(19)中华人民共和国国家知识产权局

(12)发明专利

(10)授权公告号 CN 101924934 B
(45)授权公告日 2012.12.05

(21)申请号 201010169982.9
(22)申请日 2004.09.03
(30)优先权数据
60/501,081 2003.09.07 US
10/882,135 2004.06.29 US
(62)分案原申请数据
200480024621.8 2004.09.03
(73)专利权人 微软公司
地址 美国华盛顿州
(72)发明人 K·慕克吉 T·W·赫尔科比
(74)专利代理机构 上海专利商标事务所有限公司 31100
代理人 钱静芳
(51)Int.Cl.
H04N 7/26(2006.01)
H04N 7/46(2006.01)
H04N 7/50(2006.01)
(56)对比文件
CN 1201332 A,1998.12.09,
CN 1244001 A,2000.02.09,
CN 1226781 A,1999.08.25,
CN 1143884 A,1997.02.26,
审查员 马丽莉

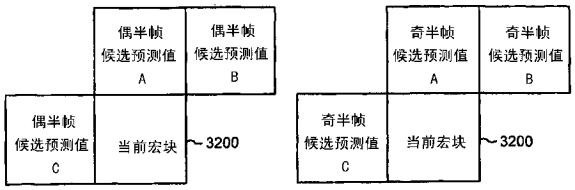
权利要求书 2 页 说明书 54 页 附图 69 页

(54)发明名称

视频编码和解码方法

(57)摘要

对于隔行扫描B-半帧或隔行扫描B-帧,正向运动向量使用来自正向运动向量缓冲区的正向运动向量来预测,而反向运动向量使用来自反向运动向量缓冲区的反向运动向量来预测。结果的运动向量被添加到相应的缓冲区中。运动向量缓冲区中的空穴可用所估计的运动向量值填充。在隔行扫描B-帧的半帧编码宏块中的半帧之间切换预测模式。对于隔行扫描B-半帧或隔行扫描B-帧,计算直接模式运动向量。对于隔行扫描B-半帧或隔行扫描B-帧,使用4MV编码。编码器/解码器使用"自参考"B-帧。编码器发送二进制信息,表示对隔行扫描B-半帧的一个或多个宏块的预测模式是正向还是非正向的。编码器/解码器使用帧内编码的B-半帧("BI-半帧")。



CN 101924934 B

1. 一种在实现视频解码器的计算设备中解码视频的方法，所述计算设备包括处理器和存储器，所述方法包括：

用所述实现视频解码器的计算设备，对当前隔行扫描的双向预测半帧中的当前直接模式宏块进行解码，包括：

如果时间上在将来的隔行扫描预测锚半帧中的共处宏块是使用四运动向量来编码的：

确定所述共处宏块的最多达四个运动向量；以及

计算要在对所述当前直接模式宏块的直接模式缩放操作中使用的运动向量，包括通过以下步骤，在计算所述运动向量时偏爱所述共处宏块的最多达四个运动向量的主极性：

确定所述共处宏块的最多达四个运动向量中的更多运动向量是引用相同极性参考半帧还是相反极性参考半帧；

如果所述共处宏块的最多达四个运动向量中的更多运动向量引用所述相反极性参考半帧，使用所述共处宏块的最多达四个运动向量中引用所述相反极性参考半帧的那些运动向量来计算要在直接模式缩放操作中使用的运动向量；以及

否则，使用所述共处宏块的最多达四个运动向量中引用所述相同极性参考半帧的那些运动向量来计算要在直接模式缩放操作中使用的运动向量；以及

用所述实现视频解码器的计算设备，使用所解码的当前直接模式宏块来重构所述当前隔行扫描的双向预测半帧。

2. 如权利要求 1 所述的方法，所述将来的隔行扫描预测锚半帧具有与带有所述当前直接模式宏块的当前隔行扫描双向预测半帧相同的极性。

3. 如权利要求 1 所述的方法，其特征在于，所述计算在直接模式缩放操作中使用的运动向量还包括：

对所述共处宏块的最多达四个运动向量中的相同极性运动向量计数，并对相反极性运动向量计数。

4. 如权利要求 1 所述的方法，其特征在于，所述对当前直接模式宏块进行解码还包括：

使用所计算的运动向量来导出正向运动向量和反向运动向量以在对所述当前直接模式宏块的运动补偿中使用；以及

使用所述正向运动向量和所述反向运动向量来对所述当前直接模式宏块进行运动补偿。

5. 一种在实现视频编码器的计算设备中编码视频的方法，所述计算设备包括处理器和存储器，所述方法包括：

用所述实现视频编码器的计算设备，对当前隔行扫描 B- 半帧进行编码以产生已编码视频信息，其中所述编码包括计算要在对所述当前隔行扫描 B- 半帧中的当前宏块的直接模式缩放中使用的运动向量，包括：

如果时间上在将来的隔行扫描锚 P- 半帧的共处宏块是具有最多达四个运动向量的 4MV 宏块：

从所述共处宏块的最多达四个运动向量中计算主极性运动向量，作为在所述直接模式缩放中使用的运动向量：

确定所述共处宏块的最多达四个运动向量中的更多运动向量是引用相同极性参考半

帧还是相反极性参考半帧；

如果所述共处宏块的最多达四个运动向量中的更多运动向量引用所述相反极性参考半帧，使用所述共处宏块的最多达四个运动向量中引用所述相反极性参考半帧的那些运动向量来计算所述主极性运动向量；以及

否则，使用所述共处宏块的最多达四个运动向量中引用所述相同极性参考半帧的那些运动向量来计算所述主极性运动向量；以及

用所述实现视频编码器的计算设备，在比特流中输出所述已编码视频信息。

6. 如权利要求 5 所述的方法，其特征在于，所述计算主极性运动向量包括：

对所述最多达四个运动向量中的相同极性运动向量计数，以及对相反极性运动向量计数。

7. 如权利要求 5 所述的方法，其特征在于，所述计算主极性运动向量包括对两个运动向量求平均。

8. 如权利要求 5 所述的方法，其特征在于，所述计算主极性运动向量包括计算三个或四个运动向量的中值。

9. 如权利要求 5 所述的方法，其特征在于，所述计算主极性运动向量包括在所述最多达四个运动向量中选择单个可用运动向量。

10. 如权利要求 5 所述的方法，其特征在于，包括在所述当前隔行扫描 B- 半帧的当前宏块的处理期间执行对所述主极性运动向量的计算。

11. 如权利要求 5 所述的方法，其特征在于，包括在所述当前隔行扫描 B- 半帧的当前宏块的处理开始之前在所述共处宏块的处理期间执行对所述主极性运动向量的计算。

12. 如权利要求 5 所述的方法，其特征在于，所述将来的隔行扫描锚 P- 半帧具有与所述当前隔行扫描 B- 半帧相同的极性。

13. 如权利要求 5 所述的方法，其特征在于，所述编码还包括：

使用所述主极性运动向量来计算正向运动向量和反向运动向量以在对所述当前宏块的运动补偿中使用；以及

使用所述正向运动向量和所述反向运动向量来对所述当前宏块进行运动补偿。

视频编码和解码方法

[0001] 本申请是申请日为2004年9月3日、国际申请号为PCT/US2004/029000、中国国家申请号为200480024621.8、发明名称为“隔行扫描视频的高级双向预测编码”的专利申请的分案申请。

[0002] 相关申请

[0003] 本申请要求对申请号为60/501,081题为“视频编码和解码工具和技术”(Video Encoding and Decoding Tools and Techniques)并于2003年9月7日提交的美国临时专利申请的权利,该申请通过参考结合于此。

[0004] 以下共同待批的美国专利申请涉及本申请,并通过参考结合于此:1)序列号为10/622,378题为“视频帧的高级双向预测编码”(Advanced Bi-Directional Predictive Coding of Video Frames)并于2003年7月18日提交的美国专利申请;2)序列号为10/622,284题为“帧内和帧间隔行扫描编码和解码”(Intraframe and Interframe Interlace Coding and Decoding)并于2003年7月18日提交的美国专利申请;3)序列号为10/622,841题为“运动向量信息的编码”(Coding of Motion Vector Information)并于2003年7月18日提交的美国专利申请;4)序列号为10/857,473题为“预测用于正向预测的隔行扫描视频帧的各个半帧的运动向量”(Predicting Motion Vectors for Fields of Forward-predicted Interlaced Video Frames)并于2004年5月27日提交的美国专利申请。

技术领域

[0005] 描述了用于隔行扫描视频编码和解码的技术和工具。例如,视频编码器对隔行扫描视频中的双向预测宏块进行编码。

背景技术

[0006] 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包括每秒15或30个图片。每个图片可包括数万或数十万个像素(也称为pel)。每个像素表示图片的小元素。在原始形式中,计算机通常用24个比特或以上来表示像素。因而,典型的原始数字视频序列的每秒比特数或比特率可以是5百万比特/秒或以上。

[0007] 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。为此,工程师们使用压缩(也称为译码或编码)来降低数字视频的比特率。压缩可以是无损的,其中视频的质量未受损,但比特率的降低受视频复杂性的限制。或者,压缩可以是有损的,其中视频的质量受损,但可获得的比特率降幅更大。解压缩是压缩的反向过程。

[0008] 一般而言,视频压缩技术包括“帧内”压缩和“帧间”或预测压缩。帧内压缩技术压缩通常称为I-帧或关键帧的各个图片。帧间压缩技术参考前面和/或后续的各帧来压缩各个帧,而经帧间压缩的帧通常称为预测帧、P-帧、或B-帧。

[0009] I.Windows Media Video版本8和9中的帧间压缩

[0010] 微软公司的Windows Media Video版本8[“WMV8”]包括视频编码器和视频解码

器。WMV8 编码器使用帧内压缩和帧间压缩，且 WMV8 解码器使用帧内解压缩和帧间解压缩。Windows Media Video 版本 9[“WMV9”]则将类似的体系结构用于许多操作。

[0011] WMV8 编码器中的帧内压缩使用基于块的运动补偿预测编码，然后使用残差的变换编码。图 1 和 2 示出 WMV8 编码器中预测帧的基于块的帧间压缩。特别地，图 1 示出对预测帧 110 的运动估计，而图 2 示出对预测帧的运动补偿块的预测残差压缩。

[0012] 例如，在图 1 中，WMV8 编码器计算用于预测帧 110 中的宏块 115 的运动向量。为了计算该运动向量，编码器在参考帧 130 的搜索区域 135 中进行搜索。在搜索区域 135 内，编码器对来自预测帧 110 的宏块 115 和各个候选宏块作比较，以便于寻找是好匹配的候选宏块。编码器输出指定匹配宏块的（经熵编码的）运动向量。

[0013] 因为运动向量值常常相关于空间上位于四周的运动向量的值，所以用来传送运动向量信息的数据的压缩可通过从相邻宏块中选择运动向量预测值，并使用该预测值来预测当前宏块的运动向量来获得。编码器可编码运动向量和预测值之前的差值。在通过将差值添加到预测值中重构运动向量之后，解码器使用该运动向量来用来自参考帧 130 的信息计算用于宏块 115 的预测宏块，该参考帧是在编码器和解码器上可用的先前重构帧。预测很少是完美的，所以编码器常常编码预测宏块和宏块 115 本身之间的像素差异（也称为误差或残差块）的各个块。

[0014] 图 2 示出 WMV8 编码器中误差块 235 的计算和编码的一个示例。该误差块 235 是预测块 215 和原始当前块 225 之间的差异。编码器将离散余弦变换 [“DCT”]240 应用于误差块 235，得到系数的 8x8 块 245。然后编码器量化 (250)DCT 系数，得到经量化的 DCT 系数的 8x8 块 255。编码器将 8x8 块 255 扫描 (260) 成一维数组 265，从而系数通常从最低频率到最高频率进行排序。编码器使用运行长度编码 270 的变体来熵编码经扫描的系数。编码器从一个或多个 run/level/last 表格 275 中选择熵码，并输出该熵码。

[0015] 图 3 示出对经帧间编码块的相应解码过程 300 的一个示例。从图 3 总地看来，解码器使用具有一个或多个 run/level/last 表格 315 的可变长度解码 310 和运行长度解码 320 来解码 (310、320) 表示预测残差的经熵编码的信息。解码器将存储了经熵编码信息的一维数组 325 逆扫描 (330) 成二维块 335。解码器逆量化并逆离散余弦变换（一起编号为 340）该数据，产生经重构的误差块 345。在独立的运动补偿路径中，解码器使用运动向量信息 355 来计算预测块 365，用于相对参考帧的位移。解码器将预测块 365 与经重构的误差块 345 组合 (370) 在一起，以形成重构块 375。

[0016] 原始和重构帧之间的改变量是失真，且编码该帧所需的比特数表示该帧的速率。失真量大致与速率成反比。

[0017] II. 隔行扫描视频和逐行扫描视频

[0018] 视频帧包含视频信号的各行空间信息。对于逐行扫描视频，这些行包含从一时刻开始并继续后续行直到帧的底部的样本。逐行扫描 I 帧是帧内编码的逐行扫描视频帧。逐行扫描 P 帧是使用正向预测编码的逐行扫描视频帧，而逐行扫描 B 帧是使用双向预测编码的逐行扫描视频帧。

[0019] 典型的隔行扫描视频帧由在不同时间开始扫描的两个半帧组成。例如，参照图 4，隔行扫描的视频帧 400 包括上半帧 410 和下半帧 420。通常，偶数编号行（上半帧）在一时间（例如时间 t）开始扫描而奇数编号行（下半帧）在不同（通常为后续）时间（例如时

间 t+1）开始扫描。该定时可在隔行扫描视频帧的区域中创建锯齿状特征，其中因为两个半帧在不同时间开始扫描而呈现运动。因此，各个隔行扫描的视频帧可根据半帧结构来重新排列，其中奇数行组合在一个半帧中而偶数行组合在另一个半帧中。称为半帧编码的该排列对在高运动图片中减少这种锯齿边缘人工效应是有用的。另一方面，在静态区域中，隔行扫描视频帧中的图像细节无需这种重新排列就可更为有效地保留。因此，帧编码常用于静态或低运动的隔行扫描视频帧，其中原始的交替半帧行排列得以保留。

[0020] 典型的逐行扫描视频帧由具有非交替行的一内容帧组成。与隔行扫描视频相比，逐行扫描视频不将视频帧分成各个半帧，并且整个帧从某个时间开始进行从左到右、从上到下的扫描。

[0021] III. 先前 WMV 编码器和解码器中的 P- 帧编码和解码

[0022] 先前的 WMV 编码器和解码器使用 P- 帧中的逐行扫描及隔行扫描的编码和解码。在隔行扫描和逐行扫描的 P- 帧中，运动向量通过计算运动向量和运动向量预测值之间的差值来在编码器中编码，其中运动向量预测值基于相邻的运动向量进行计算。并且，在解码器中，运动向量通过将运动向量差值添加到运动向量预测值来重新构建，其中运动向量预测值再次基于相邻的运动向量进行计算（这次是在解码器中）。当前宏块的预测值或当前宏块的半帧基于候选预测值来选择，而运动向量差值基于该预测值来计算。通过将运动向量差值添加到编码器侧或解码器侧的选定运动向量预测值，可重构运动向量。通常，亮度运动向量从经编码的运动信息中重构，而色度运动向量从经重构的亮度运动向量中导出。

[0023] A. 逐行扫描 P- 帧的编码和解码

[0024] 例如，在先前的 WMV 编码器和解码器中，逐行扫描 P- 帧可包含以一运动向量（1MV）模式或四运动向量（4MV）模式编码的宏块，或被跳过的宏块，其中决定通常在逐个宏块的基础上作出。仅具有 1MV 宏块（以及可能被跳过的宏块）的 P- 帧被称为 1MVP- 帧，而都具有 1MV 和 4MV 宏块（以及可能被跳过的宏块）的 P- 帧被称为混合 MV 的 P- 帧。一个运动向量与每个 1MV 宏块相关联，而四个运动向量与每个 4MV 宏块相关联（每个块一个）。

[0025] 图 5A 和 5B 是示出被视作用于 1MV 逐行扫描 P- 帧中宏块的候选运动向量预测值的宏块的位置的示图。候选预测值从左边、上方和右上方宏块中取得，除了宏块是行中最后一个宏块的情形。在该情形中，Predicator（预测值）B 从左上方宏块而不是从右上方中取得。对于帧为一个宏块宽的特定情形，预测值总是预测值 A（顶部预测值）。当因为宏块在首行中从而预测值 A 在界限之外时，预测值为预测值 C。各种其它规则解决其它特定情形，诸如帧内编码的预测值。

[0026] 图 6A-10 示出被视作用于混合 MV 帧中的 1MV 或 4MV 宏块的运动向量的至多 3 个候选运动向量的块或宏块的位置。在以下附图中，较大的正方形是宏块边界而较小的正方形是块边界。对于帧为一个宏块宽的特定情形，预测值总是 PredictorA（顶部预测值）。各种其它规则解决其它特定情形，诸如用于首行 4MV 宏块的首行块、首行 1MV 宏块、以及帧内编码预测值。

[0027] 图 6A 和 6B 是示出被视作用于混合 MV 帧中的 1MV 当前宏块的候选运动向量预测值的块的位置的示图。相邻的各个宏块可以是 1MV 或 4MV 宏块。图 6A 和 6B 示出假设邻居都是 4MV 的候选运动向量的位置（即预测值 A 是当前宏块上面的宏块中块 2 的运动向量，而预测值 C 是当前宏块左侧紧邻的宏块中块 1 的运动向量）。如果邻居的任一个是 1MV 宏

块，则图 5A 和 5B 中示出的运动向量预测值被视为整个宏块的运动向量预测值。如图 6B 所示，如果宏块是该行中的最后一个宏块，则预测值 B 来自左上方宏块的块 3 而不像其它情形一样来自右上方宏块的块 2。

[0028] 图 7A-10 示出被视作用于 4MV 宏块中 4 个亮度块的每一个的候选运动向量预测值的块的位置。图 7A 和 7B 是示出被视作用于位置 0 上一个块的候选运动向量预测值的块的位置的示图；图 8A 和 8B 是示出被视作用于位置 1 上一个块的候选运动向量预测值的块的位置的示图；图 9 是示出被视作用于位置 2 上一个块的候选运动向量预测值的块的位置的示图；而图 10 是示出被视作用于位置 3 上一个块的候选运动向量预测值的块的位置的示图。再一次，如果邻居是 1MV 宏块，则该宏块的运动向量预测值用于该宏块的各个块。

[0029] 对于宏块是行中第一宏块的情形，块 0 的预测值 B 与该行中剩余宏块的块 0 进行不同的处理（参见图 7A 和 7B）。在该情形中，预测值 B 从当前宏块上面紧邻宏块的块 3 中取得，而不像其它情形一样从当前宏块左上方的宏块的块 3 中取得。类似地，对于宏块是行中最后一个宏块的情形，对块 1 的预测值 B 进行不同的处理（参见图 8A 和 8B）。在该情形中，预测值从当前宏块上面紧邻宏块的块 2 中取得，而不像其它情形一样从当前宏块右上方的宏块的块 2 中取得。一般而言，如果该宏块在第一宏块列中，则块 0 和 2 的预测值 C 被设置为等于 0。

[0030] B. 先前 WMV 编码器和解码器中隔行扫描 P- 帧的编码和解码

[0031] 先前 WMV 编码器和解码器将 4 ： 1 ： 1 宏块格式用于隔行扫描的 P- 帧，该宏块格式可包含用半帧模式或帧模式编码的宏块，或被跳过的宏块，其中决定通常在逐个宏块的基础上作出。两个运动向量关联于每个半帧编码的宏块（每个半帧一个运动向量），而一个运动向量关联于每个帧编码的宏块。编码器联合编码运动信息，包括水平和垂直运动向量差值分量，以及可能的其它信令信息。

[0032] 图 11 和 12A-B 示出分别用于先前 WMV 编码器和解码器的隔行扫描 P- 帧中的帧编码 4 ： 1 ： 1 宏块和半帧编码 4 ： 1 ： 1 宏块的运动向量预测的候选预测值的示例。图 11 示出用于隔行扫描 P- 帧的内部位置中当前帧编码 4 ： 1 ： 1 宏块的候选预测值 A、B 和 C（不是宏块行中第一或最后一个宏块、也不是首行）。预测值可从不同于那些标有 A、B 和 C 的不同候选方向中获得（例如在诸如当前宏块是行中第一宏块或最后一个宏块或在首行中的特定情形，因为对于这些情形得不到某些预测值）。对于当前的帧编码宏块，候选预测值取决于相邻宏块是半帧编码还是帧编码的来进行不同的计算。对于相邻的帧编码宏块，运动向量简便地取作候选预测值。对于相邻的半帧编码宏块，候选运动向量通过平均上半帧和下半帧运动向量来确定。

[0033] 图 12A-B 示出用于半帧内部位置中半帧编码的 4 ： 1 ： 1 宏块内的当前半帧的候选预测值 A、B 和 C。在图 12A 中，当前半帧是下半帧，且相邻宏块中的下半帧运动向量被用作候选预测值。在图 12B 中，当前半帧是上半帧，且相邻宏块中的上半帧运动向量被用作候选预测值。因而，对于当前半帧编码宏块中的每个半帧，每个半帧的候选运动向量预测值的数量最多为 3，其中每个候选值来自与当前半帧相同的同一半帧类型（例如上半帧或下半帧）。再一次，各种特定情形（未示出）在当前宏块是行中第一宏块或最后一个宏块，或在首行中时应用，因为对于这些情形得不到某些预测值。

[0034] 为了从候选预测值集中选择一预测值，讨论中的先前 WMV 编码器和解码器使用不

同的选择算法，诸如三者中值算法或四者中值算法。三者中值预测的过程在图 13 的伪码 1300 中说明。四者中值预测的过程在图 14 的伪码 1400 中说明。

[0035] IV. 双向预测

[0036] 双向预测帧（或 B- 帧）使用来自源视频的两个帧作为参考（或锚）帧，而不是在 P- 帧中使用的一个锚。在典型 B- 帧的锚帧中，一个锚帧来自时间上的过去而另一个锚帧来自时间上的将来。参照图 15，视频序列中的 B- 帧 1510 具有时间上在先的参考帧 1520 和时间上将来的参考帧 1530。使用 B- 帧提供了根据更大的比特率节约（例如出现某些类型的移动，诸如闭塞）而有效压缩的优点。具有 B- 帧的经编码比特流通常比没有 B- 帧的经编码比特流使用较少的比特，同时提供相似的视觉质量。B- 帧还提供在较小设备空间中使用时的更多选项和灵活性。例如，解码器可通过选择不解码或显示 B- 帧来容许空间和时间限制，因为 B- 帧通常不被用作参考帧。对视频序列中使用 B- 帧来使速率 - 失真改进的估计的范围为从 0 到 50％。

[0037] V. 先前 WMV 编码器和解码器中 B- 帧的编码和解码

[0038] 先前的 WMV 编码器和解码器使用 B- 帧。尽管正向预测帧（例如 P- 帧）中的宏块仅具有一个方向模式的预测（正向，从先前的 I- 或 P- 帧），但 B- 帧中的宏块可使用五种不同预测模式来预测：正向、反向、直接、插值和帧内。编码器选择并用信号表示比特流中不同的预测模式。例如，讨论中的先前 WMV 编码器在帧级别上发送经压缩的位平面，表示用于 B- 帧的每个宏块的直接 / 非直接模式决定，而非直接模式（诸如正向、反向和插值模式）在宏块级别中示出。

[0039] 正向模式类似于常规的 P- 帧预测。在正向模式中，宏块从时间上在先的锚中导出。在反向模式中，宏块从时间上后续的锚中导出。以直接或插值模式预测的宏块在预测中使用正向和反向锚。直接和插值模式使用将两个参考的像素值组合到一个宏块像素集中的舍入平均，如以下公式所示：

[0040] 平均像素值＝（正向插值 + 反向插值 +1）＞＞ 1

[0041] A. 共处 (co-located) 运动向量的分数编码 (Fraction Coding) 和缩放

[0042] 在讨论中的先前 WMV 编码器和解码器中，编码器通过缩放正向锚的共处运动向量来隐式地导出直接模式的运动向量。该缩放运算依赖于当前 B- 帧相对于其锚的时间位置。为了编码参考图片的时间位置，编码器使用分数编码。

[0043] 在分数编码中，编码器显式地将当前 B- 帧的时间位置编码为其两个锚之间距离的一分数。变量 BFRACTION 用来表示不同的各个分数，并在帧级别上发送。该分数在 0 和 1 之间离散值的有限集上取值。对于直接模式的运动向量，编码器和解码器使用该分数来缩放参考帧中的共处运动向量 (MV)，从而通过实现以下缩放运算来导出当前 B- 帧的隐式的直接模式运动向量 (MV_F 和 MV_B)：

[0044] MV_F ＝分数 *MV

[0045] MV_F ＝（分数 -1)*MV

[0046] 图 16 示出分数编码如何使编码器能任意缩放周围参考帧之间的运动。为了导出在 B- 帧 1620 中编码的当前宏块的 MV_F 和 MV_B，编码器和解码器使用分数编码来缩放将来参考帧 1630 中相应宏块的运动向量 (MV)。在图 16 所示的示例中，对于分数 p 和 q，p+q ＝ 1。该编码器和解码器使用两个隐式的运动向量来处理在先参考帧 1640 和将来参考帧 1630 中

的宏块，并使用这些的平均来预测当前宏块 1610。例如，在图 16 中，MV_F = (dx*p, dy*p) 而 MV_B = (-dx*q, -dy*q)。

[0047] 图 17 中的表格 1700 是用于比特流元素 BFRACTION 的可变长度代码 (VLC) 表格。在表格 1700 所示示例中，3- 比特代码字是“短”代码字而 7- 比特代码字是“长”代码字。该解码器根据图 18 中所示伪码 1800 基于分数的分子和分母来寻找缩放系数。

[0048] 一旦缩放系数已被确定，解码器就用它来缩放用于共处宏块的运动向量的 x- 和 y- 元素。给定后续的锚帧是 P- 帧（对于 I- 帧，所有的运动向量被假定为 (0,0)）且共处宏块包含运动向量 (MV_X, MV_Y)，则解码器导出两个运动向量，其中一个 (MV_X_F, MV_Y_F) 参考正向（在先）锚帧，而另一个 (MV_X_B, MV_Y_B) 参考反向（后续）锚帧。

[0049] 解码器根据图 19 中所示伪码 1900 来执行缩放。在伪码 1900 的函数 Scale_Direct_MV 中，输入 MV_X 和 MV_Y 是来自将来参考图片的共处宏块的运动向量的 x- 和 y- 元素，而输出 MV_X_F、MV_Y_F、MV_X_B 和 MV_Y_B 是用于被解码宏块的正向和反向指示运动向量的 x- 和 y- 元素。

[0050] B. B/I 帧

[0051] 讨论中的先前 WMV 编码器和解码器还在逐行扫描编码和解码中使用帧内 B- 帧（“B/I- 帧”）。B/I 帧像 I- 帧一样编码，因为它们不依赖于参考帧。但与 I- 帧不一样的是，B/I 帧不是关键帧；不允许其它帧将 B/I 帧用作锚。

