



(12) 发明专利

(10) 授权公告号 CN 102316320 B

(45) 授权公告日 2014. 07. 09

(21) 申请号 201110159665. 3

H04N 19/132(2014. 01)

(22) 申请日 2002. 12. 16

H04N 19/82(2014. 01)

(30) 优先权数据

H04N 19/61(2014. 01)

60/341, 674 2001. 12. 17 US

H04N 19/593(2014. 01)

60/377, 712 2002. 05. 03 US

H04N 19/46(2014. 01)

(62) 分案原申请数据

(56) 对比文件

02825191. 1 2002. 12. 16

CN 1201577 C, 2005. 05. 11,

US 5552832 A, 1996. 09. 03,

(73) 专利权人 微软公司

审查员 闫晓宁

地址 美国华盛顿州

(72) 发明人 S·斯里尼瓦杉 P·苏

(74) 专利代理机构 上海专利商标事务所有限公

司 31100

代理人 蔡悦

(51) Int. Cl.

H04N 19/523(2014. 01)

H04N 19/196(2014. 01)

H04N 19/146(2014. 01)

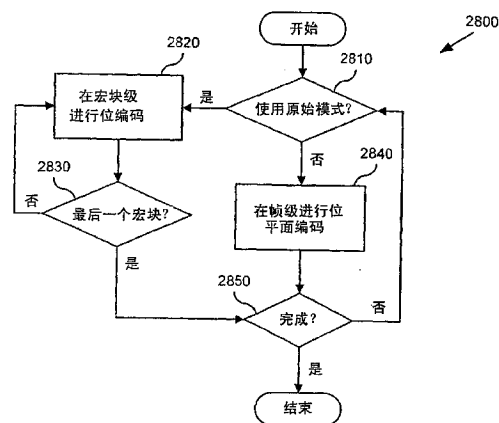
权利要求书3页 说明书19页 附图18页

(54) 发明名称

处理视频图像的方法

(57) 摘要

描述了各种用于对二进制信息（如跳过宏块信息）进行编码和解码（如在视频编码器/解码器中）的技术和工具。在一些实施例中，二进制信息在位平面上排列，并且该位平面在图像/帧层上编码。编码器和解码器处理二进制信息，并且在一些实施例中，切换编码模式。例如，编码器和解码器使用普通、行跳过、列跳过或差分模式，或其它和/或另外的模式。在一些实施例中，编码器和解码器将跳过宏块定义为其运动等于其因果预测的运动并且具有零剩余误差的预测宏块。在一些实施例中，编码器和解码器使用原始编码模式来允许低等待时间应用。



1. 在计算机系统中,一种处理一个或多个视频图像的计算机实现的方法,其特征在于,所述方法包括:

从一组多个可用的编码模式中选择一编码模式;以及

对于一个或多个视频图像中 P 图像的多个预测宏块根据所选择的编码模式处理跳过信息,其中跳过信息指示 P 图像的多个预测宏块是被跳过还是不被跳过,并且如果满足以下条件,P 图像的多个预测宏块中的一个预测宏块被跳过:

所述预测宏块使用基于 P 图像中一个或多个其它宏块的运动的预测运动;以及
所述预测宏块没有剩余信息。

2. 在实现一种视频解码器的计算机系统中,一种对包含多个预测宏块的视频图像进行解码的方法,其中所述多个预测宏块的每个块根据不多于一个参考视频图像来预测,其特征在于,所述方法包括:

从比特流中接收编码数据;以及

对视频图像进行解码,解码包括:

从多个可用编码模式中选择一种编码模式;

对于多个预测宏块中的一个或多个跳过的宏块,对跳过的宏块信息进行解码,所述跳过的宏块信息已根据从多个可用编码模式中所选择的编码模式进行了编码,其中所述跳过的宏块信息指示跳过/非跳过状态;以及

处理所述一个或多个跳过的宏块,其中所述一个或多个跳过的宏块中的每一个被跳过是由于:

使用基于所述跳过的宏块周围的一个或多个其它的预测宏块的运动的预测运动;以及
缺少剩余信息。

3. 如权利要求 2 所述的方法,其特征在于,所述处理所述一个或多个跳过的宏块包括:对于视频图像的当前宏块,使用与从所述一个或多个其它的预测宏块中获取的预测器运动矢量相等的运动矢量来重建当前宏块,并且其中所述一个或多个其它的预测宏块与所述视频图像中的当前宏块相邻。

4. 如权利要求 2 所述的方法,其特征在于,所述比特流包括多个分级层,其中所述多个分级层至少包括图像层和宏块层,并且其中所述跳过的宏块信息在所述图像层传输信号。

5. 在实现一种视频编码器的计算机系统中,一种对包含多个预测宏块的视频图像进行编码的方法,其中所述多个预测宏块的每个块根据不多于一个参考视频图像来预测,其特征在于,所述方法包括:

对视频图像进行编码以生成编码数据,编码包括:

从多个可用编码模式中选择一种编码模式;

处理多个预测宏块中的一个或多个跳过的宏块,其中所述一个或多个跳过的宏块中的每一个被跳过是由于:

使用基于所述跳过的宏块周围的一个或多个其它的预测宏块的运动的预测运动;以及
缺少剩余信息;

对一个或多个跳过的宏块的跳过的宏块信息进行编码,其中所述跳过的宏块信息指示跳过/非跳过状态,并且其中所述编码包括根据从所述多个可用编码模式中选择的一种编码模式来对所述跳过的宏块信息进行编码;以及在比特流中输出编码数据。

6. 如权利要求 5 所述的方法,其特征在于,所述处理所述一个或多个跳过的宏块包括:对于视频图像的当前宏块,使用与从所述一个或多个其它的预测宏块中获取的预测器运动矢量相等的运动矢量来预测当前宏块,并且其中所述一个或多个其它的预测宏块与所述视频图像中的当前宏块相邻。

7. 如权利要求 6 所述的方法,其特征在于,所述处理所述一个或多个跳过的宏块包括:确定所述跳过的宏块的预测的运动是否等于从与所述跳过的宏块相邻一个或多个其它的预测宏块的运动矢量中获取的预测器运动矢量;以及确定所述跳过的宏块缺少剩余信息。

8. 如权利要求 5 所述的方法,其特征在于,所述比特流包括多个分级层,其中所述多个分级层至少包括图像层和宏块层,并且其中所述跳过的宏块信息在所述图像层传输信号。

9. 一种对包含多个预测宏块的视频图像进行解码的系统,其中所述多个预测宏块的每个块根据不多于一个参考视频图像来预测,其特征在于,所述系统包括:

用于从比特流中接收编码数据的装置;以及

用于对视频图像进行解码的装置,解码包括:

从多个可用编码模式中选择一种编码模式;

对于多个预测宏块中的一个或多个跳过的宏块,对跳过的宏块信息进行解码,所述跳过的宏块信息已根据从多个可用编码模式中所选择的编码模式进行了编码,其中所述跳过的宏块信息指示跳过/非跳过状态;以及

处理所述一个或多个跳过的宏块,其中所述一个或多个跳过的宏块中的每一个被跳过是由于:

使用基于所述跳过的宏块周围的一个或多个其它的预测宏块的运动的预测运动,以及缺少剩余信息。

10. 如权利要求 9 所述的系统,其特征在于,所述处理所述一个或多个跳过的宏块包括:对于视频图像的当前宏块,使用与从所述一个或多个其它的预测宏块中获取的预测器运动矢量相等的运动矢量来重建当前宏块,并且其中所述一个或多个其它的预测宏块与所述视频图像中的当前宏块相邻。

11. 如权利要求 9 所述的系统,其特征在于,所述比特流包括多个分级层,其中所述多个分级层至少包括图像层和宏块层,并且其中所述跳过的宏块信息在所述图像层传输信号。

12. 一种对包含多个预测宏块的视频图像进行编码的系统,其中所述多个预测宏块的每个块根据不多于一个参考视频图像来预测,其特征在于,所述系统包括:用于对视频图像进行编码以生成编码数据的装置,编码包括:

从多个可用编码模式中选择一种编码模式;

处理多个预测宏块中的一个或多个跳过的宏块,其中所述一个或多个跳过的宏块中的每一个被跳过是由于:

使用基于所述跳过的宏块周围的一个或多个其它的预测宏块的运动的预测运动,以及缺少剩余信息;

对一个或多个跳过的宏块的跳过的宏块信息进行编码,其中所述跳过的宏块信息指示跳过/非跳过状态,并且其中所述编码包括根据从所述多个可用编码模式中选择

式来对所述跳过的宏块信息进行编码；以及

用于在比特流中输出编码数据的装置。

13. 如权利要求 12 所述的系统,其特征在于,所述处理所述一个或多个跳过的宏块包括:对于视频图像的当前宏块,使用与从所述一个或多个其它的预测宏块中获取的预测器运动矢量相等的运动矢量来预测当前宏块,并且其中所述一个或多个其它的预测宏块与所述视频图像中的当前宏块相邻。

14. 如权利要求 13 所述的系统,其特征在于,所述处理所述一个或多个跳过的宏块包括:

确定所述跳过的宏块的预测的运动是否等于从与所述跳过的宏块相邻一个或多个其它的预测宏块的运动矢量中获取的预测器运动矢量;以及

确定所述跳过的宏块缺少剩余信息。

15. 如权利要求 12 所述的系统,其特征在于,所述比特流包括多个分级层,其中所述多个分级层至少包括图像层和宏块层,并且其中所述跳过的宏块信息在所述图像层传输信号。

处理视频图像的方法

[0001] 本申请是 2002 年 12 月 16 日提交的、申请号为 02825191.1、发明名称为《跳过宏块编码》的发明专利申请的分案（该分案于 2008 年 11 月 20 日提交的，申请号为 200810176684.5、发明名称为《处理视频图像的方法》）的分案申请。

技术领域

[0002] 描述了用于在视频编码 / 解码应用中对二进制信息进行编码 / 解码的技术和工具。例如，视频编码器对跳过宏块信息进行编码。

背景技术

[0003] 数字视频消耗大量的存储和传输容量。典型的原始数字视频序列每秒包括 15 或 30 帧。每一帧能够包括几万或几十万个像素（也称为 pel）。每一像素表示图像的一个微小元素。在原始的形式中，计算机通常用 24 比特来表示一个像素。由此，典型的原始数字视频序列的每秒的比特数，或比特率可以是 500 万比特 / 秒或更高。

[0004] 大多数计算机和计算机网络缺乏处理原始数字视频的资源。鉴于此原因，工程师使用压缩（也称为译码或编码）来降低数字视频的比特率。在视频的质量不受损害但比特率的降低受视频的复杂性限制的情况下，压缩是无损的。或者，在视频的质量受损害但比特率的降低更显著的情况下，压缩是有损的。解压缩倒转了压缩。

[0005] 一般来说，视频压缩技术包括帧内压缩和帧间压缩。帧内压缩技术压缩单个的帧，通常称为 I 帧或主帧。帧间压缩技术在压缩帧时参考前帧和 / 或后帧，通常称为预测帧、P 帧或 B 帧。

[0006] 微软公司的 Windows Media Video, 版本 7 [“WMV7”] 包括视频编码器和视频解码器。WMV7 编码器使用帧内和帧间压缩，WMV7 解码器使用帧内和帧间解压。

[0007] A. WMV7 的帧内压缩

[0008] 图 1 说明了 WMV7 编码器中对主帧中像素的块 (105) 的基于块的帧内压缩 (100)。块是一组像素，例如，像素的 8×8 排列。WMV7 编码器将主视频帧拆分为 8×8 的像素块，并对各单个块，如块 (105) 应用 8×8 的离散余弦变换 [“DCT”] (110)。DCT 是一种频率变换，将 8×8 的像素（空间信息）块转换为 8×8 的 DCT 系数块 (115)，即频率信息。DCT 操作本身是无损或接近无损的。然而，与原始像素值相比较，DCT 系数对编码器来说能更有效地用来压缩，因为大多数重要信息集中在低频系数（常规地位于块 (115) 的左上角）中，并且许多高频系数（常规地位于块 (115) 的右下角）的值为零或接近零。

[0009] 编码器然后将 DCT 系数量化 (120)，得到一个 8×8 的量化 DCT 系数 (125) 块。例如，编码器对每一系数应用统一标量量化步长，类似于将每一系数除以同一值并舍入成整数。例如，如果一个 DCT 系数值是 163 并且步长为 10，则量化 DCT 系数值为 16。量化是有损的。重建的 DCT 系数值将是 160，而非 163。由于低频 DCT 系数往往具有较大的值，量化会导致精度损耗，但不会完全丢失系数的信息。另一方面，由于高频 DCT 系数的值往往为零或接近零，对高频系数的量化通常导致零值的邻接区域。此外，在某些情况下，高频 DCT 系

数比低频 DCT 系数更粗糙地量化,导致高频 DCT 系数的精度 / 信息的大量损耗。

[0010] 编码器然后准备 8×8 的量化 DCT 系数 (125) 块来进行熵编码,这是一种无损压缩的形式。确切的熵编码类型根据系数是否是 DC 系数 (最低频率)、顶行或左列中的 AC 系数 (其它频率) 或另一 AC 系数而不同。

[0011] 编码器将 DC 系数 (126) 编码为对相邻的 8×8 块的 DC 系数的差分,该相邻块是正在编码的块的相邻的 (如,上方或左边) 先前已编码的块。(图 1 示出了帧内相邻块 (135) 位于正在编码的块的左边。) 编码器对该差分进行熵编码 (140)。

[0012] 熵编码器能够将 AC 系数的左列或顶行编码为对相邻的 8×8 的块的相应列或行的差分。图 1 示出将 AC 系数的左列 (127) 编码为对相邻 (左边) 块 135 的左列 (137) 的差分。差分编码提高了差分系数具有零值的机率。剩余的 AC 系数来自量化 DCT 系数块 (125)。

[0013] 编码器将 8×8 的预测的量化 AC DCT 系数块 (145) 扫描 (150) 为一维数组 (155), 然后使用行程编码 (160) 的一种变异来对扫描的 AC 系数进行熵编码。编码器从一个或多个游程 / 级 / 最后 (run/level/last) 表 (165) 中选择熵码并输出该熵码。

[0014] 主帧对比特率比预测帧起到更大的作用。在低或者中比特率应用中,主帧经常是性能的关键瓶颈,因此主帧的有效压缩是关键。

[0015] 图 2 说明了如图 1 所示的帧内压缩的缺点。特别地,对主帧的块之间的冗余的充分利用被限制在对来自块 (210) 的左边邻块 (220) 或上方邻块 (230) 的频率系数的子集 (例如,DC 系数和 AC 系数的左列 (或顶行)) 的预测上。DC 系数表示块的平均值,AC 系数的左列表示块的行平均值,顶行表示列平均值。事实上,如 WMV7 中的对 DC 和 AC 系数的预测限制了向左面 (或上方) 邻块的行范围 (或列范围) 平均信号的外插。对于左块 (220) 的一个特定的行 (221),左块 (220) 的左 DCT 系数列中的 AC 系数用来预测块 (210) 的整个相应行 (211)。

[0016] B. WMV7 中的帧间压缩

[0017] WMV7 编码器中的帧间压缩使用基于块的运动补偿预测编码,随后为剩余误差的变换编码。图 3 和 4 说明了 WMV7 编码器中对预测帧的基于块的帧间压缩。特别地,图 3 说明了对预测帧 (310) 的运动估计,图 4 说明了对预测帧的已估计运动的块的预测剩余的压缩。

[0018] WMV7 编码器将一个预测帧拆分为 8×8 的像素块。一组 4 个 8×8 的块组成宏块。对每一宏块,执行运动估计进程。运动估计近似与参考帧,如先前被编码的预测帧相关的像素宏块的运动。在图 3 中,WMV7 编码器计算预测帧 (310) 中的宏块 (315) 的运动矢量。为计算该运动矢量,编码器在参考帧 (330) 的搜索范围 (335) 中进行搜索。在搜索范围 (335) 内,编码器将来自预测帧 (310) 的宏块 (315) 与不同候选宏块进行比较,来找出较佳匹配的候选宏块。编码器能够在搜索范围 (335) 中对候选块每一像素或每 $1/2$ 像素进行检查,取决于编码器的所期望的运动估计分辨率。其它视频编码器以其它增量进行检查,如每 $1/4$ 像素。对于一个候选宏块,编码器检查预测帧 (310) 的宏块 (315) 和候选宏块之间的差异以及对该宏块进行运动矢量编码的成本。当编码器找到较佳匹配的宏块之后,块匹配进程结束。编码器输出匹配宏块的运动矢量 (已被熵编码),使得解码器能够在解码过程中找到匹配的宏块。当对预测帧 (310) 进行解码时,解码器使用运动矢量,利用来自参考帧 (330) 的信息来计算宏块 (315) 的预测宏块。对宏块 (315) 的预测很少是理想的,因此编码器通常对预测宏块和宏块 (315) 其本身之间 8×8 的像素差异块 (也称为误差或剩余块) 来进

行编码。

[0019] 图 4 说明了 WMV7 编码器中对已进行运动估计的块进行误差块 (435) 的计算和编码。误差块 (435) 是预测块 (415) 和原始当前块 (425) 之间的差异。编码器对误差块 (435) 应用 DCT (440), 得到 8×8 的系数块 (445)。与像素值的 DCT 系数的情况相比, 误差块 (435) 的重要信息更显著得多地集中在低频系数 (常规地位于块 (445) 的左上角) 中, 许多高频系数的值为零或接近零 (常规地位于块 (445) 的右下角)。

