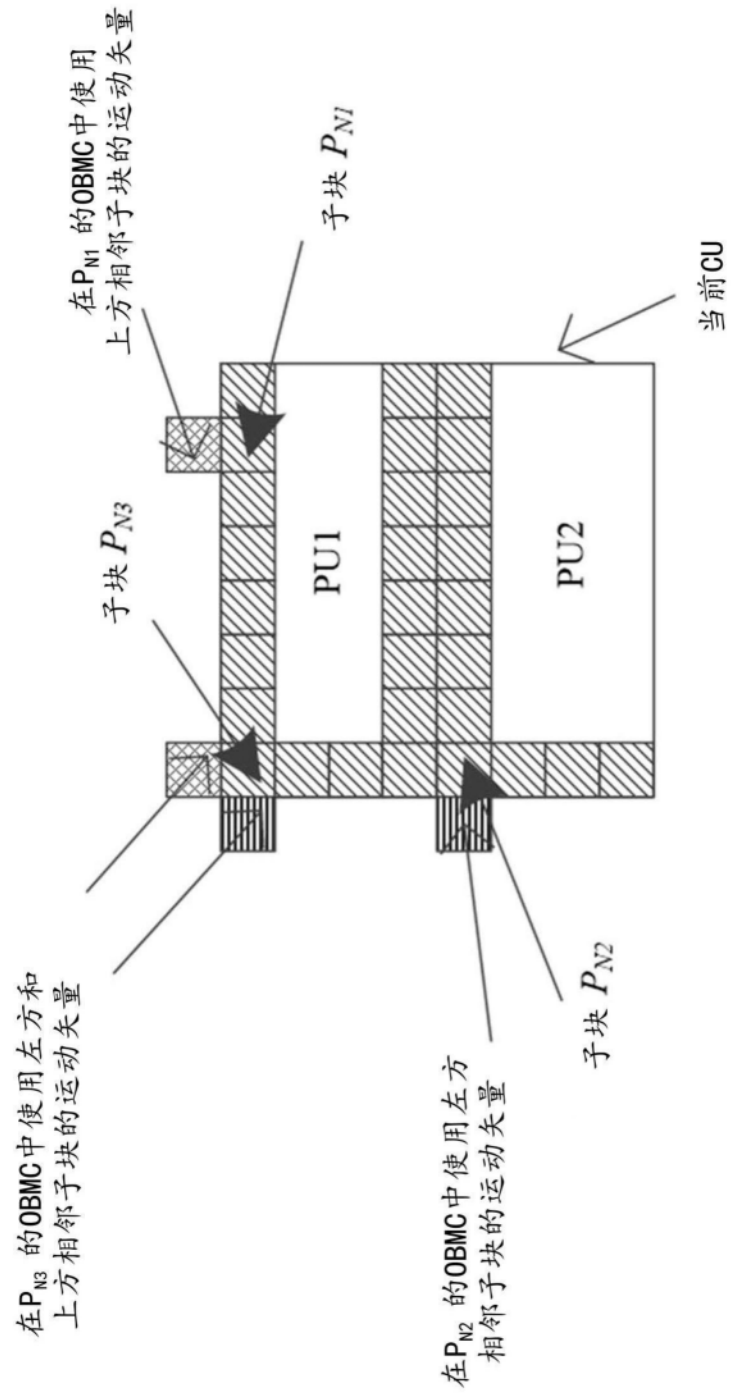


图12A

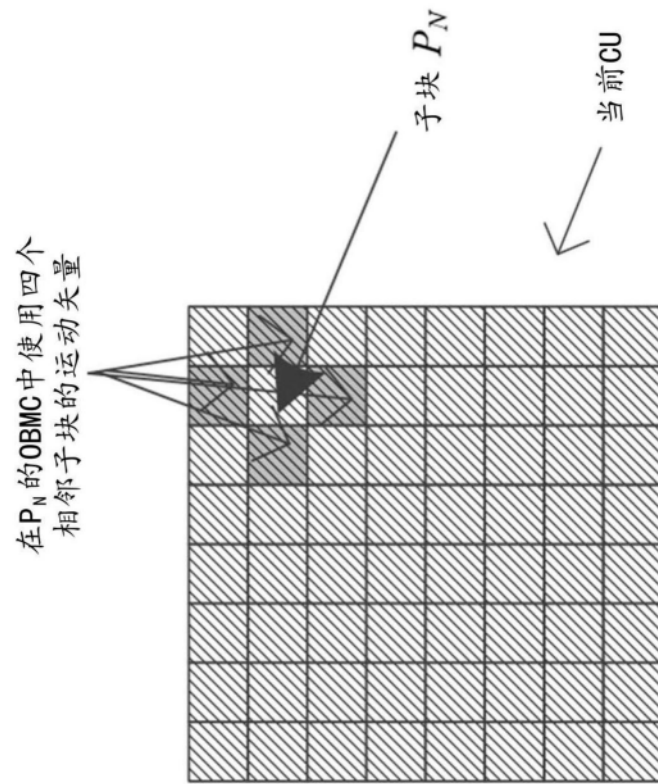


图12B

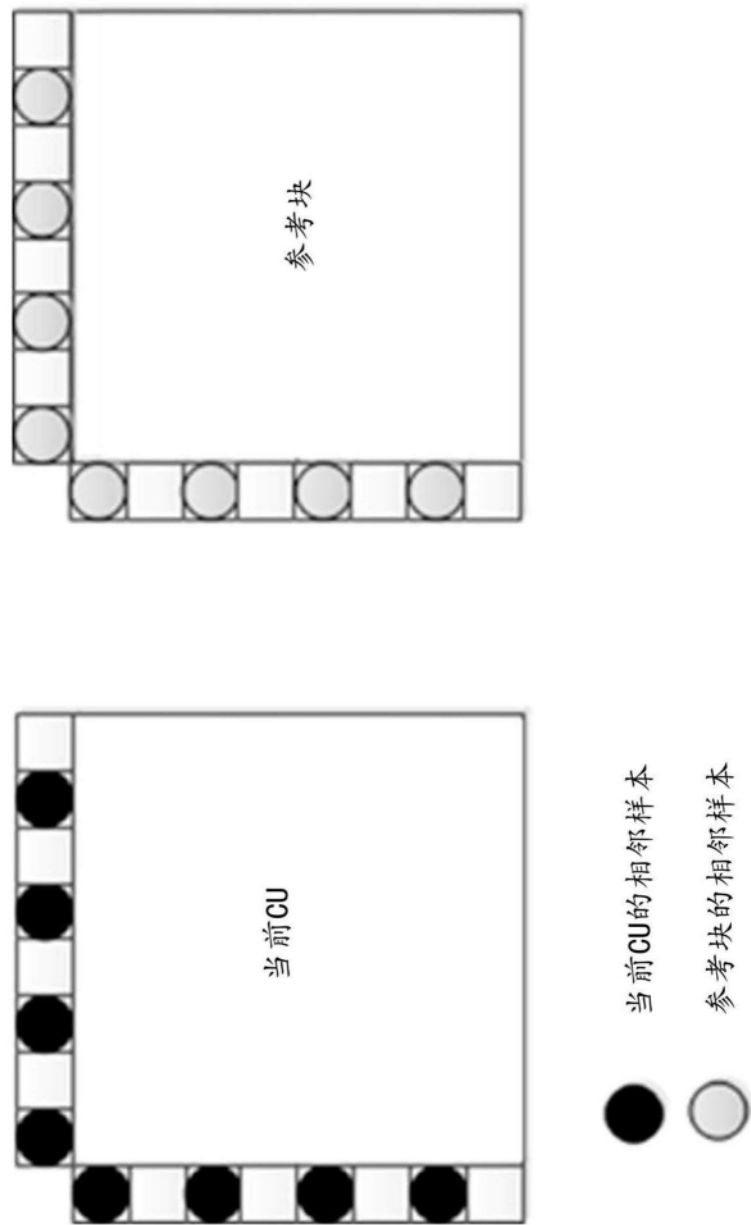


图13