[0052] C. 隔行扫描的 B- 帧

[0053] 讨论中的先前 WMV 编码器和解码器还使用隔行扫描的 B- 帧。隔行扫描 B- 帧中的宏块可进行半帧编码或帧编码。经帧编码的宏块可具有一个、两个（例如插值模式的正向和反向运动向量，直接模式的导出正向和反向运动向量）、或没有运动向量，而半帧编码的宏块可取决于预测模式具有多达四个运动向量。例如，在直接模式的半帧编码宏块中，导出四个隐式运动向量：上半帧的正向和反向运动向量，以及下半帧的正向和反向运动向量。

[0054] 尽管讨论中的先前 WMV 编码器和解码器使用隔行扫描的 B- 帧，但它们在若干重要方面受限。例如，每个宏块只允许一种宏块预测模式（例如直接模式、正向模式等），不使用 4MV 编码（即，对宏块中的每个块使用一个运动向量），且没有任何 B- 帧部分可以是任何帧的运动补偿的参考。作为另一示例，讨论中的先前 WMV 编码器和解码器的隔行扫描编码和解码（包括隔行扫描的 B- 帧）仅使用 4 ： 1 ： 1 宏块格式执行。

[0055] VI. 用于视频压缩和解压缩的标准

[0056] 除了先前的 WMV 编码器和解码器之外，若干种国际标准涉及视频压缩和解压缩。这些标准包括来自国际电信同盟 [“ITU”] 的运动图象专家组 [“MPEG”]1、2 和 4 标准以及 H.261、H.262、H.263 和 H.264 标准。在国际标准中用来获得数字视频序列的数据压缩的主要方法之一是减少图片之间的时间冗余。这些流行压缩方案 (MPEG-1、MPEG-2、MPEG-4、H.261、H.263 等）使用运动估计和补偿。例如，当前帧被分成均匀的正方形区域（例如各个块和 / 或宏块）。每个当前区域的匹配区域通过发送该区域的运动向量信息来指定。该运动向量指示在先前编码（和重构）帧中要用作当前区域的预测值的区域的位置。当前区域和参考帧中区域之间称为误差信号逐个像素差值被导出。该误差信号通常具有比原始信号更低的熵。因此，信息可以较低速率进行编码。因为在先前的 WMV 编码器和解码器中，运动向量值常常相关于空间上位于四周的运动向量，所以用来表示运动向量信息的数据的压缩

可通过编码当前运动向量与基于先前编码的、相邻运动向量的预测值之间的差异来获得。

[0057] 一些国际标准描述隔行扫描视频帧的运动估计和补偿。H.262 标准使隔行扫描的视频帧被编码为单个帧或两个半帧，其中帧编码或半帧编码可自适应地在逐帧基础上选择。H.262 标准描述基于半帧的预测，这是一种仅使用参考帧的一个半帧的预测模式。H.262 标准还描述双基预测，这是两个正向的基于半帧预测值对隔行扫描 P- 图片中的 16x16 块平均的预测模式。H.262 标准的 7.6 节描述“半帧预测”，包括在两个参考半帧中进行选择，以用作隔行扫描视频帧的当前半帧的宏块的运动补偿。小节 7.6.3 描述运动向量预测和重构，其中给定宏块的重构运动向量变成经后续编码 / 解码宏块的运动向量预测值。这种运动向量预测在许多情形中未能充分预测用于隔行扫描视频帧的半帧的宏块的运动向量。

[0058] 此外，H.262 标准的 7.6 节描述 B- 图片的“半帧预测”和“帧预测”。在“半帧预测”和“帧预测”中，对 B- 图片的预测使用两个最新重构的参考帧（略去其它介入的 B- 图片）执行，这些参考帧可被编码为两个半帧或单个帧。

[0059] 知道了视频压缩和解压缩对数字视频的关键重要性，视频压缩和解压缩是得以丰富开发的领域就不足为奇了。然而，不管先前的视频压缩和解压缩技术的优点是什么，它们并不具有以下技术和工具的优点。

发明内容

[0060] 总而言之，详细描述涉及用于编码和解码双向预测的隔行扫描视频帧（例如隔行扫描的 B- 半帧、隔行扫描的 B- 帧）的各种技术和工具。所述技术和工具改进了速率 / 失真性能，并便于更好地支持具有较低 CPU 资源的设备（例如具有较小形状系数的设备）。

[0061] 所述各实施例实现一种或多种用于编码和 / 或解码隔行扫描 B- 图片的所述技术和工具，如下包括但不限于：

[0062] 在一方面中，对于隔行扫描 B- 帧，编码器 / 解码器在隔行扫描 B- 帧的半帧编码宏块中的半帧之间切换预测模式。例如，编码器 / 解码器在半帧编码宏块中的上半帧的正向预测模式和下半帧的反向模式之间切换。同一半帧编码宏块内正向和反向预测之间的切换使发现对隔行扫描 B- 帧的不同部分的有效预测模式更为灵活。

[0063] 在另一方面中，对于隔行扫描 B- 帧，编码器 / 解码器通过为先前解码的时间后续锚的共处宏块的上半帧和下半帧的每一个选择至多一个代表性运动向量，计算当前宏块的直接模式运动向量。例如，至少部分地基于编码当前隔行扫描 B- 帧的宏块的模式（例如 1MV 模式、2 半帧 MV 模式等）执行选择。

[0064] 在又一方面中，对于隔行扫描 B- 半帧或隔行扫描 B- 帧，编码器 / 解码器使用 4MV 编码。例如，4MV 被用于单向预测模式（正向或反向模式），但不用于其它可用预测模式（例如直接、插值）。使用 4MV 允许对隔行扫描 B- 半帧和隔行扫描 B- 帧进行更准确的运动补偿；将 4MV 限制在正向和反向模式减少了编码开销，并避免了关联于组合 4MV 与诸如直接和插值的模式的解码复杂性。

[0065] 在另一方面中，对于隔行扫描 B- 半帧或隔行扫描 B- 帧，正向运动向量使用来自正向运动向量缓冲区的先前重构（或估计）的正向运动向量由编码器 / 解码器进行预测，而反向运动向量使用来自反向运动向量缓冲区的先前重构（或估计）的反向运动向量进行预测。结果的运动向量被添加到相应的缓冲区中。运动向量缓冲区中的空穴可用估计的运动

向量值来填充。例如，对于隔行扫描 B- 帧，当正向预测用来预测运动向量且运动向量被添加到正向运动向量缓冲区中时，反向运动向量缓冲区中的相应位置以仅将反向运动向量用作预测值的预测运动向量来填充（“空穴填充”）。作为另一示例，对于隔行扫描 B- 半帧，要为空穴填充在不同极性的运动向量（例如“相同极性”或“相反极性”）之间选择，编码器 / 解码器选择主极性半帧运动向量。锚和当前帧之间的距离使用各种语法元素计算，且计算出的距离被用于缩放参考半帧运动向量。各个运动向量缓冲区和各个运动向量缓冲区中的空穴填充使得对隔行扫描 B- 半帧和隔行扫描 B- 帧的运动向量预测更为准确。

[0066] 在又一方面中，对于隔行扫描的 B- 半帧，编码器 / 解码器使用“自参考”帧。例如，当前帧中的第二个 B- 半帧在运动补偿预测中参考来自当前帧的第一 B- 半帧。使帧中第一个 B- 半帧用作该帧中第二个 B- 半帧的参考使得第二个半帧的预测更加准确，同时还保留在当前帧中具有 B- 半帧的时间可缩放性优点。

[0067] 在另一方面中，对于隔行扫描的 B- 半帧，编码器发送指示用于隔行扫描 B- 半帧中的一个或多个宏块的预测模式是正向还是非正向的二进制信息。例如，编码器在经压缩位平面的 B- 半帧级别上发送正向 / 非正向决定信息。在经压缩位平面的 B- 半帧级别上发送正向 / 非正向预测模式决定信息可减少预测模式编码的编码开销。解码器执行相应的解码。

[0068] 在又一方面中，对于隔行扫描的 B- 半帧，如果下一锚图片的相应半帧中的相应宏块使用四个运动向量编码，则编码器 / 解码器使用利于主极性的逻辑来选择直接模式的运动向量。例如，如果相应宏块的相同极性运动向量数量上超过其相反极性的运动向量，则编码器 / 解码器计算相同极性运动向量的中值，以获得用于导出直接模式运动向量的运动向量。该选择过程允许导出准确的用于具有 4MV 宏块锚的隔行扫描 B- 半帧的直接模式运动向量。

[0069] 各种技术和工具可组合或单独使用。

[0070] 从以下参照附图进行的不同实施例的详细描述中，本发明的其它特征和优点将变得显而易见。

附图说明

[0071] 图 1 是示出根据现有技术的视频编码器的运动估计的示图。

[0072] 图 2 是示出根据现有技术的视频编码器中预测残差的 8x8 块的基于块的压缩。

[0073] 图 3 是示出根据现有技术的视频编码器中预测残差的 8x8 块的基于块的解压缩。

[0074] 图 4 是示出根据现有技术的隔行扫描帧的示图。

[0075] 图 5A 和 5B 是示出用于根据现有技术的逐行扫描 P- 帧中 1MV 宏块的候选运动向量预测值的宏块的位置的示图。

[0076] 图 6A 和 6B 是示出用于根据现有技术的混合 1MV/4MV 逐行扫描 P- 帧中的 1MV 宏块的候选运动向量预测值的块的位置的示图。

[0077] 图 7A、7B、8A、8B、9 和 10 是示出用于根据现有技术的混合 1MV/4MV 逐行扫描 P- 帧中的 4MV 宏块中各位置上块的候选运动向量预测值的块的位置的示图。

[0078] 图 11 是示出用于根据现有技术的隔行扫描 P- 帧中当前帧编码宏块的候选运动向量预测值的示图。

[0079] 图 12A-12B 是示出用于根据现有技术的隔行扫描 P- 帧中当前半帧编码宏块的候选运动向量预测值的示图。

[0080] 图 13 和 14 是示出根据现有技术的用于分别执行三者中值和四者中值计算的伪码的代码示图。

[0081] 图 15 是示出根据现有技术的具有过去和将来参考帧的 B- 帧的示图。

[0082] 图 16 是示出根据现有技术的使用分数编码的直接模式预测的示图。

[0083] 图 17 示出根据现有技术的比特流元素 BFRACTION 的 VLC 表格。

[0084] 图 18 是示出根据现有技术的用于寻找缩放直接模式预测中共处宏块的运动向量的缩放系数的伪码的代码清单。

[0085] 图 19 是示出根据现有技术的用于根据缩放系数缩放共处宏块中运动向量的 x- 和 y- 元素的伪码的代码清单。

[0086] 图 20 是适当计算环境的框图，结合该环境可实现若干所述实施例。

[0087] 图 21 是通用视频编码器系统的框图，结合该系统可实现若干所述实施例。

[0088] 图 22 是通用视频解码器系统的框图，结合该系统可实现若干所述实施例。

[0089] 图 23 是在若干所述实施例中使用的宏块格式的示图。

[0090] 图 24A 是一部分隔行扫描视频帧的示图，其中示出上半帧和下半帧的交替行。图 24B 是为编码 / 解码组织成帧的隔行扫描视频帧的示图，而图 24C 是为编码 / 解码组织成半帧的隔行扫描视频帧的示图。

[0091] 图 25 和 26 是示出具有两个参考半帧的隔行扫描 P- 半帧的示图。

[0092] 图 27 和 28 是示出使用所允许的最新参考半帧的隔行扫描 P- 半帧的示图。

[0093] 图 29 和 30 是示出使用所允许的第二最新参考半帧的隔行扫描 P- 半帧的示图。

[0094] 图 31 是示出运动向量的各垂直分量与当前和参考半帧极性的不同组合的相应空间位置之间的关系的示图。

[0095] 图 32 是示出当前宏块的两组三个候选运动向量预测值的示图。

[0096] 图 33A-33F 是示出用于计算两个参考半帧的隔行扫描 P- 或 B- 半帧中的运动向量预测值的伪码的代码清单。

[0097] 图 34A-34B 是示出用于缩放来自一个半帧的预测值以导出来自另一个半帧的预测值的伪码的代码清单。

[0098] 图 35 和 36 是示出关联于不同参考帧距离的缩放运算值的表格。

[0099] 图 37 是示出隔行扫描 P- 帧的 2 半帧 MV 宏块中亮度块的运动向量和色度块的导出运动向量的示图。

[0100] 图 38 是示出隔行扫描 P- 帧的 4 帧 MV 宏块中 4 个亮度块的每一个的不同运动向量和 4 个色度子块的每一个的导出运动向量的示图。

[0101] 图 39 是示出隔行扫描 P- 帧的 4 半帧 MV 宏块中亮度块的运动向量和色度块的导出运动向量的示图。

[0102] 图 40A-40B 是示出隔行扫描 P- 帧的当前宏块的候选预测值的示图。

[0103] 图 41 是示出使用不同预测模式预测隔行扫描 B- 帧的半帧编码宏块中的各个半帧的运动向量的技术的流程图。

[0104] 图 42 是示出计算隔行扫描 B- 帧的宏块的直接模式运动向量的技术的流程图。

[0105] 图 43 是先前已解码的时间上后续锚帧的共处宏块的各个块的缓存运动向量的示图，这些缓存运动向量用于计算隔行扫描 B- 帧的宏块的直接模式运动向量。

[0106] 图 44 是示出使用正向和 / 或反向运动向量缓冲区预测隔行扫描 B- 帧的当前宏块的运动向量的技术的流程图。

[0107] 图 45 是示出用于预测宏块运动向量的正向运动向量缓冲区和反向运动向量缓冲区中的运动向量的示图。

[0108] 图 46 是示出正向运动向量缓冲区和反向运动向量缓冲区中重构宏块的上半帧和下半帧运动向量的示图。

[0109] 图 47 是示出描述用于隔行扫描 B- 半帧运动向量预测中的实值缓存和空穴填充的极性选择过程的伪码的代码清单。

[0110] 图 48A-48B 是示出缩放来自一个半帧的预测值以导出来自另一半帧的预测值用于经反向预测的隔行扫描 B- 半帧的伪码的代码清单。

[0111] 图 49 是示出关联于第一隔行扫描 B- 半帧的不同参考帧距离的缩放运算值的表格。

[0112] 图 50A 和 50B 是示出隔行扫描 B- 半帧的参考半帧的示图。

[0113] 图 51 是示出用于在具有一个或多个位平面编码模式的视频编码器中编码隔行扫描 B- 半帧的宏块的正向 / 非正向预测模式决定信息的技术的流程图。

[0114] 图 52 是示出用于解码隔行扫描 B- 半帧的宏块的正向 / 非正向预测模式决定信息的技术的流程图，其中该决定信息由具有一个或多个位平面编码模式的视频编码器编码。

[0115] 图 53 是显示描述对用作隔行扫描 B- 半帧中直接模式运动向量的基础的运动向量的选择过程的伪码的代码清单。

[0116] 图 54 是示出用于组合实现中隔行扫描 B- 半帧或 BI- 半帧的帧 - 层比特流语法的示图。

[0117] 图 55 是示出用于组合实现中隔行扫描 B- 半帧的半帧 - 层比特流语法的示图。

[0118] 图 56 是示出用于组合实现中隔行扫描 BI- 半帧的半帧 - 层比特流语法的示图。

[0119] 图 57 是用于组合实现中隔行扫描 B- 半帧的宏块的宏块 - 层比特流语法的示图。

[0120] 图 58 是用于组合实现中隔行扫描 BI- 半帧的宏块的宏块 - 层比特流语法的示图。

[0121] 图 59 是用于组合实现中隔行扫描 B- 帧的帧 - 层比特流语法的示图。

[0122] 图 60 是用于组合实现中隔行扫描 B- 帧的宏块的宏块 - 层比特流语法的示图。

[0123] 图 61A-61B 是示出用于组合实现中解码运动向量差值和主 / 非主预测值信息的伪码的代码清单。

[0124] 图 62A-62F 是示出用于组合实现中计算双参考隔行扫描 P- 半帧中的运动向量预测值的伪码的代码清单。

[0125] 图 63 是示出用于组合实现中确定隔行扫描 B- 半帧的参考半帧的伪码的代码清单。

[0126] 图 64 是示出用于组合实现中收集隔行扫描 P- 帧的 1MV 宏块的候选运动向量的伪码的代码清单。

[0127] 图 65、66、67 和 68 是示出用于组合实现中收集隔行扫描 P- 帧的 4 帧 MV 宏块的候选运动向量的伪码的代码清单。

[0128] 图 69 和 70 是示出用于组合实现中收集隔行扫描 P- 帧的 2 半帧 MV 宏块的候选运动向量的伪码的代码清单。

[0129] 图 71、72、73 和 74 是示出用于组合实现中收集隔行扫描 P- 帧的 4 半帧 MV 宏块的候选运动向量的伪码的代码清单。

[0130] 图 75 是示出用于组合实现中计算隔行扫描 P- 帧的帧运动向量的运动向量预测值的伪码的代码清单。

[0131] 图 76 是示出用于组合实现中计算隔行扫描 P- 帧的半帧运动向量的运动向量预测值的伪码的代码清单。

[0132] 图 77A 和 77B 是示出用于组合实现中解码隔行扫描 P- 帧和 B- 帧的运动向量差值的伪码的代码清单。

[0133] 图 78 是示出用于组合实现中导出隔行扫描 P- 帧的色度运动向量的伪码的代码清单。

[0134] 图 79A-79C 是示出用于隔行扫描 B- 半帧的宏块的正向 / 非正向预测模式决定信息的 Norm-6 和 Diff-6 位平面编码模式的平铺显示的示图。

具体实施方式

[0135] 本申请涉及用于隔行扫描视频的有效压缩和解压缩的技术和工具。在各个所述实施例中，视频编码器和解码器结合用于编码和解码双向预测的隔行扫描视频帧的技术，和以包括各种层或级别（例如序列级别、帧级别、半帧级别、宏块级别、和 / 或块级别）的比特流格式或语法使用的相应信号表示技术。

[0136] 对所述实现的各种可选方案是可能的。例如，参照流程图描述的方法可通过改变各流程图中所示的各阶段的顺序、或重复或略去某些阶段等来变更。作为另一示例，尽管某些实现参照特定宏块格式进行了描述，但是也可使用其它格式。此外，参照双向预测描述的技术和工具也可应用于其它类型的预测。

[0137] 各种技术和工具可组合或独立地使用。不同实施例实现所述技术和工具的一种或多种。在此所述的一些技术和工具可用于视频编码器或解码器，或者在一些其它系统中并非具体受限于视频编码或解码。

[0138] I. 计算环境

[0139] 图 20 示出可实现若干所述实施例的适当计算环境 2000 的通用示例。计算环境 2000 并非旨在提出对本发明使用范围或功能的任何限制，因为本发明的技术和工具可在不同的通用或专用计算环境中实现。

[0140] 参照图 20，计算环境 2000 包括至少一个处理单元 2010 和存储 2020。在图 20 中，该最基本配置 2030 包括在虚线内。处理单元 2010 执行计算机可执行指令，且可以是实际或虚拟处理器。在多处理系统中，多个处理单元执行计算机可执行指令以增加处理能力。存储器 2020 可以是易失性存储器（例如寄存器、高速缓存、RAM）、非易失性存储器（例如 ROM、EEPROM、闪存等）或两者的组合。存储器 2020 存储实现进行隔行扫描视频帧的双向预测的视频编码器或解码器的软件 2080。

[0141] 计算环境可具有附加特征。例如，计算环境 2000 包括存储 2040、一个或多个输入设备 2050、一个或多个输出设备 2060、以及一个或多个通信连接 2070。诸如总线、控制器

或网络的互连机制（未示出）使计算环境 2000 的各个组件互相连接。通常，操作系统软件（未示出）向在计算环境 2000 中执行的其它软件提供一个操作环境，并协调计算环境 2000 的各个组件的动作。

[0142] 存储 2040 可以是可移动或不可移动的，并包括磁盘、磁带或盒式磁带、CD-ROM、CD-RW、DVD、或任何其它可用于存储信息并可在计算环境 2000 内访问的媒体。存储 2040 为软件 2080 存储实现视频编码器或解码器的指令。

[0143] 输入设备 2050 可以是诸如键盘、鼠标、电子笔、或跟踪球、语音输入设备、扫描设备、或向计算环境 2000 提供输入的另一设备。对于音频或视频编码，输入设备 2050 可以是接受模拟或数字形式的音频或视频输入的声卡、视频卡、TV 调谐器卡或相似设备、或将音频或视频样本读入计算环境 2000 中的 CD-ROM 或 CD-RW。输出设备 2060 可以是显示器、打印机、扬声器、CD 刻录机、或从计算环境 2000 提供输出的另一设备。

[0144] 通信连接 2070 允许经通信媒体与另一计算实体的通信。通信媒体在已调制数据信号中传送诸如计算机可执行指令、音频或视频输入或输出、或其它数据等信息。已调制数据信号是使其特征的一个或多个以在信号中编码信息的方式进行设置或改变的信号。作为示例且非限制，通信媒体包括用电子、光学、RF、红外线、声学或其它载体实现的有线或无线技术。

[0145] 这些技术和工具可在计算机可读媒体的一般上下文中描述。计算机可读媒体是可在计算环境内访问的任何可用媒体。作为示例，但非限制，对于计算环境 2000，计算机可读媒体包括存储器 2020、存储 2040、通信媒体、以及以上任一个的组合。

[0146] 这些技术和工具可在计算机可执行指令的一般上下文中描述，诸如在目标实际或虚拟处理器上的计算环境中执行的程序模块所包括的计算机可执行指令。通常，程序模块包括执行具体任务或实现具体抽象数据类型的例程、程序、库、对象、类、组件、数据结构等。程序模块的功能可在各个实施例中按需在程序模块之间组合或划分。程序模块的计算机可执行指令可在本地或分布式的计算环境内执行。

[0147] 为说明起见，详细描述使用像“估计”、“补偿”、“预测”和“应用”的术语来描述计算环境中的计算机操作。这些术语是由计算机执行的操作的高层抽象，且不应与人执行的动作相混淆。对应于这些术语的实际计算机操作取决于实现而变化。

[0148] II. 通用视频编码器和解码器

[0149] 图 21 是通用视频编码器 2100 的框图，结合该系统可实现若干所述实施例。图 22 是通用视频解码器 2200 的框图，结合该系统可实现若干所述实施例。

[0150] 编码器 2100 和解码器 2200 内各模块之间的所示关系表示编码器和解码器中的一般信息流；其它关系为简化起见并未示出。特别地，图 21 和 22 通常并未示出表示用于视频序列、图片、宏块、块等的编码器设置、模式、表格等的辅助信息。这些辅助信息通常在辅助信息的熵编码之后在输出比特流中发送。输出比特流的格式可以是 Windows Media Video 版本 9 格式或其它格式。

[0151] 编码器 2100 和解码器 2200 处理视频图片，它们可以是视频帧、视频半帧、或帧和半帧的组合。图片和宏块级别上的比特流语法和语义可依赖于是使用帧还是半帧。也可以有对宏块组织和整体定时的改变。编码器 2100 和解码器 2200 是基于块的，并对帧使用 4 ： 2 ： 0 的宏块格式，其中每个宏块包括 4 个 8x8 亮度块（常被视为一个 16x16 宏块）和

2 个 8x8 色度块。对于半帧，可使用相同或不同的宏块组织和格式。这些 8x8 块可在不同阶段上作进一步的细分，例如在频率变换和熵编码阶段。示例视频帧组织如下进行更详细的描述。或者，编码器 2100 和解码器 2200 是基于对象的、使用不同的宏块或块格式、或对不同于 8x8 块和 16x16 宏块的大小或配置的像素集执行操作。

[0152] 取决于所需的压缩实现和类型，编码器或解码器的模块可添加、略去、划分成多个模块、与其它模块相组合、和 / 或用类似模块替换。在替换实施例中，具有不同模块和 / 或模块其它配置的编码器或解码器执行一种或多种所述技术。

[0153] A. 视频帧组织

[0154] 在一些实现中，编码器 2100 和解码器 2200 处理如下组织的视频帧。帧包含视频信号的各行空间信息。对于逐行扫描视频，这些行包含从一时刻开始并继续后续行直到帧的底部的样本。逐行扫描视频帧被分成诸如图 23 中所示宏块 2300 的宏块。宏块 2300 包括 4 个 8x8 亮度块 (Y1 到 Y4) 和与 4 个亮度块共处的 2 个 8x8 色度块，这些色度块符合常规的 4 ： 2 ： 0 宏块格式但在水平和垂直方向分辨率减半。这些 8x8 块可在不同阶段上作进一步的细分，例如在频率变换 (例如 8x4、4x8 或 4x4 DCT) 和熵编码阶段。逐行扫描 I- 帧是帧内编码的逐行扫描视频帧。逐行扫描 P- 帧是使用正向预测编码的逐行扫描视频帧，而逐行扫描 B- 帧是使用双向预测编码的逐行扫描视频帧。逐行扫描 P- 和 B- 帧可包括帧内编码宏块以及不同类型的经预测宏块。

[0155] 隔行扫描视频帧由一个帧的两次扫描组成 - 一次包括该帧的偶数行 (上半帧) 而另一次包括该帧的奇数行 (下半帧)。这两个半帧可表示两个不同时段，或者它们可来自同一时段。图 24A 示出隔行扫描视频帧 2400 的一部分，包括隔行扫描视频帧 2400 中左上方的上半帧和下半帧的交替行。

[0156] 图 24B 示出为编码 / 解码而组织为帧 2430 的图 24A 的隔行扫描视频帧 2400。隔行扫描视频帧 2400 已被分成诸如宏块 2431 和 2432 的宏块，它们使用如图 23 所示的 4 ： 2 ： 0 格式。在亮度平面中，每个宏块 2431、2432 包括来自上半帧的 8 行、与之交替的来自下半帧的 8 行 (一共为 16 行)，且每一行为 16 像素长。(宏块 2431、2432 内亮度块和色度块的实际组织和放置并未示出，且实际上对于不同的编码决定可变化。) 在给定宏块内，上半帧信息和下半帧信息在各个阶段的任一个上可联合或独立地编码。隔行扫描的 I- 帧是隔行扫描视频帧的两个帧内编码半帧，其中宏块包括两个半帧的信息。隔行扫描 P- 帧是使用正向预测编码的隔行扫描视频帧的两个半帧，而隔行扫描 B- 帧是使用双向预测编码的隔行扫描视频帧的两个半帧，其中宏块包括两个半帧的信息。隔行扫描 P- 和 B- 帧可包括帧内编码宏块以及不同类型的预测宏块。

[0157] 图 24C 示出为编码 / 解码而组织为半帧 2460 的图 24A 的隔行扫描视频帧 2400。隔行扫描视频帧 2400 的两个半帧的每一个被分成各个宏块。上半帧被分成诸如宏块 2461 的宏块，而下半帧被分成诸如宏块 2462 的宏块。(再一次，各宏块使用如图 23 所示的 4 ： 2 ： 0 格式，且各宏块内亮度块和色度块的组织和放置并未示出。) 在亮度平面中，宏块 2461 包括来自上半帧的 16 行、宏块 2462 包括来自下半帧的 16 行，且每一行为 16 像素长。隔行扫描的 I- 半帧是隔行扫描视频帧的单个独立表示半帧。隔行扫描 P- 半帧是使用正向预测编码的隔行扫描视频帧的单个独立表示半帧，而隔行扫描 B- 半帧是使用双向预测编码的隔行扫描视频帧的单个独立表示半帧。隔行扫描 P- 和 B- 半帧可包括帧内编码宏