[0020] 编码器然后量化 (450) DCT 系数, 得到 8×8 的量化 DCT 系数块 (455)。量化步长是可调节的。再一次, 由于低频 DCT 系数往往具有较大的值, 量化导致精度的损耗, 但未完全损耗系数的信息。另一方面, 由于高频 DCT 系数的值往往为零或接近零, 高频系数的量化导致零值的邻接区域。另外, 在某些情况下, 高频 DCT 系数比低频 DCT 系数更粗糙地量化, 导致高频 DCT 系数的精度 / 信息的更大损耗。

[0021] 编码器然后准备 8×8 的量化 DCT 系数块 (455) 用于熵编码。编码器将 8×8 的块 (455) 扫描 (460) 为具有 64 个元素的一维数组 (465), 使得系数一般从低频到高频地排列, 这通常令零值位于最后。

[0022] 编码器使用行程编码 (470) 的一个变异对扫描的系数进行熵编码。编码器从一个或多个游程 / 级 / 最后表 (475) 中选择熵码并输出该熵码。

[0023] 当宏块的运动矢量为零 (即没有运动), 并且对该宏块没有发送剩余块信息, 编码器对该宏块使用 1 比特的跳过宏块标志。对多种视频内容 (如, 低运动和 / 或低比特率视频) 来说, 这通过避免运动矢量和剩余块信息的发送降低了比特率。编码器将宏块的跳过宏块标志与该宏块的其它信息一起放置在输出比特流的宏块层。

[0024] 图 5 示出了帧间编码块的解码进程 (500)。由于 DCT 系数的量化, 重建的块 (575) 与相应的原始块不相同。该压缩是有损的。

[0025] 在图 5 的概述中, 解码器使用可变长度解码和一个或多个游程 / 级 / 最后表 (515) 对已经过熵编码的表示预测剩余的信息进行解码 (510, 520)。解码器将储存已进行熵解码的信息的一维数组 (525) 逆扫描为二维块 (535)。解码器对数据进行反量化和反离散余弦变换 (同时进行, 540), 得到重建的误差块 (545)。在一单独的路径中, 解码器使用对参考帧的位移的运动矢量信息 (555) 来计算预测块 (565)。解码器将预测块 (565) 和重建的误差块 (545) 组合 (570) 来组成重建块 (575)。

[0026] 当解码器接收到宏块的跳过宏块标志时, 解码器跳过对该宏块的预测计算和剩余块信息解码。取而代之的是, 解码器使用参考帧中宏块位置的相应像素数据。

[0027] 原始和重建帧之间的变化量被称为失真, 对帧进行编码所需要的比特数被称为率。失真量一般说来与率成反比。换言之, 用较少的比特来对帧进行编码 (较大压缩) 将导致较大的失真, 反之亦然。视频压缩模式的目的之一是试图改进率 - 失真 - 换言之, 试图采用较少的比特来达到同样的失真 (或使用相同的比特来达到较低的失真)。

[0028] 尽管 WMV7 中使用的跳过宏块标志对多种视频内容来说通常能够降低比特率, 然而在某些情况下却远非最佳。在许多情况下, 未能充分利用从宏块到宏块的跳过宏块标志中的冗余, 例如, 当跳过宏块成群地出现在图像中时。同样, WMV7 在跳过宏块时忽略了对预测帧中宏块的运动预测, 在某些情况下会损伤预测帧的压缩的效率。

[0029] C. 视频压缩和解压的标准

[0030] 除 WMV7 之外,若干种国际标准与视频压缩和解压相关。这些标准包括运动图像专家组 [“MPEG”]1、2 和 4 标准以及国际电信联盟(“ITU”)的 H. 261、H. 262 和 H. 263 标准。与 WMV7 类似,这些标准使用帧内和帧间压缩的组合,尽管这些标准通常在所使用的压缩技术细节上与 WMV7 不同。

[0031] 一些国际标准认可将宏块的跳过编码作为一种工具在视频压缩和解压中使用。欲知这些标准中有关跳过宏块编码的另外的细节,参见该标准的说明书本身。

[0032] 上述标准中的跳过宏块编码通常降低了多种视频内容的比特率,但是在某些情况下远非最佳。在许多情况下,未能充分利用从宏块到宏块的跳过宏块标志中的冗余,例如,当跳过宏块成群地出现在图像中时。同样,当跳过宏块时,它忽略了对预测宏块 / 图像中的宏块的运动预测,从而在某些情况下会损伤预测宏块 / 图像的压缩效率。

[0033] 鉴于对数字视频的视频压缩和解压的关键重要性,视频压缩和解压是充分开发的领域并不令人惊讶。然而,不论先前的视频压缩和解压技术的好处如何,它们都没有以下技术和工具的优点。

[0034] 发明概述

[0035] 总的来说,详细描述针对用于对二进制信息进行编码和解码(如,在视频编码器 / 解码器中)的各类技术和工具。二进制信息可能包括指示视频编码器或解码器在视频帧中是否跳过了某一宏块的位。或者,二进制信息可能包括指示对宏块的运动矢量分辨率(如 1-MV 或 4-MV)、隔行扫描方式(如半帧或帧)或一些其它信息的位。二进制信息可能在逐帧的基础上或者在一些其它基础上编码。

[0036] 在一些实施例中,二进制信息在位平面上排列。例如,位平面在图像 / 帧层被编码。可选地,二进制信息以其它方法排列和 / 或在不同的层编码。编码器和解码器处理二进制信息。二进制信息可能包括宏块级信息。可选地,编码器和解码器处理块级、子块级或像素级信息的位平面。

[0037] 在一些实施例中,编码器和解码器切换编码模式。例如,编码器和解码器使用普通、行跳过或列跳过模式。不同的模式使得编码器和解码器可以充分利用二进制信息中的冗余。可选地,编码器和解码器使用其它和 / 或另外的模式,如差分模式。为提高效率,编码器和解码器可能在某些模式中使用位平面倒置技术。

[0038] 在一些实施例中,编码器和解码器将跳过宏块定义为运动等于其因果预测运动并具有零剩余误差的预测宏块。可选地,编码器和解码器将跳过宏块定义为具有零运动和零剩余误差的预测宏块。

[0039] 在一些实施例中,编码器和解码器使用原始编码模式来允许低等待时间应用。例如,在原始编码模式中,已编码的宏块能够立即发送到解码器,而不需要等到帧 / 图像中的所有宏块都被编码。编码器和解码器能够在原始编码模式和其它模式之间切换。

[0040] 各类技术和工具可以组合使用或单独使用。特别地,该应用连同相应的比特流语法一起描述了跳过宏块编码和解码的两种实现。不同的实施例实现了一种或多种所描述的技术和工具。

[0041] 参考附图阅读以下对不同实施例的详细描述,可以更清楚其它特点和优点。

[0042] 附图的简要描述

[0043] 图 1 所示是依照现有技术的对一个 8×8 的像素块的基于块的帧内压缩的图。

- [0044] 图 2 所示是依照现有技术的频率系数的预测的图。
- [0045] 图 3 所示是依照现有技术的视频编码器中的运动估计的图。
- [0046] 图 4 所示是依照现有技术的视频编码器中对一个 8×8 的预测剩余块的基于块的帧间压缩的图。
- [0047] 图 5 所示是依照现有技术的对一个 8×8 的预测剩余块的基于块的帧内解压的图。
- [0048] 图 6 是适于在其中实现若干所描述的实施例的计算环境的结构图。
- [0049] 图 7 是若干所描述的实施例中使用的—般化的视频编码器系统的结构图。
- [0050] 图 8 是若干所描述的实施例中使用的—般化的视频解码器系统的结构图。
- [0051] 图 9 所示是依照第一实现的组成 P 图像层的比特流元素的图。
- [0052] 图 10 所示是用于在具有多个跳过宏块编码模式的视频编码器中对跳过宏块的信息进行编码的技术的流程图。
- [0053] 图 11 所示是用于对由具有多个跳过宏块编码模式的视频编码器编码的跳过宏块的信息进行解码的技术的流程图。
- [0054] 图 12 所示是跳过宏块编码帧的一个示例。
- [0055] 图 13 所示是用于在普通跳过宏块编码模式中进行编码的技术的流程图。
- [0056] 图 14 所示是用于在行预测跳过宏块编码模式中进行编码的技术的流程图。
- [0057] 图 15 所示是用于对跳过宏块的信息进行行预测解码的伪代码的代码列表。
- [0058] 图 16 所示是用于在列预测跳过宏块编码模式中进行编码的技术的流程图。
- [0059] 图 17 所示是用于对跳过宏块的信息进行列预测解码的伪代码的代码列表。
- [0060] 图 18 所示是用于确定在视频编码器中是否跳过特定宏块的编码的技术的流程图。
- [0061] 图 19 所示是用于以行跳过编码模式在位平面中对二进制信息进行编码的技术的流程图。
- [0062] 图 20 所示是用于以列跳过编码模式在位平面中对二进制信息进行编码的技术的流程图。
- [0063] 图 21 所示是用于以普通 -2 编码模式在位平面中对二进制信息进行编码的技术的流程图。
- [0064] 图 22、23 和 24 所示是以普通 -6 模式平铺的二进制信息的帧的示例。
- [0065] 图 25 所示是用于以普通 -6 模式在位平面中对二进制信息进行编码的技术的流程图。
- [0066] 图 26 所示是用于以差分编码模式对二进制信息进行编码的技术的流程图。
- [0067] 图 27 所示是用于以差分编码模式对二进制信息进行解码的技术的流程图。
- [0068] 图 28 所示是用于对低等待时间应用以原始编码模式选择性地对二进制信息进行编码的技术的流程图。
- [0069] 详细描述
- [0070] 所描述的实施例与用于对二进制信息进行编码和解码（如，在视频编码器 / 解码器中）的技术和工具相关。二进制信息可能包括指示视频编码器或解码器是否跳过了视频帧中的特定宏块的位。或者，二进制信息可包括指示宏块的运动矢量分辨率（如 1-MV 或 4-MV）、隔行扫描模式（如半帧或帧）或其它信息的位。二进制信息可能在逐帧的基础上或

在一些其它基础上进行编码。

[0071] 在一些实施例中,二进制信息在位平面中排列。位平面在图像/帧层被编码。可选地,二进制信息以一些其它方式排列和/或在不同的层编码。

[0072] 在一些实施例中,编码器和解码器切换编码模式。例如,编码器和解码器使用普通、行跳过或列跳过模式。不同的模式使得编码器和解码器可以充分利用二进制信息中的冗余。可选地,编码器和解码器使用其它和/或另外的模式。

[0073] 在一些实施例中,编码器和解码器将跳过宏块定义为运动等于其因果预测的运动并具有零剩余误差的预测宏块。可选地,编码器和解码器将跳过宏块定义为具有零运动和零剩余误差的预测宏块。

[0074] 在一些实施例中,作为对有效帧/图像级编码的替代,准许使用原始编码模式来允许低等待时间应用。在原始编码模式中,已编码的宏块能够立即发送到解码器,而不需要等待直到帧/图像中的所有宏块都编码完成。

[0075] 在一些实施例中,编码器和解码器处理宏块级信息的位平面。可选地,编码器和解码器处理块、子块或像素级信息的位平面。

[0076] 各类技术和工具可以组合使用或单独使用。具体而言,本申请连同相应的比特流语法一起描述了跳过宏块编码和解码的两种实现。不同的实施例实现一种或多种所描述的技术和工具。

[0077] 在所描述的实施例中,视频编码器和解码器执行各类技术。尽管这些技术的操作通常为演示目的,以特定顺序来描述,应当理解,这一描述方式包含操作顺序中较小的重排列,除非需要特定的顺序。例如,顺序地描述的操作在某些情况下可以重新排列或并发执行。此外,为简化的目的,流程图通常不显示特定技术可以与其它技术结合使用的不同方法。

[0078] 在所描述的实施例中,视频编码器和解码器在比特流中使用不同的标志和信号。尽管描述了特定的标志和信号,应当理解,这一描述方式包含了对标志和信号的不同约定(如0的约定而不是1的约定)。

[0079] I. 计算环境

[0080] 图6说明了适于在其中实现若干所描述的实施例的计算环境(600)的一般化示例。该计算环境(600)并不意味着对使用或功能的范围的限制,这些技术和工具可以在不同的通用或专用计算系统中实现。

[0081] 参考图6,计算环境(600)包括至少一个处理单元(610)和存储器(620)。在图6中,这一最基本的配置(630)包括在虚线框中。处理单元(610)执行计算机可执行指令并可以是真实或虚拟的处理器。在多处理系统中,多处理单元执行计算机可执行指令来提高处理能力。存储器(620)可以是易失存储器(如,寄存器、高速缓存、RAM)、非易失存储器(如,ROM、EEPROM、闪存等等)或两者的某一组合。存储器(620)储存实现编码器或解码器,如视频编码器或解码器的软件(680)。

[0082] 计算环境可以具有其它特性。例如,计算系统(600)包括存储(640)、一个或多个输入设备(650)、一个或多个输出设备(660)以及一个或多个通信连接(670)。互连机制(未示出)如总线、控制器或网络将计算环境(600)的组件互连。通常,操作系统软件(未示出)为其它在计算环境(600)中执行的软件提供了操作环境,并协调计算环境(600)的

组件的活动。

[0083] 存储 (640) 可以是可移动或不可移动的,包括磁盘、磁带或盒式磁带、CD-ROM、DVD 或其它任一可以用来储存信息并在计算环境 (600) 内可访问的媒质。存储 (640) 储存软件 (680) 用来实现编码器或解码器的指令。

[0084] (多个) 输入设备 (650) 可以是触摸输入设备,如键盘、鼠标、笔、或轨迹球、语音输入设备、扫描设备或另一提供向计算环境 (600) 的输入的设备。对音频或视频编码来说,(多个) 输入设备 (650) 可以是声卡、视频卡、电视调谐卡或以模拟或数字形式接受音频或视频输入的类似设备、或将音频或视频样本读入计算环境 (600) 的 CD-ROM 或 CD-RW。(多个) 输出设备 (660) 可以是显示器、打印机、扬声器、CD 记录器或从计算环境 (600) 提供输出的另一设备。

[0085] (多个) 通信连接 (670) 启用通过通信媒质向另一计算实体的通信。通信媒质传送信息如计算机可执行指令、音频或视频输入或输出或其它已调制数据信号中的数据。已调制数据信号是以某一方式设定或改变其一个或多个特征来对信号中的信息进行编码的信号。作为示例而非限制,通信媒质包括用电子、光学、RF、红外、声学或其它载体实现的有线或无线技术。

[0086] 本技术和工具可以在计算机可读媒质的一般语境中描述。计算机可读媒质是任一在计算环境中可访问的可用媒质。作为示例而非局限,在计算环境 (600) 中,计算机可读媒质包括存储器 (620)、存储 (640)、通信媒质以及上述任一件的组合。

[0087] 本技术和工具可以在计算机可执行指令的一般语境下描述,如包括在程序模块中、在目标真实或虚拟处理器上的计算环境中执行的计算机可执行指令。通常,程序模块包括例程、程序、库、对象、类、组件、数据结构等等,执行特定的任务或实现特定的抽象数据类型。如不同实施例的期望,程序模块的功能可以在程序模块之间组合或分离。程序模块的计算机可执行指令可以在本地或分布式计算环境中执行。

[0088] 为演示目的,详细描述使用术语如“确定”、“选择”、“重建”及“通知”,来描述计算环境中的计算机操作。这些术语是对计算机执行的操作的高级抽象,不应与人类执行的行动混淆。与这些术语相应的实际计算机操作可根据实现的不同而不同。

[0089] II. 一般化的视频编码器和解码器

[0090] 图 7 是一般化的视频编码器 (700) 的结构图,图 8 是一般化的视频解码器 (800) 的结构图。

[0091] 编码器和解码器内的模块之间示出的关系指示了编码器和解码器内的信息的主流程;为简化目的,未示出其它关系。特别地,图 7 和 8 通常未示出指示用于视频序列、帧、宏块、块等等的编码器设置、模式、表等等的侧面信息。这类侧面信息在输出比特流中发送,通常在对侧面信息的熵编码之后。输出比特流的格式可以是 Windows Media Video 版本 8 的格式或另一格式。

[0092] 编码器 (700) 和解码器 (800) 是基于块的,并使用 4:2:0 的宏块格式,每一宏块包括 4 个 8×8 的亮度块(有时也作为一个 16×16 的宏块来看待)以及两个 8×8 的色度块。可选地,编码器 (700) 和解码器 (800) 可以是基于对象的,使用不同的宏块或块格式,或在尺寸和构造不同于 8×8 的块和 16×16 的宏块的像素集的基础上执行操作。

[0093] 根据所期望的实现和压缩类型,编码器或解码器的模块可以被添加、省略、拆分成

多个模块、与其它模块组合和 / 或用相似模块代替。在替代实施例中,使用不同模块和 / 或其它模块构造的编码器或解码器执行所描述的技术的一个或多个。

[0094] A. 视频编码器

[0095] 图 7 是一般视频编码器系统 (700) 的结构图。编码器系统 (700) 接收包括当前帧 (705) 的一系列视频帧并生成压缩的视频信息 (795) 作为输出。视频编码器的具体实施例通常使用一般化的编码器 (700) 的变异或补充版本。