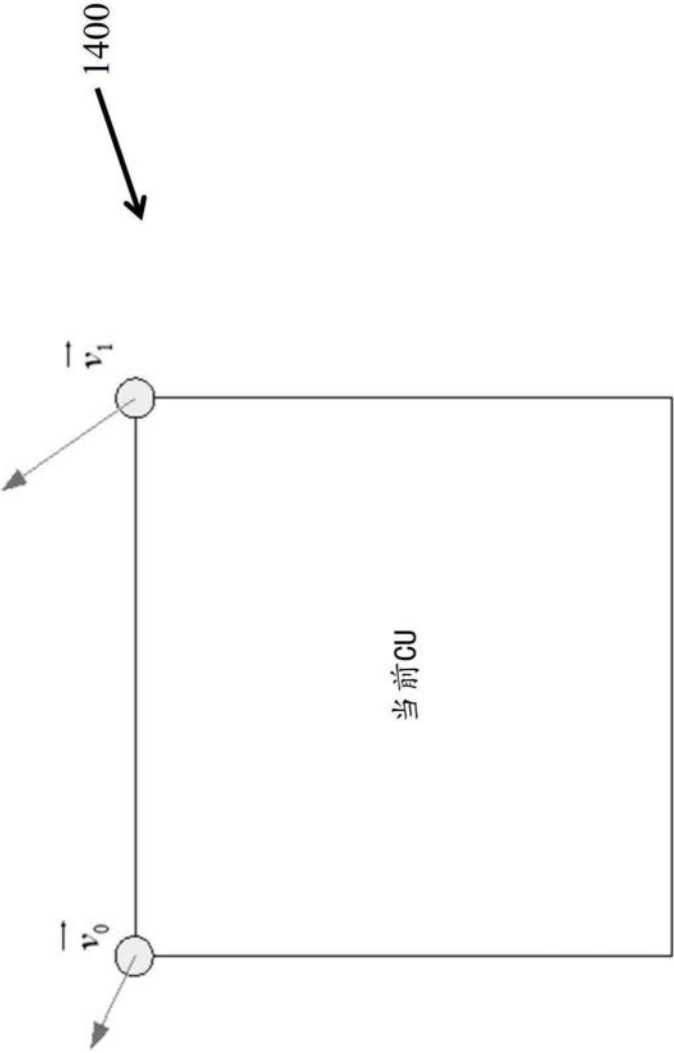


图14

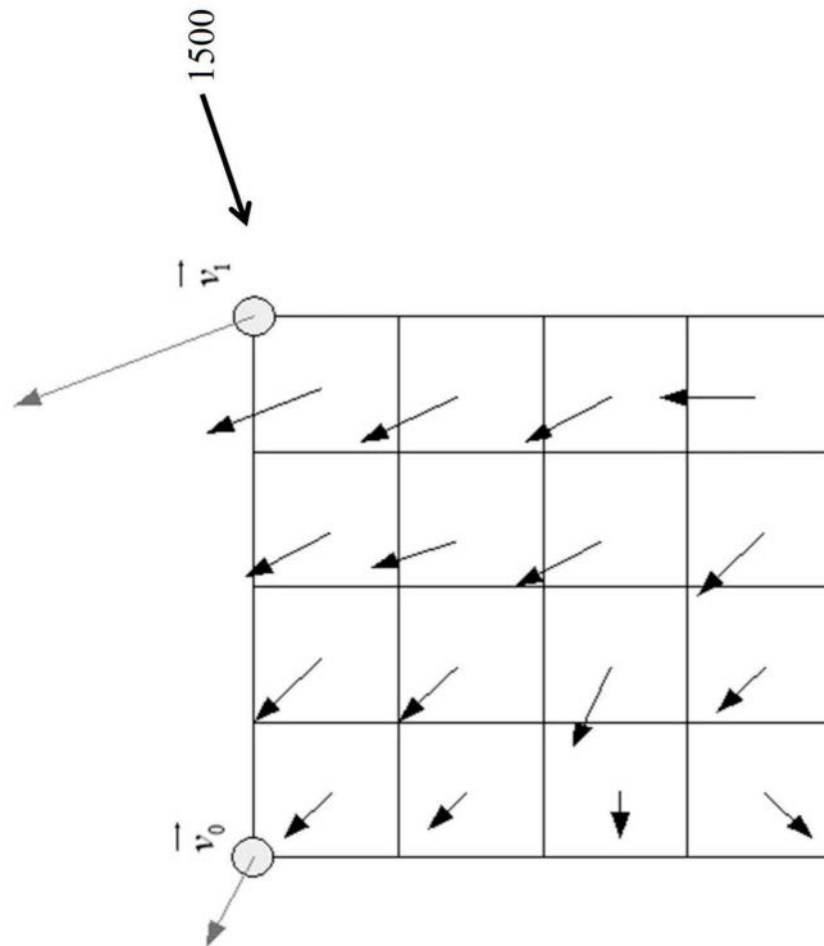


图15

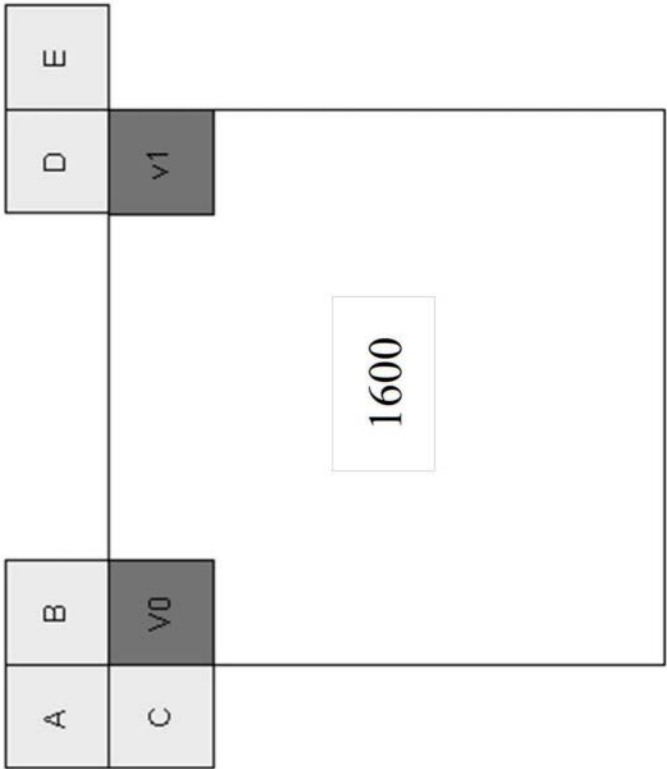


图16