块以及不同类型的预测宏块。

[0158] 术语图片通常指源、经编码或重构的图像数据。对于逐行扫描视频，图片是逐行扫描视频帧。对于隔行扫描视频，取决于上下文，图片可指隔行扫描视频帧、帧的上半帧、或帧的下半帧。

[0159] 或者，编码器 2100 或解码器 2200 是基于对象的、使用不同的宏块或块格式、或对不同于 8x8 块和 16x16 宏块的大小或配置的像素集执行操作。

[0160] B. 视频编码器

[0161] 图 21 是通用视频编码器系统 2100 的框图。编码器系统 2100 接收包括当前图片 2105 的一个视频图片序列（例如，逐行扫描视频帧、隔行扫描视频帧、或隔行扫描视频帧的半帧），并产生压缩后视频信息 2195 作为输出。视频编码器的特定实施例通常使用通用编码器 2100 的变体或补充版本。

[0162] 编码器系统 2100 压缩预测图片和关键图片。为演示起见，图 21 示出关键图片通过编码器系统 2100 的路径，以及预测图片的路径。编码器系统 2100 的许多组件用于压缩关键图片和预测图片。由那些组件执行的确切操作可取决于所压缩的信息类型而变化。

[0163] 预测图片（例如逐行扫描 p- 帧或 b- 帧、隔行扫描 p- 半帧或 b- 半帧、或隔行扫描 p- 帧或 b- 帧）根据来自一个或多个其它图片（通常称为参考图片或锚）的预测（或差值）来表示。预测残差是所预测图片和原始图片之间的差值。相反，关键图片（例如逐行扫描 I- 帧、隔行扫描 I- 半帧、或隔行扫描 I- 帧）不参照其它图片进行压缩。

[0164] 如果当前图片 2105 是正向预测图片，则运动估计器 2110 参照一个或多个参考图片估计当前图片 2105 的宏块或其它像素集的运动，其中参考图片是例如缓存在图片存储 2120 中的重构的先前图片 2125。如果当前图片 2105 是双向预测图片，则运动估计器 2110 参照（例如隔行扫描 B- 半帧的）多达 4 个重构参考图片来估计当前图片 2105 中的运动。通常，运动估计器参照一个或多个时间上在先的参考图片和一个或多个时间上将来的参考图片来估计 B- 图片中的运动。因此，编码器系统 2100 可将分别的存储 2120 和 2122 用于多个参考图片。对于有关逐行扫描 B- 帧的更多信息，参见序列号为 10/622,378 题为“Advanced Bi-Directional Predictive Coding of Video Frames”（视频帧的高级双向预测编码）并于 2003 年 7 月 18 日提交的美国专利申请。

[0165] 运动估计器 2110 可按像素、1/2 像素、1/4 像素或其它增量进行估计，并可在逐个图片基础或其它基础上切换运动估计的分辨率。运动估计器 2110（和补偿器 2130）还可在每帧或其它基础上在参考图片像素插值类型之间（例如双三次和双线性）切换。运动估计的分辨率可水平地或垂直地相同或不同。运动估计器 2110 输出诸如差值运动向量信息的辅助信息运动信息 2115。编码器 2100 通过例如计算运动向量的一个或多个预测值、计算运动向量和预测值之间的差值、并熵编码这些差值，来编码运动信息。为了重构运动向量，运动补偿器 2130 组合预测值与差值运动向量信息。用于计算运动向量预测值、计算差值运动向量、并重构隔行扫描 B- 半帧和隔行扫描 B- 帧的运动向量的各种技术如下所述。

[0166] 运动补偿器 2130 将重构运动向量应用于重构图片 2125，以形成经运动补偿的当前图片 2135。然而，预测很少是完美的，且经运动补偿的当前图片 2135 和原始的当前图片 2105 之间的差值为预测残差 2145。在图片的随后重构中，预测残差 2145 被添加到经运动补偿的当前图片 2135 中，以获得更接近于原始当前图片 2105 的重构图片。然而，在有损压

缩中，仍有一些信息从原始当前图片 2105 中丢失。或者，运动估计器和运动补偿器应用另一类型的运动估计 / 补偿。

[0167] 频率变换器 2160 将空间域视频信息转换成频域（即频谱）数据。对于基于块的视频图片，频率变换器 2160 将 DCT、DCT 的变体、或其它块变换应用于像素数据或预测残差数据的块，从而产生频率变换系数的块。或者，频率变换器 2160 应用诸如傅立叶变换的另一种常规频率变换、或使用小波或子频带分析。频率变换器 2160 可应用 8x8、8x4、4x8、4x4 或其它大小的频率变换。

[0168] 然后量化器 2170 量化频谱数据系数的各个块。该量化器将均匀的标量量化应用于频谱数据，其中步长在逐个图片或其它基础上变化。或者，量化器将另一种类型的量化应用于频谱数据系数，例如不均匀的向量或非自适应量化，或者直接在不使用频率变换的编码器系统中量化空间域数据。除了自适应量化之外，编码器 2100 可将帧丢弃、自适应滤波或其它技术用于速率控制。

[0169] 编码器 2100 可将特定的信号表示用于被跳过的宏块，该宏块是没有特定类型信息的宏块（例如没有该宏块的运动信息和没有残差信息）。

[0170] 当需要重构后的当前帧用于随后的运动估计 / 补偿时，逆量化器 2176 对量化后的频谱数据系数执行逆量化。然后反向频率变换器 2166 执行频率变换器 2160 的反向操作，产生重构后预测残差（用于预测图片）或重构后的关键图片。如果当前图片 2105 是关键图片，则重构后的关键图片被取作重构后的当前图片（未示出）。如果当前图片 2105 是预测图片，则重构后的预测残差被添加到经运动补偿的当前图片 2135 中以形成重构后的当前图片。一个或多个图片存储 2120、2122 缓存重构后的当前图片，以用于运动补偿预测。在某些实施例中，编码器将解块滤波器应用于重构后的帧以自适应地使图片中的间断点和其它人工效应平滑。

[0171] 熵编码器 2180 压缩量化器 2170 的输出以及某些辅助信息（例如运动信息 2115、量化步长）。典型的熵编码技术包括算术编码、差分编码、哈夫曼编码、运行长度编码、LZ 编码、字典编码以及以上的组合。熵编码器 2180 通常使用用于不同类型信息（例如 DC 系数、AC 系数、不同类型的辅助信息）的不同编码技术，并可从特定编码技术内的多个代码表中选择。

[0172] 熵编码器 2180 向多路复用器 [“MUX”]2190 提供压缩后的视频信息 2195。MUX 2190 可包括缓冲区，且缓冲区级别指示符被反馈给比特率自适应模块作速率控制。在 MUX 2190 之前或之后，压缩后的视频信息 2195 可被信道编码用于在网络上传输。信道编码可将差错检测和纠正数据应用于压缩后的视频信息 2195。

[0173] C. 视频解码器

[0174] 图 22 是一般视频解码器系统 2200 的框图。解码器系统 2200 接收用于视频图片的经压缩序列的信息 2295，并产生包括重构图片 2205 的输出（例如，逐行扫描视频帧、隔行扫描视频帧、或隔行扫描视频帧的半帧）。视频解码器的特定实施例通常使用通用解码器 2200 的变体或补充版本。

[0175] 解码器系统 2200 解压缩预测图片和关键图片。为演示起见，图 22 示出关键图片通过解码器系统 2200 的路径，以及正向预测图片的路径。解码器系统 2200 的许多组件用于解压缩关键图片和预测图片。由那些组件执行的确切操作取决于所解压缩的信息类型而

变化。

[0176] DEMUX(多路分解器)2290 接收压缩后视频序列的信息 2295,并使接收到的信息可用于熵解码器 2280。DEMUX 2290 可包括一抖动缓冲区和其它缓冲区。在 DEMUX 2290 之前或之后,经压缩的视频信息可进行信道解码和用于差错检测和纠正处理。

[0177] 熵解码器 2280 通常应用编码器中执行的熵编码的逆,对熵编码后的量化数据以及熵编码后的辅助信息(例如运动信息 2215、量化步长)进行熵解码。熵解码技术包括算术解码、差分解码、哈夫曼解码、运行长度解码、LZ 解码、字典解码以及以上的组合。熵解码器 2280 经常使用用于不同类型信息(例如 DC 系数、AC 系数、不同类型的辅助信息)的不同解码技术,并可从特定解码技术内的多个代码表中选择。

[0178] 解码器 2200 通过例如计算运动向量的一个或多个预测值、熵解码差值运动向量、并组合解码后的差值运动向量和用于重构运动向量的预测值,来解码运动 2215。用于计算运动向量预测值、计算差值运动向量、并重构隔行扫描 B- 半帧和隔行扫描 B- 帧的各种技术如下所述。

[0179] 运动补偿器 2230 将运动信息 2215 应用于一个或多个参考图片 2225,以形成被重构的图片 2205 的预测值 2235。例如,运动补偿器 2230 使用一个或多个宏块运动向量来寻找参考图片 2225 中的宏块。一个或多个图片存储(例如图片存储 2220、2222)存储先前重构后的图片,以用作参考图片。通常,B- 图片具有一个以上的参考图片(例如至少一个时间上在先的参考图片和至少一个时间上将来的参考图片)。因此,解码器系统 2200 可将分别的图片存储 2220 和 2222 用于多个参考图片。运动补偿器 2230 可按像素、1/2 像素、1/4 像素或其它增量补偿运动,并可在逐个图片基础或其它基础上切换运动补偿的分辨率。运动补偿器 2230 还可在每帧或其它基础上在参考图片像素插值类型之间(例如双三次和双线性)切换。运动补偿的分辨率可水平地或垂直地相同或不同。或者,运动补偿器应用另一类型的运动补偿。由运动补偿器进行的预测很少是完美的,因此解码器 2200 还重构预测残差。

[0180] 逆量化器 2270 对熵解码后的数据执行逆量化。一般而言,该逆量化器将均匀的标量逆量化应用于熵解码后的数据,其中步长在逐帧或其它基础上变化。或者,该逆量化器将另一种类型的逆量化应用于数据,例如在不均匀的向量或非自适应量化后重构,或者在不使用逆频率变换的解码器系统中直接逆量化空间域数据。

[0181] 逆频率变换器 2260 将量化后的频域数据转换成空间域视频信息。对于基于块的视频图片,逆频率变换器 2260 将逆 DCT[“IDCT”]、IDCT 的变体、或其它逆向块变换应用于频率变换系数的块,从而分别产生关键图片或预测图片的像素数据或预测残差数据。或者,反向频率变换器 2260 应用诸如傅立叶逆变换的另一种常规反向频率变换、或使用小波或子频带分析。反向频率变换器 2260 可应用 8x8、8x4、4x8、4x4 或其它大小的反向频率变换。

[0182] 对于预测图片,解码器 2200 组合重构后的预测残差 2245 和经运动补偿的预测 2235 来形成重构后的图片 2205。当解码器需要用于随后运动补偿的重构后图片 2205 时,一个或两个图片存储(例如图片存储 2220)缓存重构后图片 2205,以用于预测下一图片。在一些实施例中,解码器 2200 将解块滤波器应用于重构后图片以自适应地使图片中的间断点和其它人工效应平滑。

[0183] III. 隔行扫描 P- 半帧和隔行扫描 P- 帧

[0184] 典型的隔行扫描视频帧由在不同时间扫描的两个半帧（例如上半帧和下半帧）组成。一般而言，通过一起编码半帧来编码隔行扫描视频帧的静态区域是更为有效的（“帧模式”编码）。另一方面，通过分开编码半帧来编码隔行扫描视频帧的移动区域通常更为有效（“半帧模式”编码），因为这两个半帧倾向于具有不同的运动。正向预测的隔行扫描视频帧可被编码为两个独立的正向预测半帧－隔行扫描 P- 半帧。例如，当在隔行扫描视频帧上有较高运动从而在各半帧之间有很大差异时，分开编码正向预测的隔行扫描视频帧的半帧会比较有效。

[0185] 或者，正向预测的隔行扫描视频帧可使用半帧编码和帧编码的混合来编码为隔行扫描 P- 帧。对于隔行扫描 P- 帧的宏块，该宏块包括上半帧和下半帧的像素行，且各行可用帧编码模式一起编码，或用半帧编码模式分开编码。

[0186] A. 隔行扫描 P- 半帧

[0187] 隔行扫描 P- 半帧参考一个或多个先前解码的半帧。例如，在一些实现中，隔行扫描 P- 半帧参考一个或两个先前解码的半帧，而隔行扫描 B- 半帧参考多达两个先前的和两个将来的参考半帧（即多达总共四个参考半帧）。（用于隔行扫描 B- 半帧的编码和解码技术在下面详细描述。）

[0188] 图 25 和 26 示出具有两个参考半帧的隔行扫描 P- 半帧的示例。在图 25 中，当前半帧 2510 参考时间上在先的隔行扫描视频帧中的上半帧 2520 和下半帧 2530。因为半帧 2540 和 2550 是隔行扫描的 B- 半帧，所以它们不用作参考半帧。在图 26 中，当前半帧 2610 参考在包含当前半帧 2610 的隔行扫描视频帧前面紧邻的隔行扫描视频帧中的上半帧 2620 和下半帧 2630。对于有关双参考的隔行扫描 P- 半帧的更多信息，参见序列号为 xx/yyy，zzz 题为“Predicting Motion Vectors for Fields of Forward-predicted Interlaced Video Frames”（预测正向预测的隔行扫描视频帧的半帧的运动向量）并于 2004 年 5 月 27 日提交的美国专利申请。

[0189] 图 27 和 28 示出具有一个参考半帧－所允许的时间上最新参考半帧的隔行扫描 P- 半帧的示例。在图 27 中，当前半帧 2710 参考时间上在先的隔行扫描视频帧的下半帧 2730，但不参考该隔行扫描视频帧上不那么新的上半帧 2720。在图 27 所示示例中，半帧 2740 和 2750 是隔行扫描 B- 半帧，且不是所允许的参考半帧。在图 28 中，当前半帧 2810 参考在包含当前半帧 2810 的隔行扫描视频帧前面紧邻的隔行扫描视频帧中的下半帧 2830，而不参考不那么新的上半帧 2820。

[0190] 图 29 和 30 示出使用所允许的第二最新参考半帧的隔行扫描 P- 半帧的示例。在图 29 中，当前半帧 2910 参考时间上在先的隔行扫描视频帧的上半帧 2920，但不参考更新的下半帧 2920。在图 29 所示示例中，半帧 2940 和 2950 是隔行扫描 B- 半帧，且不是所允许的参考半帧。在图 30 中，当前半帧 3010 参考上半帧 3020，而不参考更新的下半帧 3030。

[0191] 在一实现中，图 25-30 中示出的全部情形在隔行扫描 P- 半帧语法中都是许可的。其它实现也是可能的。例如，图片可将来自不同类型或时间位置的其它图片的半帧用作参考半帧。

[0192] 1. 半帧图片坐标系统和半帧极性

[0193] 运动向量表示以 1/4 像素为单位的水平和垂直位移。例如，如果运动向量的垂直分量表示 6 个 1/4 像素单位的位移，则这表示参考块是当前块位置下方 1.5 个半帧行处

(6*1/4 = 1 1/2)。

[0194] 图 31 示出在一实现中运动向量的垂直分量和空间位置之间的关系。图 31 所示示例示出当前和参考半帧类型（例如上和下）的三种不同组合的三种不同情形 3110、3120 和 3130。如果半帧类型对当前和参考半帧不同，则极性“相反”。如果半帧类型相同，则极性“相同”。对于每种情形，图 31 示出当前半帧中一个垂直像素列和参考半帧中的第二垂直像素列。实际上，该两列是水平对齐的。圆圈表示实际整数像素位置，而 X 表示插值的 1/2 或 1/4 像素位置。水平分量值（未示出）无需说明因为隔行扫描的任何偏移量，因为各个半帧都是水平对齐的。负值表示在相反方向上比所示正值垂直偏移量更偏上的偏移量。

[0195] 在情形 3110 中，极性“相反”。当前半帧为上半帧，而参考半帧为下半帧。相对于当前半帧，参考半帧的位置因为隔行扫描而在向下方向上偏移 1/2 像素。垂直运动向量分量值为 0 是“无垂直运动”偏移，并表示参考半帧中与当前半帧中位置在同一垂直水平上（绝对值）的位置；垂直运动向量分量值为 +2 表示参考半帧中偏移到当前半帧中位置下方 1/2 像素（绝对值）的位置，该位置是参考半帧中的实际值；而垂直分量值为 +4 表示参考半帧中偏移到当前半帧中位置下方 1 整个像素（绝对值）的位置，该位置是参考半帧中的插值。

[0196] 在情形 3120 中，极性也“相反”。当前半帧是下半帧，而参考半帧是上半帧。相对于当前半帧，参考半帧的位置因为隔行扫描而在向上方向上偏移 1/2 像素。垂直运动向量分量值为 -2 表示参考半帧中偏移到当前半帧中位置上方 1/2 像素（绝对值）的位置；垂直分量值为 0 表示参考半帧中与当前半帧中位置在同一水平上（绝对值）的位置；而垂直分量值为 +2 表示参考半帧中偏移到当前半帧中位置下方 1/2 像素（绝对值）的位置。

[0197] 在情形 3130 中，极性“相同”。相对于当前半帧，参考半帧的位置在垂直方向上相同。垂直运动向量分量值为 0 是“无垂直运动”偏移，并表示参考半帧中与当前半帧中位置在同一垂直水平上（绝对值）的位置；垂直运动向量分量值为 +2 表示参考半帧中偏移到当前半帧中位置下方 1/2 像素（绝对值）的位置，该位置是参考半帧中的插值；而垂直分量值为 +4 表示参考半帧中偏移到当前半帧中位置下方 1 整个像素（绝对值）的位置，该位置是参考半帧中的实际值。

[0198] 或者，运动向量的位移根据不同惯例来表达。

[0199] 2. 双参考半帧的隔行扫描 P- 半帧中的运动向量预测

[0200] 双参考半帧的隔行扫描 P- 半帧参考相同时间方向上的两个半帧（例如两个最近的先前参考半帧）。对每个宏块计算两个运动向量预测值。在一些实现中，一个预测值来自相同极性的参考半帧，而另一个预测值来自相反极性的参考半帧。极性的其它组合也是可能的。（以下描述每个方向上使用双参考半帧的隔行扫描 B- 半帧。在一些实现中，这些隔行扫描 B- 半帧将与隔行扫描 P- 半帧相同的技术用于计算运动向量预测值。）

[0201] 在一些实现中，编码器 / 解码器通过寻找奇半帧预测值和偶半帧预测值、并选择处理预测值之一用于处理宏块，来计算当前块或宏块的运动向量预测值。例如，编码器 / 解码器确定奇半帧运动向量预测值和偶半帧运动向量预测值。运动向量预测值之一因而具有与当前半帧相同的极性，而另一运动向量预测值具有相反极性。编码器 / 解码器从奇半帧运动向量预测值和偶半帧运动向量预测值中选择一运动向量预测值。例如，编码器基于哪个给出较佳预测在各运动向量预测值之间选择。编码器使用简单的选择信号或使用较复杂的结合改进编码效率的上下文信息的信号表示来用信号表示要使用哪个运动向量预测值。

该上下文信息可表示奇半帧或偶半帧的哪一个、或相同极性半帧或相反极性半帧的哪一个已主要用于块或宏块周围的邻域中。解码器基于选择信号和/或上下文信息选择要使用哪个运动向量预测值。然后，编码器/解码器使用选定的运动向量预测值处理该运动向量。例如，编码器编码运动向量和运动向量预测值之间的差值。或者，解码器通过组合运动向量差值和运动向量预测值来解码运动向量。

[0202] 或者，编码器和/或解码器可跳过确定奇半帧运动向量预测值、或跳过确定偶半帧运动向量预测值。例如，如果编码器确定奇半帧将用于特定块或宏块的运动补偿，则编码器只确定奇半帧运动向量预测值。或者，如果解码器从上下文和/或信号表示信息中确定奇半帧将用于运动补偿，则解码器只确定奇半帧运动向量预测值。这样，编码器和解码器可避免不必要的运算。

[0203] 解码器可采用以下技术来确定当前隔行扫描 P- 半帧的运动向量预测值。

[0204] 对于隔行扫描 P- 半帧中具有运动向量的每个块或宏块，可获得两组三个候选运动向量预测值。从中获得这些候选运动向量预测值的相邻宏块相对于当前宏块 3200 的位置如图 32 所示。这些候选值的三个来自偶参考半帧，而另三个来自奇参考半帧。因为每个候选方向中的相邻宏块 (A、B 和 C) 是帧内编码的或具有参考偶半帧或奇半帧的实际运动向量，所以需要导出其它半帧的运动向量（或导出帧内编码宏块的奇半帧和偶半帧运动向量候选值）。例如，对于给定宏块，假设预测值 A 具有参考奇半帧的运动向量。在该情形中，“偶半帧”候选预测值 A 从“奇半帧”候选预测值 A 的运动向量中导出。该导出使用缩放运算来完成。（参见例如以下图 34A 和 34B 的解释。）或者，导出用另一种方式完成。

[0205] 一旦已得到三个奇半帧候选运动向量预测值，就使用中值运算来从三个奇半帧候选值中导出奇半帧运动向量预测值。类似地，一旦已得到三个偶半帧候选运动向量预测值，就使用中值运算来从三个偶半帧候选值中导出偶半帧运动向量预测值。或者，使用另一种机制来基于候选半帧运动向量预测值选择半帧运动向量预测值。解码器判定是将偶半帧还是奇半帧用作运动向量预测值（例如通过选择主预测值），及偶还是奇运动向量预测值被用来重构运动向量。

[0206] 图 33A-33F 中的伪码 3300 示出用来从如图 32 所示排列的预测值 A、B 和 C 中产生运动向量预测值的过程。尽管图 32 示出当前隔行扫描 P- 半帧中典型宏块的邻域，但图 33A-33F 的伪码 3300 解决了宏块位置的各个特定情形。此外，伪码 3300 可用来计算各个位置上块的运动向量的运动向量预测值。

[0207] 在伪码 3300 中，术语“相同半帧”和“相反半帧”被理解成与当前编码或解码的半帧相关。例如，如果当前半帧是偶半帧，则“相同半帧”是偶参考半帧而“相反半帧”是奇参考半帧。伪码 3300 中的变量 samefieldpred_x 和 samefieldpred_y 表示来自相同半帧的运动向量预测值的水平和垂直分量，而变量 oppositefieldpred_x 和 oppositefieldpred_y 表示来自相反半帧的运动向量预测值的水平和垂直分量。变量 samecount 和 oppositecount 分别跟踪当前块或宏块的邻域的多少运动向量参考当前半帧的“相同”极性参考半帧，及多少参考“相反”极性参考半帧。变量 samecount 和 oppositecount 在伪码开始时初始化为 0。

[0208] 伪码 3300 中所提及的缩放运算 scaleforsame() 和 scaleforopposite() 被用来从邻居的实际运动向量值中导出“另一”半帧的候选运动向量预测值。缩放运算是实现无

关的。示例缩放运算在以下参照图 34A、34B、35 和 36 进行描述。或者，其它缩放运算可用来例如补偿诸如图 31 中所示的垂直位移。（具体用于隔行扫描 B- 半帧的缩放运算如下进行详细描述。）

[0209] 图 33A 和 33B 示出用于计算帧内的内部位置中典型块或宏块的运动向量预测值的伪码。“帧内”邻居的运动向量被设置为 0。对于每个邻居，都设置相同半帧运动向量预测值和相反半帧运动向量预测值，在其中一个通过邻居运动向量的实际值设置时，另一个从中导出。对相同半帧运动向量预测值和相反半帧运动向量预测值计算候选值的中值，而“主”预测值从 samecount 和 oppositecount 中确定。变量 dominantpredictor 表示哪个半帧包含主运动向量预测值。如果运动向量预测值具有与三个候选预测值的大部分相同的极性，则它是主要预测值。（用运动向量差值数据解码的表示信号的值 predictor_flag 表示使用主还是非主预测值。）

[0210] 图 33C 中的伪码解决每行只有一个宏块（它没有邻居 B 或 C）的隔行扫描 P- 半帧的宏块的情形。图 33D 或 33E 中的伪码解决块或宏块在隔行扫描 P- 半帧的左边缘（没有邻居 C）的情形。在此，如果运动向量预测值具有与两个以上候选预测值相同的极性，则它是主预测值；而在不分上下的情形中，相反半帧运动向量预测值为主预测值。最后，图 33F 中的伪码解决例如宏块在隔行扫描 P- 半帧的首行中的情形。

[0211] 3. 一个半帧运动向量预测值从另一个半帧运动向量预测值导出的缩放

[0212] 在一实现中，编码器 / 解码器使用图 34A 和 34B 的伪码 3400 中示出的缩放运算，将一个半帧运动向量预测值从另一个半帧运动向量预测值导出。SCALEOPP、SCALESAME1、SCALESAME2、SCALEZONE1_X、SCALEZONE1_Y、ZONE1OFFSET_X 和 ZONE1OFFSET_Y 的值是实现相关的。示出两个可能的值集，其中当前半帧是隔行扫描视频帧中的第一半帧的情形在图 35 的表格 3500 中示出，而当前半帧是隔行扫描视频帧中的第二半帧的情形在图 36 的表格 3600 中示出。对于 P- 帧，参考帧距离被第一为当前 P- 帧及其参考帧之间的 B- 帧（即包含两个 B- 半帧的视频帧）数量。如果不出现 B- 帧，则参考距离为 0。例如，编码器使用可变大小的语法元素（例如在以下 XIV 节详细描述的 REFDIST 语法元素）来编码参考帧距离。

[0213] 在表格 3500 和 3600 中示出的各个示例中，N 的值（用作表格中 SCALE ZONE1_X、SCALEZONE1_Y、ZONE1OFFSET_X 和 ZONE1OFFSET_Y 值的乘数）取决于运动向量范围。例如，经扩展的运动向量范围可通过语法元素 EXTENDED_MV ＝ 1 来用信号表示。如果 EXTENDED_MV ＝ 1，则 MVRANGE 语法元素在图片头中出现，并用信号表示运动向量范围。如果 EXTENDED_MV ＝ 0，则使用缺省运动向量范围。以下的表格 1 示出 N 和 MVRANGE 之间的关系。

[0214] 表格 1 ：图 35 和 36 中 N 的导出

[0215]