[0096] 编码器系统 (700) 压缩预测帧和主帧。为了演示,图 7 示出了主帧通过编码器系统 (700) 的路径以及前向预测帧的路径。编码器系统 (700) 的许多组件既用来压缩主帧也用来压缩预测帧。这些组件所执行的确切操作可根据压缩的信息类型的不同而不同。

[0097] 预测帧 (也称为双向预测中的 p 帧、b 帧或帧间编码帧 (inter-coded frame)) 按照从一个或多个其它帧的预测 (或差异) 来表示。预测剩余是所预测的和原始帧之间的差异。相反,主帧 (也称为 i 帧或帧内编码帧 (intra-coded frame)) 不参考其它帧而压缩。

[0098] 如果当前帧 (705) 是前向预测帧,运动估计器 (710) 估计当前帧的像素宏块或其它像素集对于参考帧的运动,参考帧是缓存在帧存储 (720) 中的重建的前一帧 (725)。在替代实施例中,参考帧是后一帧,或者当前帧是双向预测的。运动估计器 (710) 能够根据像素、1/2 像素、1/4 像素或其它增量来估计运动,并在逐帧或其它基础上切换运动估计的分辨率。运动估计的分辨率可以在水平上和垂直上相同或不同。运动估计器 (710) 将运动信息 (715),如运动矢量作为侧面信息输出。运动补偿器 (730) 将运动信息 (715) 应用到重建的前一帧 (725) 来组成已经过运动补偿的当前帧 (735)。然而,预测很少是完美的,已经过运动补偿的当前帧 (735) 和原始当前帧 (705) 之间的差异为预测剩余 (745)。可选地,运动估计器和运动补偿器应用另一类型的运动估计 / 补偿。

[0099] 频率转换器 (760) 将空间域视频信息转换为频域 (即,频谱的) 数据。对于基于块的视频帧,频率转换器 (760) 对像素数据或预测剩余数据的块应用离散余弦变换 [“DCT”] 或 DCT 的变异,生成 DCT 系数块。可选地,频率变换器 (760) 应用另一常规频率变换,如傅立叶变换或使用小波或子带分析。在编码器使用空间外插 (图 7 未示出) 来对主帧的块进行编码的实施例中,频率转换器 (760) 可以对主帧的预测剩余的块应用重定向的频率变换,如偏斜 DCT。在其它实施例中,频率变换器 (760) 对预测帧的预测剩余应用 8×8 、 8×4 、 4×8 或其它尺寸的频率变换 (如 DCT)。

[0100] 量化器 (770) 然后量化空间数据系数块。量化器对空间数据应用统一的标量量化,其步长在逐帧或其它基础上变化。可选地,量化器对空间数据系数应用另一种量化,例如,非统一、矢量或非自适应量化,或直接在不使用频率变换的编码器系统内量化空间域数据。除自适应量化之外,编码器 (700) 可以使用帧丢弃、自适应滤波或其它技术来进行率控制。

[0101] 如果预测帧中一个给定的宏块没有特定类型的信息 (如,没有宏块的运动信息并且没有剩余信息),则编码器 (700) 将该宏块编码为跳过宏块。如果是这样,编码器在压缩的视频信息 (795) 的输出比特流中发信号通知该跳过宏块。

[0102] 当需要重建的当前帧用于随后的运动估计 / 补偿时,反量化器 (776) 在量化的空间数据系数上执行反量化。反频率变换器 (766) 然后执行频率变换器 (760) 的反操作,生成重建的预测剩余 (对预测帧) 或重建的主帧。如果当前帧 (705) 是主帧,则重建的主帧

被用作重建的当前帧（未示出）。如果当前帧（705）是预测帧，则将重建的预测剩余添加到已经过运动补偿的当前帧（735）来组成重建的当前帧。帧存储（720）缓存重建的当前帧用于预测下一帧。在一些实施例中，编码器对重建的帧应用数据分块滤波器来自适应地平滑帧的块中的不连续。

[0103] 熵编码器（780）压缩量化器（770）的输出以及某些侧面信息（例如，运动信息（715）、空间外插模式、量化步长）。典型的熵编码技术包括算术编码、差分编码、哈夫曼（Huffman）编码、行程编码、LZ 编码、字典编码以及上述的组合。熵编码器（780）通常对不同种类的信息（例如，DC 系数、AC 系数、不同种类的侧面信息）使用不同的编码技术，并能够从特定编码技术之内的多个代码表中进行选择。

[0104] 熵编码器（780）将压缩的视频信息（795）放置在缓存（790）中。缓存级别指示符反馈至比特率自适应模块。

[0105] 压缩的视频信息（795）从缓存（790）中以恒定或相对恒定的比特率排出，并被储存用于该比特率的随后的流中。因此，缓存（790）的级别主要是已滤波的量化视频信息的熵的函数，它影响熵编码的效率。可选地，编码器系统（700）在压缩之后立即将压缩的视频信息形成流，并且缓存（790）的级别也取决于信息从缓存（790）中排出进行发送的率。

[0106] 在缓存（790）之前或之后，可以对压缩的视频信息（795）进行信道编码用于通过网络来发送。信道编码可以将误差检测和纠正数据应用到压缩视频信息（795）中。

[0107] B. 视频解码器

[0108] 图 8 是一般视频解码器系统（800）的结构图。解码器系统（800）接收视频帧的已压缩序列的信息（895），并生成包括重建帧（805）的输出。视频解码器的特定实施例通常使用一般化的解码器（800）的变异或补充版本。

[0109] 解码器系统（800）对预测帧和主帧进行解压。为了演示，图 8 示出了主帧通过解码器系统（800）的路径以及前向预测帧的路径。解码器系统（800）的许多组件既用来对主帧进行解压也用来对预测帧进行解压。这些组件的确切操作可以根据所要解压的信息类型的不同而不同。

[0110] 缓存（890）接收压缩视频序列的信息（895）并使接收的信息对熵解码器（880）可用。缓存（890）通常以随时间变化相当恒定的率接收信息，并包括抖动缓存来平滑带宽或传输中的短期变化。缓存（890）可以包括回放缓存以及其它缓存。可选地，缓存（890）以变化的率来接收信息。在缓存（890）之前或之后，可以对压缩的视频信息进行信道解码和误差检测和纠正的处理。

[0111] 熵解码器（880）对熵编码的量化数据和熵编码的侧面信息（例如，运动信息（815）、空间外插模式、量化步长）进行熵解码，通常应用编码器中执行的熵编码的反操作。熵解码技术包括算术解码、差分解码、哈夫曼解码、行程解码、LZ 解码、字典解码以及上述的组合。熵解码器（880）经常对不同的信息种类（如 DC 系数、AC 系数、不同的侧面信息种类）使用不同的解码技术，并能够在特定的解码技术中从多个代码表中作出选择。

[0112] 如果要重建的帧（805）是前向预测帧，则运动补偿器（830）对参考帧（825）应用运动信息（815）来组成要重建的帧（805）的预测（835）。例如，运动补偿器（830）使用宏块运动矢量来找出参考帧（825）中的宏块。帧缓存（820）储存前一重建的帧用作参考帧。运动补偿器（830）能够按照像素、1/2 像素、1/4 像素或其它增量来补偿运动，并能够在逐帧或

者其它基础上切换运动补偿的分辨率。运动补偿的分辨率可以在水平上和垂直上相同或不同。可选地,运动补偿器应用另一类型的运动补偿。运动补偿器的预测很少是完美的,因此解码器(800)也重建预测剩余。

[0113] 当解码器需要重建帧用于随后的运动补偿时,帧存储(820)缓存重建帧用于预测下一帧。在一些实施例中,编码器对重建帧应用数据分块滤波器来自适应地平滑帧的块中的不连续。

[0114] 反量化器(870)反量化熵解码的数据。一般来说,反量化器对熵解码的数据应用统一标量反量化,其步长在逐帧或者其它基础上变化。可选地,反量化器对数据应用另一类型的反量化,如非统一、矢量或非自适应量化,或直接在不使用反频率变换的解码器系统中反量化空间域。

[0115] 反频率转换器(860)将量化的频域数据转换为空间域视频信息。对于基于块的视频帧,反频率转换器(860)对DCT系数块应用反DCT[“IDCT”]或IDCT的变异,生成分别用于主帧或预测帧的像素数据或预测剩余数据。可选地,频率转换器(860)应用另一常规反频率变换,如傅立叶变换或使用小波或子带合成。在解码器使用空间外插(图8未示出)来对主帧的块进行解码的实施例中,反频率转换器(860)能够对主帧的预测剩余块应用重定向的反频率转换,如偏斜IDCT。在其它实施例中,反频率转换器(860)对预测帧的预测剩余应用 8×8 、 8×4 、 4×8 或其它尺寸的反频率转换(如IDCT)。

[0116] 当在压缩的视频帧序列的信息(895)的比特流中发信号通知跳过宏块时,解码器(800)重建跳过宏块,而不使用通常包含在非跳过宏块的比特流中的信息(例如,运动信息和/或剩余信息)。

[0117] III. 第一实现

[0118] 在第一实现中,视频编码器和解码器分别以提高的效率对跳过宏块的信息进行编码和解码。在视频比特流中的图像层上发信号通知跳过宏块的信息,使得编码器能够充分利用跳过宏块的信息中的冗余。同样,编码器和解码器在多个编码模式之间进行选择以对跳过宏块的信息进行编码和解码。

[0119] A. 跳过宏块的信息的图像层编码

[0120] 在第一实现中,压缩视频序列由构造成四个分级层的数据组成。从上到下层分别为:1)序列层;2)图像层;3)宏块层;以及4)块层。在图像层,每一图像的数据包括图像头,随后是宏块层的数据。(类似地,在宏块层,每一宏块的数据包括宏块头,随后是块层的数据。)当I图像和P图像的一些比特流元素相同时,其它元素仅出现在P图像,反之亦然。

[0121] 图9示出了组成P图像层(900)的比特流元素。表1简要地描述了P图像层(900)的比特流元素。

[0122]

字段	描述
PTYPE(910)	图像类型
PQUANT(912)	图像量化器比例

[0123]

SMBC(920)	跳过宏块代码
SMB(930)	跳过宏块字段
CPBTAB(940)	已编码块模式表
MVRES(942)	运动矢量分辨率
TTMBF(944)	宏块级转换类型标志
TTFRM(946)	帧级转换类型
DCTACMBF(948)	宏块级 DCT AC 编码置位标志
DCTACCFRM(950)	帧级 DCT AC 编码置位索引
DCTDCTAB(952)	帧内 DCT DC 表
MVTAB(954)	运动矢量表
MB LAYER(960)	宏块层

[0124] 表 1 :第一实现中 P 图像层的比特流元素

[0125] 特别地, P 图像层 (900) 包括用于 P 图像中宏块的跳过宏块字段 (“SMB”) (930) 以及发信号通知跳过宏块字段 (930) 的编码模式的跳过宏块代码 (“SMBC”) 字段 (920)。SMBC 字段 (920) 仅出现在 P 图像头中。SMBC(920) 是一个 2 位的值,发信号通知用于指示帧中跳过的宏块的四种模式之一。在第一实现中,用于跳过宏块编码模式的固定长度代码 (“FLC”) 如下:

[0126]

SMBC FLC	跳过位编码模式
00	无跳过位编码
01	普通跳过位编码
10	行预测 (或,“行跳过”) 跳过位编码
11	列预测 (或,“列跳过”) 跳过位编码

[0127] 表 2 :第一实现中跳过宏块编码模式代码表

[0128] 如果编码模式为普通、行预测或列预测,则比特流的下一字段是包含跳过宏块信息的 SMB 字段 (930)。因此, SMB 字段仅出现在 P 图像头中,并且仅当 SMBC 发信号通知普通、行预测或列预测跳过宏块编码时才出现。如果 SMBC 发信号通知普通编码,则 SMB 字段的尺寸等于帧中宏块的数量。如果 SMBC 发信号通知行预测或列预测,则 SMB 的尺寸如下所述地变化。

[0129] 跳过宏块信息通知解码器帧中哪一宏块不在宏块层中。对于这些宏块,解码器在重建该宏块时将从参考帧复制相应的宏块像素数据。

[0130] B. 切换跳过宏块信息的编码模式

[0131] 如上所述,SMB 字段 (920) 发信号通知跳过宏块字段 (930) 的编码模式。更一般地,图 10 示出了一种用于在具有多个跳过宏块编码模式的视频编码器中对跳过宏块的信息进行编码的技术 (1000)。图 11 示出了一种用于对由具有多个跳过宏块编码模式的视频编码器编码的跳过宏块的信息进行解码的相应技术 (1100)。

[0132] 参考图 10,编码器选择一种跳过宏块编码模式来对跳过宏块的信息进行编码 (1010)。例如,在第一实现中,跳过宏块编码模式包括无跳过宏块的模式、普通模式、行预测 (或,“行跳过”) 模式以及列预测 (或,“列跳过”) 模式。在选择编码模式之后,编码器对跳过宏块的信息进行编码 (1020)。编码器在逐图的基础上选择编码模式。可选地,编码器在其它基础 (例如,在序列级) 上选择编码模式。当编码器完成对跳过宏块的信息进行编码之后 (1030),编码结束。

[0133] 参考图 11,解码器确定编码器用来对跳过宏块的信息进行编码的跳过宏块编码模式 (1110)。解码器然后对跳过宏块的信息 (1120) 进行解码。解码器在逐图的基础上确定编码模式。可选地,解码器在其它基础 (例如,在序列级别) 上确定编码模式。当解码器完成对跳过宏块的信息进行解码之后 (1130),解码结束。

[0134] C. 编码模式

[0135] 在第一实现中,跳过宏块编码模式包括无跳过宏块的模式、普通模式、行预测 (或,“行跳过”) 模式以及列预测 (或“列跳过”) 模式。以下部分参考图 12 描述了每一模式中如何对跳过宏块的信息进行编码,并示出了跳过宏块编码帧的一个示例 (1200)。

[0136] 1. 普通跳过宏块编码模式

[0137] 在普通模式中,每一宏块的跳过 / 未跳过状态由一个位来表示。因此,以位表示的 SMB 字段的尺寸等于帧中宏块的数量。SMB 字段中的位的位置与在帧中从左上宏块开始对宏块进行光栅扫描的顺序相应。位的值为 0 指示相应的宏块未跳过 ;位的值为 1 指示相应的宏块被跳过。

[0138] 图 13 示出了一种用于以普通跳过宏块编码模式进行编码的技术 (1300)。首先,编码器检查是否会跳过对一个宏块的编码 (1310)。如果是这样,编码器将值为 1 的位添加到 SMB 字段来指示相应的宏块被跳过 (1320)。否则,编码器将值为 0 的位添加到 SMB 字段来指示相应的宏块未跳过 (1330)。当编码器完成向 SMB 字段添加位之后 (1340),跳过宏块编码结束。

[0139] 作为示例,使用普通模式编码,图 12 中示例帧 (1200) 的 SMB 字段被编码为 : 0100101111111111111010010。

[0140] 2. 行预测跳过宏块编码模式

[0141] 在行预测模式中,每一宏块行 (从上到下) 的状态由一个位来指示。如果该位为 1,则该行所包含的全部为跳过宏块,并且跟随其后的是下一行的状态。如果该位等于 0,则用一个位来发信号通知该行中每一宏块的跳过 / 未跳过状态。因此,跟随其后的是长度等于行中宏块数的位字段。该位字段中的位表示从左到右顺序的宏块。再一次,值为 0 指示相应的宏块未跳过 ;值为 1 指示相应的宏块被跳过。

[0142] 图 14 示出了一种用于以行预测（或，“行跳过”）宏块编码模式编码的技术（1400）。首先，编码器检查一行所包含的是否全部为跳过宏块（1410）。如果是，则编码器将值为 1 的指示符位添加到 SMB 字段（1420），并且跟随其后的是下一行的状态。如果该行所包含的不全为跳过宏块，则编码器将值为 0 的指示符位添加到 SMB 字段，并且用一个位来发信号通知该行中每一宏块的跳过 / 未跳过状态（1430）。当编码器完成帧（1440）中所有行的处理之后（1440），行预测编码结束。

[0143] 对于解码，图 15 所示为对跳过宏块的信息进行行预测解码的伪代码（1500）。在伪代码（1500）中，函数 `get_bits(n)` 从比特流读取 `n` 位并返回其值。

[0144] 作为示例，使用行预测模式编码，图 12 的示例帧（1200）的 SMB 字段被编码为：0010010110010010。

[0145] 3. 列预测跳过宏块编码模式

[0146] 在列预测模式中，每一宏块列（从左到右）的状态用一个位来指示。如果该位为 1，则该列所包含的全部为跳过宏块，并且跟随其后的是下一列的状态。如果该位为 0，则用一个位来发信号通知该列中每一宏块的跳过 / 未跳过状态。因此，跟随其后的是长度等于该列中宏块数的位字段。该位字段中的位表示从上到下顺序的宏块。再一次，值为 0 指示相应的宏块未跳过；值为 1 指示相应的宏块被跳过。