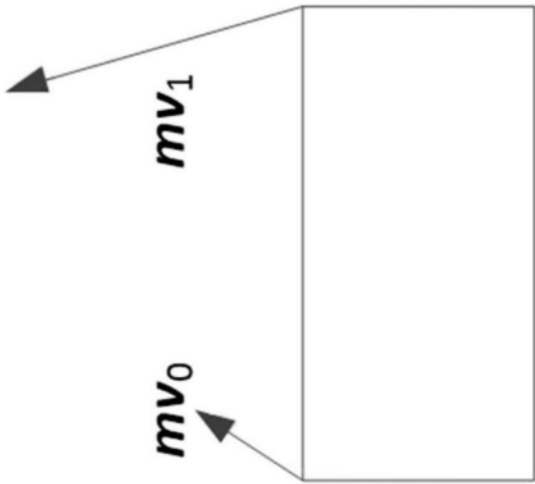


图17A

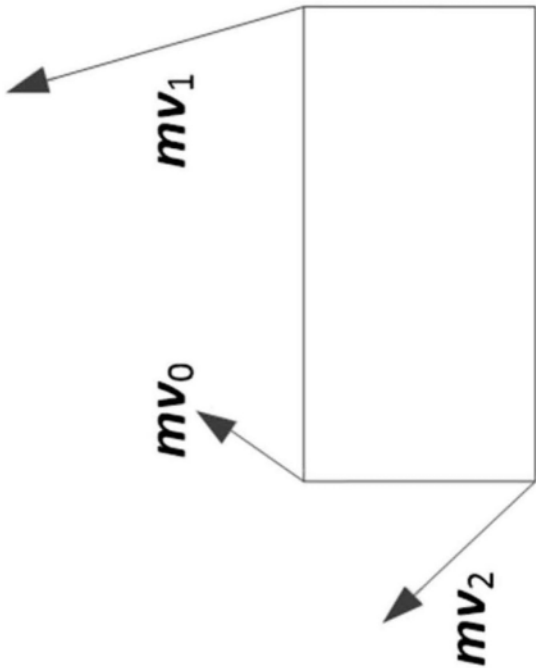


图17B

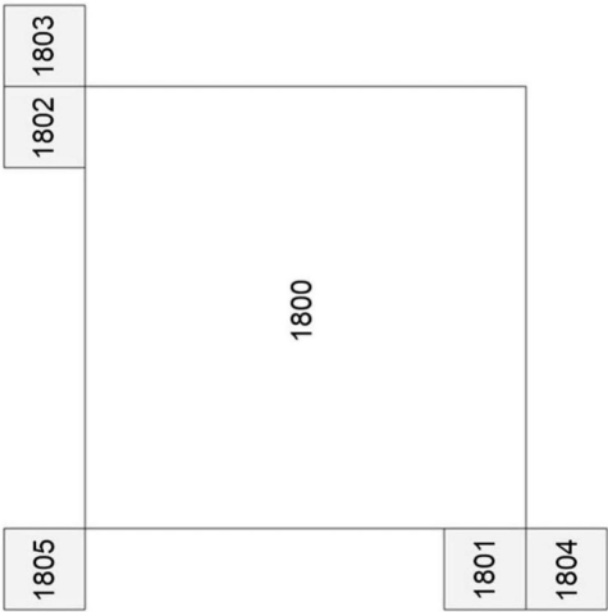


图18A

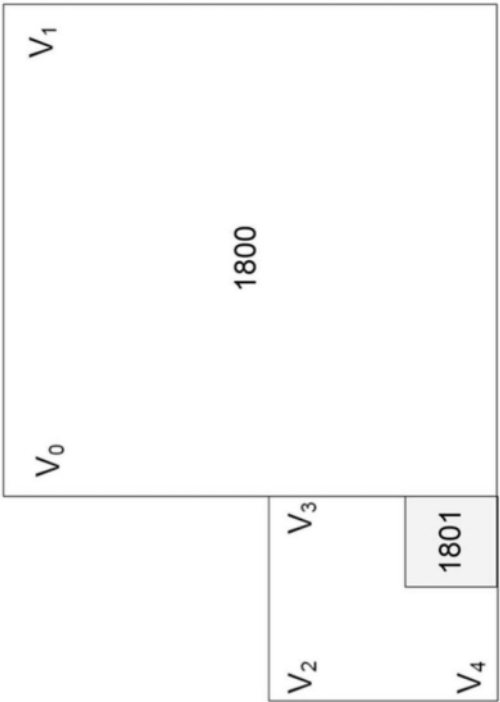