MVRANGE	N
0 或缺省值	1
10	2
110	8

111	16

[0216] 表格 3500 和 3600 中示出的各个值可依赖于实现而更改。

[0217] 或者，N 被假设为 1（即缩放不依赖于 N），或者缩放可用一些其它方式来执行。

[0218] B. 隔行扫描 P- 帧

[0219] 在一些实现中，隔行扫描 P- 帧中的宏块可以是 5 种类型之一：1MV、2 半帧 MV、4 帧 MV、4 半帧 MV 和帧内。

[0220] 在 1MV 宏块中，宏块中 4 个亮度块的位移通过单个运动向量表示。相应的色度运动向量可从亮度运动向量导出，以表示运动向量的 2 个 8x8 色度块的每一个的位移。例如，再参看图 23 中示出的宏块排列，1MV 宏块 2300 包括 4 个 8x8 亮度块和 2 个 8x8 色度块。亮度块 (Y1 到 Y4) 的位移通过单个运动向量表示，且相应的色度运动向量可从亮度运动向量导出，以表示 2 个色度块 (U 和 V) 的每一个的位移。

[0221] 在 2 半帧 MV 宏块中，宏块中 4 个亮度块的每个半帧的位移通过不同运动向量描述。例如，图 37 示出，上半帧运动向量描述全部 4 个亮度块的偶数行的位移，下半帧运动向量描述全部 4 个亮度块的奇数行的位移。使用上半帧运动向量，编码器可导出相应的上半帧色度运动向量，它描述色度块偶数行的位移。类似地，编码器可导出下半帧色度运动向量，它描述色度块奇数行的位移。

[0222] 参照图 38，在 4 帧 MV 宏块中，4 个亮度块的每一个的位移通过不同运动向量 (MV1、MV2、MV3 和 MV4) 描述。每个色度块可通过使用 4 个导出色度运动向量 (MV1、MV2、MV3 和 MV4) 来进行运动补偿，这些色度运动向量描述 4 个 4x4 色度子块的位移。每个 4x4 色度子块的运动向量可从空间上相应的亮度块的运动向量中导出。

[0223] 参照图 39，在 4 半帧 MV 宏块中，亮度块的每一个半帧的位移通过两个不同运动向量来描述。亮度块的偶数行被垂直细分以形成 2 个 8x8 区域。对于偶数行，左边区域的位移通过左上方半帧块的运动向量描述，而右边区域的位移通过右上方半帧块的运动向量描述。亮度块的奇数行也被垂直细分以形成 2 个 8x8 区域。左边区域的位移通过左下方半帧块的运动向量描述，而右边区域的位移通过右下方半帧块的运动向量描述。每个色度块也可用与亮度块相同的方式分成 4 个区域，且每个色度块区域可使用导出运动向量进行运动补偿。

[0224] 对于帧内宏块，运动假设为 0。

[0225] 一般而言，计算隔行扫描 P- 帧中当前宏块的运动向量预测值的过程包括两个步骤。首先，从其相邻宏块收集当前宏块的三个候选运动向量。例如，在一实现中，候选运动向量基于图 40A-40B 中示出的排列（和首行宏块等的各种特定情形）来收集。或者，候选运动向量可在一些其它顺序或排列中收集。其次，当前宏块的运动向量预测值从候选运动向量集中计算。例如，预测值可使用 3 个预测值的中值、或通过其它方法来计算。

[0226] 对于有关隔行扫描 P- 帧的宏块的预测值计算和色度运动向量导出的其它细节，参见申请号为 60/501,081 题为“Video Encoding and Decoding Tools and Techniques”（视频编码和解码工具及技术）并于 2003 年 9 月 7 日提交的美国临时专利申请，如以下 XIV 节所述。

[0227] IV. 逐行扫描视频帧的双向预测

[0228] 如上所述，逐行扫描 B- 帧中的宏块可使用 5 种不同预测模式来预测：正向、反向、直接、插值和帧内。编码器选择并用信号表示在宏块级别或一些其它级别上比特流中的不同预测模式。在正向模式中，当前逐行扫描 B- 帧中的宏块从时间上在先的锚中导出。在反向模式中，当前逐行扫描 B- 帧中的宏块从时间上后续的锚中导出。用直接或插值模式预测的宏块将正向和反向锚用于预测。因为有直接和插值模式的两个参考帧，所以对于每个宏块通常都有至少两个运动向量（显式编码或导出的）。（用于逐行扫描 B- 帧的编码、信号表示和解码的各个方面也可用于隔行扫描 B- 帧，如下所述。）

[0229] 在一些实现中，编码器通过使用分数值缩放正向锚的共处运动向量，隐式地用直接模式导出运动向量。该分数可反映当前逐行扫描 B- 帧在通过其锚形成的间隔内的相对时间位置，但不需要反映真实的帧间距离。因而，编码器无需采取固定速度。这使得编码器有附加的自由度，来通过改变来自“实际”时间位置的分数准确并容易地描述锚和当前逐行扫描 B- 帧之间的真实运动，以便于改进运动补偿预测。变量 BFRACTION 表示可在比特流中发送（例如在图片级别或一些其它级别上）以表示该相对时间位置的不同分数。不同分数是 0 和 1 之间的有限离散值集。

[0230] 再参看图 17，表格 1700 是用于比特流元素 BFRACTION 的可变长度代码 (VLC) 表格。在相同两个锚之间的逐行扫描 B- 帧中对 BFRACTION 的唯一性没有限制；具有相同锚的不同逐行扫描 B- 帧可具有相同的 BFRACTION 值。表格 1700 中的代码可改变或重新安排，以用不同代码表示不同分数。未在表格 1700 中示出的其它可能代码（例如，1111110 或 1111111）可被认为是无效代码，或可用于其它目的。例如，条目 1111110 可用来显式地编码定点格式的 BFRACTION。作为另一示例，条目 1111111 可用于用信号表示特定帧类型（例如，帧内编码的逐行扫描 B- 帧）。

[0231] 再参看图 18，解码器根据伪码 1800 寻找缩放系数。再参看图 19，解码器使用该缩放系数来缩放后续参考图片中共处宏块的运动向量的 x 和 y 元素。伪码 1900 中的函数 Scale_Direct_MV 取输入 MV_X 和 MV_Y，并用直接模式来导出两个运动向量，其中一个运动向量参考正向（先前）锚图片 (MV_X_F、MV_Y_F)，而另一个运动向量参考反向（后续）锚图片 (MV_X_B、MV_Y_B)。

[0232] 逐行扫描 B- 帧中的“跳过”宏块信号表示对给定宏块未出现运动向量预测误差。所预测的运动向量将精确地等同于编码器 / 解码器在重构宏块时使用的运动向量（即不应用运动向量预测误差）。编码器仍然用信号表示宏块的预测模式，因为该宏块可使用直接、正向、反向或插值预测来跳过。

[0233] V. 对隔行扫描 B- 图片的预测编码 / 解码的创新的纵览

[0234] 各所述实施例包括用于编码和解码隔行扫描 B- 图片（例如隔行扫描 B- 半帧、隔行扫描 B- 帧）的技术和工具。各所述实施例实现所述用于编码和 / 或解码双向预测的隔行扫描图片的技术和工具的一种或多种，如下包括但不限于：

[0235] 1. 对于隔行扫描 B- 帧，编码器 / 解码器在隔行扫描 B- 帧的宏块中的上半帧和下半帧之间切换预测模式。

[0236] 2. 对于隔行扫描 B- 帧，编码器 / 解码器通过为先前解码的时间后续锚的共处宏块的上和下半帧的每一个选择一个代表性运动向量，计算当前宏块的直接模式运动向量。该选择至少可部分地基于编码当前隔行扫描 B- 帧的宏块的模式（例如 1MV 模式、2 半帧 MV 模

式等）执行。

[0237] 3. 对于隔行扫描 B- 半帧或隔行扫描 B- 帧，编码器 / 解码器使用 4MV 编码。例如，4MV 可用于单向预测模式（正向或反向模式），但不用于其它可用预测模式（例如直接、插值）。

[0238] 4. 对于隔行扫描 B- 半帧或隔行扫描 B- 帧，正向运动向量使用来自正向运动向量缓冲区的先前重构（或估计）的正向运动向量进行预测，而反向运动向量使用来自反向运动向量缓冲区的先前重构（或估计）的反向运动向量进行预测。结果的运动向量被添加到相应的缓冲区中，且运动向量缓冲区中的空穴可用估计的运动向量值来填充。

[0239] a. 对于隔行扫描 B- 帧，当正向预测用来预测运动向量且运动向量被添加到正向缓冲区中时，反向缓冲区中的相应位置以仅将反向运动向量用作预测值的预测运动向量来填充（“空穴填充”）。类似地，当反向预测用来预测运动向量且运动向量被添加到反向缓冲区中时，正向运动向量缓冲区中的相应位置以仅将正向运动向量用作预测值的预测运动向量来填充。

[0240] b. 对于隔行扫描 B- 半帧，要为空穴填充在不同极性的运动向量（例如“相同极性”或“相反极性”）之间选择，编码器 / 解码器选择主极性半帧运动向量。锚和当前帧之间的距离使用各种语法元素计算，且计算出的距离被用于缩放参考半帧运动向量。

[0241] 5. 对于隔行扫描的 B- 半帧，编码器 / 解码器使用“自参考”帧。例如，当前帧中的第二个 B- 半帧在运动补偿预测中参考来自当前帧的第一 B- 半帧。

[0242] 6. 对于隔行扫描的 B- 半帧，编码器发送指示用于隔行扫描 B- 半帧中的一个或多个宏块的预测模式是正向还是非正向的二进制信息（例如，在经压缩位平面的 B- 半帧级别上）。解码器执行相应的解码。

[0243] 7. 对于隔行扫描的 B- 半帧，如果下一锚图片的相应半帧中的相应宏块使用四个运动向量编码，则编码器 / 解码器使用利于主极性的逻辑来选择直接模式的运动向量。

[0244] 8. 帧内编码半帧：当没有好的运动补偿对 B 半帧可能时，它可被编码为帧内（即非预测的）B- 半帧（“BI- 半帧”）。

[0245] 各种所述技术和工具可彼此组合、或与其它技术组合、或可单独使用。

[0246] VI. 切换隔行扫描 B- 帧中半帧编码宏块内的预测模式

[0247] 在一些实现中，编码器在隔行扫描 B- 帧的宏块内执行预测模式切换。例如，编码器允许在隔行扫描 B- 帧的宏块中从上半帧去到下半帧时，预测模式可从正向切换成反向，或从反向切换成正向。与用一种预测方向模式编码整个宏块相反，预测方向模式的组合被用来编码单个宏块。在宏块的各个半帧中改变预测方向模式的能力在许多情形中导致隔行扫描 B- 帧的更有效编码。

[0248] 图 41 示出使用不同预测模式预测隔行扫描 B- 帧的半帧编码宏块中的各个半帧的运动向量的一种技术 4100。在 4110，在隔行扫描 B- 帧中，编码器 / 解码器使用第一预测模式预测半帧编码宏块中第一半帧的运动向量。在一些实现中，该“第一半帧”可以是上半帧或下半帧，对其的判定独立地用信号表示。在 4120，编码器 / 解码器使用一不同预测模式预测同一宏块中第二半帧的运动向量。

[0249] 例如，对于使用两个运动向量半帧编码的宏块，上半帧可以是正向预测的（即，上半帧运动向量参考先前的锚图片），而下半帧可以是反向预测的（即，下半帧参考后续的锚

图片）。在一些实现中，隔行扫描 B- 帧中的半帧编码宏块不使用 4 个运动向量编码。或者，如果该宏块使用 4 个运动向量进行半帧编码（例如每个半帧两个运动向量），则上半帧的这两个运动向量将参考一个锚（正向或反向），而下半帧运动向量将参考另一个锚。

[0250] 这种预测模式的切换在该宏块类型不是以直接或插值开始的情形中仅需一个附加比特，如在以下隔行扫描 B- 帧的伪码中进一步示出：

[0251] IfMB 是半帧编码的 AND MB 类型是正向或反向 then

[0252] IfMVSwitch ＝ 1 then 预测模式在上半帧和下半帧之间切换（从正向到反向或反之）

[0253] 因此，将预测模式切换限于正向和反向模式避免对用信号表示第二模式的更多比特的需要，因为该第二模式从第一模式（在先前用信号表示的）和切换值中隐含。

[0254] 如果在由隔行扫描 B- 帧的宏块覆盖的区域中有较高运动，则宏块有可能用半帧模式编码。在这些情形中，正向或反向预测更可能比直接或插值模式（包括像素平均）给出准确的运动补偿结果。因为在平滑中平均了各个结果（例如损失了伴随高运动的高频率元素），所以直接和插值模式并非是编码这些宏块的最佳方法。试验结果表明，因为在半帧编码宏块内的半帧级别上将全部四种预测模式用信号表示为切换选项的增加开销使其低效。

[0255] 或者，编码器可在隔行扫描 B- 帧的半帧编码宏块内切换两种以上的预测模式，或可在不同预测模式之间切换。

[0256] VII. 计算隔行扫描 B- 帧中的直接模式运动向量

[0257] 在一些实现中，编码器 / 解码器缓存来自经先前解码锚的 I- 帧或 P- 帧（它是时间上正向的参考帧，用作反向预测参考帧）的运动向量，并选择一个或多个缓存运动向量，以用于计算隔行扫描 B- 帧中当前宏块的直接模式运动向量。例如，编码器 / 解码器缓存来自锚帧的每个宏块的上半帧和下半帧的每一个的代表性运动向量，并使用一个或多个所缓存的运动向量来计算当前直接模式宏块的运动向量。该选择至少部分地基于当前宏块的编码模式执行（例如 1MV 模式、2 半帧 MV 模式等）。

[0258] 图 42 示出用于在一实现中计算隔行扫描 B- 帧中宏块的直接模式运动向量的一种技术 4200。在 4210，编码器 / 解码器缓存在先前重构的时间上将来的锚帧中共处宏块的每个宏块的多个运动向量。如果该共处宏块仅具有一个运动向量，则如果需要，该运动向量将被缓存为共处宏块的各个块的运动向量值。在 4220，编码器 / 解码器部分地依赖于当前宏块所需运动向量的数量，选择共处宏块的一个或多个缓存运动向量，用于隔行扫描 B- 帧中当前宏块的直接模式预测。

[0259] 在一实现中，解码器缓存共处宏块中的两个运动向量，或缓存来自将来锚帧的最大可能数量的经解码亮度运动向量的一半。锚帧中的各宏块可用不同方法编码，其中每个宏块最多达 4 个运动向量，但只能缓存最多达两个运动向量，如下所述。而且，为当前宏块产生的正向 / 反向运动向量对的数量取决于当前宏块的编码模式，而不是仅仅取决于经先前解码的将来锚帧的共处宏块的编码模式。

[0260] 例如，如果当前直接模式宏块是 1MV 编码的，则解码器从锚帧的共处宏块的上半帧中取得被缓存的运动向量，并产生一对直接运动向量 - 一个正向另一个反向。如果当前直接模式宏块是半帧编码的，则解码器从锚帧的共处宏块中取得被缓存的上半帧和下半帧运动向量，并产生两对运动向量，一共有当前直接模式宏块的 4 个运动向量 - 对每个半帧都

有一个正向另一个反向的运动向量。

[0261] 图 43 示出经先前解码的时间上将来的锚帧的共处宏块 4300 的各个块的运动向量 MV1、MV2、MV3 和 MV4。如果该共处宏块是 1MV 宏块，则 MV1、MV2、MV3 和 MV4 都相等。如果该共处宏块是 2 半帧 MV 宏块，则 MV1 和 MV2 等于一个值，而 MV3 和 MV4 等于另一个值。如果该锚帧的共处宏块是 4 半帧 MV 或 4 帧 MV 宏块，则 MV1、MV2、MV3 和 MV4 可能都是不同的值。然而，即使 MV1、MV2、MV3 和 MV4 都可用，解码器仍然仅缓存 MV1 和 MV3。

[0262] 在图 43 所示示例中，解码器缓存 MV1 和 MV3。如果当前宏块使用 1MV 模式，则解码器选择 MV1 来计算当前宏块的正向和反向直接模式运动向量，并略去 MV3。如果当前宏块使用 2 半帧 MV 模式，则解码器使用 MV1 和 MV3 来计算 4 个直接模式运动向量。该运算产生当前宏块的上下半帧的运动的良好表示。

[0263] 在已选择来自锚帧中共处宏块的运动向量时，解码器应用缩放逻辑来导出相应的正向和反向指示运动向量，用于 B 帧宏块的直接模式预测。例如，解码器可应用图 19 中的函数 Scale_Direct_MV。或者，解码器应用不同的缩放函数。

[0264] 或者，编码器 / 解码器可缓存每个锚帧宏块的 4 个运动向量。例如，如果当前宏块是 1MV 编码的，则编码器 / 解码器可取锚帧中共处宏块的左上运动向量，并产生一对直接运动向量，或者可取锚帧宏块的 4 个运动向量的均值。如果当前宏块是半帧编码的，则编码器 / 解码器可取左上和左下运动向量，并产生两对（一个半帧一对），或者可取锚帧宏块的上运动向量的均值和下运动向量的均值。

[0265] 当锚帧中的共处宏块为帧内、或当锚帧为 I- 帧时，直接模式运动向量被视为 (0, 0)。

[0266] VIII. 隔行扫描 B- 半帧和隔行扫描 B- 帧中的 4MV 编码

[0267] 在一些实现中，编码器使用 4 运动向量 (4MV) 编码模式编码隔行扫描的 B- 半帧和隔行扫描的 B- 帧。4MV 编码可允许复杂运动轨迹的表示比 1 运动向量 (1MV) 编码更准确（例如，通过允许宏块中的 4 个亮度块独立地进行预测和运动补偿）。使用 4MV 可受限于某些预测模式。例如，在一些实现中，编码器将 4MV 用于正向和反向模式（包括半帧和帧变化两者），而不用于直接或插值模式。当 4MV 不用于逐行扫描 B- 帧时，这不同于逐行扫描编码模式。

[0268] 直接和插值模式涉及计算经运动补偿预测时的像素平均，它用于平滑细微细节。如果这种平滑是可接受的，则可能可使用 1MV 模式而不是 4MV 模式，因为编码 1MV 较容易，且 1MV 可用来准确地描述平滑运动轨迹。实验已显示，在隔行扫描 B- 半帧和隔行扫描 B- 帧的宏块中使用 4MV 模式，而将 4MV 模式限于正向和反向预测宏块中是有利的。有利于将 4MV 限于正向和反向模式中的另一个因素是组合 4MV 与直接或插值模式将导致每种情形中一共有 8 个运动向量。信号表示开销（用于插值模式）和实现以及关联于 8 个运动向量的解码复杂性通常抵销了准确性优点。此外，当通常在较高质量设置上编码的（即较不强烈量化的）P- 图片通常可仅将一个或四个运动向量用于运动补偿时，用 8 个运动向量来编码隔行扫描 B- 图片通常是不实用的。

[0269] 将 4MV 限于某些预测模式还具有其它优点。例如，如果 4MV 仅限于正向和反向预测模式，并且如果正向 / 非正向模式决定已用信号表示（例如用诸如在以下 XI 节所述的位平面编码技术），编码器无需发送任何附加比特来用信号表示 4MV 宏块的预测模式。

[0270] 以下伪码可应用于隔行扫描 B- 半帧的宏块，其中正向 / 非正向决定是位平面编码的，并在任何宏块级别信息之前发送（例如在图片级别上发送）：

[0271] IfMB 是 4MV 编码的 AND 预测模式是非正向的

[0272] then 预测模式＝反向（不发送任何更多比特来用信号表示模式）

[0273] 在一些实现中，直接 / 非直接预测模式决定在任何宏块级别信息之前发送（例如在图片级别上的经压缩位平面中）。（对于有关编码直接 / 非直接信息的更多信息，参见序列号为 10/622,378 的题为“Advanced Bi-Directional Predictive Coding of Video Frames”（视频帧的高级双向预测编码）并于 2003 年 7 月 18 日提交的美国专利申请。）以下伪码可应用于隔行扫描 B- 帧的宏块，其中 4MV 在这些实现中受限于正向和反向模式：

[0274] IfMB 是 4MV 编码的 AND 预测模式是非正向的

[0275] then 发送一附加比特来用信号表示预测模式（正向或反向）

[0276] 或者，4MV 用于不同于或除正向或反向模式之外的预测模式、不用于正向模式、不用于反向模式、或不用于任何预测模式。例如，在一些实现中，4MV 用于隔行扫描 B- 半帧，但不用于隔行扫描 B- 帧。在其它可选实现中，其它代码或代码长度可用于用信号表示结合 4MV 编码的预测模式。

[0277] IX. 使用分开的正向和反向运动向量缓冲区来预测隔行扫描 B- 图片中的运动向量

[0278] 隔行扫描 B- 图片的运动向量使用分开的正向和反向运动向量上下文来预测。一般而言，正向运动向量使用存储在正向运动向量缓冲区中的运动向量来预测，而反向运动向量使用存储在反向运动向量缓冲区中的运动向量来预测。然后当前宏块的结果运动向量存储在适当的缓冲区中，并可用于其它宏块的后续运动向量预测值中。通常，正向和反向运动向量缓冲区中的相应空间都为每个宏块填充，即使给定宏块仅用正向运动向量（在正向预测宏块的情形中）或仅用反向运动向量（在反向预测宏块的情形中）预测。以下各节描述用于预测隔行扫描 B- 图片（如，隔行扫描 B- 半帧，隔行扫描 B- 帧）中的运动向量，以及用于为“遗漏”的正向或反向运动向量“填充”运动向量缓冲区中的相应空间的技术。

[0279] A. 正向和反向缓冲区

[0280] 当预测隔行扫描 B- 图片的运动向量时，编码器 / 解码器使用正向运动向量缓冲区和 / 或反向运动向量缓冲区中的先前重构的运动向量。在正向模式中，编码器 / 解码器使用来自正向运动向量缓冲区的经重构正向运动向量，来预测用于正向运动补偿的当前运动向量。在反向模式中，编码器 / 解码器使用来自反向运动向量缓冲区的经重构反向运动向量，来预测用于反向运动补偿的当前运动向量。对于直接模式或插值模式宏块，编码器 / 解码器使用正向运动向量缓冲区来预测正向运动向量分量（或可能多个正向运动分量），并使用反向运动向量缓冲区来预测反向运动向量分量（或可能多个反向运动分量）。

[0281] 在重构隔行扫描 B- 图片的运动向量之后，编码器 / 解码器将重构后的正向运动向量缓存在正向运动向量缓冲区中，并将重构后的反向运动向量缓存在反向运动向量缓冲区中。在正向模式中，编码器 / 解码器将重构后的正向运动向量存储在正向运动向量缓冲区中。在反向模式中，编码器 / 解码器将重构后的反向运动向量存储在反向运动向量缓冲区中。对于使用直接或插值预测模式的宏块，编码器 / 解码器将（各）正向运动向量分量存储在正向运动向量缓冲区中，并将（各）反向运动向量分量存储在反向运动向量缓冲区中。

[0282] 例如，如果编码器在隔行扫描 B- 图片中的宏块坐标位置 (12,13) 上编码正向预测宏块时，则编码器计算正向运动向量预测值并在比特流中发送该正向运动向量的残差（假设该宏块未被“跳过”）。解码器解码该残差（即差值），并重构运动向量。编码器 / 解码器将重构后的运动向量插入正向运动向量缓冲区。然后编码器 / 解码器使用运动向量预测逻辑来计算要填充反向运动向量的反向运动向量预测值，并将反向运动向量置于反向运动向量缓冲区中的位置 (12,13) 上。例如，在三者中值的预测情形中，编码器 / 解码器可取位置 (11,13)、(12,12) 和 (13,12) 上的缓存反向运动向量的中值（当前正向预测宏块的左、上、右上邻居），来填充 (12,13) 的反向运动向量。

[0283] 图 44 示出使用正向和 / 或反向运动向量缓冲区来预测隔行扫描 B- 图片中当前宏块的运动向量的一种技术 4400。在 4410，取决于要预测的运动向量是正向还是反向运动向量，编码器 / 解码器选择使用正向还是反向运动向量缓冲区。如果当前运动向量是正向运动向量，则编码器 / 解码器在 4420 从正向运动向量缓冲区选择一个候选运动向量预测值集。如果当前运动向量是反向运动向量，则编码器 / 解码器在 4430 从反向运动向量缓冲区选择一个候选运动向量预测值集。在 4440，编码器 / 解码器基于候选运动向量预测值集计算运动向量预测值。例如，编码器 / 解码器计算候选运动向量预测值集的中值。在简单情形中，编码器 / 解码器基于都是 1MV 宏块的预测值计算 1MV 当前宏块的运动向量预测值。更复杂的变化描述如下，其中当前宏块和 / 或邻居宏块具有不同模式。

[0284] 图 45 示出正向运动向量缓冲区 4510 和反向运动向量缓冲区 4520 中的运动向量。在图 45 所示示例中，对于重构宏块 4530-4570，编码器 / 解码器将正向运动向量存储在正向运动向量缓冲区 4510 中，并将反向运动向量存储在反向运动向量缓冲区 4520 中。为了预测当前宏块 4580 的运动向量，编码器 / 解码器使用来自相邻宏块的候选预测值。例如，如果当前宏块 4580 用正向模式预测，则编码器使用正向运动向量缓冲区中的相邻正向运动向量来预测正向运动向量（例如使用三者中值预测），随后用重构后的运动向量值填充正向运动向量缓冲区中的当前宏块位置。为了填充反向运动向量缓冲区 4520 中的相应当前宏块位置，编码器 / 解码器可使用反向运动向量缓冲区中的相邻反向运动向量来预测反向运动向量，并将该预测值置于反向运动向量缓冲区的当前宏块的位置中。

[0285] B. 隔行扫描 B- 帧中的运动向量预测

[0286] 在一些实现中，编码器 / 解码器采用以下方案来预测隔行扫描 B- 帧中宏块（包括其不同半帧）的运动向量，它使用分开的正向和反向运动向量上下文。图 40A-40B 示出从中收集候选运动向量的相邻宏块。

[0287] 如果 1MV 宏块是正向预测的，则编码器 / 解码器从正向运动向量缓冲区的候选运动向量中预测其正向运动向量（例如使用诸如图 40A 和 40B 中或别处示出的三者中值预测和预测模式）。编码器 / 解码器（在添加运动向量预测误差后）将该正向运动向量存储在正向运动向量缓冲区中。编码器 / 解码器通过从反向运动向量缓冲区的候选运动向量中预测反向运动向量来填充“空穴”（例如与正向预测情形中一样），并将该反向运动向量（在此为预测值）存储在反向运动向量缓冲区中。

[0288] 如果 1MV 宏块是反向预测的，则编码器 / 解码器从反向运动向量缓冲区的候选运动向量中预测其反向运动向量（例如与正向预测情形中一样）。编码器 / 解码器（在添加预测误差后）将该反向运动向量存储在反向运动向量缓冲区中。编码器 / 解码器通过从正

向运动向量缓冲区的候选运动向量中预测正向运动向量来填充“空穴”，并将该正向运动向量（在此为预测值）存储在正向运动向量缓冲区中。

[0289] 在正向和反向运动向量缓冲区中略去作为帧内编码宏块的邻居。

[0290] 各种特定情形确定隔行扫描 B- 帧中 1MV 和半帧编码 2MV 宏块的组合。如果当前 1MV 宏块的位置 A、B 或 C 中的相邻宏块是半帧编码的 2MV 宏块，则编码器 / 解码器取 2MV 宏块的半帧运动向量的均值作为该位置的运动向量预测值。