[0147] 图 16 示出了一种用于以列预测（或，“列跳过”）宏块编码模式编码的技术（1600）。首先，编码器检查该列所包含的是否全部为跳过宏块（1610）。如果是，则编码器将值为 1 的指示符位添加到 SMB 字段（1620），并且跟随其后的是下一列的状态。如果该列包含的不全为跳过宏块，则编码器将值为 0 的指示符位添加到 SMB 字段，并且用一个位来发信号通知该列中每一宏块的跳过 / 未跳过状态。当编码器完成帧中所有列的处理之后（1640），列预测编码结束。

[0148] 对于解码，图 17 所示为对跳过宏块的信息进行列预测解码的伪代码（1700）。

[0149] 作为示例，使用行预测模式编码，图 12 的示例帧（1700）的 SMB 字段被编码为：0011010011000110100110。

[0150] IV. 第二实现

[0151] 在第二实现中，视频编码器和解码器以提高的效率分别对跳过宏块的信息和 / 或其它 2-D 二进制数据进行编码和解码。编码器和解码器将跳过宏块定义为具有缺省运动（不一定是零运动）的块，使得编码器和解码器可以在许多情况下跳过更多的宏块。有效的帧级位平面编码指示了跳过宏块的信息和 / 或其它 2-D 二进制数据。同样，编码器和解码器可以使用跳过宏块的原始（MB 级）编码选项用于低等待时间的应用。

[0152] A. 跳过位定义（跳过宏块的定义）

[0153] 第二实现包括跳过宏块的概念的新定义。“跳过”指比特流中没有进一步的信息需要在该粒度级上发送的情况。跳过宏块（块）是具有缺省类型、缺省运动和缺省剩余误差的宏块（块）。（作为比较，在其它实现和标准中，跳过宏块是具有零运动和零剩余的预测宏块。）

[0154] 跳过宏块的新定义是其运动等于其因果预测运动并具有零剩余误差的预测宏块。（与其它定义的不同点是缺省运动等于运动预测器，这不一定要为零。）

[0155] 例如，在一些实施例中，从紧靠当前宏块上方或左边的宏块中得到当前宏块的预

测运动矢量。或者,从当前宏块的左边、上方或者右上的宏块的水平 and 垂直分量范围内的中项生成预测器的水平和垂直分量。

[0156] 具有四个运动矢量 (4MV) 的跳过宏块的运动矢量由其顺序地以自然扫描顺序执行的各预测给出。在一个运动矢量 (1MV) 的情况下,误差剩余为零。

[0157] 图 18 示出了一种用于在视频编码器中依照跳过宏块的新定义确定是否要跳过对特定宏块的编码的技术 (1800)。首先,编码器检查当前帧是否是 I 帧或 P 帧 (1810)。如果当前帧是 I 帧,则当前帧中不跳过任何宏块 (1820),对该帧的跳过宏块编码结束。

[0158] 另一方面,如果当前帧是 P 帧,编码器检查当前帧中可以跳过的宏块。对于一个给定的宏块,编码器检查该宏块的运动矢量是否等于该宏块因果预测的运动矢量 (例如,该宏块的差分运动矢量是否等于零) (1830)。如果宏块的运动不等于因果预测的运动,则编码器不跳过该宏块 (1840)。否则,编码器检查该宏块是否有任何剩余需要编码 (1850)。如果有需要编码的剩余,编码器不跳过该宏块 (1860)。然而,如果该宏块没有剩余,编码器跳过该宏块 (1870)。编码器继续编码或跳过宏块直到完成编码 (1880)。

[0159] B. 位平面编码

[0160] 在第二实现中,某一宏块特定信息 (包括发信号通知跳过宏块) 可以以每宏块一位来编码。帧内所有宏块的状态可以一起编码为位平面并在帧头中发送。

[0161] 在第二实现中,编码器在三种情况下使用位平面来发信号通知关于帧中宏块的信息。这三种情况为:1) 发信号通知跳过宏块,2) 发信号通知半帧或帧宏块模式,以及 3) 发信号通知每一宏块的 1-MV 或 4-MV 运动矢量。这一部分描述了对三种情况的任一种的位平面编码以及相应的解码。

[0162] 帧级位平面编码用来对二维二进制数组进行编码。每一数组的尺寸是 $rowMB \times colMB$, 其中, $rowMB$ 和 $colMB$ 分别为宏块行和列的数目。在比特流中,每一数组被编码为一组连续的位。使用七种模式之一来对每一数组进行编码,如表 3 列举并在下文描述的。

[0163]

编码模式	描述
原始	以每码元一位编码
普通 -2	两个码元共同编码
差分 -2	位平面的差分编码,随之为对两个剩余码元共同编码
普通 -6	六个码元共同编码
差分 -6	位平面的差分编码,随之为对六个剩余码元共同编码
行跳过	一位跳跃来发信号通知没有置位的位的行
列跳过	一位跳跃来发信号通知没有置位的位的列

[0164] 表 3: 第二实现中的编码模式

[0165] 在第二实现中,编码器使用三个语法元素:MODE、INVERT 和 DATABITS,以在位平面中嵌入信息。

[0166] MODE 字段是对位平面的编码模式进行编码的可变长度代码(“VLC”)。例如,MODE 字段中的 VLC 表示表 3 所列的七种编码模式中的任一种。为节省位,编码器可以向可能性较大的编码模式分配较短的代码,并向可能性较小的编码模式分配较长的代码。如上所述,MODE 字段在帧头中发送。

[0167] 编码器和解码器在逐帧的基础上在编码模式之间切换。例如,编码器和解码器分别如图 10 和 11 中第一实现的编码器和解码器在跳过宏块编码模式之间切换一样在编码模式之间切换。可选地,编码器和解码器使用其它技术和 / 或在其它基础上进行切换。

[0168] 如果模式不是原始模式,则发送一位的 INVERT 字段。在可能执行条件倒置的若干个编码模式中,INVERT 字段指示编码器中位平面中的位是否在编码发生之前被倒置以及解码器中的解码输出是否被倒置。当位平面中的大多数位等于 1 时,INVERT 字段为 1,当位平面中的大多数位等于 0 时,INVERT 字段为 0。当出现较多的 0 时,编码器采用若干种消耗更少位的编码模式(如普通 -2 和普通 -6)。如果要编码的位平面中 1 比 0 多,则编码器可以倒置位平面来增加位平面中 0 的比例并增加用于节省位的潜力。其它模式(如差分 -2 和差分 -6)使用 INVERT 的值来计算预测器位平面。因此,在某些编码模式中,解码器上最终的重建位平面取决于 INVERT。

[0169] DATABITS 字段是包含重建位平面所需要的信息的 VLC 码元的熵编码流,在给定 MODE 和 INVERT 字段的情况下。

[0170] C. 编码模式

[0171] 在第二实现中,编码器以七种不同编码模式的任一种对二进制信息(例如,跳过宏块信息)进行编码:行跳过模式、列跳过模式、普通 -2 模式、普通 -6 模式、差分 -2 模式、差分 -6 模式以及原始模式。解码器对七种编码模式的任一种执行相应的解码。每一模式在下文详细描述。

[0172] 可选地,编码器和解码器使用其它和 / 或另外的编码模式。

[0173] 1. 行跳过和列跳过模式

[0174] 行跳过编码模式通过当行中每一二进制码元是一个特定值时用单个位来表示位平面中的一行来节省位。例如,编码器在位平面中用 0 来表示跳过宏块,并使用以单个位来表示全 0 行的行跳过编码模式。因此,当宏块的全部行都跳过时,编码器节省了位。解码器执行相应的解码。

[0175] 在第二实现中,全零行用设为 0 的一位来指示。当该行不是全为 0 时,一位指示符设为 1,跟随其后的是按顺序排列的包含该位平面行的 colMB 位。行以自然顺序扫描。

[0176] 同样地,对于列跳过模式,如果整个行为零,则发送一个 0 位。否则,则发送 1,跟随其后的是按顺序排列包含整个列的 rowMB 位。列以自然顺序扫描。

[0177] 对以差分 -6 和普通 -6 模式(下文描述)对剩余的行和 / 或列的编码,应用相同的逻辑。一个一位的标志指示行或列是否全为 0。如果不是,则使用每码元一位来发送整个行或列。

[0178] 当编码器对主要包含 1 的位平面编码时,行跳过和列跳过编码通常效率较低,因为行 / 列全部包含 0 的概率较小。然而,编码器能够在这一情况下在位平面上执行倒置来

增加 0 的比例并潜在地提高位节省。由此,当通过 INVERT 位来指示条件倒置时,编码器在平铺位平面并对其进行编码之前,对位平面进行预倒置 (pre-invert)。在解码器端,通过对最终输出执行倒置来实现条件倒置。(对差分 -2 和差分 -6 模式不执行该步骤)。

[0179] 图 19 示出了一种用于以行跳过模式在位平面中对二进制信息进行编码的技术 (1900)。编码器首先检查位平面的倒置是否适合,如果是,则执行倒置 (1910)。编码器然后检查位平面中的一行来看行中的每一位是否等于 0 (1920)。如果是,则编码器将该行的指示符位设为 0 (1930)。如果行中任一位不为 0,则编码器将该行的指示符位设为 1,并用一位来对该行中每一位进行编码 (1940)。当编码器完成对位平面中所有行的编码之后 (1950),位平面编码结束。

[0180] 对行跳过编码模式,解码器执行相应的解码。

[0181] 图 20 示出了一种用于以列跳过模式对二进制信息进行编码的技术 (2000)。编码器首先检查位平面的倒置是否适合,如果是,则执行倒置 (2010)。编码器然后检查位平面中的一列来看该列中的每一位是否等于 0 (2020)。如果是,则编码器将该列的指示符位设为 0 (2030)。如果列中任一位不为 0,则编码器将该列的指示符位设为 1,并用一位来对该列中的每一位进行编码 (1940)。当编码器完成位平面中每一列的编码之后 (1950),位平面编码结束。

[0182] 对列跳过编码模式,解码器执行相应的解码。

[0183] 2. 普通 -2 模式

[0184] 编码器使用普通 -2 模式来对位平面中多个码元进行共同编码 (例如,通过使用矢量哈夫曼或其它可变长度编码模式)。编码器使用可变长度代码对二进制码元对进行编码。解码器执行相应的解码。

[0185] 如果 $\text{rowMB} \times \text{colMB}$ 是奇数,则将第一个码元编码为单个位。随后的码元以自然扫描顺序成对地进行编码。使用 VLC 表来对码元对进行编码以降低总熵。

[0186] 当通过 INVERT 位指示条件倒置时,编码器在成对地对位平面进行编码之前对其进行预倒置。在解码器端,通过对最终输出执行倒置来实现条件倒置。(当使用差分 -2 模式时,在这一步骤不执行条件倒置)。

[0187] 图 21 示出了一种用于以普通 -2 模式对二进制信息进行编码的技术 (2100)。编码器执行初始检查来确定位平面倒置是否适合以提高编码效率,如果是,执行倒置 (2110)。编码器然后确定要编码的位平面是否具有奇数个二进制码元 (2120)。如果是,则编码器将第一个码元用单个位来编码 (2130)。编码器然后使用可变长度代码来对码元对进行编码,使用较短的代码来表示可能性较大的对,使用较长的代码来表示可能性较小的对 (2140)。当完成对码元对的编码时 (2150),编码结束。

[0188] 对普通 -2 模式,解码器执行相应的解码。

[0189] 3. 普通 -6 模式

[0190] 编码器也使用普通 -6 模式来对位平面中多个二进制码元进行共同编码 (例如,使用矢量哈夫曼或其它可变长度编码模式)。编码器平铺以六个二进制码元组成的组,并用可变长度代码来表示每一组。解码器执行相应的解码。

[0191] 在普通 -6 模式 (以及差分 -6 模式) 中,位平面以六个像素为一组进行编码。这些像素分组成 2×3 或 3×2 的平铺块。使用一组规则最大化地对位平面进行平铺,剩余的

像素使用行跳过或列跳过模式的变异来进行编码。

[0192] 在第二实现中,当且仅当 rowMB 为 3 的倍数且 colMB 不是 3 的倍数时使用 3×2 “垂直”平铺块。否则,则使用 2×3 “水平”平铺块。图 22、23 和 24 示出了普通 -6 编码模式中平铺的帧的示例。图 22 示出了帧 (2200),具有 3×2 垂直平铺块以及将使用列跳过模式进行编码的一个码元宽度的剩余(以阴影部分显示)。图 23 示出了帧 (2300),具有 2×3 水平平铺块以及将要使用行跳过模式进行编码的一个码元宽度的剩余。图 24 示出了帧 (2400),具有 2×3 水平平铺块以及将要使用行跳过和列跳过模式进行编码的一个码元宽度的剩余。

[0193] 尽管在该示例中使用 3×2 和 2×3 平铺块,在其它实施例中,可以使用不同的平铺块构造和 / 或不同的平铺规则。

[0194] 首先对 6 元素平铺块进行编码,跟随其后的是用列跳过和行跳过编码的线性平铺块。如果数组尺寸是 3×2 或 2×3 的倍数,则后面的线性平铺块不存在,位平面能完美地平铺。使用 VLC 表来对 6 元素矩形平铺块进行编码。

[0195] 当通过 INVERT 位来指示条件倒置时,编码器在对位平面进行平铺和编码之前对其进行预倒置。在解码器端,通过对最终输出执行倒置来实现条件倒置。(当使用差分 -6 模式时,在这一步骤不执行条件倒置。)

[0196] 图 25 示出了一种用于以普通 -6 模式对二进制信息进行编码的技术 (2500)。编码器执行初始检查来确定位平面的倒置是否适合以提高编码效率,如果是,则执行倒置 (2510)。编码器然后检查位平面的行数是否为 3 的倍数 (2520)。如果行数不是 3 的倍数,则将位平面中的码元分组为 2×3 的水平平铺块 (2530)。

[0197] 如果行数为 3 的倍数,则编码器检查位平面的列数是否为 3 的倍数 (2540)。如果列数为 3 的倍数,则编码器将位平面中的码元分组为 2×3 的水平平铺块 (2530)。如果列数不是 3 的倍数,则编码器将码元分组为 3×2 垂直平铺块 (2550)。

[0198] 在将码元分组为 3×2 或 2×3 的平铺块之后,编码器使用如六维矢量哈夫曼编码技术或其它编码技术来对以 6 平铺的码元组进行编码 (2560)。编码器使用上述行跳过和 / 或列跳过编码技术来对任何剩余的未平铺码元进行编码 (2570)。

[0199] 对普通 -6 编码模式,解码器执行相应的解码。

[0200] 在其它实施例中,编码器使用其它技术来对平铺和未平铺的码元进行编码。

[0201] 4. 差分 -2 和差分 -6 模式

[0202] 差分编码模式,如差分 -2 或差分 -6 模式,通过基于要编码的位平面的预测器首先对要编码的位平面生成差分(或剩余)位的位平面来对位平面进行编码。然后使用如普通 -2 或普通 -6 模式来对剩余位平面进行编码,而不需要条件倒置。

[0203] 在第二实现中,差分 -2 和差分 -6 模式使用以 diff 操作来表示的差分编码。如果使用了任一差分模式,首先通过检查位平面 $b(i, j)$ 的预测器 $\hat{b}(i, j)$ 来生成差分位的位平面,预测器由以下因果操作来定义:

[0204]

$$\hat{b}(i, j) = \begin{cases} INVERT & i = j = 0 \text{ 或 } b(i, j-1) \neq b(i-1, j) \\ b(0, j-1) & i = 0 \\ b(i-1, j) & \text{其它} \end{cases} \quad (1)$$

[0205] 换言之,给定二进制码元 $b(i, j)$ 的预测器 $\hat{b}(i, j)$ 是紧靠 $b(i, j)$ 左边的二进制码元,下列特殊情况除外:

[0206] 1) 如果 $b(i, j)$ 在位平面的左上角,或其上方码元 $b(i, j-1)$ 不等于左边二进制码元 $b(i-1, j)$,则预测器 $\hat{b}(i, j)$ 等于 INVERT 的值;或者

[0207] 2) 如果 1) 不适用并且 $b(i, j)$ 位于最左列 ($i = 0$),则预测器 $\hat{b}(i, j)$ 为上方二进制码元 $b(i, j-1)$ 。

[0208] 在编码器端, diff 操作根据以下来计算剩余位平面 r :

$$[0209] \quad r(i, j) = b(i, j) \oplus \hat{b}(i, j) \quad (2)$$

[0210] 其中 \oplus 是异或操作。使用普通 -2 或普通 -6 模式对剩余位平面进行编码,而不需要条件倒置。

[0211] 在解码器端,使用适当的普通模式来重新生成剩余位平面。随后,使用剩余位来将原始位平面作为二进制 2-D 差分重新生成:

$$[0212] \quad b(i, j) = r(i, j) \oplus \hat{b}(i, j) \quad (3)$$

[0213] 图 26 示出了一种用于以差分编码模式对二进制信息进行编码的技术 (2600)。编码器对位平面计算预测器 (2610),如等式 (1) 所示。然后编码器如通过在位平面及其预测器上执行 XOR 操作来计算剩余位平面 (2620)。然后编码器对剩余位平面进行编码(例如,用普通 -2 或普通 -6 模式) (2630)。

[0214] 图 27 示出了一种用于对以差分编码模式编码的二进制信息进行解码的技术 (2700)。解码器根据用于对剩余位平面进行编码的模式(例如,普通 -2 或普通 -6 模式)使用适当的解码技术对剩余位平面进行解码 (2710)。解码器也使用编码器中使用的同一技术对位平面计算预测器 (2720)。然后解码器如通过在剩余位平面及其预测器位平面上执行 XOR 操作来重建原始位平面 (2730)。