图18B

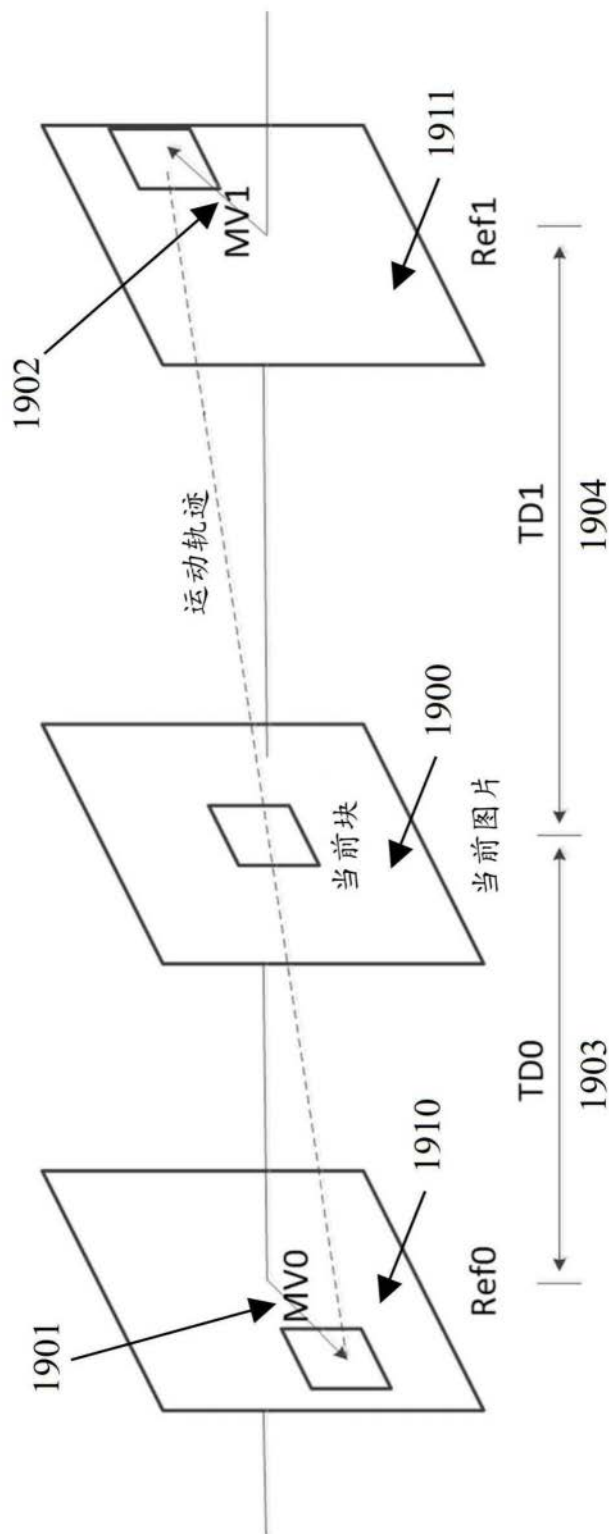


图19

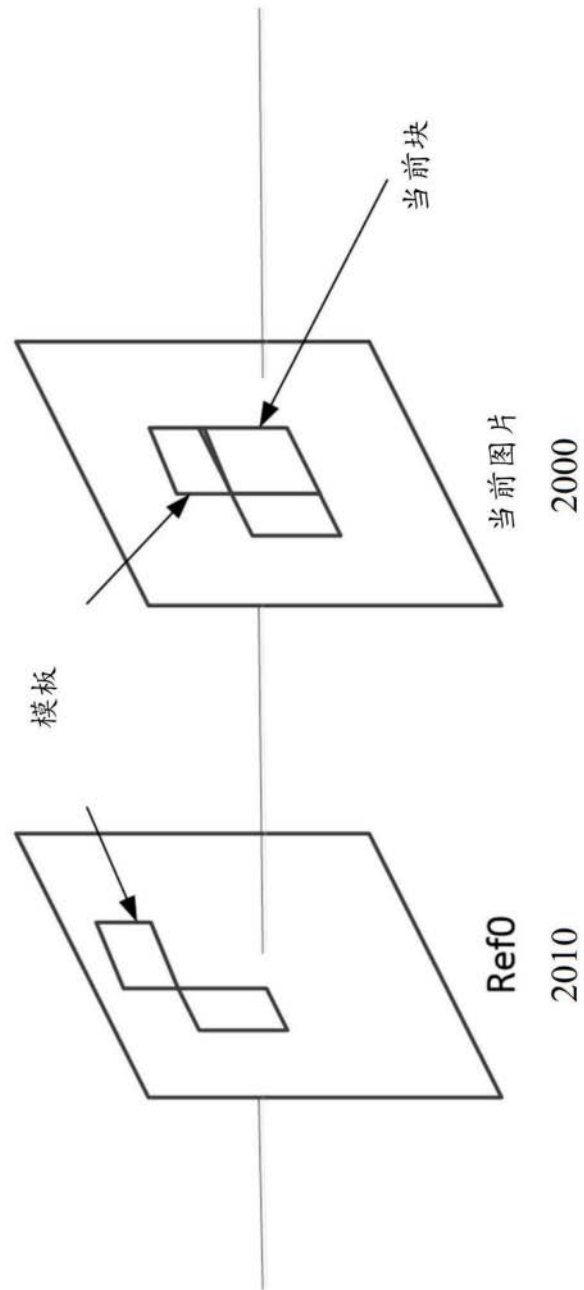


图20

2100

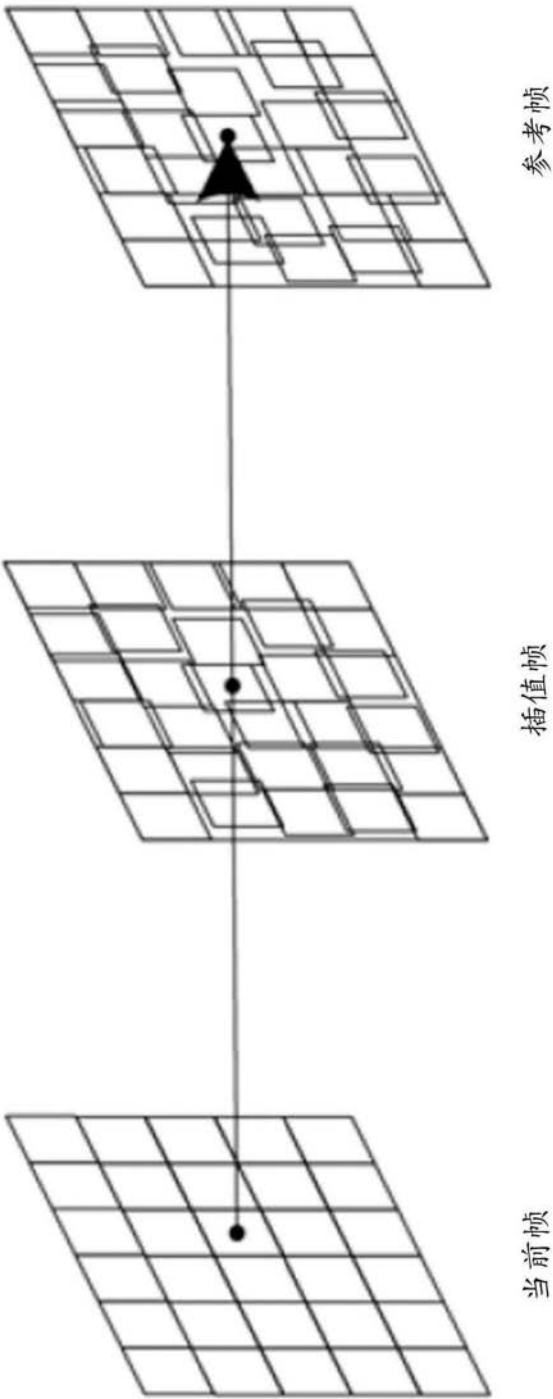