[0291] 对于正向预测的当前 2 半帧 MV 宏块，例如对于两个正向预测半帧运动向量的每一个，来自邻居的候选运动向量从正向运动向量缓冲区中收集。编码器 / 解码器基于相邻宏块的编码模式（例如帧内、1MV、2 半帧 MV）选择一个候选运动向量集，这些相邻宏块存储在正向运动向量缓冲区中。如果相邻宏块存在且不是帧内编码的，则编码器 / 解码器注意该宏块的运动向量以添加到候选集中去。在一些实施例中，编码器 / 解码器继续如下动作。对于上半帧正向运动向量，如果位置 A、B 或 C 中的相邻宏块是 1MV 宏块，则编码器将来自正向运动向量缓冲区的相应位置的宏块的运动向量添加到候选集中。对于是 2 半帧 MV 宏块的位置 A、B 或 C 中的相邻宏块，编码器 / 解码器将来自正向运动向量缓冲区的相应位置的上半帧 MV 添加到该集中。

[0292] 对于下半帧正向运动向量，如果位置 A、B 或 C 中的相邻宏块是 1MV 宏块，则编码器将来自正向运动向量缓冲区的相应位置的宏块的运动向量添加到候选集中。对于是 2 半帧 MV 宏块的位置 A、B 或 C 中的相邻宏块，编码器 / 解码器将来自正向运动向量缓冲区的相应位置的下半帧 MV 添加到该集中。

[0293] 为了计算 2 半帧 MV 宏块的半帧运动向量的预测值，编码器 / 解码器然后计算候选集的中值。

[0294] 为了计算 2 半帧 MV 宏块的反向预测运动向量，逻辑与正向预测情形相同，但来自邻居的候选运动向量是从反向运动向量缓冲区中收集的。

[0295] 再一次，对于运动向量预测，略去位置 A、B 或 C 中帧内编码的邻居。

[0296] 在重构 2 半帧 MV 宏块的运动向量之后（例如通过添加运动向量差值信息），重构后的实际运动向量按适合重构后运动向量的预测方向，被置入正向运动向量缓冲区或反向运动向量缓冲区。运动向量缓冲区的用于缺少方向的相应空槽通过计算缺少方向的运动向量预测值并将该运动向量预测值存储在空槽中来填充。

[0297] 如果使用预测模式切换（参见以上 VI 节），一例外牵涉到隔行扫描 B- 帧内半帧编码宏块的空穴填充。在该情形中，给定半帧编码的 2MV 宏块具有一个正向运动向量和一个反向运动向量。在重构隔行扫描 B- 帧的半帧编码宏块之后，在半帧编码宏块在上半帧和下半帧之间切换预测方向时，编码器 / 解码器用正向运动向量填充正向运动向量缓冲区的上下运动向量“槽”，并用反向运动向量填充反向缓冲区的上下运动向量槽。尽管正向运动向量仅对一个半帧（例如上半帧）发送，编码器将同一运动向量置入正向运动向量缓冲区的上下半帧运动向量槽。类似地，尽管反向运动向量仅对下半帧发送，编码器将它置入反向运动向量缓冲区的上下半帧槽。

[0298] 例如，图 46 示出正向运动向量缓冲区 4610 和反向运动向量缓冲区 4620 中重构宏块 4680 的上下半帧的运动向量。在图 46 所示示例中，对于重构宏块 4630-4670，编码器 / 解码器将正向运动向量存储在正向运动向量缓冲区 4610 中，并将反向运动向量存储在反

向缓冲区 4620 中。重构宏块 4680 是用预测切换进行半帧编码的，且其上半帧运动向量被存储在正向或反向运动向量缓冲区中（取决于上半帧运动向量的预测方向）的上下位置。宏块 4680 的下半帧运动向量被存储在其它运动向量缓冲区的上下位置中。在该示例中，重构宏块 4680 使用预测模式切换。尽管正向运动向量和方向运动向量都仅对一个半帧发送，但编码器将同一运动向量置入相应正向和反向运动向量缓冲区的上下半帧运动向量槽中。

[0299] 如果当前宏块被插值，则编码器 / 解码器使用正向运动向量缓冲区来预测正向运动向量（或 2 半帧 MV 宏块的正向运动向量），使用反向运动向量缓冲区来预测反向运动向量（或 2 半帧 MV 宏块的反向运动向量），并（在添加了一计算好就添加的预测误差之后）将该正向和反向运动向量分别存储在正向和反向运动向量缓冲区中。

[0300] 如果宏块是在隔行扫描 B- 帧中直接预测的，则编码器 / 解码器可使用以上 VII 节描述的技术。

[0301] 在一些实现中，1MV 宏块、2 半帧 MV 宏块和帧内宏块被允许用于隔行扫描 B- 帧（但不是其它 MV 宏块类型），因为只需要确定较少的当前 / 邻居模式组合而简化了用于预测运动向量的逻辑。或者，允许其它和 / 或附加 MV 模式，诸如 4 帧 MV 宏块和 4 半帧 MV 宏块。例如，图 64、69 和 70 中示出的伪码的一部分可用来确定隔行扫描 B- 帧的这种其它组合。

[0302] C. 隔行扫描 B- 半帧的运动向量预测

[0303] 一般而言，对于隔行扫描 B- 半帧，先前重构（或导出）的正向半帧运动向量被用作当前正向半帧运动向量的预测值，且先前重构（或导出）的反向半帧运动向量被用作当前反向半帧运动向量的预测值。在正向或反向模式中，当前正向或反向半帧的运动向量被添加到适当的运动向量缓冲区中，且另一（缺少）方向的运动向量（例如正向模式中的反向方向，或反向模式中的正向方向）被导出用作将来使用的预测值。

[0304] 在一些实现中，半帧运动向量预测选择根据以上 III.A.2 节的详细描述和以下 XIV.B.3 节描述的双参考半帧运动向量预测逻辑进行。例如，图 33A-33F 中示出的伪码用来计算隔行扫描 B- 半帧的宏块的两个半帧的正向运动向量预测值，且一个运动向量预测值被选择用于重构正向半帧的运动向量。然后将重构的运动向量置于正向运动向量缓冲区中。该伪码也用来计算该宏块的两个半帧的反向运动向量预测值，且一个预测值被选择用作反向运动向量缓冲区的填充值。对于隔行扫描 B- 半帧，为了填充缺少方向的运动向量缓冲区中的“空穴”，编码器 / 解码器在相同极性和相反极性的运动向量预测值之间选择。这种极性之间的选择因为两个预测值在给定缺少方向中产生 - 一个与当前半帧极性相同，另一个与当前半帧极性相反。因此，在一些实现中，编码器 / 解码器选择用于缺少方向运动向量的主要或“主”极性预测值。这样，正向和方向运动向量的完整集就被提供用于运动向量预测。或者，确定主极性、首先进行预测值选择、并只计算选定运动向量预测值。

[0305] 在一实现中通过从不同极性的半帧运动向量预测值中选择来进行实际值缓存和空穴填充的过程如图 47 中的伪码 4700 所示。伪码 4700 示出，在空穴填充预测期间，没有实际运动向量被提供为缺少方向，因此具有主极性的经预测的缺少方向运动向量由编码器 / 解码器来选择。

[0306] 在一些实现中，隔行扫描 B- 半帧的运动向量预测的整个方案如下所述。

[0307] 如果宏块是正向预测的，则编码器 / 解码器从正向运动向量缓冲区的候选相同和 / 或相反极性运动向量（例如大多数情形中使用来自左、上和右上邻居的三者中值预测）或

从缓存运动向量中导出的运动向量中预测其正向运动向量。编码器 / 解码器将重构后的正向运动向量存储在正向运动向量缓冲区中，计算主反向运动向量预测值（类似于用来自反向运动向量缓冲区的空间邻域的三者中值预测的），并将其存储在反向运动向量缓冲区中的相应位置。

[0308] 如果宏块是反向预测的，则编码器 / 解码器从反向运动向量缓冲区的候选相同和 / 或相反极性运动向量（例如大多数情形中使用来自左、上和右上邻居的三者中值预测）或从缓存运动向量中导出的运动向量中预测其反向运动向量。编码器 / 解码器将重构后的反向运动向量存储在反向运动向量缓冲区中，计算主正向运动向量预测值（类似于用来自正向运动向量缓冲区的空间邻域的三者中值预测的），并将其存储在正向运动向量缓冲区中的相应位置。

[0309] 如果该宏块是插值的，则编码器 / 解码器使用正向运动向量缓冲区来预测正向运动向量分量，使用反向运动向量缓冲区来预测反向运动向量分量，并（在添加了一计算好就添加的预测误差之后）将重构后的正向和反向运动向量分别存储在正向和反向运动向量缓冲区中。

[0310] 如果宏块是直接预测的，则编码器 / 解码器计算当前半帧的直接模式运动向量，并将正向和反向运动向量分量存储在相应的运动向量缓冲区中。

[0311] 在运动向量预测中，略去位置 A、B 或 C 中帧内编码的邻居。

[0312] 各种特定情形确定隔行扫描 B- 半帧中 1MV 和 4MV 宏块的组合。图 6A-10 示出用于逐行扫描 P- 帧的运动向量预测的预测值模式。这些相同模式示出被视为用于混合 MV 的隔行扫描 B- 半帧的 1MV 或 4MV 宏块的运动向量的运动向量预测的候选运动向量的块或宏块的位置。对于帧是一个宏块宽的特定情形，预测值总是 Predicator A（顶部预测值）。各种其它规则解决其它特定情形，诸如首行 4MV 宏块、首行 1MV 宏块、和帧内编码预测值。

[0313] 图 6A-10 中所示的预测值模式用来使用来自正向运动向量缓冲区中各位置的候选值进行正向预测，并使用来自反向运动向量缓冲区中位置的候选值进行反向预测。此外，图 6A-10 中所示的预测值模式结合以上所述的对隔行扫描 B- 半帧的双参考半帧运动向量预测逻辑来使用。

[0314] 图 6A 和 6B 示出被视作用于混合 MV 的隔行扫描 B- 半帧中的 1MV 当前宏块的候选运动向量预测值的块的位置。相邻的各个宏块可以是 1MV 或 4MV 宏块。图 6A 和 6B 示出假设邻居是 4MV 的候选运动向量的位置（即预测值 A 是当前宏块上面的宏块中块 2 的运动向量，而预测值 C 是当前宏块左侧紧邻的宏块中块 1 的运动向量）。如果邻居的任一个是 1MV 宏块，则图 5A 和 5B 中示出的运动向量预测值被视为整个宏块的运动向量预测值。如图 6B 所示，如果宏块是该行中的最后一个宏块，则预测值 B 来自左上方宏块的块 3 而不像其它情形一样来自右上方宏块的块 2。

[0315] 图 7A-10 示出被视作用于混合 MV 的隔行扫描 B- 半帧的 4MV 宏块中 4 个亮度块的每一个的候选运动向量预测值的块的位置。图 7A 和 7B 是示出被视作用于位置 0 上一个块的候选运动向量预测值的块的位置的示图；图 8A 和 8B 是示出被视作用于位置 1 上一个块的候选运动向量预测值的块的位置的示图；图 9 是示出被视作用于位置 2 上一个块的候选运动向量预测值的块的位置的示图；而图 10 是示出被视作用于位置 3 上一个块的候选运动向量预测值的块的位置的示图。再一次，如果邻居是 1MV 宏块，则该宏块的运动向量预测值

用于该宏块的各个块。

[0316] 对于宏块是行中第一宏块的情形，块 0 的预测值 B 与该行中剩余宏块的块 0 进行不同的处理（参见图 7A 和 7B）。在该情形中，预测值 B 从当前宏块上面紧邻宏块的块 3 中取得，而不像其它情形一样从当前宏块左上方的宏块的块 3 中取得。类似地，对于宏块是行中最后一个宏块的情形，块 1 的预测值 B 进行不同的处理（参见图 8A 和 8B）。在该情形中，预测值从当前宏块上面紧邻宏块的块 2 中取得，而不像其它情形一样从当前宏块右上方的宏块的块 2 中取得。一般而言，如果该宏块在第一宏块列中，则块 0 和 2 的预测值 C 被设置为等于 0。

[0317] 再一次，对于运动向量预测，略去位置 A、B 或 C 中帧内编码的邻居。

[0318] 在重构 4MV 宏块的运动向量之后（例如通过添加运动向量差值信息），重构后的实际运动向量按适合重构后运动向量的预测方向，被置入正向运动向量缓冲区或反向运动向量缓冲区。运动向量缓冲区的用于缺少方向的相应空槽通过计算缺少方向的相同和相反极性的运动向量预测值、在不同极性运动向量预测值之间选择、并将该运动向量预测值存储在空槽中来填充。

[0319] 再参看图 34A 和 34B，对于运动向量预测，编码器 / 解码器使用伪码 3400 中所示的缩放运算，将一个半帧运动向量预测值从另一个半帧运动向量预测值导出。示出两个可能的值集，其中当前半帧是隔行扫描视频帧中的第一半帧的情形在图 35 的表格 3500 中示出，而当前半帧是隔行扫描视频帧中的第二半帧的情形在图 36 的表格 3600 中示出。在表格 3500 和 3600 中，SCALEOPP、SCALESAME1、SCALESAME2、SCALEZONE1_X、SCALEZONE1_Y、ZONE1OFFSET_X 和 ZONE1OFFSET_Y 取决于参考帧距离。

[0320] 在一些实现中，使用分数编码来计算用于隔行扫描 B- 半帧中的正向和反向参考的参考帧距离。BFRACTION 语法元素（它用信号表示隔行扫描 B- 半帧的正向或反向预测模式宏块，而不只是隔行扫描 B- 帧的直接模式宏块）用来导出正向和反向参考图片距离，如以下伪码所示：

[0321] 正向参考帧距离 (FRFD) ＝

[0322] NINT((BFRACTION 分子)/(BFRACTION 分母)* 参考帧距离 -1

[0323] If(FRFD ＜ 0)then FRFD ＝ 0

[0324] 反向参考帧距离 (BRFD) ＝

[0325] 参考帧距离 -FRFD-1（其中 NINT 是最近整数操作符）

[0326] BFRACTION 分子和分母从 BFRACTION 语法元素中解码。元素 BFRACTION 可用来表示可在比特流中发送的不同分数（例如在隔行扫描 B- 半帧的帧级别上）。该分数在 0 和 1 之间的有限离散值集中取值，并在通过其锚形成的间隔内标示 B- 图片的相对时间位置。

[0327] 对于具有隔行扫描 B- 半帧的帧中的第二半帧的正向预测和反向预测，编码器 / 解码器根据图 34A 和 34B 中的伪码 3400 执行运动向量缩放。然而，在一些实现中，执行第一半帧的反向运动向量预测的编码器 / 解码器使用如在图 48 中所示伪码 4800 中定义的函数 scaleforopposite_x、scaleforopposite_y、scaleforsame_x、scaleforsame_y。在一实现中用于第一隔行扫描 B- 半帧的 SCALESAME、SCALEOPP1、SCALEOPP2、SCALEZONE1_X、SCALEZONE1_Y、ZONE1OFFSET_X 和 ZONE1OFFSET_Y 在图 49 的表格 4900 中示出。在表格 4900 中，变量 N 和运动向量范围之间的关系与以上参考图 35 和 36 和表格 1 所述的关系相同。

[0328] 或者，参考帧距离用另一种方法计算，或者缩放根据不同算法执行。例如，缩放与 N 的值无关地执行（即 N 取为 1）。

[0329] X. 具有隔行扫描 B- 半帧的“自参考”帧

[0330] 具有隔行扫描 B- 半帧的帧被编码为两个独立（且某种程度上独立编码的）的半帧。上半帧由帧的偶光栅行（从行 0 开始）组成，而下半帧由帧的奇光栅行组成。因为“半帧图片”中的半帧可独立解码，所以它们不需要以任何预先设置顺序发送。例如，编码器可先发送下半帧再发送上半帧，或反之。在有些实现中，两个半帧的顺序由“先上半帧”的语法元素表示，该语法元素取决于解码帧的两个半帧的准确时间顺序而为真或假。

[0331] 现有的编码器和解码器已将前后锚帧（例如 I- 或 P- 帧）或前后锚帧中的半帧用作“参考”图片，以执行对当前 B- 图片的运动补偿。现有的编码器和解码器还限制 B- 图片或其任何部分用作任何图片的运动补偿参考。然而，在所述技术和工具的一些实现中，一种或多种这些“规则”被放松。

[0332] 例如，在一些实现中，第一隔行扫描 B- 半帧参考来自前后锚图片的第一和第二半帧。第二隔行扫描 B- 半帧参考作为“相反极性”半帧的来自当前图片的第一隔行扫描 B- 半帧和作为“相同极性”半帧的前一锚帧的相同极性半帧，以及来自下一锚图片的第一和第二半帧。

[0333] 图 50B 是示出隔行扫描视频帧 B2 中两个隔行扫描 B- 半帧的每一个的参考半帧的示图。在图 50B 所示示例中，要解码的第一 B- 半帧（在此为上半帧）被允许参考正向（时间上过去）锚 P1 中的两个参考半帧和来自反向（时间上将来）锚 P3 中的两个参考半帧，总共为 4 个参考半帧。B2 的要解码的第二隔行扫描 B- 半帧被允许参考来自同一隔行扫描视频帧（因而打破了不允许 B- 图片的各部分用作参考的惯例）的第一半帧和来自前一锚 P1 的一个参考半帧，以及来自将来锚 P3 的两个半帧。为作比较，图 50A 示出隔行扫描视频帧的隔行扫描 P- 半帧遵守的惯例。

[0334] 实现这些隔行扫描 B- 半帧参考规则的技术和工具可提供更好的压缩。隔行扫描视频的半帧编码对于编码高运动是最为有效的，高运动即为在上半帧和下半帧之间有相当运动时。例如，在该情形中，帧的上（且首先编码的）半帧对于同一帧下半帧中的像素将是比从前一帧中（来自较远的时间距离）取得的上半帧好得多的预测值。由于它们之间较大的时间距离，当运动较高时这些时间上将来的预测值提供弱得多的预测。此外，对于时间上更远的预测值来说闭塞的可能性放大，这导致更多编码昂贵的帧内编码宏块。特别地，实验证实允许帧的第二个时间隔行扫描 B- 半帧参考同一帧的第一时间隔行扫描 B- 半帧可产生显著的压缩增益。

[0335] XI. 隔行扫描 B- 半帧中正向模式的位平面编码

[0336] 如以上 X 节所述，在一些实现中，当前帧的第二已编码隔行扫描 B- 半帧可参考当前帧的第一已编码隔行扫描 B- 半帧。该“自参考”技术在具有高运动的帧的隔行扫描 B- 半帧中是有效的，因为当前帧的时间上较近的 B- 半帧常常是比时间上较远的锚半帧更好的预测值。当具有隔行扫描 B- 半帧的帧具有较高运动，且时间上第二隔行扫描 B- 半帧首选时间上第一隔行扫描 B- 半帧作为预测参考时，第二隔行扫描 B- 半帧中宏块的更有效预测模式将常常是“正向”模式。

[0337] 因为隔行扫描 B- 半帧中的正向模式预测是用于降低比特率的有效工具，所以特

别是在低比特率情形中，减少信号表示开销以降低用信号表示正向模式预测的整体成本是有利的。因此，在一些实施例中，编码器使用统一的位平面编码技术来编码正向模式预测信息。例如，编码器编码压缩位平面中的正向模式预测信息，其中位平面中的每个比特关联于一宏块，且每个比特的值用信号表示该宏块是用正向模式还是用非正向预测模式编码。

[0338] 经压缩位平面可在帧级别、半帧级别或在一些其它级别上发送。与用于隔行扫描 B- 半帧的其它预测模式相比，位平面编码技术偏向于利用正向模式。例如，如果隔行扫描 B- 半帧上的大多数宏块使用正向预测，则编码器通过位平面编码正向 / 非正向决定，可将信号表示开销降为每个宏块小于一个比特。

[0339] 图 51 示出在具有一个或多个位平面编码模式的视频编码器中用于编码隔行扫描 B- 半帧的宏块的正向 / 非正向预测模式决定信息的一种技术 5100。图 52 示出用于解码由具有一个或多个位平面编码模式的视频编码器编码的正向 / 非正向预测模式决定信息的相应技术 5200。。

[0340] 参照图 51，编码器选择用于编码正向 / 非正向预测模式决定信息的位平面编码模式 (5110)。在选择编码模式之后，编码器用所选模式来编码正向 / 非正向预测模式决定信息 (5120)。该编码器在逐个半帧基础上选择位平面编码模式。或者，编码器在一些其它基础上（例如在序列级别）选择位平面编码模式。或者，如果只使用一种位平面编码模式，则不进行位平面编码模式的选择。当编码器完成编码正向 / 非正向预测模式决定信息时 (5130)，正向 / 非正向预测模式决定信息的编码结束。

[0341] 参照图 52，解码器确定编码器使用（并用信号表示）的编码正向 / 非正向预测模式决定信息的位平面编码模式 (5210)。然后该解码器用选定模式解码正向 / 非正向预测模式决定信息。该解码器在逐个半帧基础上确定位平面编码模式。或者，解码器在一些其它基础上（例如在序列级别）确定位平面编码模式。或者，如果只有一种位平面编码模式可用，则不进行位平面编码模式的选择。当解码器完成解码正向 / 非正向预测模式决定信息时 (5230)，正向 / 非正向预测模式决定信息的解码结束。

[0342] 对于有关根据若干组合实现来用信号表示和解码各种位平面编码模式的其它细节，参见以下 XIV 节。对于有关一般位平面编码的更多细节，参见序列号为 10/321,415 题为“Skip Macroblock Coding”（跳过宏块编码）并于 2002 年 12 月 16 日提交的美国专利申请，其公开内容通过引用结合在此。或者，表示正向 / 非正向模式信息的比特可未经压缩地和 / 或在一些其它级别（例如宏块级别）上发送。

[0343] 如果指示的是非正向预测，则编码器指定该宏块的非正向预测模式（例如反向模式、直接模式、插值模式、或帧内模式）。在一些实现中，编码器在宏块级别参照 VLC 表格编码非正向预测模式，如以下表格 2 所示。

[0344]

BMVTYPE VLC	运动预测模式
0	反向
10	直接

[0345]

11	插值

[0346] 表格 2. 运动预测模式 VLC 表格

[0347] 在表格 2 所示示例中，反向模式是较佳的非正向预测模式。编码器用 1- 比特信号表示反向模式，并用 2- 比特信号表示直接和插值模式。或者，编码器使用不同代码来表示不同的预测模式和 / 或首选一种不同的非正向预测模式。

[0348] 在一些实现中，帧内模式由特定运动向量差值来作信号表示，该值用预测模式是帧内模式的编码方式来表示。运动向量差值因此用来推断该宏块是帧内编码的，但是按照惯例编码器将预测类型设置为反向，以便于不会具有任何未定义预测类型。

[0349] XII. 在隔行扫描 B- 半帧中选择用于直接模式的共处运动向量

[0350] 在一些实现中，半帧编码 B- 图片中宏块的直接模式运动向量使用特殊逻辑来选择。对于隔行扫描 B- 半帧中的当前宏块，如果下一锚图片的相应半帧的共处宏块使用 4 个运动向量进行编码，则该逻辑在该共处宏块的最多达四个运动向量中偏向更主要的极性（例如相同或相反）。一旦选择了用于当前宏块的运动向量，编码器 / 解码器就可应用缩放运算以给出直接模式运动向量。

[0351] 在一些实现中，对于隔行扫描 B- 半帧的直接模式 1MV 宏块，编码器 / 解码器基于具有相同极性的参考半帧（例如时间上的下一 P- 半帧）内共处宏块的一个或多个运动向量，计算用于直接模式缩放的运动向量。如果参考半帧中的共处宏块是 1MV 宏块，则编码器 / 解码器使用单个运动向量来导出隔行扫描 B- 半帧中宏块的直接模式运动向量。另一方面，如果参考半帧中的共处宏块是 4MV 宏块，则编码器 / 解码器在选择用于导出隔行扫描 B- 半帧中宏块的直接模式运动向量的运动向量时考虑该 4 个运动向量的极性（偏向主要极性）。在解码隔行扫描 B- 半帧期间编码器 / 解码器在需要时可将该选择逻辑应用于参考半帧中的 4MV 宏块。或者，编码器 / 解码器可在解码参考半帧之后应用该选择逻辑，然后仅缓存要在后来隔行扫描 B- 半帧解码中使用的值。

[0352] 例如，对于参考半帧中的共处 4MV 宏块，如果（4 者中）来自相同极性半帧的运动向量数量超过来自相反极性半帧的运动向量数量，则如果相同极性运动向量的数量分别为 4、3、2 或 1，编码器 / 解码器可使用四者中值、三者中值、二者中值或相同极性半帧运动向量的值来计算在直接模式隔行扫描 B- 半帧解码中使用的运动向量。否则，如果来自相反极性半帧的运动向量数量超过来自相同极性半帧的运动向量，则编码器 / 解码器可使用类似运算来从相反极性半帧的运动向量中得到代表性的运动向量，以用于直接模式的隔行扫描 B- 半帧解码。如果共处宏块的四个运动向量的原始集中（不管极性）两个以上运动向量是帧内编码的，则编码器 / 解码器可将共处代表性运动向量简单地视为是帧内编码的（即 (0, 0)）。然而，在一些实现中，隔行扫描 B- 半帧中的所有帧内 MB 都被编码为 1MV，所以原始的 4 个运动向量中的两个以上为帧内编码的情形导致共处代表性运动向量被视为是帧内编码在实际上不可能。

[0353] 图 53 中的伪码 5300 示出对用作隔行扫描 B- 半帧中直接模式运动向量的基础的运动向量的选择过程。在一些实现中，该选择过程是产生正向和反向指示直接模式运动向量的缩放运算的前身。

[0354] XIII. 隔行扫描视频帧中的帧内编码 B- 半帧

[0355] 隔行扫描 BI- 半帧（或“帧内 B- 半帧”）是与其参考图片独立编码的半帧。在隔行扫描 BI- 半帧不可用作预测其它图片的锚的意义上，它们不同于其它帧内半帧（例如隔行扫描 I- 半帧）。没有对隔行扫描 BI- 半帧的图片间依赖性，且它在比特流中的出现不表示可独立解码分段或图片组的开始。然而，隔行扫描视频帧中的第一半帧如果被编码为 BI- 半帧，则可用来预测该帧中可被编码为隔行扫描 B- 半帧的第二个半帧。这种革新还通过在许多情形中仅对帧的一半（第一编码半帧）使用帧内编码而不将整个帧编码为内帧或将两个半帧编码为帧内半帧，来改进整体压缩。在一些实现中，帧可包括两个 B- 半帧、两个 BI- 半帧、或一个 B- 或一个 BI- 半帧。