[0215] 5. 原始模式

[0216] 除原始模式之外的所有模式在帧级对位平面进行编码,要求在编码过程中第二次通过该帧。然而,对低等待时间情况,第二次通过会增加不能接受的延迟(例如,由于帧头和宏块层信息的发送被延迟,直到遇到帧中的最后一个宏块为止,这是由用于对位平面进行编码的花费时间所引起的)。

[0217] 原始模式使用传统的方法,在比特流的同一位置以每二进制码元一位对位平面进行编码,作为宏块级信息的剩余。尽管码元的宏块级编码其本身不是一种新概念,然而将码元的编码从帧级切换到宏块级能够替代帧级编码提供低等待时间。

[0218] 图 28 示出了一种用于对低等待时间应用选择性地以原始编码模式对宏块的二进制信息进行编码的技术 (2800)。首先,编码器检查是否使用原始模式来对二进制信息进行编码 (2810)。如果是,则编码器对宏块在宏块级进行位的编码 (2820),并检查该宏块是否是帧中的最后一个宏块 (2830)。如果该宏块不是帧中的最后一个宏块,则编码器继续对下一宏块在宏块级进行位的编码 (2820)。

[0219] 如果编码器不使用原始编码模式,则编码器对帧中的各宏块在帧级进行位平面的编码 (2840)。当完成对帧中的宏块的编码时 (2850),编码器结束对帧的编码。

[0220] 尽管该技术 (2800) 示出了在逐帧基础上的切换模式, 编码器可选地在其它基础上进行切换。

[0221] 参考各种实施例描述并说明了本发明的原理之后, 可以认可, 在不背离这些原理的情况下可以在方案和细节上对各种实施例进行修改。应当理解, 这里描述的程序、进程或方法并非相关或局限于任一特定的计算环境类型, 除非另外指明。可以依照这里描述的指导结合各种类型的通用或专用计算环境使用或执行操作。以软件形式示出的实施例的元素可以以硬件来实现, 反之亦然。