图21

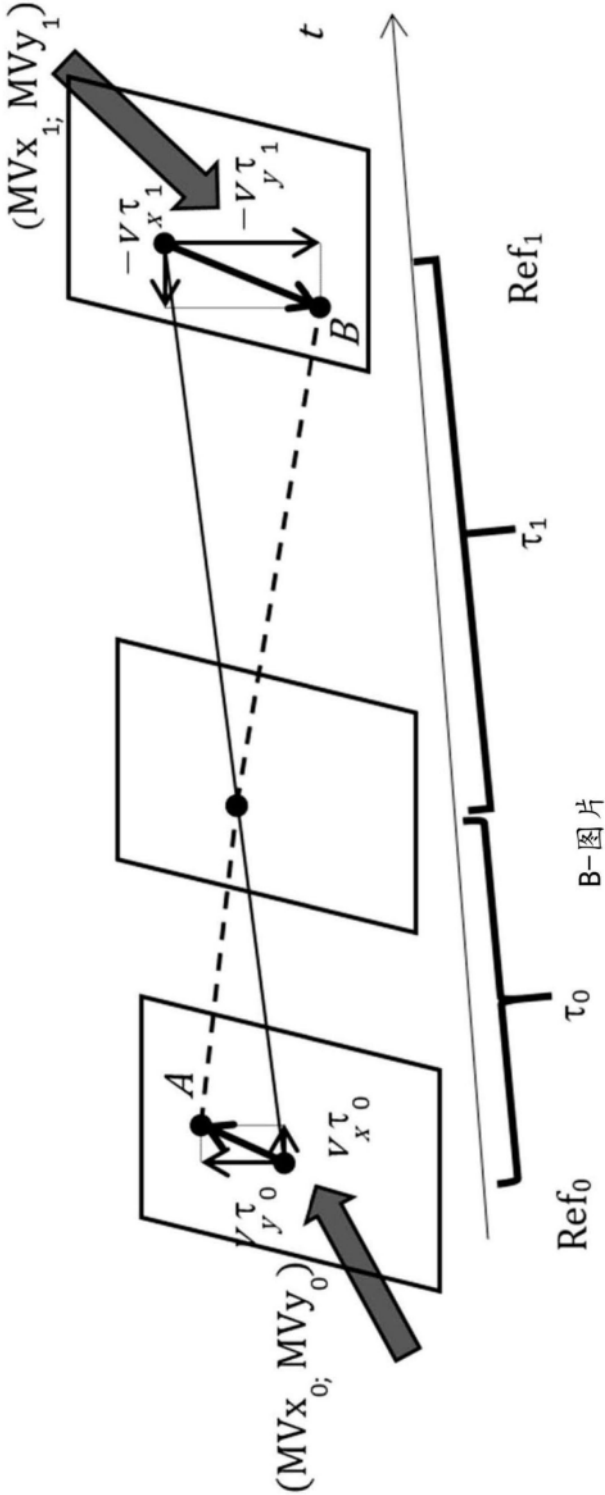


图22

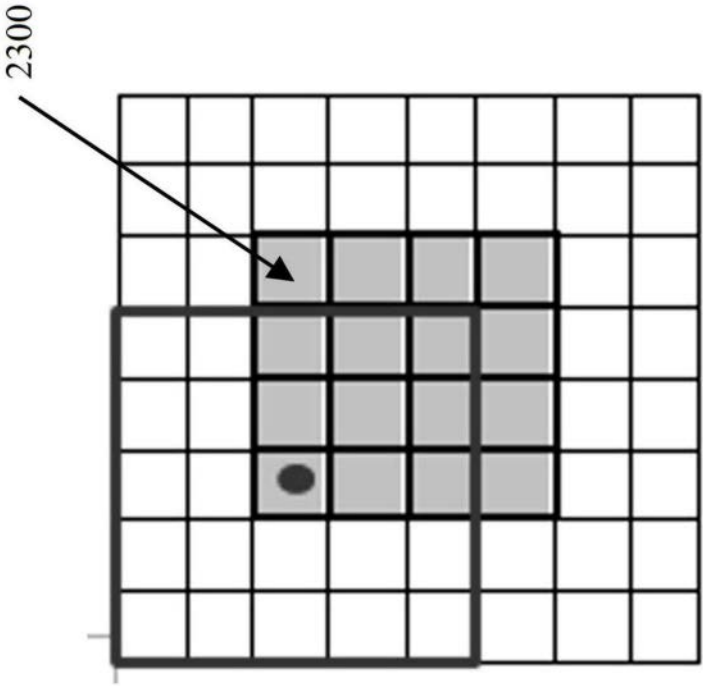


图23A

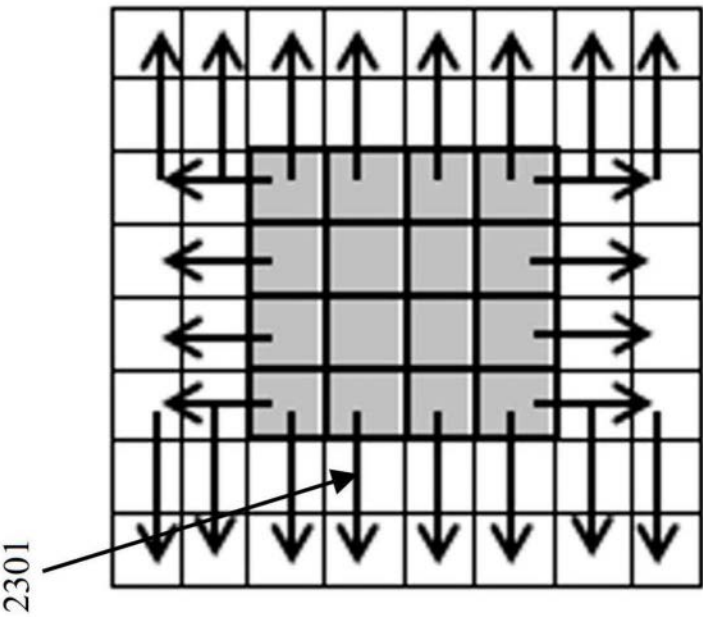