[0356] 使用隔行扫描 BI- 半帧而不使用隔行扫描 I- 半帧是有理由的。一个理由是避免牺牲时间可缩放性。例如，当解码器提交数字视频且需要立即丢弃一些图片以跟上处理需求时，它可寻找它可能丢弃的半帧序列。如果序列中的帧内半帧变成关键半帧，则解码器将被迫解码它们以用作其它半帧的参考，并且不能丢弃它们。然而，如果序列中的帧内半帧被编码为 BI- 半帧，则解码器将仍然有丢弃它们的选择，而不损害后续运动补偿。

[0357] 在隔行扫描 BI- 半帧更有效地用信号表示用于帧内编码和解码的语法元素的意义上，隔行扫描 BI- 半帧不同于具有帧内宏块的隔行扫描 B- 半帧，因为 BI- 半帧内运动补偿相关元素（或信号表示其缺失的元素）可被消除。换言之，当在视频序列中的帧间半帧预测中断点上（例如因为场景变化或复杂运动）编码隔行扫描 B- 半帧时，使用隔行扫描 BI- 半帧（而不是常规 B- 半帧）的理由产生。常常这种半帧中的大多数宏块将需要编码为帧内宏块。在该情形中，根据比特率来看，要将整个 B- 半帧编码为一个 BI- 半帧常常比发送该半帧中每个宏块的预测模式信息要容易。当对隔行扫描 B- 半帧的较好预测或运动补偿不可能时，它可被编码为 BI- 半帧。

[0358] 在一些实现中，编码器可用信号表示比特流中作为图片类型的可能值之一的 BI- 半帧的出现。或者，BI- 半帧的出现可用一些其它方法来指示。

[0359] XIV. 组合实现

[0360] 现在描述对比特流语法、语义和解码器的详细组合实现，以及与主要组合实现有细微差异的另一组合实现。

[0361] A. 比特流语法

[0362] 在各种组合实现中，用于隔行扫描 B- 图片的数据以具有多个层（例如，序列、帧、半帧、宏块、块和 / 或子块层）的比特流形式呈现。

[0363] 对于具有隔行扫描 B- 半帧和 / 或 BI- 半帧的隔行扫描视频帧，帧级别比特流元素如图 54 所示。每个帧的数据包括帧头，随后是半帧层的数据（示为每个半帧的重复“FieldPicLayer”元素）。组成隔行扫描 B- 半帧和 BI- 半帧的半帧头的比特流元素分别如图 55 和 56 所示。组成隔行扫描 B- 半帧（帧内、1MV、或 4MV 宏块）和 BI- 半帧的宏块层的比特流元素分别如图 57 和 58 所示。

[0364] 对于隔行扫描 B- 帧，帧级别比特流元素如图 59 所示。每个帧的数据包括帧头，随后是宏块层的数据。组成隔行扫描 B- 帧的宏块层的比特流元素（帧内或各种帧间类型宏块）如图 60 所示。

[0365] 以下小节描述帧、半帧和宏块层中的选定比特流元素，它们与相关于双向预测隔行扫描图片的信号表示相关。尽管选定比特流元素在特定层的上下文中描述，但一些比特

流元素可在一个以上层中使用。

[0366] 1. 选定帧层元素

[0367] 图 54 是示出包含隔行扫描 B- 半帧或 BI- 半帧（或可能其它类隔行扫描半帧）的帧的帧级别比特流语法的示图。图 59 是示出隔行扫描 B- 帧的帧级别比特流语法的示图。特定的比特流元素如下所述。

[0368] 帧编码模式 (FCM)（可变大小）

[0369] FCM 是用来表示图片编码类型的可变长度代码字 [“VLC”]。FCM 具有如以下表格 3 所示的帧编码模式的值。

[0370] 表格 3. 帧编码模式 VLC

[0371]

FCM 值	帧编码模式
0	逐行扫描
10	帧 - 隔行扫描
11	半帧 - 隔行扫描

[0372] 半帧图片类型 (FPTYPE)(3 个比特）

[0373] FPTYPE 是包括隔行扫描 P- 半帧、隔行扫描 I- 半帧、隔行扫描 B- 半帧和 / 或隔行扫描 BI 半帧的帧的帧头中提供的 3- 比特语法元素。FPTYPE 具有隔行扫描视频帧中半帧类型的不同组合的值，如下表 4 所示。

[0374] 表格 4. 半帧图片类型 FLC

[0375]

FPTYPE FLC	第一半帧类型	第二半帧类型
000	I	I
001	I	P
010	P	I
011	P	P
100	B	B
101	B	BI
110	BI	B
111	BI	BI

[0376] 参考距离 (REFDIST)（可变大小）

[0377] REFDIST 是可变大小的语法元素。该元素表示当前帧和参考帧之间帧的数量。表

格 5 示出用来编码 REFDIST 值的 VLC。

[0378] 表格 5. REFDISTVLC 表格

[0379]

参考帧距离	VLC 代码字（二进制）	VLC 大小
0	00	2
1	01	2
2	10	2
N	11[(N-3)1s]0	N

[0380] 表格 5 中的最后一行表示用来表示大于 2 的参考帧距离的代码字。这些被编码为（二进制）11，随后是 N-3 1s，其中 N 是参考帧距离。代码字中最后一个比特为 0。例如：

[0381] N = 3，VLC 代码字 = 110，VLC 大小 = 3

[0382] N = 4，VLC 代码字 = 1110，VLC 大小 = 4

[0383] N = 5，VLC 代码字 = 11110，VLC 大小 = 5

[0384] 图片类型 (PTYPE)（可变大小）

[0385] PTYPE 是隔行扫描 B- 帧（或其它类型的隔行扫描帧，诸如隔行扫描 I- 帧、或隔行扫描 P- 帧）的帧头中提供的可变大小的语法元素。PTYPE 具有不同帧类型的值，如以下表格 6 所示。

[0386] 表格 6. 图片类型 VLC

[0387]

PTYPE VLC	图片类型
110	I
0	P
10	B
1110	BI
1111	跳过

[0388] 如果 PTYPE 指示该帧被跳过，则该帧被视为与其参考帧等同的 P 帧。被跳过帧的重构在概念上等同于复制该参考帧。被跳过帧意思是没有该帧的其它数据被传送。

[0389] B- 帧直接模式 MB 比特语法元素 (DIRECTMB)（可变大小）

[0390] DIRECTMB 语法元素使用位平面编码来表示 B 图片（在此为隔行扫描 B- 帧）中以直接模式编码的宏块。DIRECTMB 语法元素还可用信号表示：该直接模式是用原始模式来作信号表示的，在该情形中直接模式在隔行扫描 B- 帧的宏块的宏块级别上作信号表示。

[0391] 经扩展的 MV 范围标记 (MVRANGE)（可变大小）

[0392] MVRANGE 是在序列层 EXTENDED_MV 比特被设置为 1 时提供的可变大小的语法元素。MVRANGE VLC 表示运动向量范围。

[0393] 经扩展的差值 MV 范围标记 (DMVRANGE)（可变大小）

[0394] DMVRANGE 是如果序列层语法元素 EXTENDED_DMV = 1 时提供的可变大小的语法元

素，该 DMVRANGE VLC 表示运动向量差值范围。

[0395] 宏块模式表格 (MBMODETAB)(2 或 3 个比特)

[0396] MBMODETAB 语法元素是固定长度的字段。对于隔行扫描 P- 半帧，MBMODETAB 是 3 比特值，表示 8 个哈夫曼表格的哪一个用来解码宏块层中的宏块模式语法元素 (MBMODE)。

[0397] 运动向量表格 (MVTAB)(2 或 3 个比特)

[0398] MVTAB 语法元素是 2 或 3 个比特的值。对于 NUMREF ＝ 1 的隔行扫描 P- 半帧，MVTAB 是表示 8 个隔行扫描哈夫曼表格的哪一个用来解码运动向量的数据的 3 比特语法元素。

[0399] 2MV 块模式表格 (2MVBPTAB)(2 个比特)

[0400] 2MVBPTAB 语法元素是 2 个比特的值，它表示 4 个哈夫曼表格的哪一个用来解码 2MV 半帧宏块中 2MV 块模式 (2MVBP) 语法元素。

[0401] 4MV 块模式表格 (4MVBPTAB)(2 个比特)

[0402] 4MVBPTAB 语法元素是 2 个比特的值，它表示 4 个哈夫曼表格的哪一个用来解码 4MV 半帧宏块中 4MV 块模式 (4MVBP) 语法元素。

[0403] 在另一组合实现中，图片类型信息在隔行扫描 B- 半帧的半帧级别的开始处，而不是在包括隔行扫描 B- 半帧的隔行扫描视频帧的帧级别上用信号表示，并可略去参考距离。

[0404] 2. 选定半帧层元素

[0405] 图 55 是示出组合实现中隔行扫描 B- 半帧的半帧级别比特流语法的示图。特定比特流元素如下所述。

[0406] 运动向量模式 (MVMODE)(可变大小或 1 比特)

[0407] MVMODE 语法元素用信号表示 4 种运动向量编码模式之一，或一种亮度补偿模式 (还有某些图片类型的较少可能)。若干后续元素提供附加运动向量模式和 / 或亮度补偿信息。

[0408] B- 半帧正向模式 MB 比特语法元素 (FORWARDMB)(可变大小)

[0409] FORWARDMB 语法元素将位平面编码用来表示用正向模式编码的 B- 半帧中宏块。FORWARDMB 语法元素还用信号表示，正向模式用原始模式来作信号表示，在该情形中正向 / 非正向模式决定在宏块级别上作信号表示。

[0410] 图 56 是示出组合实现中隔行扫描 B- 半帧的半帧级别比特流语法的示图。在该组合实现中，隔行扫描 BI- 半帧的半帧级别比特流语法使用与隔行扫描 I- 半帧一样的语法元素。

[0411] 3. 选定宏块层元素

[0412] 图 57 是示出组合实现中隔行扫描 B- 半帧的宏块的宏块级别比特流语法的示图。图 60 是示出组合实现中隔行扫描 B- 帧的宏块的宏块级别比特流语法的示图。特定比特流元素如下所述。宏块的数据包括宏块头，随后是块层数据。

[0413] 宏块模式 (MBMODE)(可变大小)

[0414] MBMODE 语法元素指示宏块类型 (例如隔行扫描 B- 半帧的 1MV、4MV 或帧内)，以及 CBP 标记和运动向量数据的出现。

[0415] 正向 B 半帧编码模式 (FORWARDBIT)(1 比特)

[0416] FORWARDBIT 是如果半帧级别语法元素 FORWARDMB 表示使用了原始模式时在隔行扫描 B- 半帧宏块中提供的 1- 比特语法元素。如果 FORWARDBIT ＝ 1，则宏块使用正向模式

编码。

[0417] B 宏块运动预测类型 (BMVTYPE)(可变大小)

[0418] BMVTYPE 是隔行扫描 B- 帧宏块和隔行扫描 B- 半帧宏块中提供的可变大小语法元素,它表示该宏块使用正向、反向还是插值预测。如表格 7 所示,对于隔行扫描 B- 帧的宏块,BFRACTION 以及 BMVTYPE 的值确定使用哪一种类型。

[0419] 表格 7. BMVTYPE VLC

[0420]

BMVTYPE	运动预测类型	
	BFRACTION ≤ 1/2	BFRACTION>1/2
0	反向	正向
10	正向	反向
11	插值	插值

[0421] 在隔行扫描 B- 半帧中,如果宏块模式不是正向(由 FORWARDMB 或 FORWARDBIT 语法元素所示)且不使用 4MV,则发送 BMVTYPE。在该情形中,BMVTYPE 用于用信号表示该 B 宏块是反向、直接、还是插值的。这是简单的 VLC,其中反向 = 0,直接 = 10,插值 = 11。在宏块模式不是正向且使用 4MV 的情形中,BMVTYPE 是反向的,因为只有正向和反向模式被允许用于 4MV。

[0422] 插值 MV 提供 (INTERPMVP)(1 比特)

[0423] INTERPMVP 是如果半帧级别语法元素 BMVTYPE 表示该宏块类型为插值时在 B- 半帧中提供的 1- 比特语法元素。如果 INTERPMVP = 1,则插值 MV 出现,否则它不出现。

[0424] B 宏块运动向量 1(BMV1)(可变大小)

[0425] BMV1 是差分编码宏块的第一运动向量的可变大小的语法元素。

[0426] B 宏块运动向量 2(BMV2)(可变大小)

[0427] BMV2 是如果使用插值模式时隔行扫描 B- 帧宏块和隔行扫描 B- 半帧宏块中提供的可变大小的语法元素。该语法元素差分地编码宏块的第二运动向量。

[0428] 4MV 块模式 (4MVBP)(4 个比特)

[0429] 该 4MVBP 语法元素指示 4 个亮度块的哪一个包含非零运动向量差值,它的使用在下面详细描述。

[0430] 块级别运动向量数据 (BLKMVDATA)(可变大小)

[0431] BLKMVDATA 是包含该块的运动信息并在 4MV 宏块中提供的可变大小的语法元素。

[0432] 半帧变换标记 (FIELDTX)(1 比特)

[0433] FIELDTX 是在隔行扫描 B- 帧的帧内编码宏块中提供的 1 比特语法。该语法元素指示宏块是帧编码还是半帧编码(基本上是宏块的内部组织)。FIELDTX = 1 指示该宏块是半帧编码的。否则,宏块是帧编码的。在帧间编码宏块中,该语法元素可从 MBMODE 推断。

[0434] 直接 B 帧编码模式 (DIRECTBBIT)(1 比特)

[0435] DIRECTBBIT 是如果帧级别语法元素 DIRECTMB 表示使用原始模式时在隔行扫描 B- 帧宏块中提供的 1- 比特语法元素。如果 DIRECTBBIT = 1,则宏块使用直接模式编码。

[0436] B 帧 MV 切换 (MVSW)(1 比特)

[0437] MVSW 是如果 MB 为半帧模式以及如果 BMVTYPE 为正向或反向时在隔行扫描 B- 帧宏块中提供的 1- 比特语法元素。如果 MVSW = 1,则 MV 类型和预测类型在从上半帧去到下半

帧时从正向变成反向（或者从反向变成正向）。

[0438] 两运动向量块模式 (2MVBP)（可变大小）

[0439] 2MVBP 是在隔行扫描 B- 帧宏块中提供的可变大小语法元素。如果 MBMODE 语法元素指示该宏块包含一个运动向量，并且如果该宏块是插值宏块，则提供该语法元素。在该情形中，2MVBP 指示提供两个运动向量（正向和反向运动向量）的哪一个。

[0440] 运动向量数据 (MVDATA)（可变大小）

[0441] MVDATA 是编码宏块的运动向量差值的可变大小语法元素，它的解码在下面详细描述。

[0442] 图 58 是示出组合实现中隔行扫描 BI- 半帧的宏块级别比特流语法的示图。在组合实现中，隔行扫描 BI- 半帧的宏块级别比特流语法使用与隔行扫描 I- 半帧的相同语法元素。

[0443] B. 解码隔行扫描的 B- 半帧

[0444] 以下各节描述用于解码组合实现中隔行扫描 B- 半帧的过程。

[0445] 1. 帧 / 半帧层解码

[0446] 隔行扫描 B- 半帧可以是两种类型之一 :1MV 或混合 MV。

[0447] 在 1MV 隔行扫描 B- 半帧中，取决于宏块的预测类型 (BMVTYPE)，用 0、1 或 2 个运动向量来表示各预测块的位移。当 BMVTYPE 等于 DIRECT（直接）时，推断出正向和反向运动向量，并且不显式地用信号表示其它的运动向量。当 BMVTYPE 是 INTERPOLATED（插值）时，解码两个运动向量 :正向和反向。在正向和反向情形中，只解码一个运动向量。1MV 模式通过 MVMODE 图片层语法元素来用信号表示。

[0448] 在混合 MV 隔行扫描 B- 半帧中，每个宏块可被编码为 1MV 或 4MV 宏块。在 4MV 宏块中，4 个亮度块的每一个都具有与之相关联的运动向量。此外，4MV 宏块只可关联于隔行扫描 B- 半帧中的正向或反向预测类型 (BMVTYPE)。每个宏块的 1MV 或 4MV 模式通过每个宏块上的 MBMODE 语法元素来指示。混合 MV 模式通过 MVMODE 图片层语法元素来用信号表示。

[0449] 2. 宏块层解码

[0450] 隔行扫描 B- 半帧中的宏块可以是三种可能类型之一 :1MV、4MV 和帧内。此外，宏块可以是四种预测类型 (BMVTYPE) 之一 :正向、反向、直接或插值。宏块类型通过宏块层中的 MBMODE 语法元素用信号表示。预测类型通过帧级别位平面 FORWARDMB 和宏块级别 BMVTYPE 语法元素的组合用信号表示，其中帧级别位平面 FORWARDMB 用信号表示每个宏块的正向 / 非正向，而宏块级别 BMVTYPE 语法元素则在预测类型为非正向的情形中用信号表示。

[0451] 以下各节描述 1MV 和 4MV 类型、以及如何用信号表示它们。

[0452] 隔行扫描 B- 半帧中的 1MV 宏块

[0453] 1MV 宏块可出现在 1MV 和混合 MV 隔行扫描 B- 半帧中。在 1MV 宏块中，单个运动向量表示宏块中全部 6 个块的当前和参考图片之间的位移。对于 1MV 宏块，宏块层中的 MBMODE 语法元素表示三点 :

[0454] 1) 该宏块类型为 1MV

[0455] 2) 是否出现 CBPCY 语法元素

[0456] 3) 是否出现 BMV1 语法元素

[0457] 如果 MBMODE 语法元素表示 BMV1 语法元素出现，则 BMV1 语法元素出现在相应位置

的宏块层中。BMV1 语法元素编码运动向量差值。运动向量差值与运动向量预测值相组合，以重构运动向量。如果 MBMODE 语法元素表示 BMV1 语法元素未出现，则运动向量差值被取为 0 且因此运动向量等于运动向量预测值。

[0458] 如果 MBMODE 语法元素表示 CBCPY 语法元素出现，则 CBCPY 语法元素出现在相应位置的宏块层中。CBCPY 语法元素表示 6 个块的哪一个在块层上编码。如果 MBMODE 语法元素表示 CBCPY 语法元素未出现，则 CBCPY 被取为等于 0 且对宏块中 6 个块的任一个都不出现块数据。

[0459] 此外，如果宏块类型为 1MV 且宏块的预测类型是插值，则编码器使用 INTERPMVP 语法元素来用信号表示是否出现第二个运动向量差值 BMV2。如果出现，则解码器在 BMV1 之后立即解码 BMV2。否则，BMV2 的运动向量差值被取为 0，而第二个运动向量等于运动向量预测值。

[0460] 当预测类型是插值时，BMV1 对应于正向运动向量而 BMV2 对应于反向运动向量。

[0461] 隔行扫描 B- 半帧中的 4MV 宏块

[0462] 4MV 宏块仅可出现在混合 MVB- 半帧图片中，且限于正向和反向预测类型。在 4MV 宏块中，4 个亮度块的每一个都具有关联运动向量。色度块的位移从 4 个亮度运动向量中导出。在混合 MV 隔行扫描 B- 半帧中，4MV 宏块仅可关联于正向和反向预测类型。

[0463] 对于 4MV 宏块，宏块层中的 MBMODE 语法元素表示三点 ：

[0464] 1) 该宏块类型为 4MV

[0465] 2) 是否出现 CBPCY 语法元素

[0466] 3) 是否出现 4MVBP 语法元素

[0467] 4MVBP 语法元素表示 4 个亮度块的哪一个包含非零运动向量差值。该 4MVBP 语法元素解码到 0 到 15 的值。对于 4MVBP 中 4 个比特位置的每一个，值 0 表示未出现该块的运动向量差值 (BLKMVDATA) 且运动向量差值取为 0。值 1 表示该块的运动向量差值 (BLKMVDATA) 出现在相应位置中。例如，如果 4MVBP 解码为值 100(二进制)，则比特流包含块 0 和 1 的 BLKMVDATA 并且未出现块 2 和 3 的 BLKMVDATA。

[0468] 如果 MBMODE 语法元素表示 4MVBP 语法元素未出现，则假设出现全部 4 个亮度块的运动向量差值数据 (BLKMVDATA)。

[0469] 取决于 MVMODE 语法元素是表示混合 MV 还是全 -1MV，MBMODE 如下用信号表示信息。以下表格 8 示出 MBMODE 元素如何用信号表示有关全 -1MV 图片中宏块的信息。

[0470] 表格 8. 全 -1MV 图片中的宏块模式

[0471]

索引	宏块类型	CBP 出现	MV 出现
0	帧内	无	-
1	帧内	有	-
2	1MV	无	无
3	1MV	无	有

4	1MV	有	无
5	1MV	有	有

[0472] 以下表格 9 示出 MBMODE 元素如何用信号表示有关混合 MV 图片中宏块的信息。

[0473] 表格 9. 混合 1MV 图片中的宏块模式

[0474]

索引	宏块类型	CBP 出现	MV 出现
0	帧内	无	-
1	帧内	有	-
2	1MV	无	无
3	1MV	无	有
4	1MV	有	无
5	1MV	有	有
6	4MV	无	-
7	4MV	有	-

[0475] 8 个编码表格之一用于用信号表示 MBMODE。所用特定表格通过 MBMODETAB 语法元素用信号表示。

[0476] 以下各节描述预测类型解码和直接模式运动向量的解码。

[0477] 隔行扫描 B- 半帧中的预测类型解码 (BMVTYPE)

[0478] 预测类型根据以下规则进行解码。如果图片级别位平面 FORWARDMB 表示宏块是正向类型，则该宏块的预测类型被设置为正向。如果 FORWARDMB 元素被编码为原始，则编码器 / 解码器在宏块级别上使用一附加比特 FORWARDBIT，来判定预测类型是否为正向。

[0479] 如果预测类型为非正向，且如 MBMODE 语法元素用信号表示的该宏块使用 4MV（仅在混合 MV B 图片中可能），则解码器可直接推断预测类型为反向，因为只有正向和反向类型可与 4MV 模式相关联。否则，解码器显式地解码 BMVTYPE 语法元素。

[0480] 解码隔行扫描 B- 半帧中的直接模式运动向量

[0481] 为了解码隔行扫描 B- 半帧中的直接模式运动向量，解码器首先缓存来自先前解码（即时间上在将来）的锚 (I 或 P) 图片的运动向量。这样，解码器将所缓存的对应于上半帧的运动向量用作预测值，以便于计算上部 B- 半帧的直接模式运动向量，并使用对应于下半帧的运动向量计算下部 B- 半帧的运动向量。例如，半帧 z(z ＝上 / 下）中的宏块 (x, y) 将参考从先前解码 I 或 P 半帧 z（即与当前半帧相同极性的锚半帧中的共处宏块）的宏块 (x, y) 中缓存的运动向量。

[0482] 如果所缓存的来自锚图片的运动向量为帧内运动向量（诸如当先前解码半帧 z 是

I- 半帧时），或者如果锚图片是 P- 半帧但宏块 (x, y) 是帧内编码的，则解码器将经缓存的运动向量视为 (0,0)。如果共处宏块为 1MV，则解码器使用该运动向量。如果共处宏块为 4MV，则解码器将图 53 中伪码 5300 所述的逻辑用来计算运动向量预测值。

[0483] 在伪码 5300 中，SelectDirectModeMVFromColocatedMB 导出在直接模式计算中使用的运动向量预测值。解码器可缓存来自先前解码的锚图片的所有运动向量，并在解码 B- 半帧期间应用以上直接模式，或者解码器在解码锚半帧的同时可应用以上直接模式规则，并缓存用于 B- 半帧的结果运动向量。

[0484] 使用以上获得的运动向量，解码器应用缩放逻辑（图 19 中的 Scale_Direct_MV）。Scale_Direct_MV 获得正向和反向指示运动向量。Scale_Direct_MV 可导致指向上下半帧的正向和反向运动向量。这是有效的，因为直接运动向量由编码器评估并仅在它们给出好预测时选择，还因为隔行扫描 B- 半帧在正向和反向方向上都使用两个参考半帧。

[0485] 在另一实现中，可使用产生以直接模式缩放的运动向量的任何其它过程，包括不涉及任何缓存的过程，它在存储器受限设备中会有用（例如使用随机数发生器来模拟偏零拉普拉斯分布）。这种过程仍将起作用，因为好的编码器将丢弃对直接模式运动向量的较差推测，而在比特流中保留更为准确的推测。

[0486] 3. 运动向量解码过程

[0487] 以下各节描述组合实现中隔行扫描 B- 半帧的块和宏块的运动向量解码过程。

[0488] 填充正向和反向预测上下文

[0489] 正向和反向运动向量被分开缓存，并分别用来预测正向和反向运动向量。例如在以上 X 节描述了单独缓冲区在正向和反向上下文中的使用。用于选择运动向量预测值的技术在 III 节背景技术、III 节详细说明、以及说明书的其它部分中描述。

[0490] 在解码正向运动向量期间用预测运动向量填充反向缓冲区（“缺少方向”缓冲区）时（或在解码反向运动向量期间填充正向缓冲区时），要添加另外两个细节。通常，编码器 / 解码器可使用运动向量类型信息（例如 1MV 等）、以及先前解码的运动向量的极性来形成预测。然而，在“空穴填充”情形中，因为编码器 / 解码器实际上并不解码缺少方向类型的运动向量，编码器 / 解码器并不具有运动向量类型信息或极性信息（例如相同极性或相反极性）。在该组合实现中，编码器 / 解码器将运动向量类型设置为 1MV，并将主半帧运动向量选为预测值。图 47 中的伪码 4700 描述该组合实现中的极性选择过程。

[0491] 对于帧内编码宏块，“帧内运动向量”用来填充正向和反向运动预测平面。“帧内运动向量”的任何一致表示可由解码器实现来选择。例如，如果运动向量被存储在 2- 字节短数组中，则“帧内运动向量”可被表示为唯一的大常数，它被填充到运动向量数组中来表示该宏块被编码为帧内宏块。

[0492] B- 半帧中的正向运动向量预测

[0493] 正向参考帧距离从 BFRACTION 语法元素和 REFDIST 语法元素中计算。正向运动向量预测如以上 X 节所述地进行。

[0494] B- 半帧中的反向运动向量预测

[0495] 正向参考帧距离从 BFRACTION 语法元素和 REFDIST 语法元素中计算。正向运动向量预测如以上 X 节所述地进行。

[0496] 解码运动向量差值

[0497] BMV1、BMV2 或 BLKMVDATA 语法元素编码宏块或宏块中各块的运动信息。1MV 宏块具有 BMV1 和 BMV2 语法元素，且 4MV 宏块可具有 0 到 4 个之间的 BLKMVDATA 元素。

[0498] 当预测类型 (BMVTYPE) 为插值时，BMV1 对应于正向而 BMV2 对应于反向运动向量残差。

[0499] 以下各节描述对于应用于 B- 图片的双参考情形如何计算运动向量差值。

[0500] 双参考半帧图片中的运动向量差值

[0501] 双参考半帧图片在使用半帧图片对隔行扫描帧的编码中出现。序列的每个帧都被分成两个半帧，且每个半帧使用实际上逐行扫描代码路径来编码。

[0502] 在具有两个参考半帧的半帧图片中 (诸如具有隔行扫描 B- 半帧的图片)，宏块层中的每个 MVDATA 或 BLKMVDATA 语法元素联合编码三种信息 :1) 水平运动向量差值分量，2) 垂直运动向量差值分量，3) 使用主还是非主预测值，即两个半帧的哪一个被运动向量参考。