[0222] 考虑到可以应用本发明的原理的许多可能的实施例, 对本发明要求权利, 所有这类实施例都包含在所附权利要求及其等效权利要求的范围和精神之内。

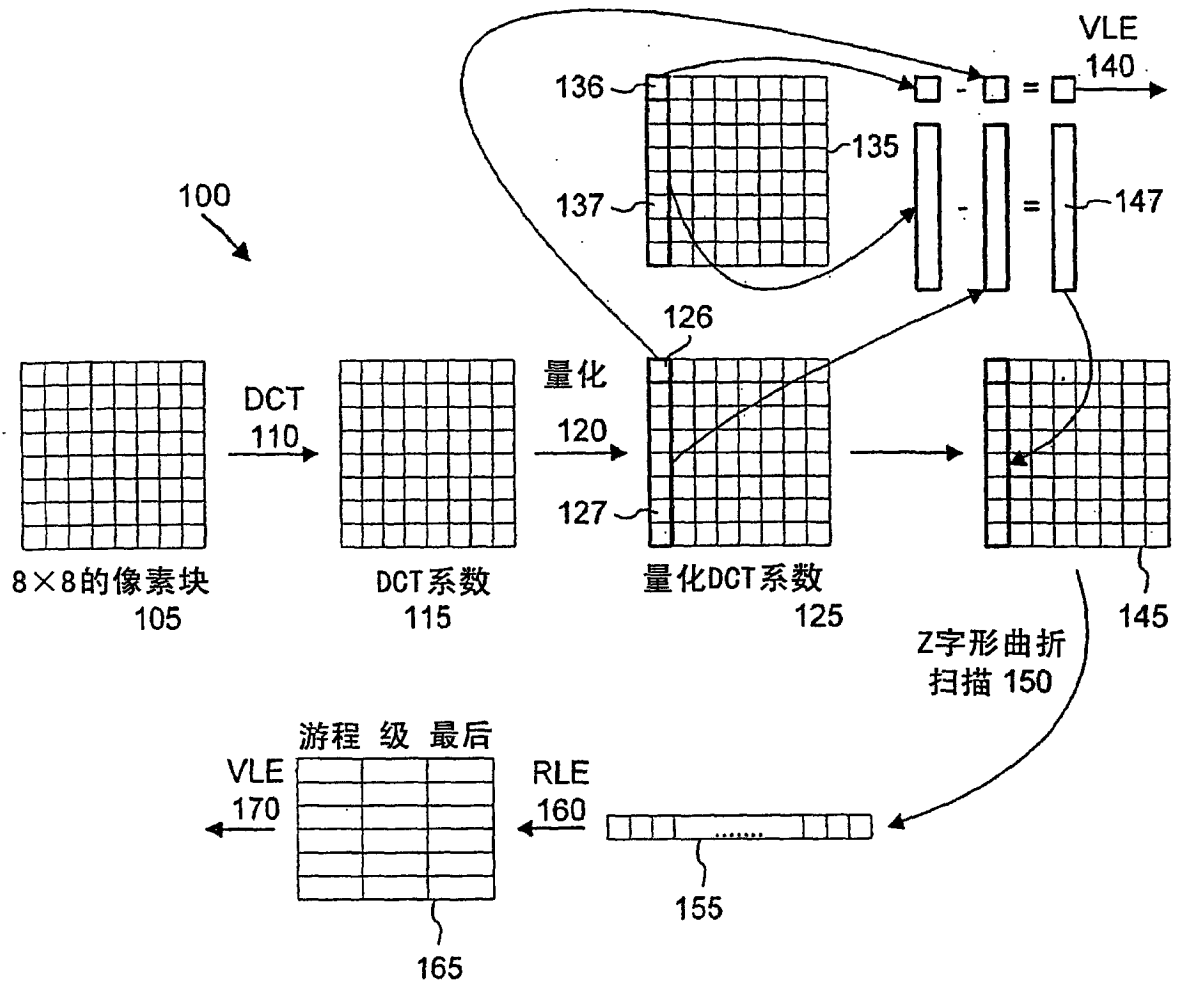


图1 现有技术

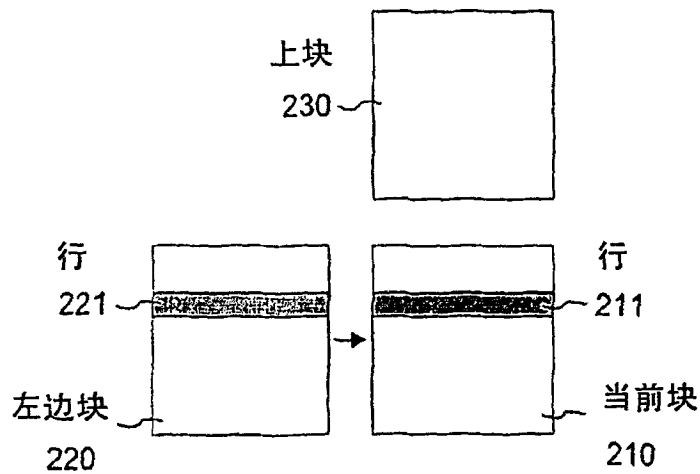


图2 现有技术

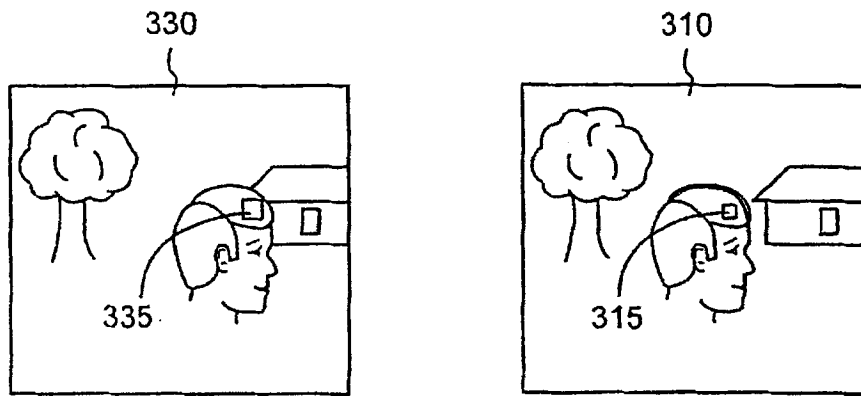


图3 现有技术

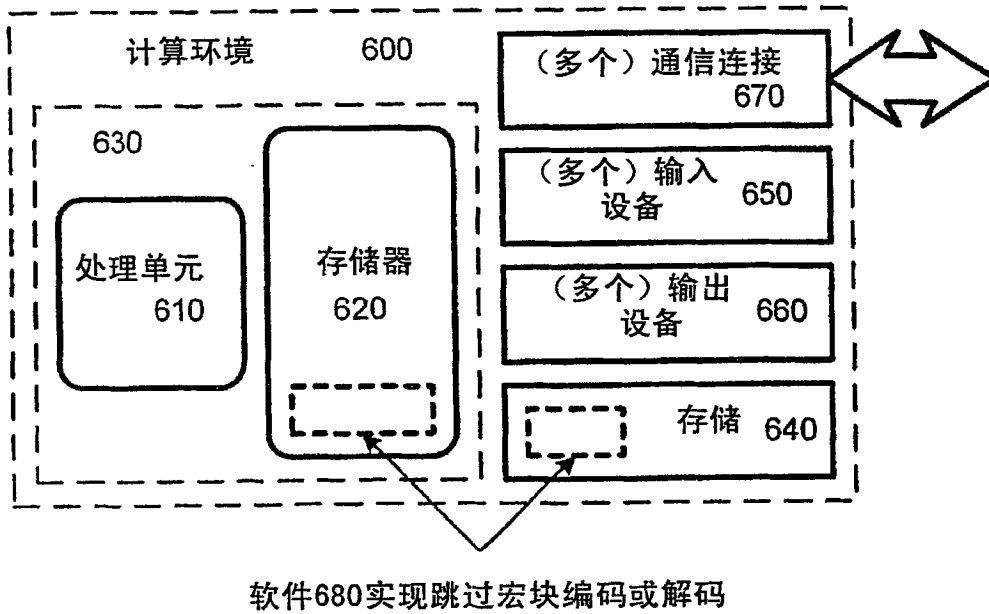


图6

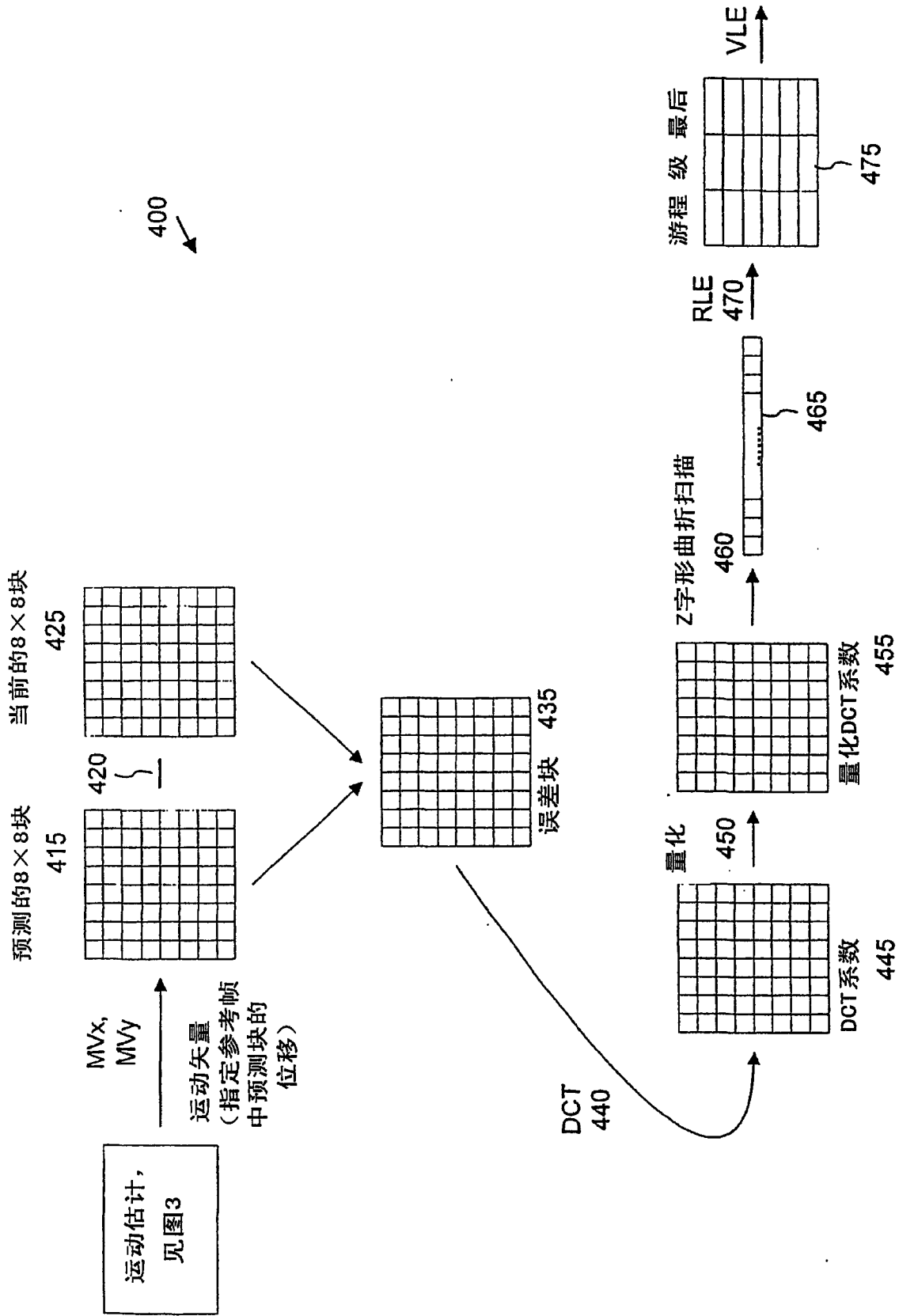


图 4

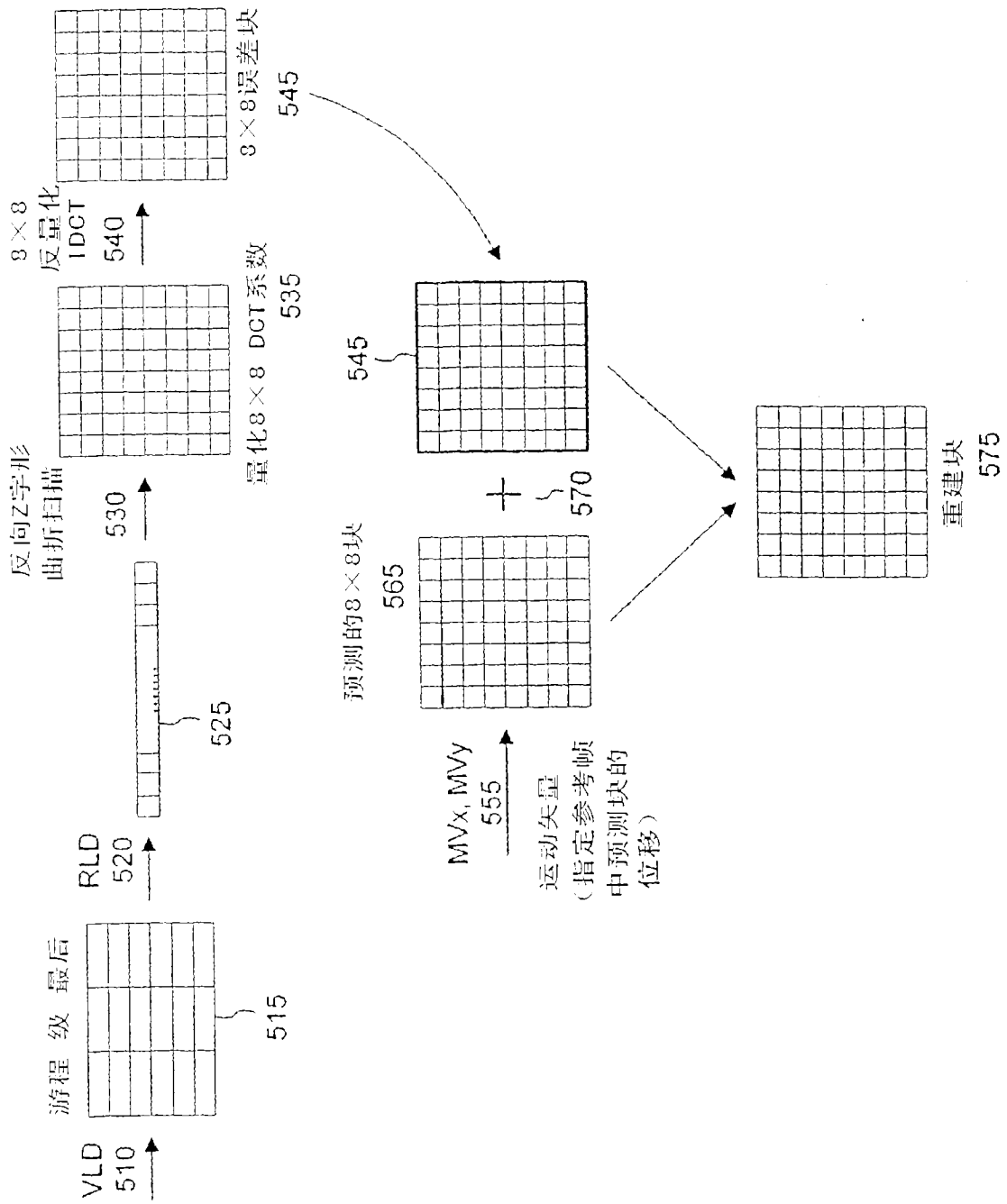


图5 现有技术

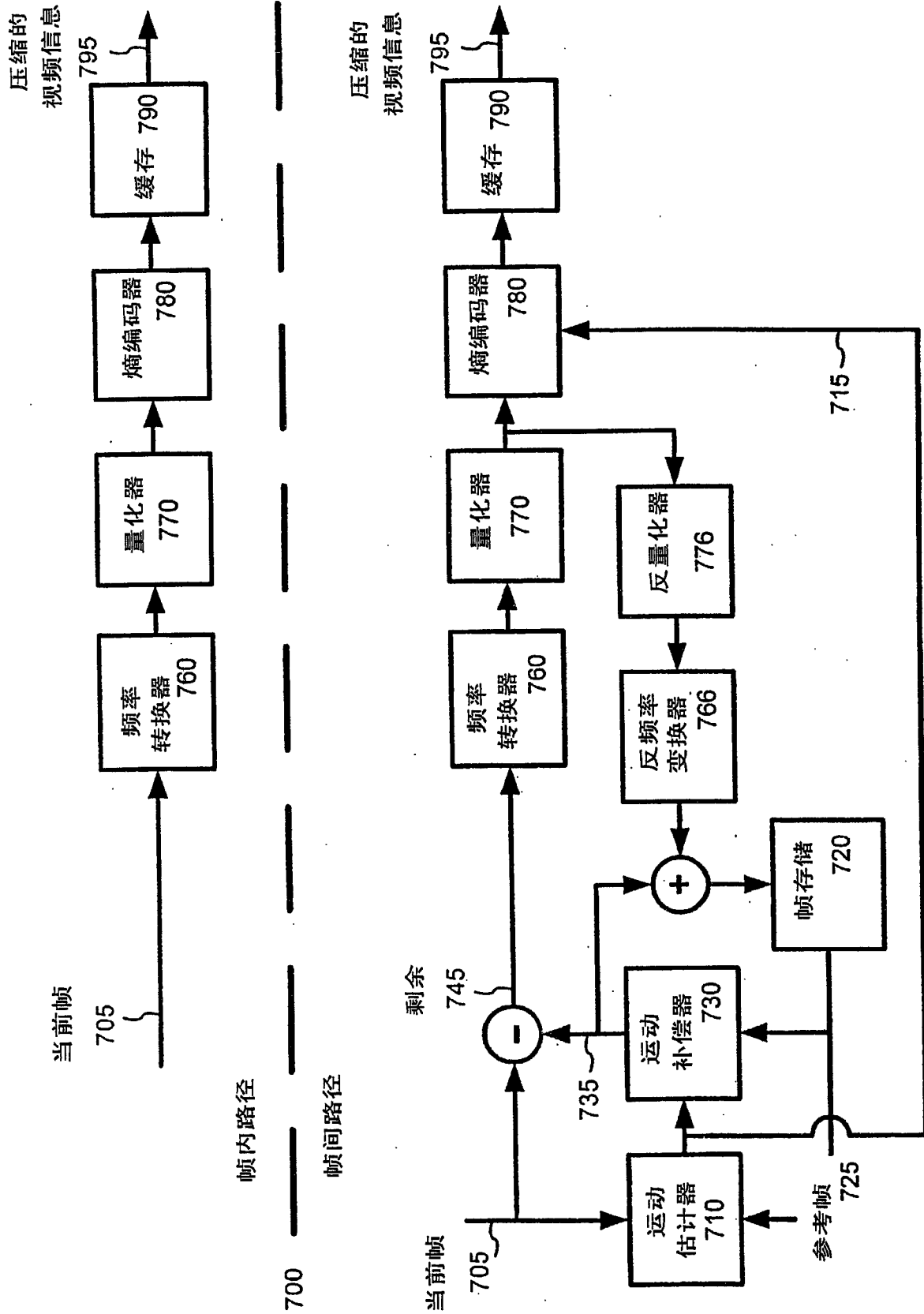


图 7

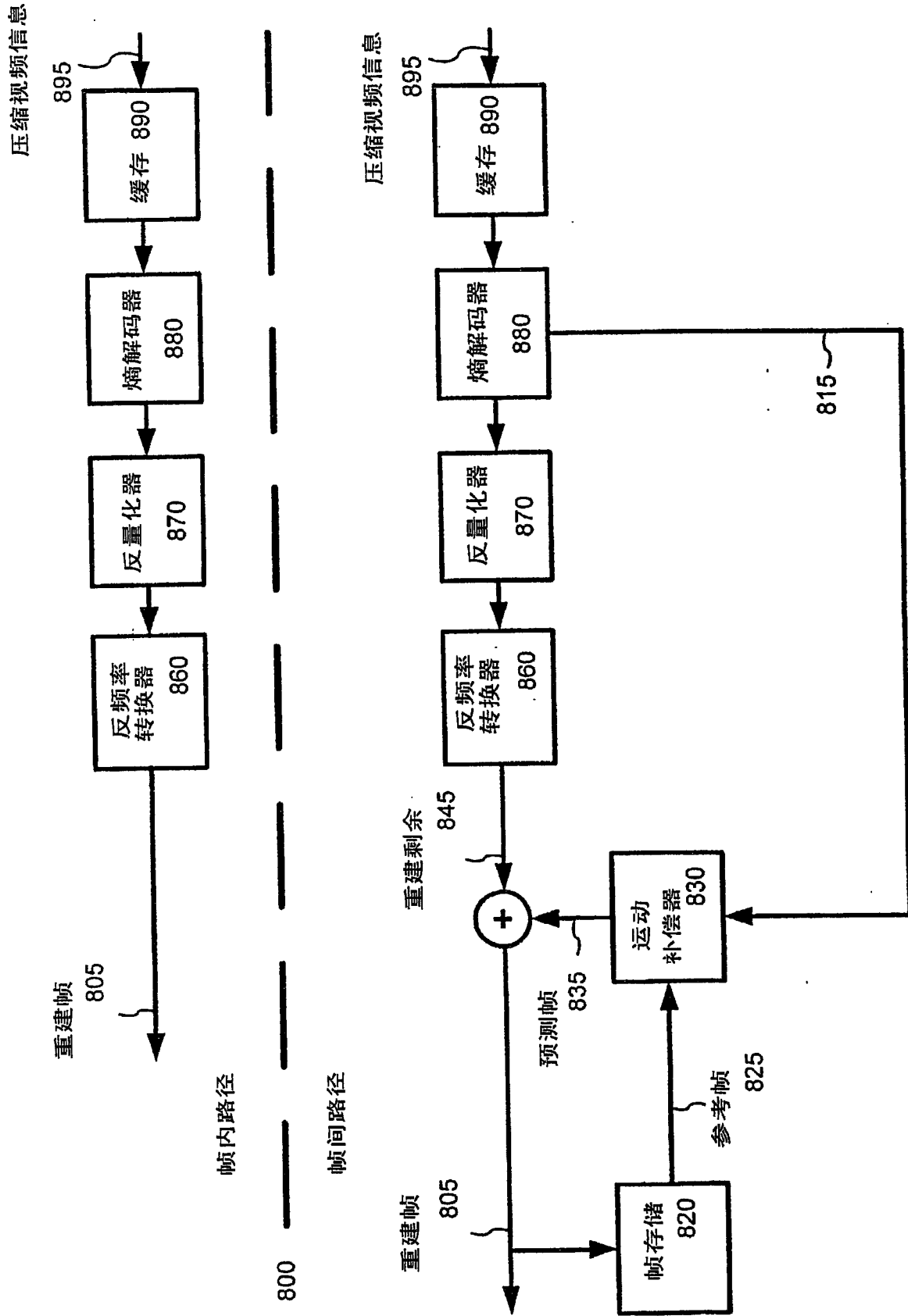


图 8

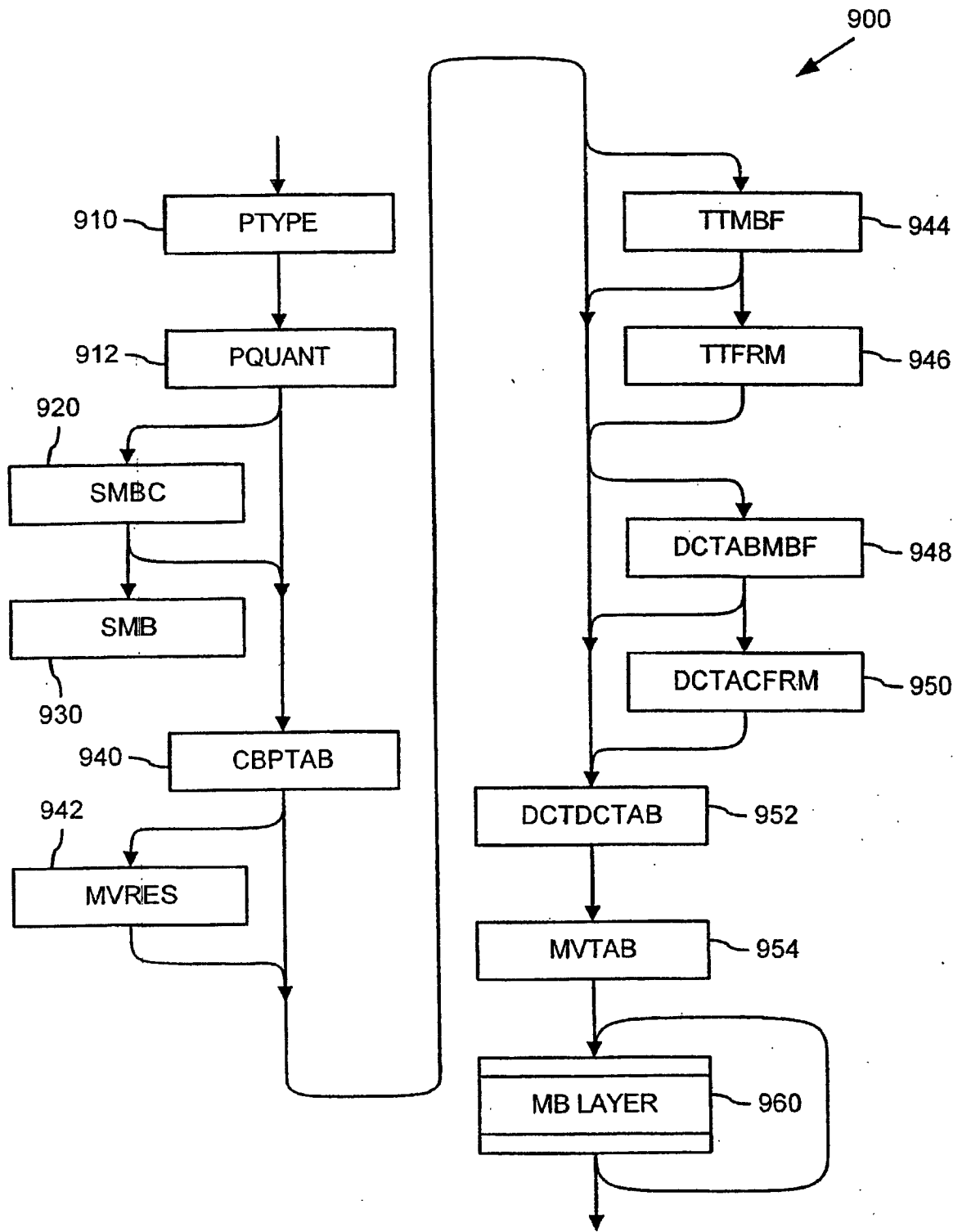


图 9

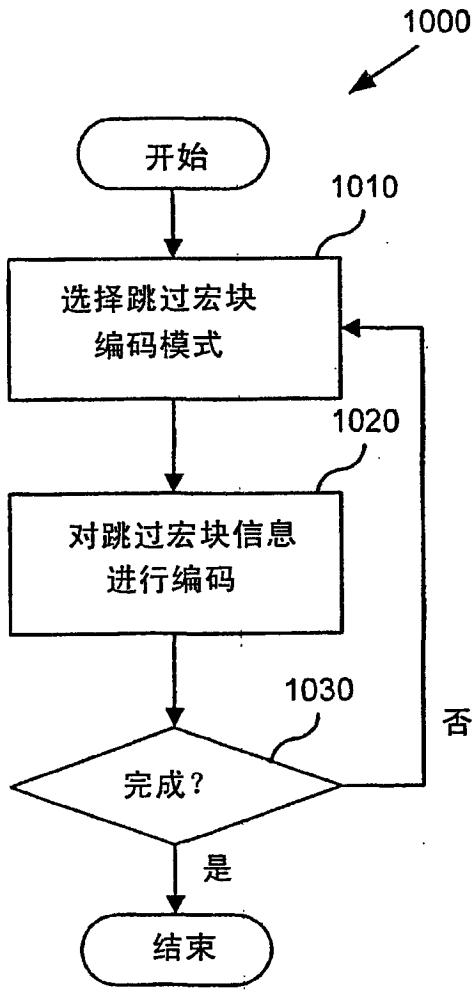


图 10

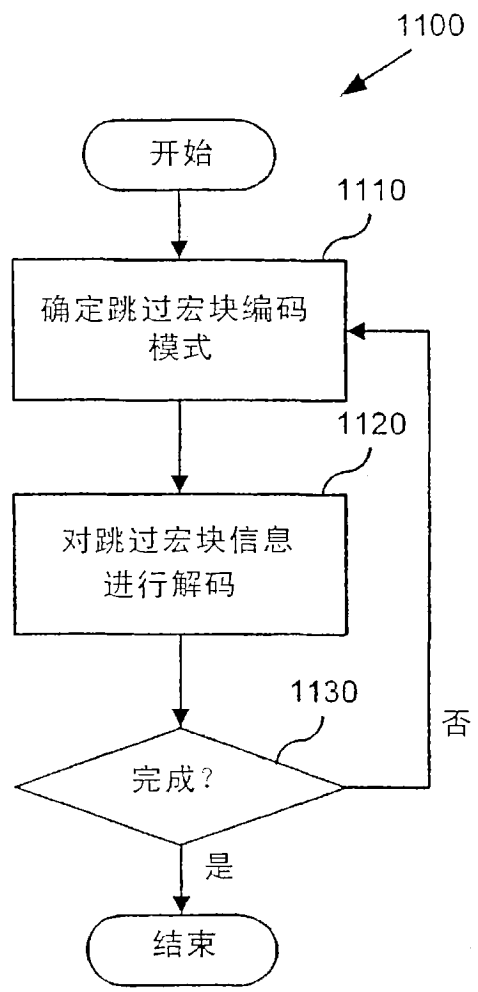


图 11

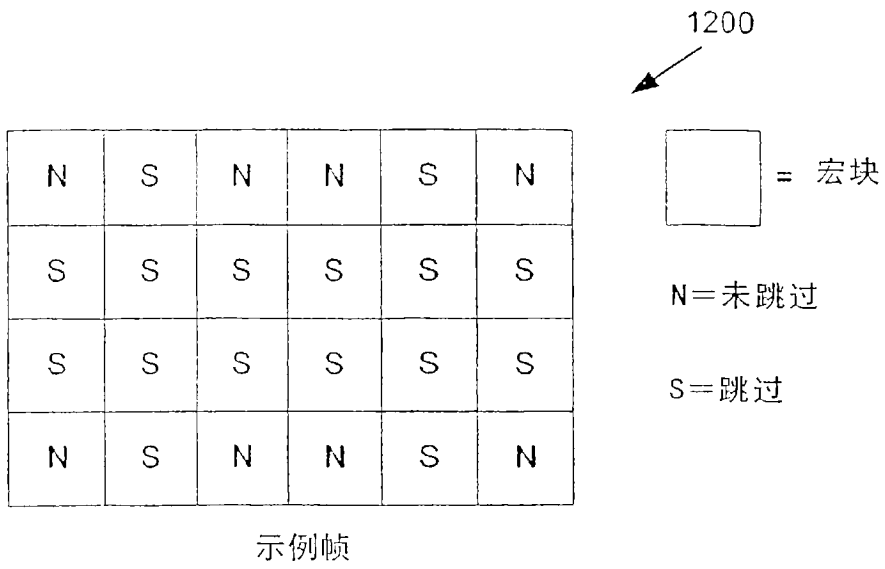


图 12

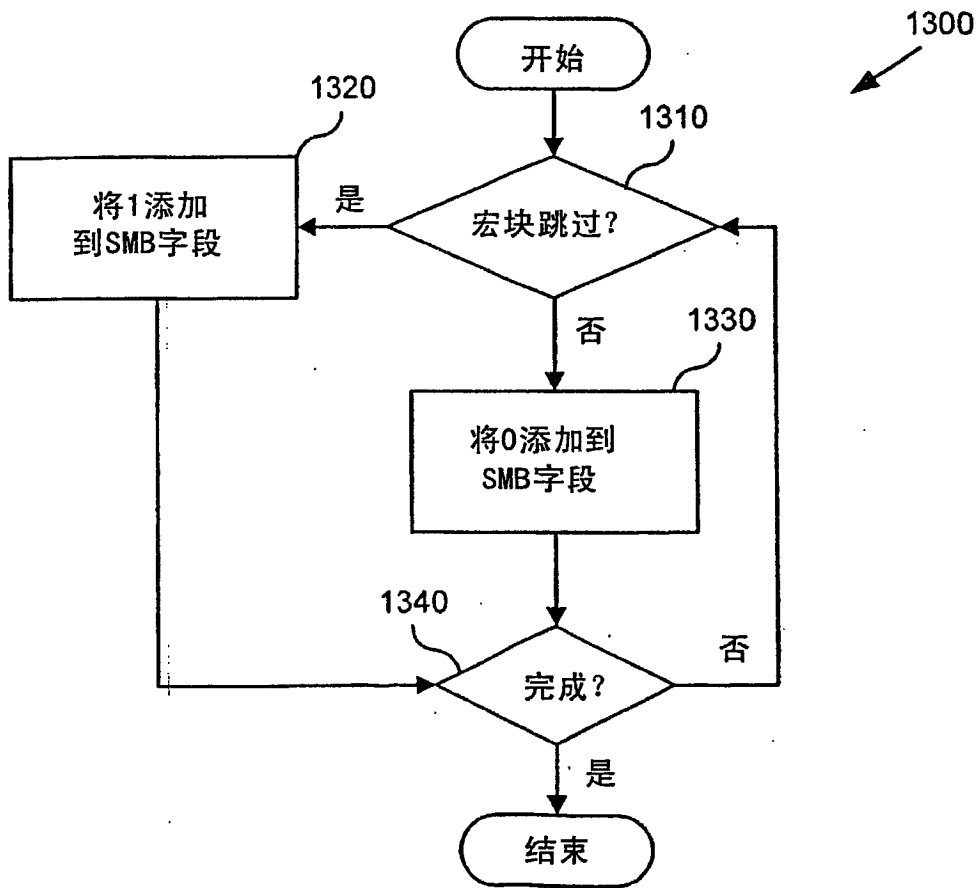


图 13

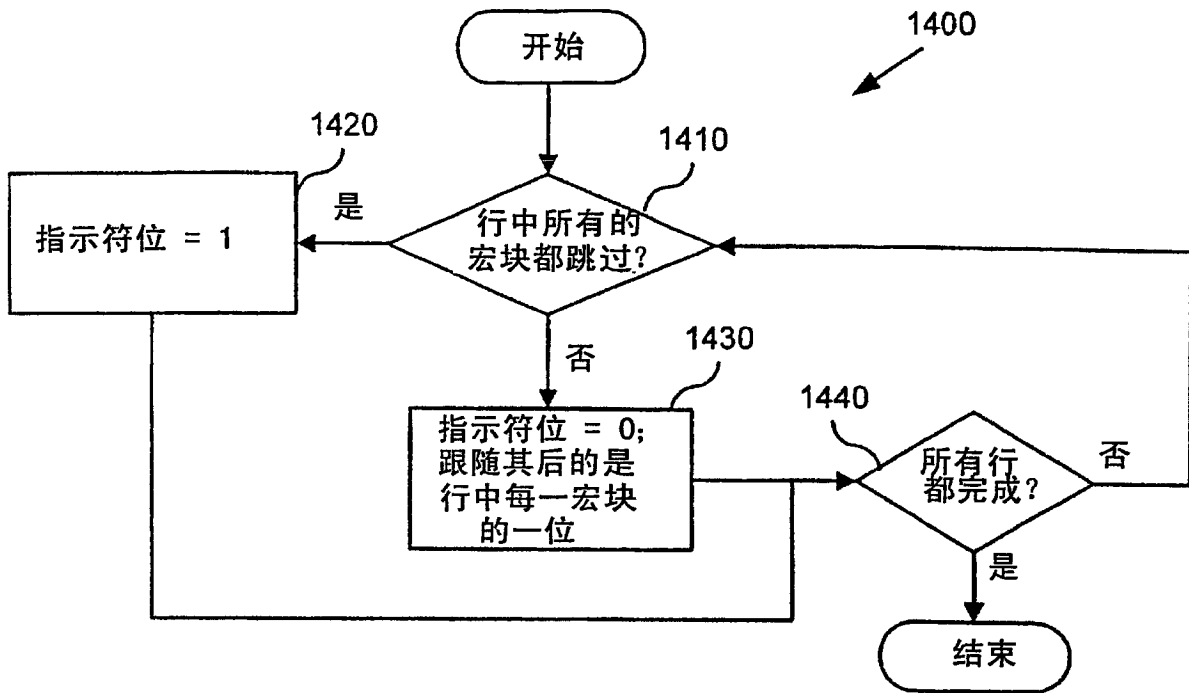


图 14

```

    for (mbrow = 0; mbrow < NumMBRows; mbrow++)
    {
        if (get_bits(1) == 1)
            ## All the macroblocks in this macroblock row are skipped
        else
        {