图23B

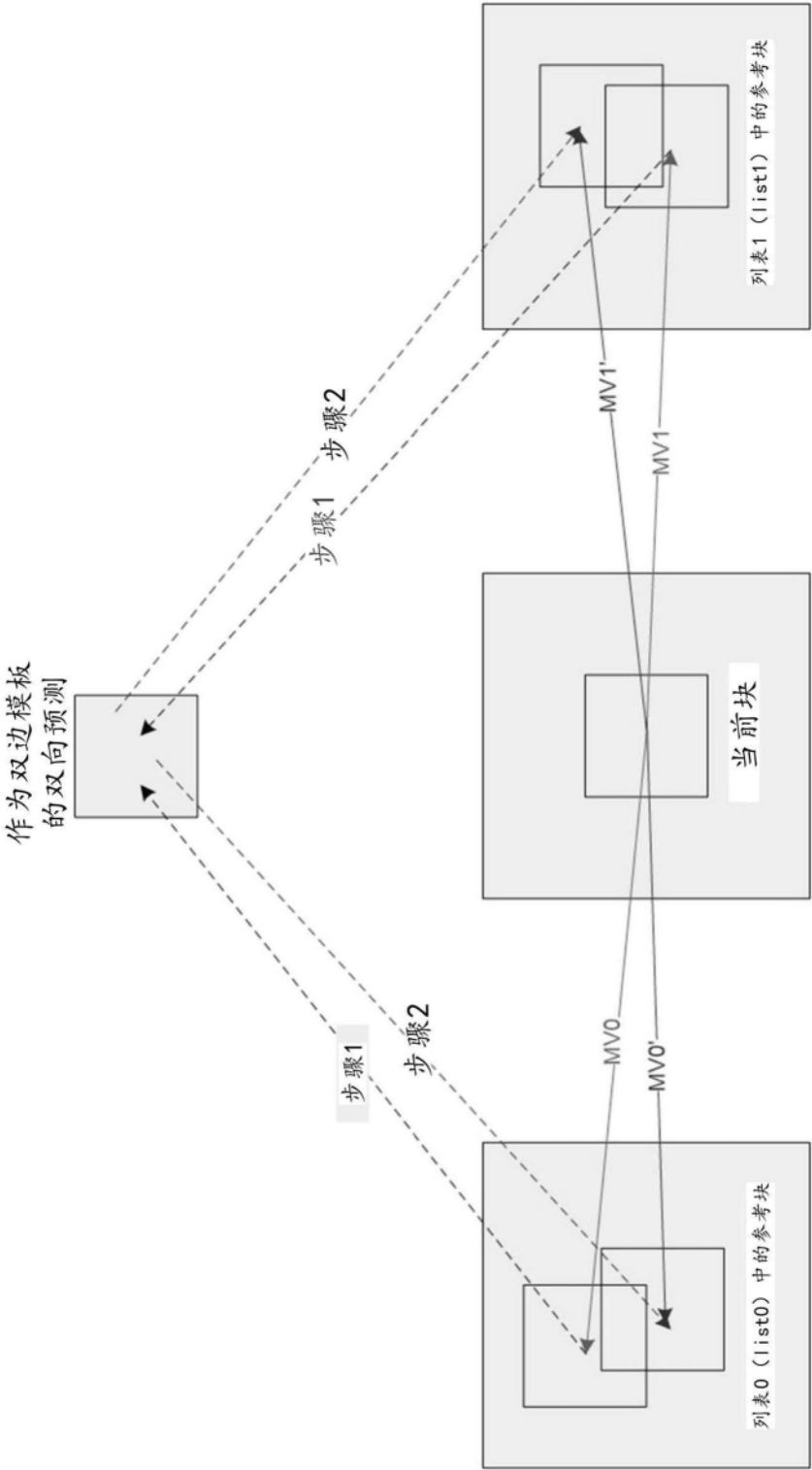


图24

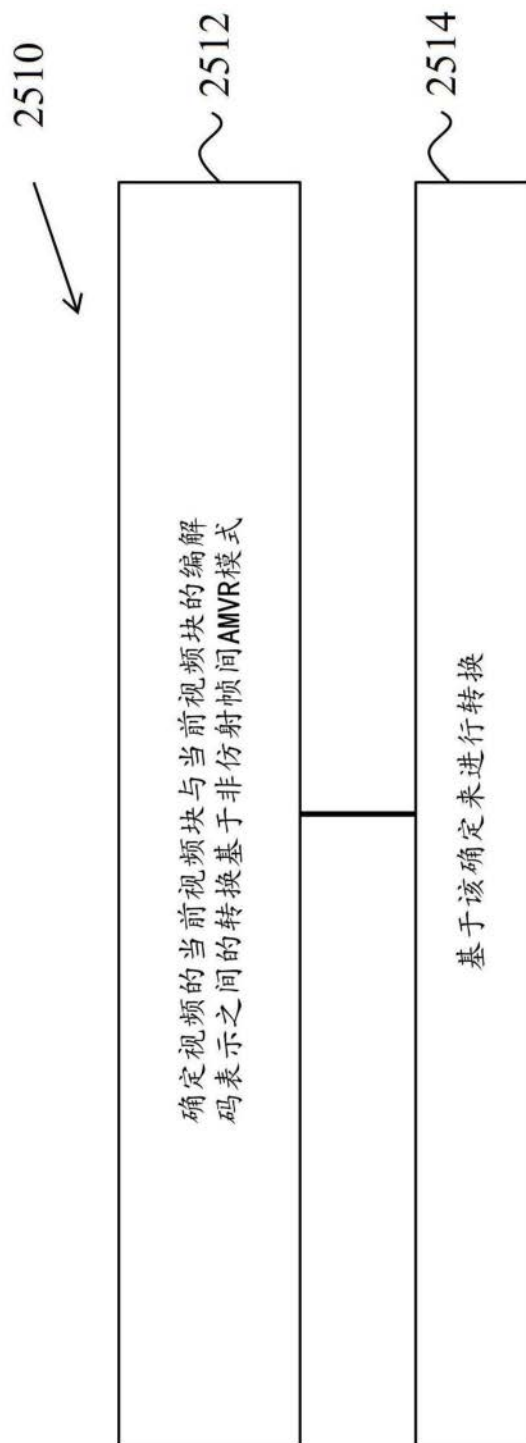


图25A

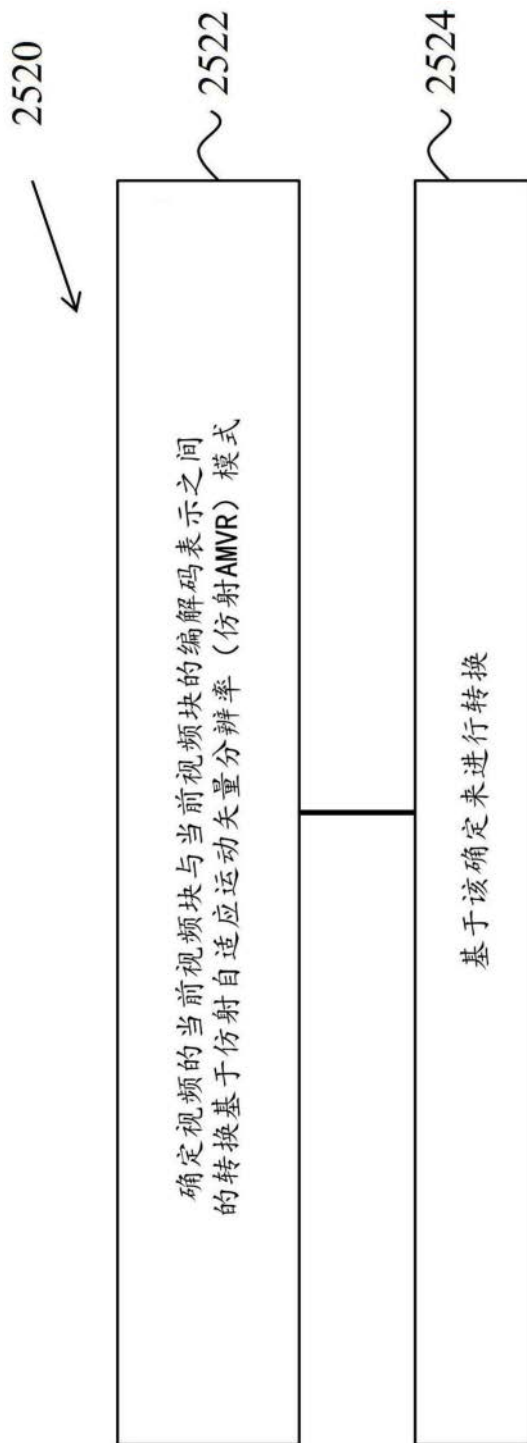