[0503] MVDATA 或 BLKMVDATA 语法元素是可变长度的哈夫曼代码字，然后是一固定长度代码字。哈夫曼代码字的值确定固定长度代码字的大小。图片层中的 MVTAB 语法元素指定用来解码可变大小代码字的哈夫曼表格。图 61A 中的伪码 6100 示出如何解码运动向量差值和主 / 非主预测值信息。

[0504] 值 predictor_flag、dmv_x 和 dmv_y 在图 61A 中的伪码 6100 中计算。伪码 6190 中的各个值定义如下 :

[0505] dmv_x :运动向量水平差值分量，

[0506] dmv_y :运动向量垂直差值分量，

[0507] k_x，k_y :长运动向量的固定长度，

[0508] k_x 和 k_y 取决于由 MVRANGE 符号定义的运动向量范围。

[0509] 表格 10. 由 MVRANGE 指定的 k_x 和 k_y

[0510]

MVRANGE	k_x	k_y	range_x	range_y
0(缺省)	9	8	256	128
10	10	9	512	256
110	12	10	2048	512
111	13	11	4096	1024

[0511] extend_x :水平运动向量差值的扩展范围，

[0512] extend_y :垂直运动向量差值的扩展范围，

[0513] extend_x 和 extend_y 从 DMVRANGE 图片半帧语法元素中导出。如果 DMVRANGE 表示使用水平分量的扩展范围，则 extend_x = 1。否则，extend_x = 0。类似地，如果 DMVRANGE 表示使用垂直分量的扩展范围，则 extend_y = 1。否则，extend_y = 0。

[0514] 变量 predictor_flag 是表示使用主还是非主运动向量预测值的二进制标记 (0 = 使用主预测值，1 = 使用非主预测值)。offset_table 和 size_table 数组被定义为如图 61A 所示。

[0515] 图 61B 中的伪码 6110 示出在另一组合实现中如何解码双参考半帧的运动向量差值。伪码 6110 用不同方法解码运动向量差值。例如，伪码 6110 略去经扩展运动向量差值范围的处理。

[0516] 运动向量预测值

[0517] 运动向量通过将前面部分中计算的运动向量差值添加到运动向量预测值中来计算。以下各节描述在该组合实现中如何计算 1MV- 和混合 MV 隔行扫描 B- 半帧中宏块的运动向量预测值。

[0518] 1MV 隔行扫描 B- 半帧中的运动向量预测值

[0519] 图 5A 和 5B 是示出被视作用于 1MV 宏块的候选运动向量预测值的宏块的位置的示图。候选预测值从左边、上方和右上方宏块中取得，除了宏块是行中最后一个宏块的情形。在该情形中，预测值 B 从左上方宏块而不是从右上方中取得。对于帧为一个宏块宽的特定情形，预测值总是预测值 A（顶部预测值）。当前宏块在首行中的特定情形（没有 A 或 B 预测值、或者根本没有预测值）在以上参照图 33A-33F 以及在以下参照图 62A-62F 解决。

[0520] 混合 MV 隔行扫描 B- 半帧中的运动向量预测值

[0521] 图 6A-10 示出被视作用于混合 MV 隔行扫描 B- 半帧中的 1MV 或 4MV 宏块的运动向量的候选运动向量的块或宏块的位置。

[0522] 隔行扫描 B- 半帧中的主和非主 MV 预测值

[0523] 对于每个帧间编码的宏块，导出两个运动向量预测值。一个来自主半帧而另一个来自非主半帧。主半帧被视为包含邻域中候选运动向量预测值的大部分实际值的半帧。在不分上下的情形中，相反半帧的运动向量预测值被视为是主预测值（因为它在时间上较为接近）。帧内编码宏块在主 / 非主预测值的计算中不作考虑。如果全部候选预测值宏块都是帧内编码的，则主和非主运动向量预测值都被设置为 0 且主预测值被视为来自相反半帧。

[0524] 计算隔行扫描 B- 半帧中的运动向量预测值

[0525] 对块或宏块的每个运动向量计算两个运动向量预测值 - 每个参考一个。图 62A-62F 中的伪码 6200 描述在组合实现中如何对双参考情形计算运动向量预测值。（图 33A-33F 中的伪码 3300 描述在另一实现中如何对双参考情形计算运动向量预测值。）在双参考图片中，当前半帧可参考两个最新半帧。一个预测值用于相同极性的参考半帧，而另一个用于相反极性的参考半帧。

[0526] 重构隔行扫描 B- 半帧中的运动向量

[0527] 以下各节描述如何重构 1MV 和 4MV 宏块的亮度和色度运动向量。在重构运动向量之后，它随后可用作邻域运动向量以预测相邻宏块的运动向量。该运动向量将具有“相同”或“相反”的关联极性，并可用来导出另一半帧极性的运动向量预测值，用于运动向量预测。

[0528] 隔行扫描 B- 半帧中的亮度运动向量重构

[0529] 在所有情形（1MV 和 4MV 宏块）中，亮度运动向量通过如下将差值添加到预测值来重构：

[0530] mv_x = (dmv_x+predictor_x)smod range_x

[0531] my_y = (dmv_y+predictor_y)smod range_y

[0532] 模运算“smod”是定义如下的有符号模：

[0533] A smod b = ((A+b)% (2*b))-b

[0534] 这确保重构向量是有效的。(A smod b) 位于 -b 和 b-1 之间。range_x 和 range_y 取决于 MVRANGE。

[0535] 因为隔行扫描 B- 半帧图片使用两个参考图片，在解码运动向量差值之后导出的 predictor_flag 与从运动向量预测导出的 dominantpredictor 值相组合，以确定哪个半帧被用作参考。图 63 中的伪码 6300 描述如何确定参考半帧。

[0536] 在 1MV 宏块中，对于组成宏块的亮度分量的 4 个块将有单个运动向量。如果 MBMODE 语法元素表示没有 MV 数据在宏块层中出现，则 dmv_x = 0 且 dmv_y = 0(mv_x = predictor_x 和 mv_y = predictor_y)。

[0537] 在 4MV 宏块中，宏块中每个帧间编码亮度块将具有它自己的运动向量。因此在每个 4MV 宏块中将有 0 到 4 个亮度运动向量。如果 4MVBP 语法元素表示未出现块的运动向量信息，则该块的 dmv_x = 0 且 dmv_y = 0(mv_x = predictor_x 和 mv_y = predictor_y)。

[0538] 色度运动向量重构

[0539] 色度运动向量从亮度运动向量中导出。此外，对于 4MV 宏块，对要将色度块编码为帧间块还是帧内块的决定是基于亮度块或半帧的状态作出的。

[0540] C. 解码隔行扫描 P- 帧

[0541] 在描述组合实现中用于解码隔行扫描 B- 帧的过程之前，描述用于解码隔行扫描 P- 帧的过程。描述用于解码隔行扫描 B- 帧的过程的小节将参照本节中讨论的各个概念进行。

[0542] 1. 隔行扫描 P- 帧的宏块层解码

[0543] 在隔行扫描 P- 帧中，每个宏块可用使用一个或四个运动向量的帧模式、或使用两个或四个运动向量的半帧模式作运动补偿。帧间编码的宏块不包含任何帧内块。此外，运动补偿之后的残差可用帧变换模式或半帧变换模式来编码。更具体地，如果用半帧变换模式编码残差，则残差的亮度分量根据各个半帧来重新排列，而在帧变换模式中当色度分量保持不变时残差保持不变。宏块也可被编码为帧内宏块。

[0544] 运动补偿可被限制为不包括四个运动向量 (半帧 / 帧)，并且它通过 4MVSWITCH 用信号表示。每个宏块的运动补偿和残差编码的类型通过 MBMODE 和 SKIPMB 联合表示。MBMODE 根据 4MVSWITCH 采用不同的表格集。

[0545] 隔行扫描 P- 帧中的各个宏块被分成 5 种类型 :1MV、2 半帧 MV、4 帧 MV、4 半帧 MV 和帧内。前四类宏块是帧间编码的，而最后一类表示该宏块是帧内编码的。宏块类型由宏块层中的 MBMODE 语法元素以及跳过比特用信号表示。MBMODE 对不同类型的宏块共同编码宏块类型以及有关该宏块的各条信息。

[0546] 用信号表示跳过的宏块

[0547] SKIPMB 字段表示宏块的跳过条件。如果 SKIPMB 字段为 1，则表示要跳过当前宏块，并且在 SKIPMB 字段之后没有发送其它信息。该跳过条件暗示当前宏块是具有 0 差值运动向量的 1MV(即，该宏块是使用其 1MV 运动预测值做运动补偿的)，并且没有经编码的块 (CBP = 0)。

[0548] 另一方面，如果 SKIPMB 字段不是 1，则 MBMODE 半帧被解码为表示宏块的类型和有关当前宏块的其它信息，诸如以下小节中描述的信息。

[0549] 用信号表示宏块模式

[0550] 有 15 种通过 MBMODE 表示的可能事件 ;MBMODE 共同指定宏块的类型 (1MV、4 帧 MV、2 半帧 MV、4 半帧 MV、或帧内)、帧间编码宏块的变换类型 (即半帧或帧或未编码的块)、以及是否有 1MV 宏块的运动向量差值。

[0551] 设 <MVP> 表示用信号表示是否有非零 1MV 运动向量差值的二进制事件。设 <Field/Frame transform>(< 半帧 / 帧变换 >) 表示用信号表示宏块的残差是帧变换编码、半帧变换编码、还是零编码块 (即 CBP = 0) 的三元事件。MBMODE 共同用信号表示以下事件集 :

[0552] MBMODE = {<1MV, MVP, 半帧 / 帧变换 >, <2 半帧 MV, 半帧 / 帧变换 >, <4 帧 MV, 半帧 / 帧变换 >, <4 半帧 MV, 半帧 / 帧变换 >, < 帧内 >} ;<1MV, MVP = 0, CBP = 0> 的事件除外, 它通过跳过条件用信号表示。

[0553] 对于帧间编码宏块, 当 MBMODE 中的半帧 / 帧变换事件表示无编码块时, 不解码 CBPCY 语法元素。另一方面, 如果 MBMODE 中的半帧 / 帧变换事件表示半帧或帧变换时, 则解码 CBPCY。所解码事件 < 半帧 / 帧变换 > 被用来设置标记 FIELDTX。如果该事件表示宏块是半帧变换编码的, 则 FIELDTX 被设置为 1。如果该事件表示宏块是帧变换编码的, 则 FIELDTX 被设置为 0。如果该事件表示 0 编码块, 则 FIELDTX 被设置为与运动向量相同的类型, 即如果它是 FIELDMV 则 FIELDTX 被设置为 1, 且如果它是 FRAMEMV 则被设置为 0。

[0554] 对于非 1MV 的帧间编码宏块, 发送表示零差值运动向量事件的另一字段。在 2 半帧 MV 宏块的情形中, 发送表示两个运动向量的哪一个包含非零运动向量差值的 2MVBP 字段。类似地, 发送表示四个运动向量的哪一个包含非零运动向量差值的 4MVBP 字段。

[0555] 对于帧内编码宏块, 半帧 / 帧变换和零编码块在各个字段中编码。

[0556] 2. 对隔行扫描 P- 帧的运动向量解码

[0557] 隔行扫描 P- 帧的运动向量预测值

[0558] 计算当前宏块的运动向量预测值的过程包括两个步骤。首先, 当前宏块的三个候选运动向量从其相邻宏块中收集。其次, 当前宏块的运动向量预测值从候选运动向量集中计算。图 40A-40B 示出从中收集候选运动向量的相邻宏块。候选运动向量的收集顺序是重要的。在该组合实现中, 收集顺序总是从 A 开始、继续到 B、并在 C 结束。注意, 如果相应块在帧边界之外或者相应块是不同片的一部分, 则候选预测值被视为不存在。因而, 不跨片边界执行运动向量预测。

[0559] 以下各节描述是否收集不同类型宏块的候选运动向量, 以及如何计算运动向量预测值。

[0560] 1MV 候选运动向量

[0561] 在该组合实现中, 图 64 中的伪码 6400 被用来收集该运动向量的最多三个候选运动向量。

[0562] 4 帧 MV 候选运动向量

[0563] 对于 4 帧 MV 宏块, 对于当前宏块中四个帧块运动向量的每一个, 收集来自相邻块的候选运动向量。在该组合实现中, 图 65 中的伪码 6500 用来收集左上帧块运动向量的最多三个候选运动向量。图 66 中的伪码 6600 用来收集右上帧块运动向量的最多三个候选运动向量。图 67 中的伪码 6700 用来收集左下帧块运动向量的最多三个候选运动向量。图 68 中的伪码 6800 用来收集右下帧块运动向量的最多三个运动向量。

[0564] 2 半帧 MV 候选运动向量的导出

[0565] 对于 2 半帧 MV 宏块，对于当前宏块中两个帧运动向量的每一个，收集来自相邻块的候选运动向量。图 69 中的伪码 6900 用来收集上半帧运动向量的最多三个候选运动向量。图 70 中的伪码 7000 用来收集下半帧运动向量的最多三个候选运动向量。

[0566] 4 半帧 MV 候选运动向量的导出

[0567] 对于 4 半帧 MV 宏块，对于当前宏块中四个半帧块的每一个，收集来自相邻块的候选运动向量。图 71 中的伪码 7100 用来收集左上半帧块运动向量的最多三个候选运动向量。图 72 中的伪码 7200 用来收集右上半帧块运动向量的最多三个候选运动向量。图 73 中的伪码 7300 用来收集左下半帧块运动向量的最多三个候选运动向量。图 74 中的伪码 7400 用来收集右下半帧块运动向量的最多三个候选运动向量。

[0568] 平均半帧运动向量

[0569] 给定两个半帧运动向量 (MVX_1, MVY_1) 和 (MVX_2, MVY_2)，用来形成候选运动向量 (MVX_A, MVY_A) 的平均运算是：

[0570] $MVX_A = (MVX_1+MVX_2+1) >> 1$；

[0571] $MVX_A = (MVY_1+MVY_2+1) >> 1$；

[0572] 从候选运动向量中计算帧的 MV 预测值

[0573] 本节描述：给定一个候选运动向量集如何计算帧运动向量的运动向量预测值。在该组合实现中，运算对计算 4 帧 MV 宏块中 4 帧块运动向量的每一个的预测值相同。

[0574] 图 75 中的伪码 7500 描述如何计算帧运动向量的运动向量预测值 (PMV_x, PMV_y)。在伪码 7500 中，TotalValidMV 表示候选运动向量集中运动向量的总数 (TotalValidMV = 0，1，2 或 3)，且 ValidMV 数组表示候选运动向量集中的运动向量。

[0575] 从候选运动向量中计算半帧的 MV 预测值

[0576] 本节描述：给定一个候选运动向量集如何计算半帧运动向量的运动向量预测值。运算对计算 2 半帧 MV 宏块中 2 个半帧运动向量的每一个、或 4 半帧 MV 宏块中 4 个半帧块运动向量的每一个的预测值相同。

[0577] 首先，候选运动向量被分成两个集，其中一个集只包含指向与当前半帧相同的半帧的候选运动向量，而另一个集包含指向相反半帧的候选运动向量。假设候选运动向量在 1/4 像素单元中表示，编码器或解码器可通过对其 y- 分量的以下检查，来检查候选运动向量是否指向相同半帧：

[0578]

```
if（ValidMVy&4）{
    ValidMV 指向相反半帧。
}else{
    ValidMV 指向相同半帧。
}
```

[0579] 图 76 中的伪码 7600 描述如何计算半帧运动向量的运动向量预测值 (PMV_x, PMV_y)。在伪码 7600 中，SameFieldMV 和 OppFieldMV 表示两个候选运动向量集，而 NumSameFieldMV 和 NumOppFieldMV 表示属于每个集的候选运动向量的数量。每个集中的候选运动向量的顺序从候选 A（如果它存在）开始，然后是候选 B（如果它存在），再后是候选 C（如果它存在）。

例如，如果 SameFieldMV 候选运动向量集只包含候选 B 和候选 C，则 SameFieldMV[0] 是候选 B。

[0580] 解码运动向量差值

[0581] MVDATA 语法元素包含宏块的运动向量差值信息。取决于运动补偿的类型和每个宏块上用信号表示的运动向量块模式，每个宏块最多可有四个 MVDATA 语法元素。更具体地，

[0582] • 对于 1MV 宏块，取决于 MBMODE 中的 MVP 字段可出现 0 或 1 个 MVDATA 语法元素。

[0583] • 对于 2 半帧 MV 宏块，取决于 2MVBP 可出现 0、1、或 2 个 MVDATA 语法元素。

[0584] • 对于 4 帧 / 半帧 MV 宏块，取决于 4MVBP 可出现 0、1、2、3 或 4 个 MVDATA 语法元素。

[0585] 在该组合实现中，运动向量差值用与隔行扫描 P- 半帧的单参考半帧运动向量差值相同的方法来解码。（图 77A 中的伪码 7700 示出如何解码单参考半帧的运动向量差值。图 77B 中的伪码 7710 示出在另一组合实现中如何解码单参考半帧的运动向量差值。伪码 7710 用一种不同方法解码运动向量差值。例如，伪码 7710 略去对经扩展的运动向量差值范围的处理。）

[0586] 重构运动向量

[0587] 给出运动向量差值 dmv，亮度运动向量通过如以上 XV.B.3 节中所述的将差值添加到预测值中来重构。给定亮度帧或半帧运动向量，导出相应的色度帧或半帧运动向量来补偿 Cb/Cr 块的一部分或全部。图 78 中的伪码 7800 描述色度运动向量 CMV 如何从隔行扫描 P- 帧中的亮度运动向量 LMV 中导出。

[0588] D. 解码隔行扫描 B- 帧

[0589] 本节参照前一节中讨论的概念描述组合实现中用于解码隔行扫描 B- 帧的过程。

[0590] 1. 隔行扫描 B- 帧的宏块级别解码

[0591] 在宏块级别上，隔行扫描 B- 帧语法类似于上述隔行扫描 P- 帧。隔行扫描 B- 帧中的宏块被分成三种类型：1MV、2 半帧 MV、和帧内。在本组合实现中 4 帧 MV 和 4 半帧 MV 模式不被允许用于隔行扫描 B- 帧。这三种模式像在隔行扫描 P- 帧中一样，与 MBMODE 语法元素共同编码。每个宏块也被预测为正向、反向、直接或插值（使用 DIRECTMB 和 BMVTYPE 语法元素）如果 1MV 宏块是正向或反向，则它使用单个运动向量。如果它是 1MV 但是直接或插值的，则它使用两个运动向量。如果它是 2 半帧 MV 类型并是正向或反向预测的，则它使用两个运动向量。如果它是 2 半帧 MV 类型并是直接或插值的，则它使用四个运动向量。

[0592] 以下各节描述隔行扫描 B- 帧中不同帧间编码的宏块类型的特征。

[0593] 隔行扫描 B- 帧中的 1MV 宏块

[0594] 在隔行扫描 B- 帧中的 1MV 宏块中，亮度块的位移在预测类型是正向或反向时由单个运动向量表示，而在类型是直接或插值时由两个运动向量表示。在每一情形中导出相应的色度向量。在插值和直接预测的情形中，平均来自正向和反向参考图片的运动补偿像素以形成最终预测。

[0595] 隔行扫描 B- 帧中的 2 半帧 MV 宏块

[0596] 在隔行扫描 B- 帧中的 2 半帧 MV 宏块中，亮度块的每个半帧的位移由一不同运动向量描述，如图 37 所示。此外，在从上半帧去到下半帧时该预测类型被允许从正向切换到反向，或反之，从而使上半帧从一参考图片中得到运动补偿，而下半帧从另一参考图片中得

到运动补偿，如在以上 VII 节中所述的。

[0597] 隔行扫描 B- 帧中的 2MVBP、4MVBP 的解释和运动向量顺序

[0598] 在 1MV 宏块中，编码器用插值模式使用 2MVBP 语法元素，以表示出现两个运动向量的哪一个。比特 1 对应于正向运动向量，而比特 0 对应于反向运动向量。

[0599] 在 2 半帧 MV 宏块中，编码器用正向和反向模式使用 2MVBP 语法元素，来表示出现两个半帧的运动向量的哪一个。比特 1 对应于上半帧运动向量而比特 0 对应于下半帧运动向量。当 MVSW 语法元素用来从用于上半帧的正向预测切换到用于下半帧的反向预测或反之时，编码器使用相同的上 / 下信号表示。编码器用插值模式使用 4MVBP 语法元素，来表示出现四个运动向量的哪一个。比特 3 对应于上半帧正向运动向量，比特 2 对应于上半帧反向运动向量，比特 1 对应于下半帧正向运动向量，而比特 0 对应于下半帧反向运动向量。

[0600] 设置为‘1’的 2MVBP 和 4MVBP 的比特表示出现相应运动向量差值，同时设置为‘0’的比特表示相应运动向量等于所预测的运动向量，即未出现相应的运动向量差值。经实际解码的运动向量用与 2MVBP 或 4MVBP 中各比特相同的顺序发送。例如，在使用差值模式的 2 半帧 MV 宏块中，要由解码器接收的第一运动向量是上半帧正向运动向量，而要接收的最后（即第四）运动向量是下半帧反向运动向量。

[0601] 用信号表示跳过的宏块

[0602] 被跳过的宏块以与 P 帧相同的方式用信号表示。然而，隔行扫描 B- 帧中的被跳过宏块限于 1MV 帧类型，即不允许半帧类型。运动向量用零差值运动向量编码（即宏块是使用其 1MV 运动补偿值进行运动补偿的）且没有已编码块 (CBP ＝ 0)。如果宏块被跳过，则编码器仅发送该宏块的 BMVTYPE 信息，从而运动向量可被准确预测为正向、反向、直接或插值。

[0603] 用信号表示宏块模式

[0604] 用信号表示宏块模式用与隔行扫描 P- 帧相同的方法执行，如以上 XV.C. 节中所述。

[0605] 预测类型解码 (BMVTYPE 和 MVSW)

[0606] 隔行扫描 B- 帧的预测类型根据以下规则解码。如果图片层位平面 DIRECTMB 表示宏块是直接类型，则该宏块的预测类型被设置成直接。如果直接 / 非直接决定用原始模式编码，则编码器使用宏块级别上的附加比特 DIRECTBBIT，来表示预测类型是否是直接。

[0607] 如果预测类型是非直接的，则解码器解码 BMVTYPE 语法元素。如果宏块模式是“2MV 半帧编码”，且如果 BMVTYPE 是正向或反向的，则解码器还解码 MVSW 比特来判定从该宏块的上半帧去到下半帧时预测类型是否将改变（即，从正向变为反向，或反之）。

[0608] 解码直接模式运动向量

[0609] 为了解码直接模式运动向量，解码器首先缓存来自先前解码锚帧的运动向量。具体地，对于先前解码的将来 P- 帧，解码器缓存来自将来 P- 帧的经解码亮度运动向量的最大可能数量的一半（即 (2*NumberOfMB) 个运动向量）。选择要缓存的来自锚帧的这些运动向量的方法在以上 XIII 节中描述。

[0610] 使用以上所获得的运动向量，解码器应用图 19 中伪码 1900 示出的 Scale_Direct_MV 中的缩放逻辑，来获得正向和反向指示运动向量，而无需回拉运动向量。

[0611] 在本组合实现中，不计算其中不使用诸如正向和反向预测宏块的直接模式预测的

宏块的直接模式运动向量。相反，非直接宏块的运动向量基于正向或反向运动向量缓冲区来预测。

[0612] 2. 对隔行扫描 B- 帧的运动向量解码

[0613] 隔行扫描 B- 帧的运动向量预测值

[0614] 与隔行扫描 P- 帧一样，计算隔行扫描 B- 帧的当前宏块的运动向量预测值的过程包括，从当前宏块的相邻宏块中收集其候选运动向量，并且从候选运动向量集中计算当前宏块的运动向量预测值。图 40A-40B 示出从中收集候选运动向量的相邻宏块。在该组合实现中，隔行扫描 B- 帧的运动向量预测值根据以上 XV.C. 节所述的用于隔行扫描 P- 帧的规则来从候选集中选择。

[0615] 不同的预测上下文被用于正向和反向模式运动向量。解码器使用正向预测上下文来预测正向运动向量，并使用反向预测上下文来预测反向运动向量。

[0616] 填充隔行扫描 B- 帧中的正向和反向预测上下文

[0617] 解码器分开缓存正向和反向运动向量，并分别用它们来预测正向和反向运动向量。对于插值宏块，解码器使用正向预测缓冲区来预测正向运动向量（第一个经解码的 MVDATA 元素），并使用反向缓冲区来预测反向运动向量（第二个经解码的 MVDATA 元素）。当宏块是直接或插值，则解码器将正向 MV 分量缓存在正向缓冲区中，并将反向 MV 分量缓存在反向缓冲区中。每种情形中（例如 1MV 宏块、2 半帧 MV 宏块等）用于从一个候选集中选择运动向量预测值的实际预测逻辑如以上 XV.C. 节中所述。

[0618] 用于填充正向和反向运动向量缓冲区、并从这些缓冲区的运动向量中预测运动向量的方案如以上 X.C. 节中所述。

[0619] 解码隔行扫描 B- 帧中的运动向量差值

[0620] 隔行扫描 B- 帧中的运动向量差值根据图 77A 和 77B 中的伪码 7700 和 7710 解码，如以上 XV.C.2 节中所述。

[0621] 重构隔行扫描 B- 帧中的运动向量

[0622] 隔行扫描 B- 帧中的运动向量根据图 78 中的伪码 7800 并如以上 XV.B.3 和 XV.C.2 节中所述地解码。

[0623] E. 位平面编码

[0624] 宏块特定的二进制信息可用每个宏块一个二进制符号来编码，这些二进制信息诸如 (1) 隔行扫描 B- 半帧的宏块的正向 / 非正向决定（即 FORWARDMB 标记），以及 (2) 隔行扫描 B- 半帧的宏块的直接 / 非直接决定（即 DIRECTMB 标记）。例如，隔行扫描 B- 半帧的宏块是否用正向模式（与诸如反向、直接或插值的另一模式相对）作运动补偿可用 1 个比特作信号表示。在这些情形中，半帧或帧中全部宏块的状态可被编码为位平面并在半帧或帧头中传送。该规则的一个例外是如果位平面编码模式被设置成原始模式时，在该情形中每个宏块的状态被编码为每个符号 1 个比特，并在宏块级别上与其它宏块级别语法元素一起传送。

[0625] 半帧 / 帧级别位平面编码被用来编码两维二进制数组。每个数组的大小是 rowMB x colMB，其中 rowMB 和 colMB 分别是讨论中半帧或帧中宏块行和列的数量。在比特流内，每个数组被编码为一个连续比特集。七种模式之一被用来编码每个数组。该七种模式是：

[0626] 1. 原始模式 - 编码为每个符号 1 个比特并作为 MB 级别语法的一部分传送的信息；

[0627] 2. 正常 -2(Norm-2) 模式 - 共同编码的两个符号 ;