            ## At least one macroblock in this row is not skipped
            ## Get coded status of each macroblock in the row starting from left-to-right
            for (mb = 0; mb < NumMBsPerRow; mb++)
                k = get_bits(1); ## k equals status of macroblock
                ## k == 0 if not skipped, k == 1 if skipped
        }
    }
    
```

图 15

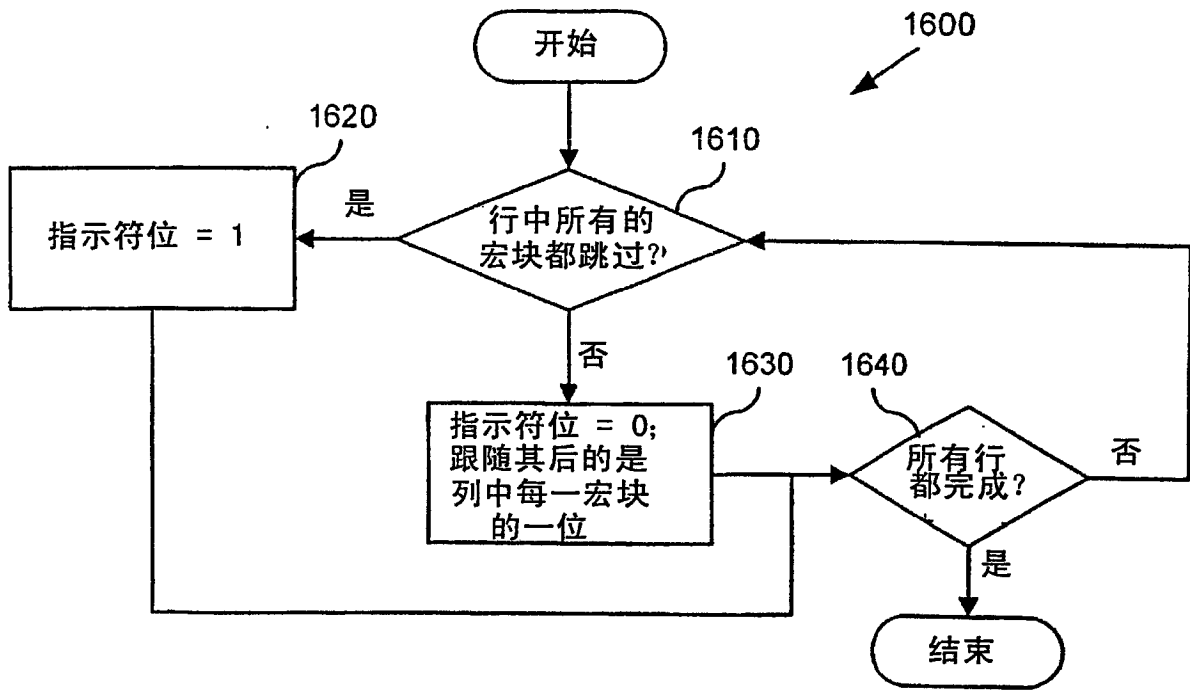


图 16

```

    for (mbcol = 0; mbcol < NumMBColumns; mbcol++)
    {
        if (get_bits(1) == 1)
            ## All the macroblocks in this macroblock column are skipped
        else
        {
            ## At least one macroblock in this column is not skipped.
            ## Get coded status of each macroblock in column starting from top-to-bottom
            for (mb = 0; mb < NumMBsPerColumn; mb++)
                k = get_bits(1); ## k equals status of macroblock
                ## k == 0 if not skipped, k == 1 if skipped
        }
    }
    
```