图25B

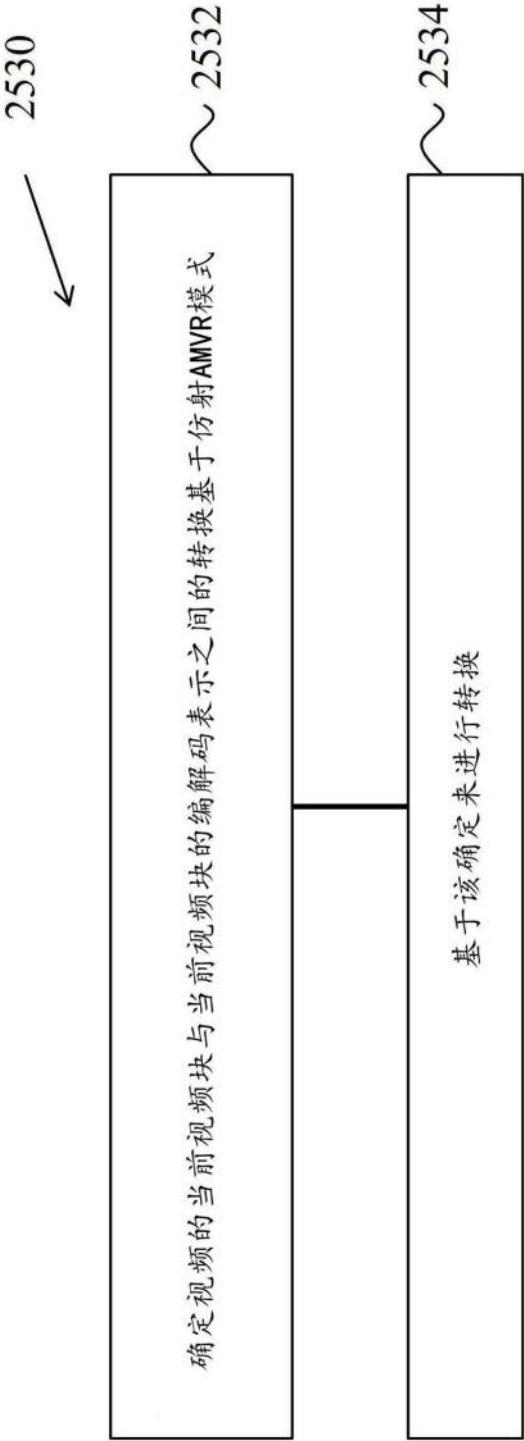


图25C

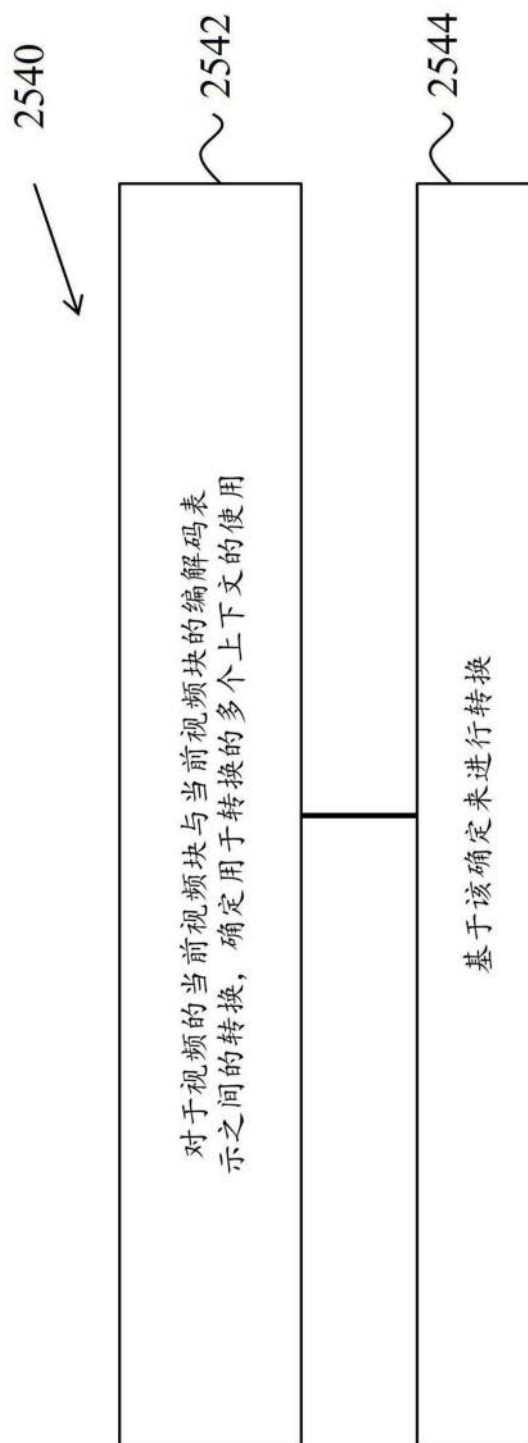


图25D

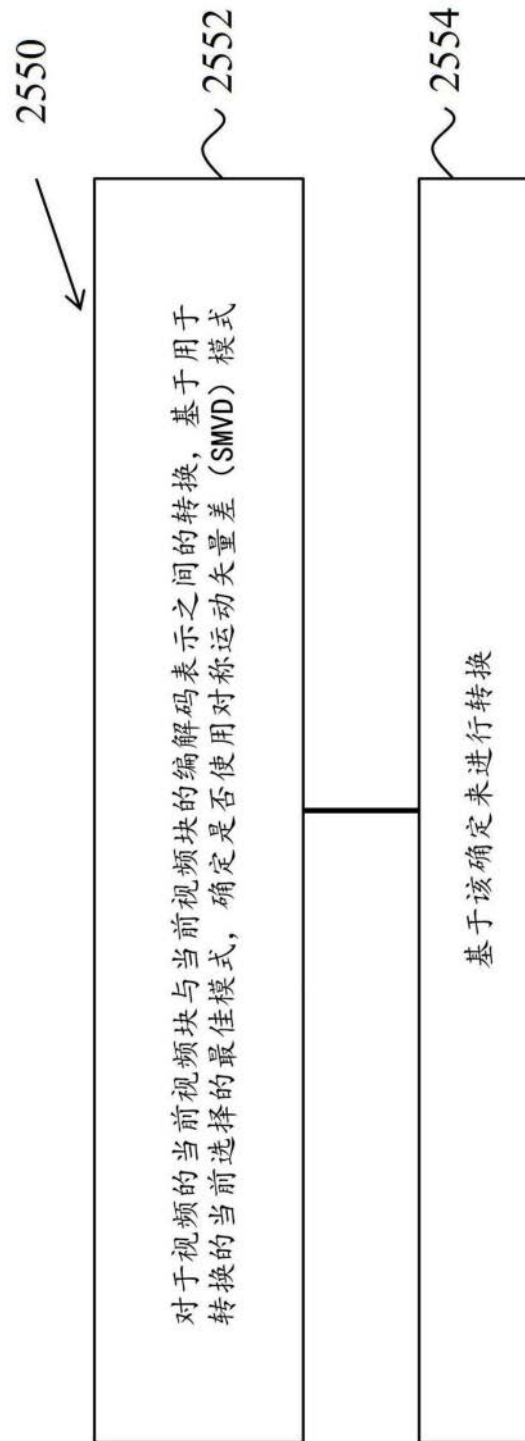