图 17

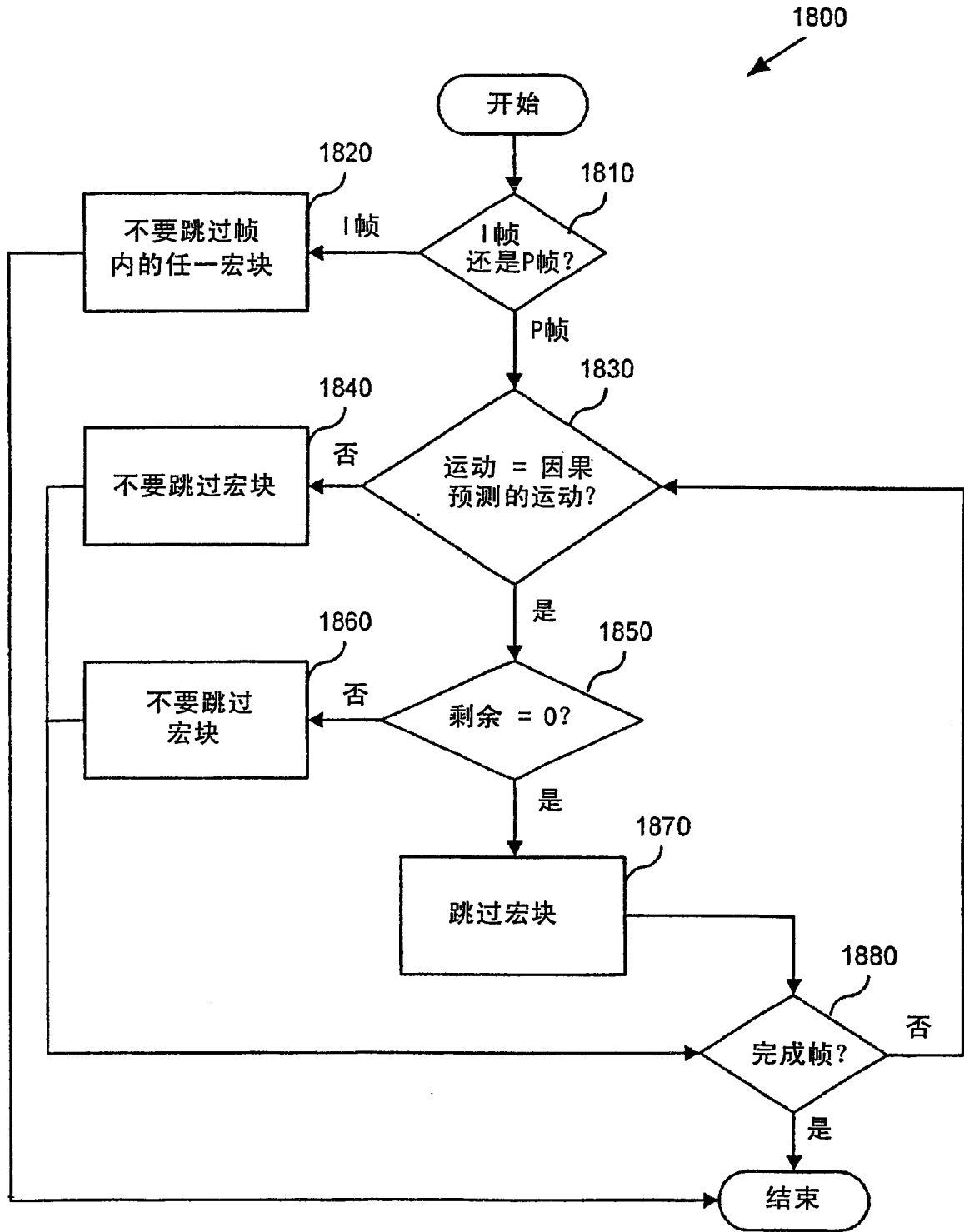


图 18

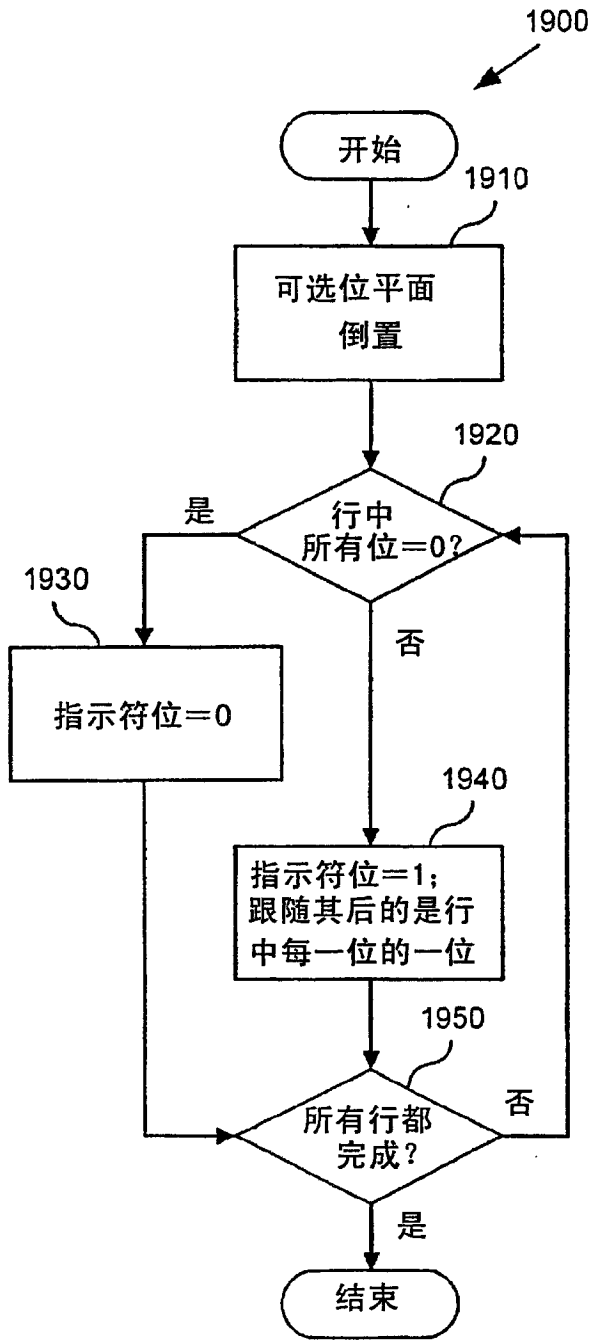


图 19

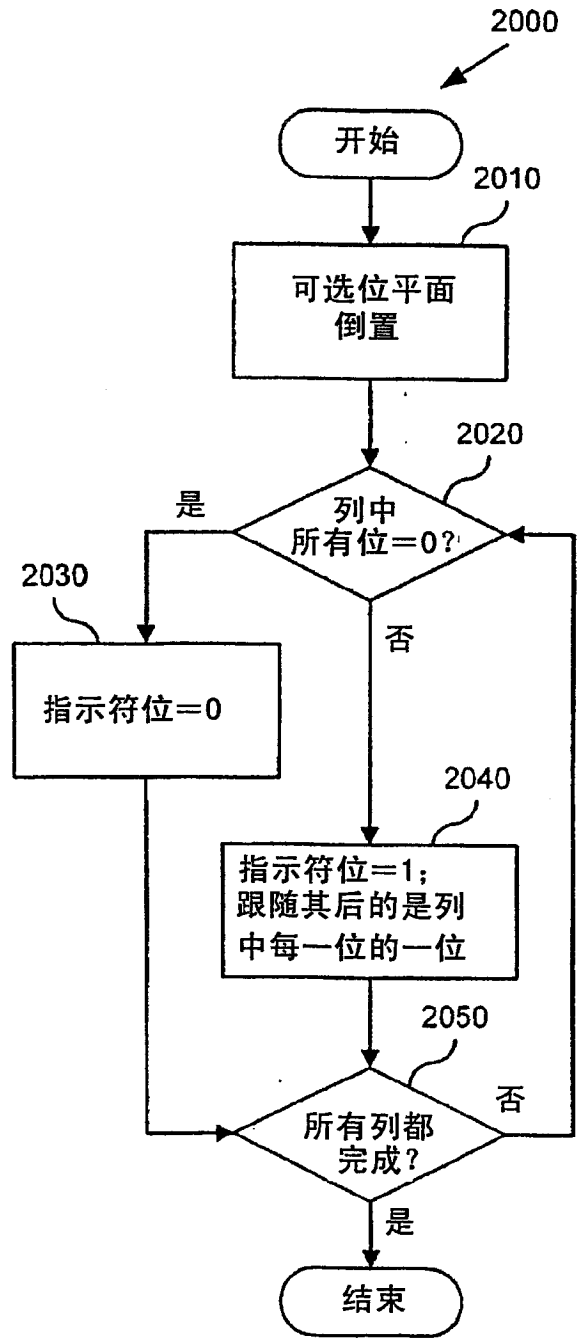


图 20

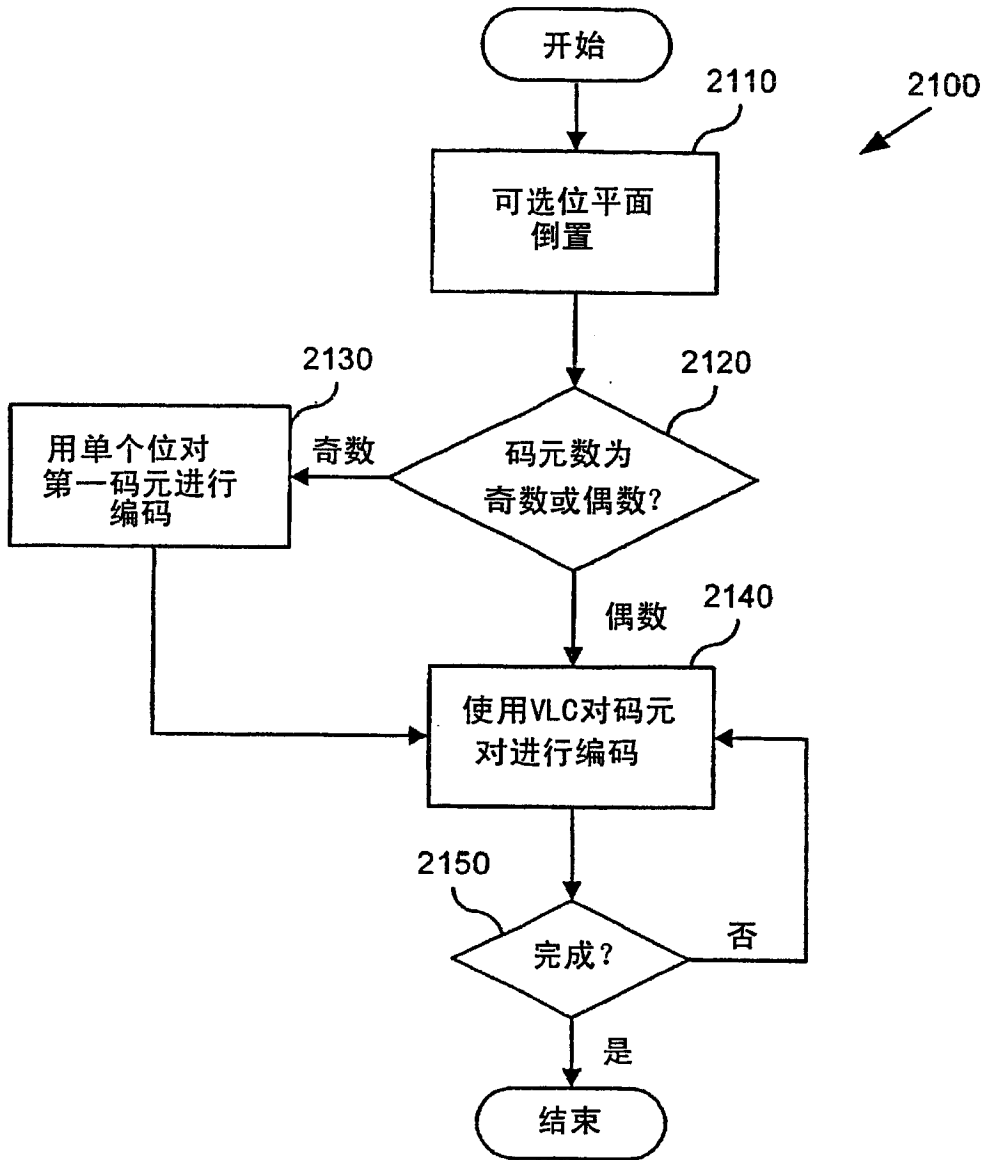


图 21

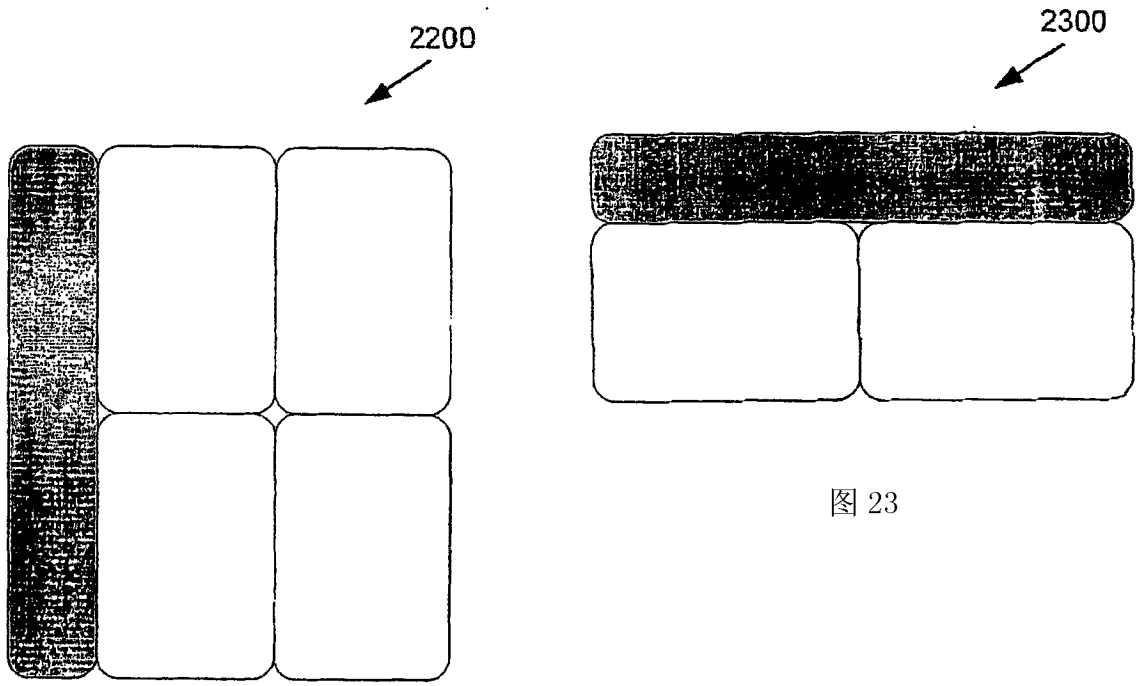


图 22

图 23

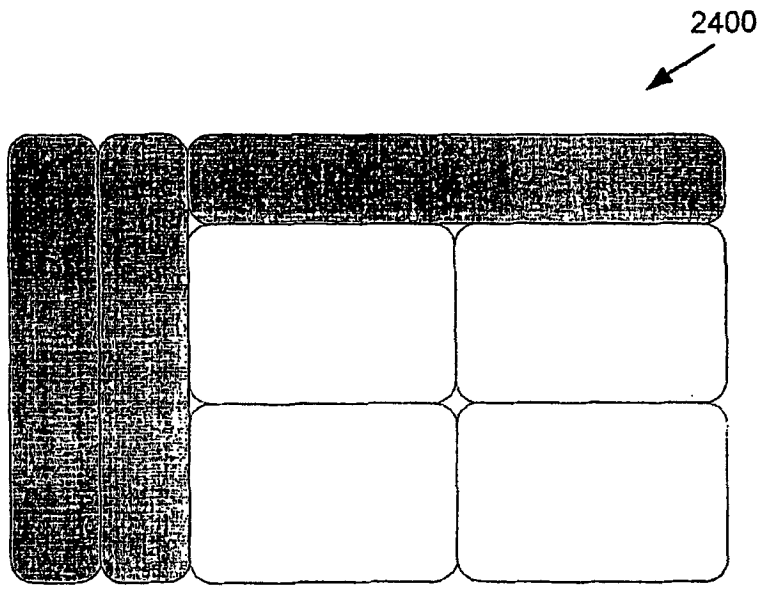


图 24

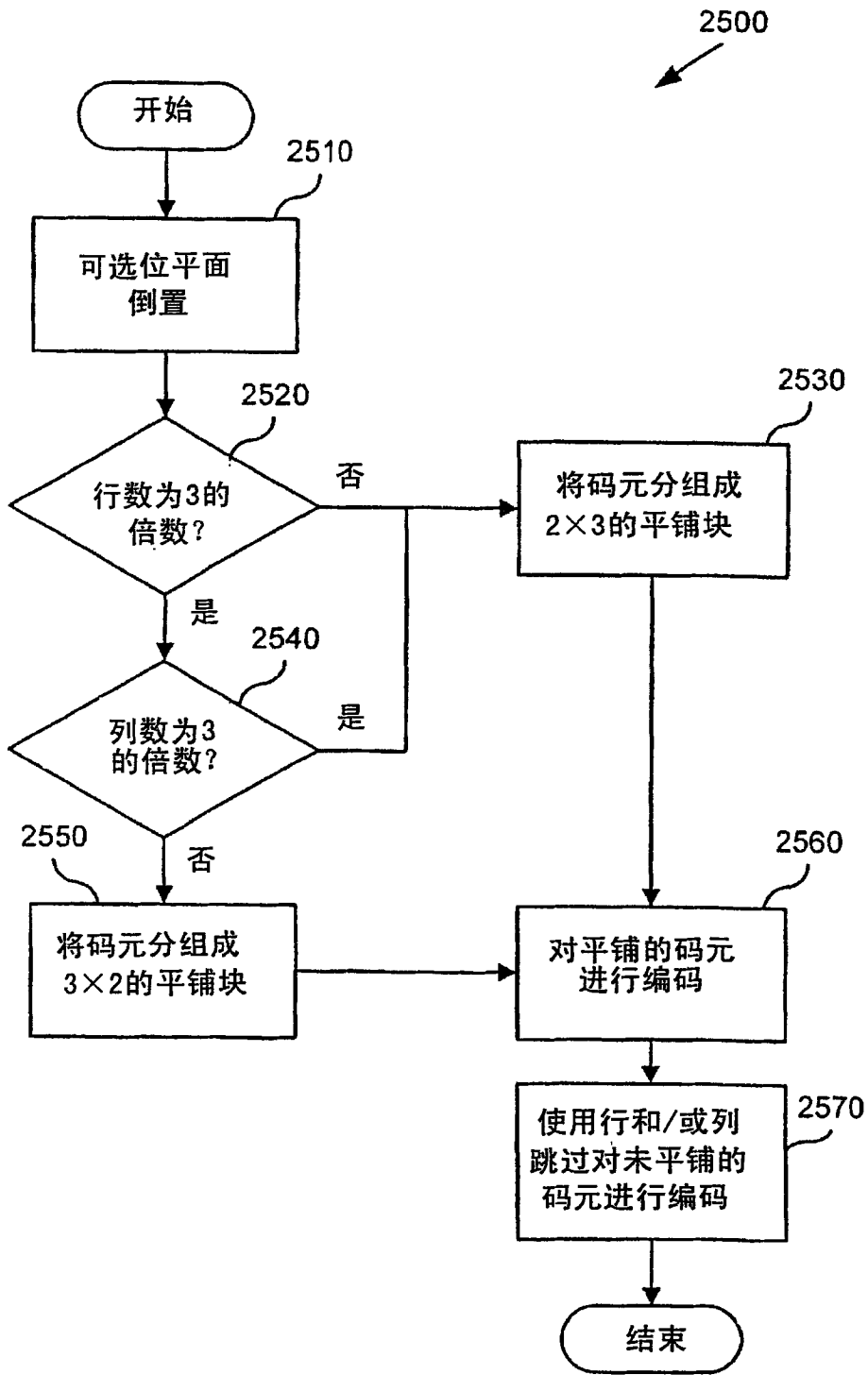


图 25

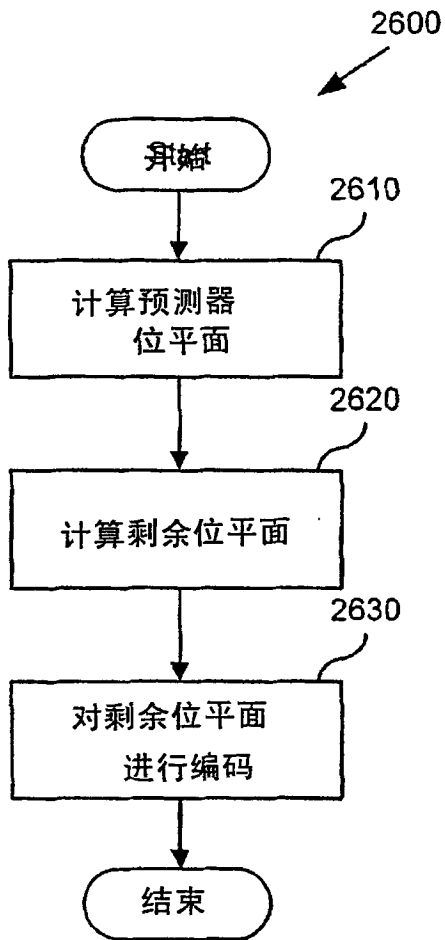


图 26

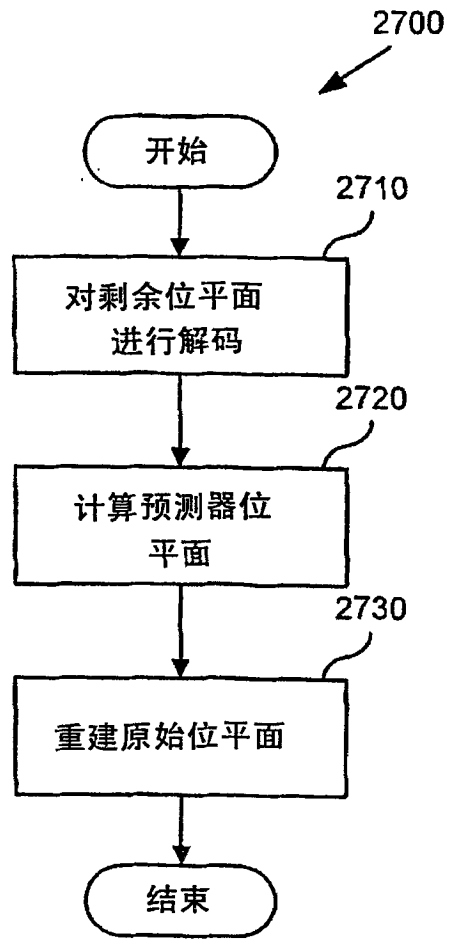


图 27

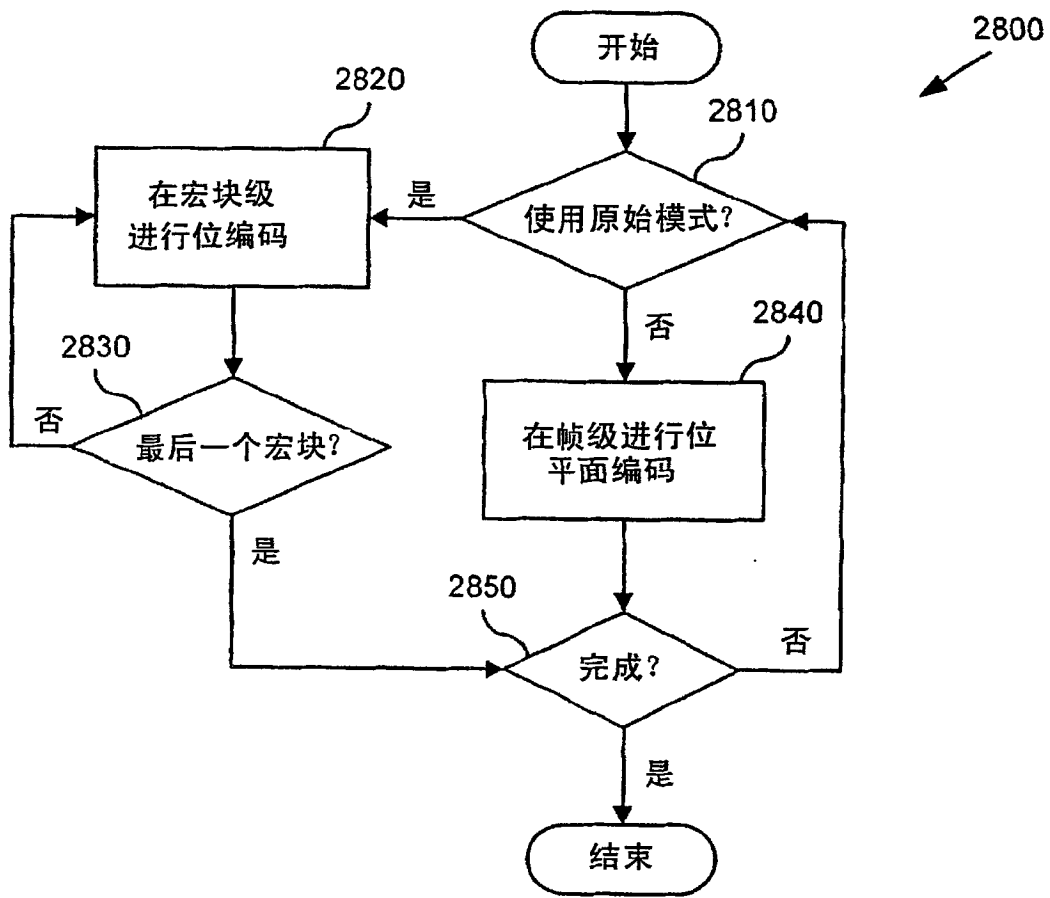


图 28