图25E



图25F

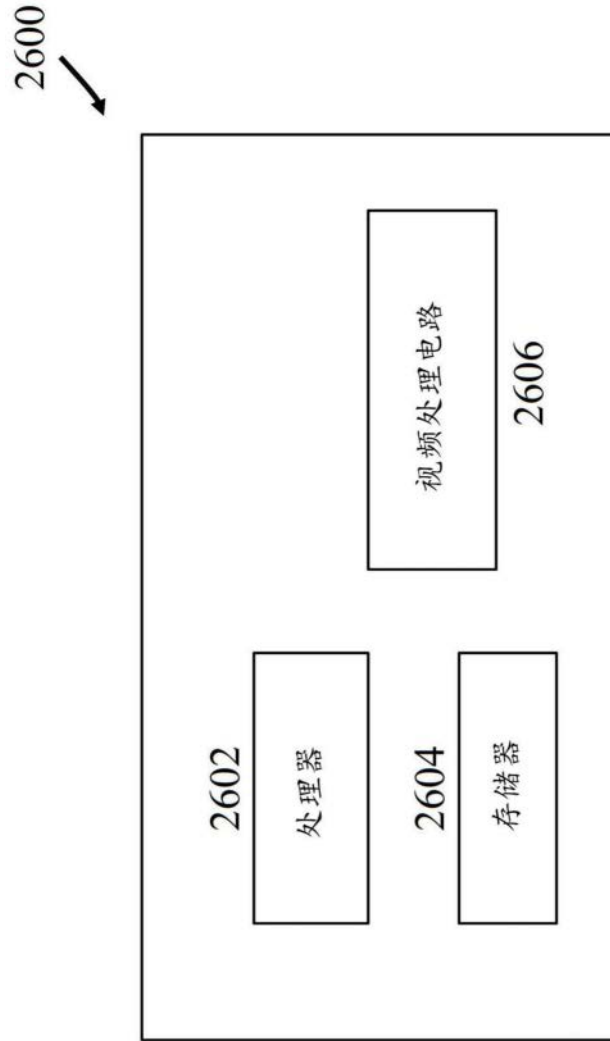


图26

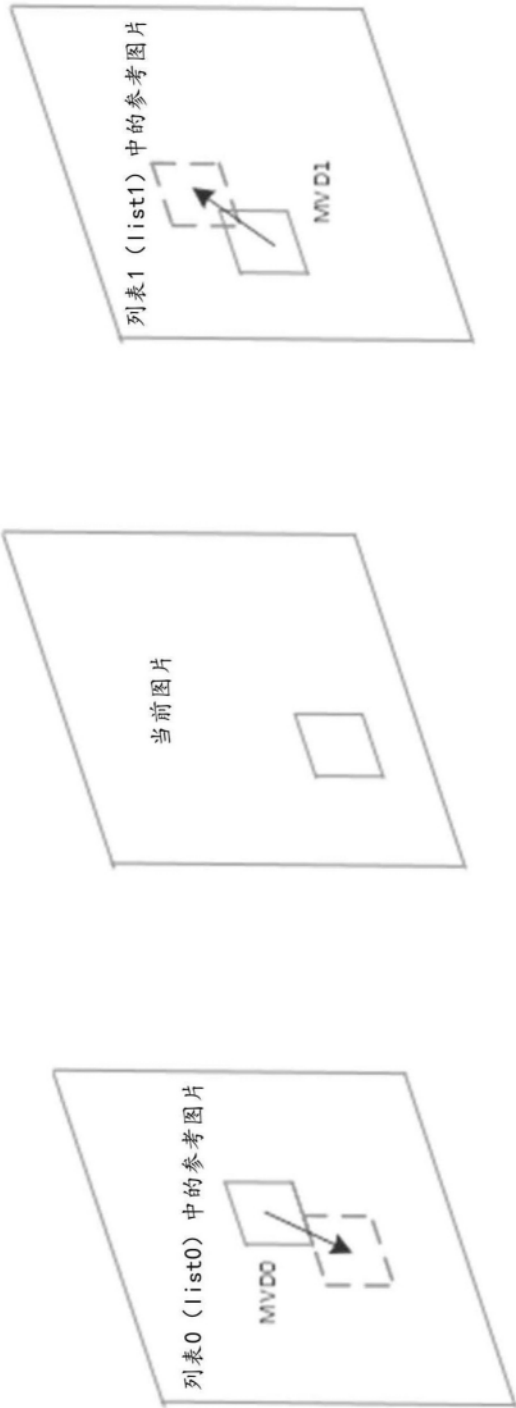


图27

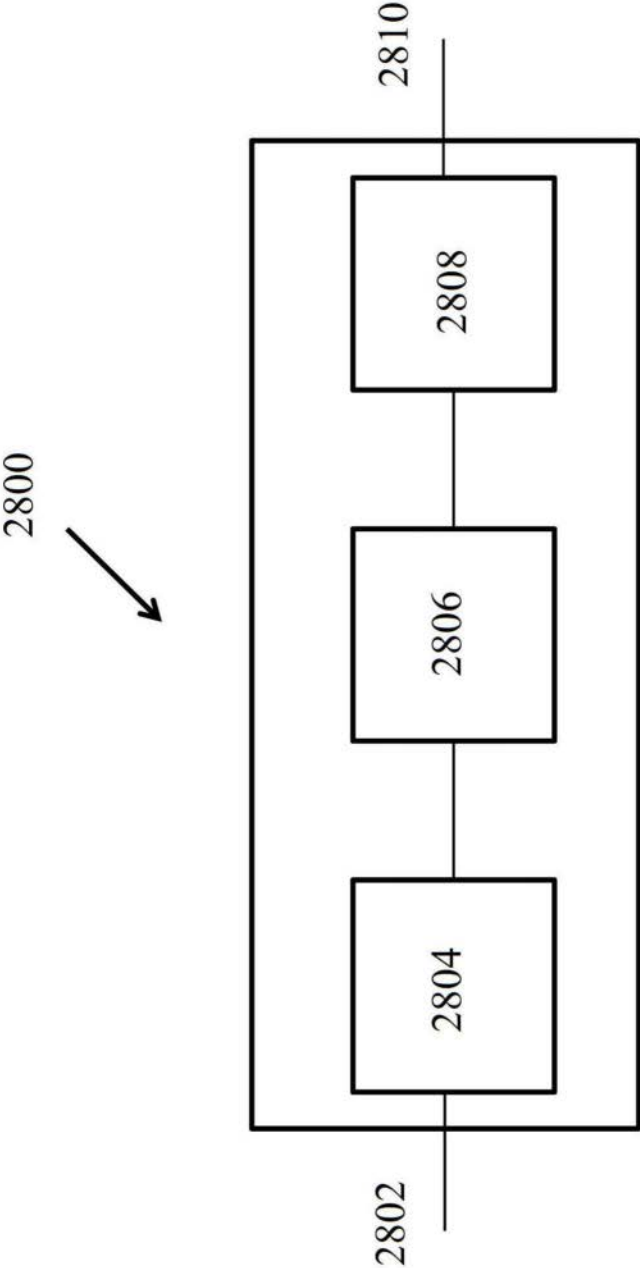


图28