[0628] 3. 差值 -2(Diff-2) 模式 - 位平面的差值编码,随后是共同编码两个残差符号 ;

[0629] 4. 正常 -6(Norm-6) 模式 - 共同编码的六个符号 ;

[0630] 5. 差值 -6(Diff-6) 模式 - 位平面的差值编码,随后是共同编码六个残差符号 ;

[0631] 6.rowskip(跳行) 模式 - 用信号表示跳过没有设置比特的行的一个比特 ;以及

[0632] 7.columnskip(跳列) 模式 - 用信号表示跳过没有设置比特的列的一个比特。

[0633] 半帧或帧级别上位平面的语法元素顺序如下 :INVERT、IMODE 和 DATABITS。

[0634] 逆转标记 (INVERT)

[0635] INVERT 语法元素是 1- 比特值,如果设置则表示该位平面具有比 0 比特更多的设置比特。取决于 INVERT 和模式,解码器将逆转所解释的位平面以重新创建原始位平面。注意,当使用原始模式时,将忽略该比特的值。以下提供对解码位平面时如何使用 INVERT 值的描述。

[0636] 编码模式 (IMODE)

[0637] IMODE 语法元素是指示用来编码位平面的编码模式的可变长度值。表格 11 示出用来编码 IMODE 语法元素的代码表。以下提供对解码位平面时如何使用 IMODE 值的描述。

[0638] 表格 11.IMODEVLC 代码表

[0639]

IMODE VLC	编码模式
10	Norm-2
11	Norm-6
010	跳行
011	跳列
001	Diff-2
0001	Diff-6
0000	原始

[0640] 位平面编码比特 (DATABITS)

[0641] DATABITS 语法元素是编码位平面的符号流的可变大小语法元素。用来编码位平面的方法根据 IMODE 的值来确定。七种编码模式在以下各节中描述。

[0642] 原始模式

[0643] 在该模式中,位平面被编码为每个以宏块的光栅扫描顺序扫描的符号 1 个比特,并作为宏块层的一部分发送。或者,该信息在半帧或帧级别上以原始模式编码,且 DATABITS 在长度上为 rowMB x colMB 比特。

[0644] Norm-2 模式

[0645] 如果 rowMB x colMB 为奇数,则第一个符号被编码为原始。后续符号成对地用自

然扫描顺序编码。表格 12 中的二进制 VLC 表格被用来编码符号对。

[0646] 表格 1.Norm-2/Diff-2 代码表

[0647]

符号 2n	符号 2n+1	代码字
0	0	0
1	0	100
0	1	101
1	1	11

[0648] Diff-2 模式

[0649] Norm-2 模式用来如上所述地产生位平面，然后 Diff-1 运算如下所述地应用于位平面。

[0650] Norm-6 模式

[0651] 在 Norm-6 和 Diff-6 模式中，位平面分 6 个像素的组进行编码。这些像素被分成 2x3 或 3x2 的块。位平面使用一系列规则来最大程度地平铺，且剩余像素使用跳行和跳列模式的变体进行编码。如果且仅当 rowMB 是 3 的倍数而 colMB 不是，则使用 2x3“竖直”块。否则，使用 3x2“水平”块。图 79A 示出 2x3“竖直”块的简化示例。图 79B 和 79C 示出 3x2“水平”块，对这些块细长的黑色矩形为 1 个像素宽并使用跳行和跳列编码进行编码。对于如图 79C 所示平铺的平面，在图片的上边缘和左边缘使用线性块，这些块的编码顺序符合以下模式。先编码 6- 元素块，然后是跳列和跳行编码的线性块。如果数组大小是 2x3 或 3x2 的倍数，则后面的线性块不存在，且位平面被完美平铺。

[0652] 该 6- 元素矩形块使用不完整的哈夫曼代码编码，即不将所有端点用于编码的哈夫曼代码。设 N 是块中设置比特的数量，即 $0 \leqslant N \leqslant 6$。对于 $N < 3$，使用 VLC 来编码该块。对于 $N = 3$，固定长度的转义码之后为 5 比特固定长度代码，对于 $N > 3$，固定长度转义码之后为该块的补码。

[0653] 该矩形块包含 6 个比特的信息。设 k 为关联于块的代码，其中 $k = b_i 2^i$，b_i 为该块内自然扫描顺序中第 i 个比特的二进制值。因此，$0 \leqslant k < 64$。VLC、转义码、加上固定长度代码被用来用信号表示 k。

[0654] Diff-6 模式

[0655] Norm-6 模式用来如上所述地产生位平面，然后 $Diff^{-1}$ 运算如下所述地应用于位平面。

[0656] 跳行模式

[0657] 在跳行编码模式中，用 1 个比特的开销跳过所有零行。语法如下：对于每一行，单个 ROWSKIP 比特表示是否跳过该行；如果跳过该行，则接着是下一行的 ROWSKIP 比特；否则（未跳过该行），则接着是 ROWBITS 比特（该行中每个宏块的比特）。因而，如果整行为零，则零比特被发送为 ROWSKIP 符号，并跳过 ROWBITS。如果在该行中有一设置比特，ROWSKIP 被设置为 1，且整个行被发送为原始 (ROWBITS)。各行从半帧或帧的顶部扫描到底部。

[0658] 跳列模式

[0659] 跳列是跳行的转置。各列从半帧或帧的顶部扫描到底部。

[0660] $Diff^{-1}$:逆向差值解码

[0661] 在使用任一差值模式(Diff-2 或 Diff-6)时,“差值比特”的位平面首先使用相应的正常模式(Norm-2 或 Norm-6)解码。差值比特被用来重新产生原始位平面。重新产生过程是二进制字母表上的 2-D DPCM。为了在位置(i,j)上重新产生比特,预测值 $b_p(i,j)$ 如下产生(从位置(i,j)上的比特 b(i,j)):

[0662]

$$b_p(i,j)=\begin{cases} A & i=j=0 \text{ 或者 } b(i,j-1)\neq b(i-1,j) \\ b(0,j-1) & i==0 \\ b(i-1,j) & \text{否则} \end{cases}$$

[0663] 对于差值编码模式,不执行基于 INVERT 的逐个比特逆转过程。然而,INVERT 标记以不同容量用来表示符号 A 的值,用于导出所示预测值。更具体地,如果 INVERT 等于 0 则 A 等于 0,且如果 INVERT 等于 1 则 A 等于 1。位平面的实际值通过异或预测值和经解码的差值比特值来获取。在以上公式中,b(i,j)是在最终解码之后(即进行 Norm-2 或 Norm-6,然后是与其预测值的差值异或之后)第(i,j)位置上的比特。

[0664] 已经参照各个实施例描述和示出了本发明的各个原理,可以理解各个实施例可在排列和细节中进行更改而不背离这些原理。应当理解,在此所述的程序、过程或方法并不相关于或限于任何特定类型的计算环境,除非另有所示。各种类型的通用或专用计算环境可根据在此所述的教授内容使用或执行操作。在软件中示出的各个实施例的元素可用硬件实现,反之亦然。

[0665] 根据本发明各原理可应用其中的许多可能实施例,我们将本发明解释为可在以下权利要求及其等效方案的范围和精神内的所有这些实施例。

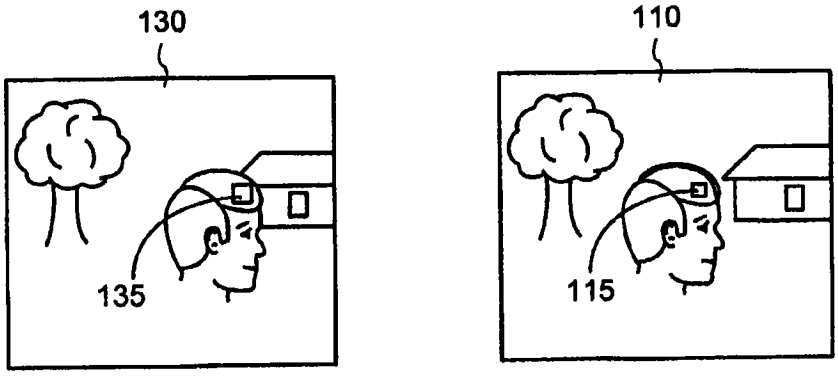


图 1

现有技术

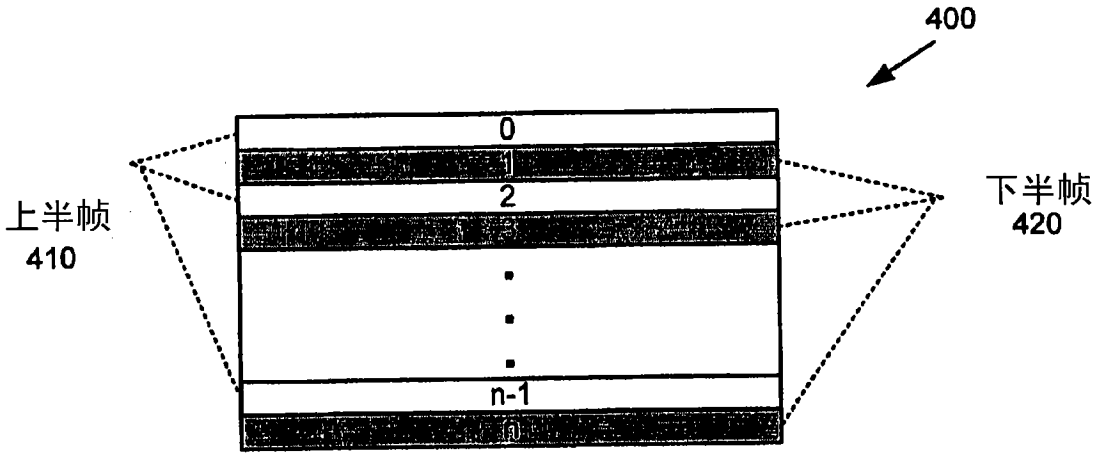


图 4

现有技术

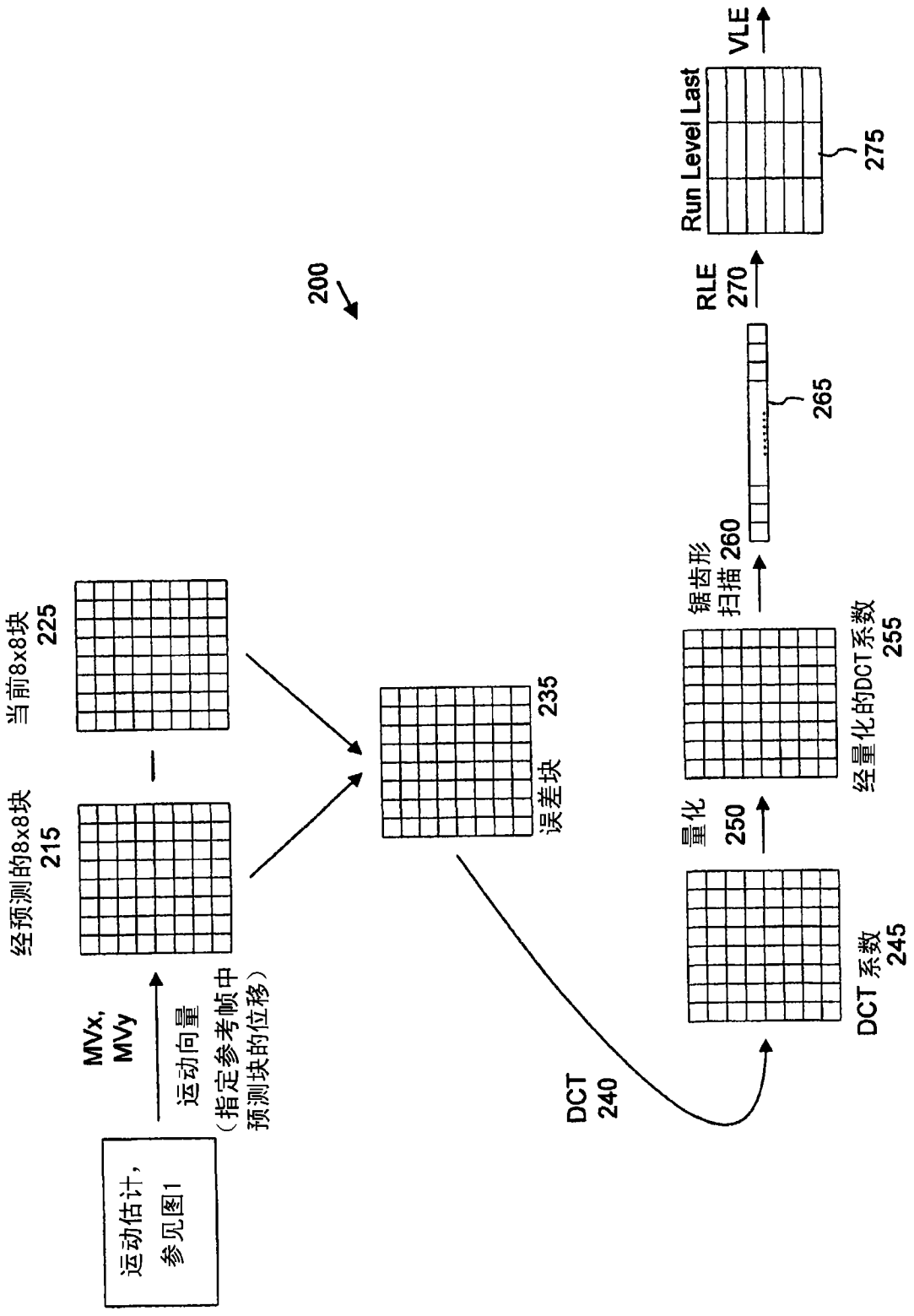

200
当前8x8块
225
经预测的8x8块
215
MVx,
MVy
运动向量
（指定参考帧中
预测块的位移）
运动估计，
参见图1
误差块 235
DCT
240
DCT 系数
245
量化
250
经量化的DCT系数
255
锯齿形
扫描 260
265
RLE
270
Run Level Last
275
VLE

图 2

现有技术

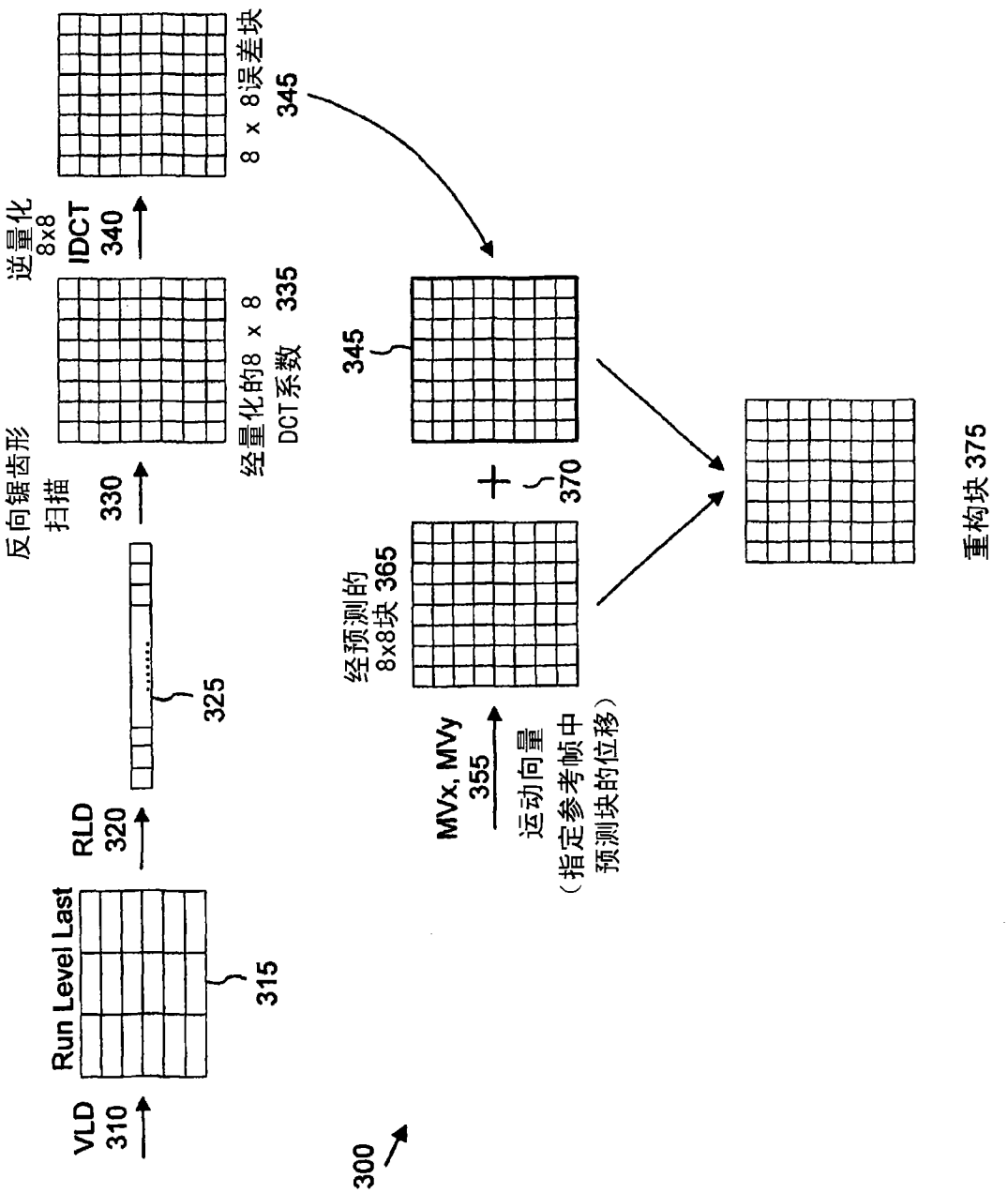

VLD
310
Run Level Last
315
RLD
320
325
反向锯齿形
扫描
330
经量化的8 x 8
DCT系数 335
逆量化
8x8
IDCT
340
8 x 8误差块
345
300
MVx, MVy
355
运动向量
（指定参考帧中
预测块的位移）
经预测的
8x8块 365
345
370
重构块 375

图 3

现有技术

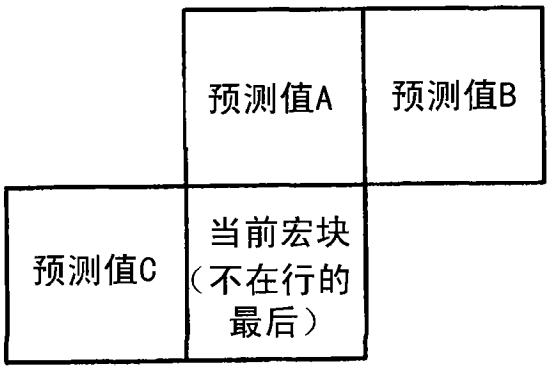


图 5A

现有技术

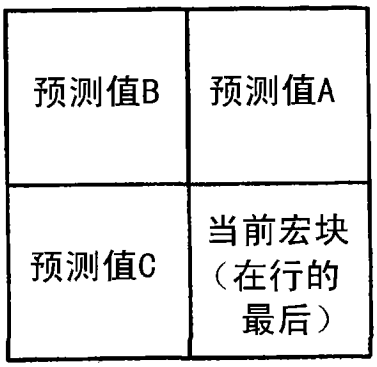


图 5B

现有技术

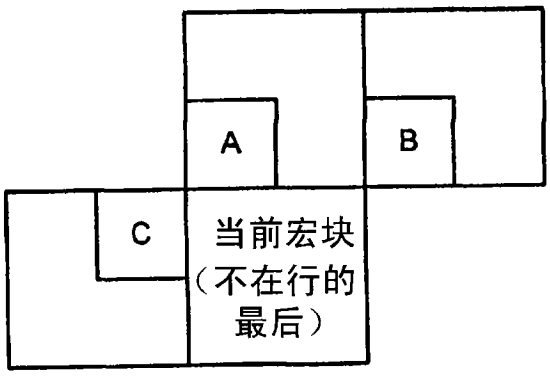


图 6A

现有技术

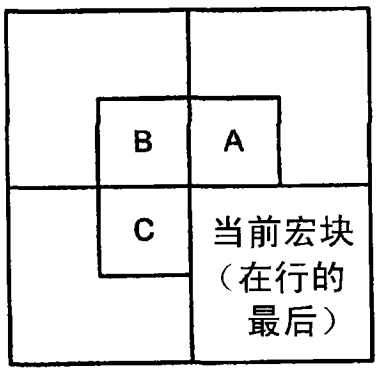


图 6B

现有技术

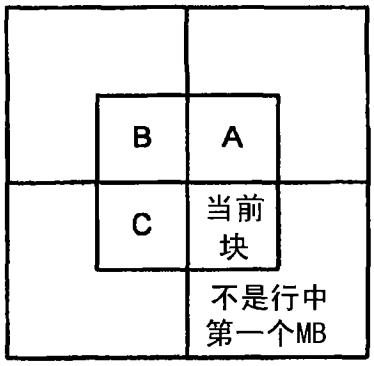


图 7A

现有技术

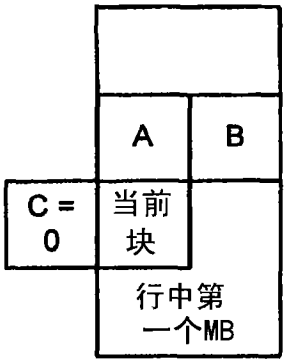


图 7B

现有技术

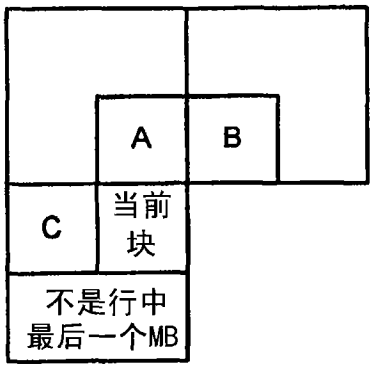


图 8A

现有技术

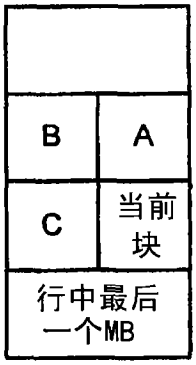


图 8B

现有技术

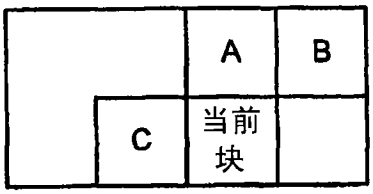


图 9

现有技术

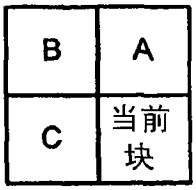


图 10

现有技术

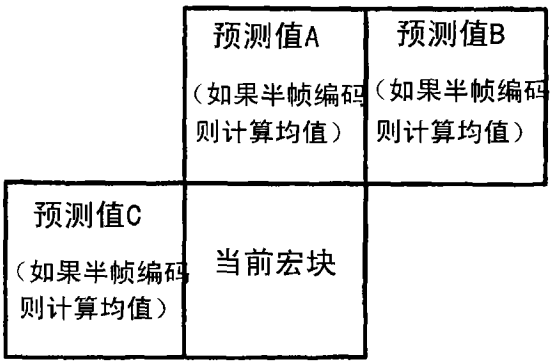


图 11

现有技术

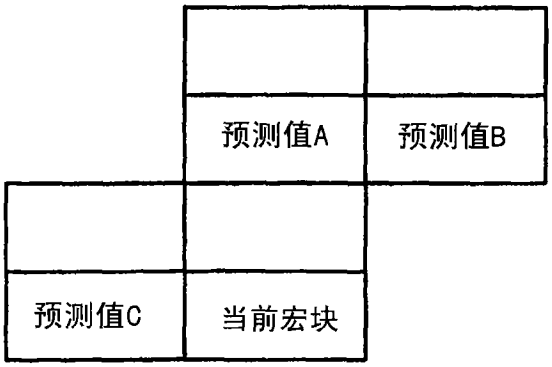


图 12A

现有技术

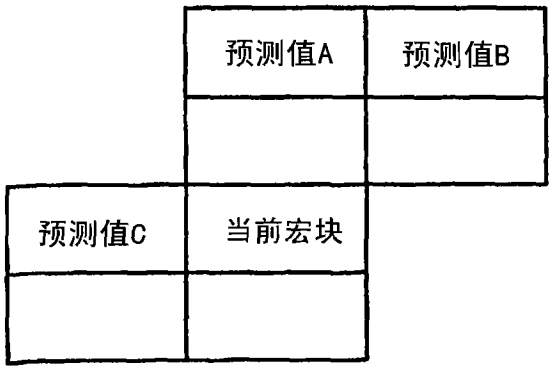


图 12B

现有技术

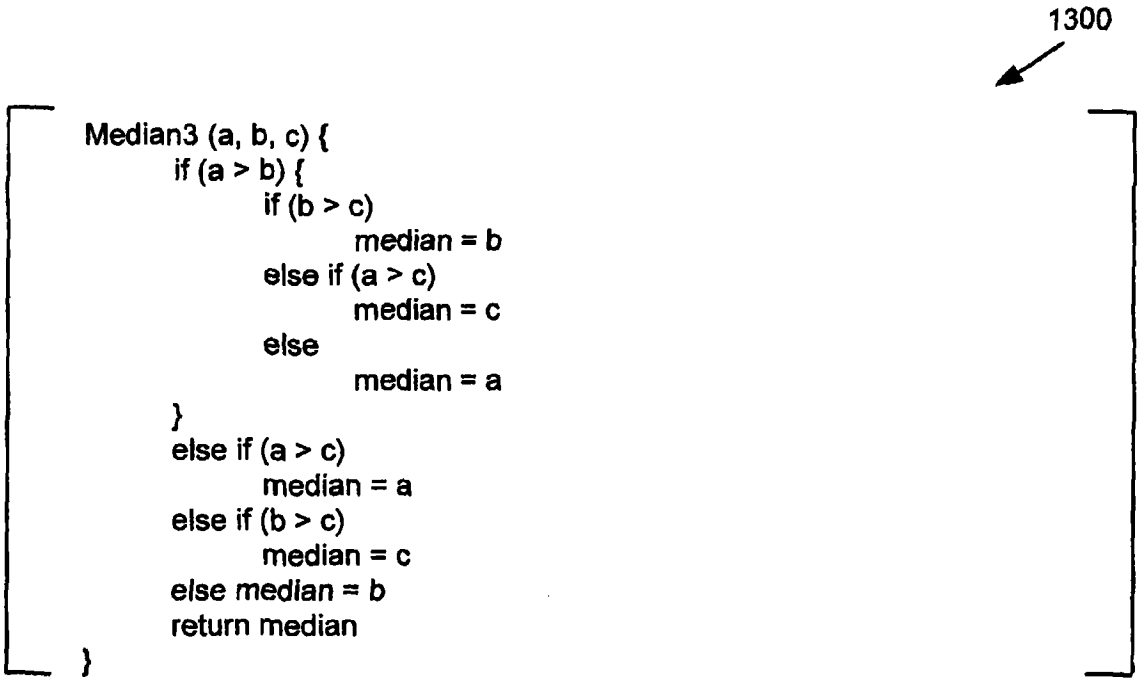


图 13

现有技术

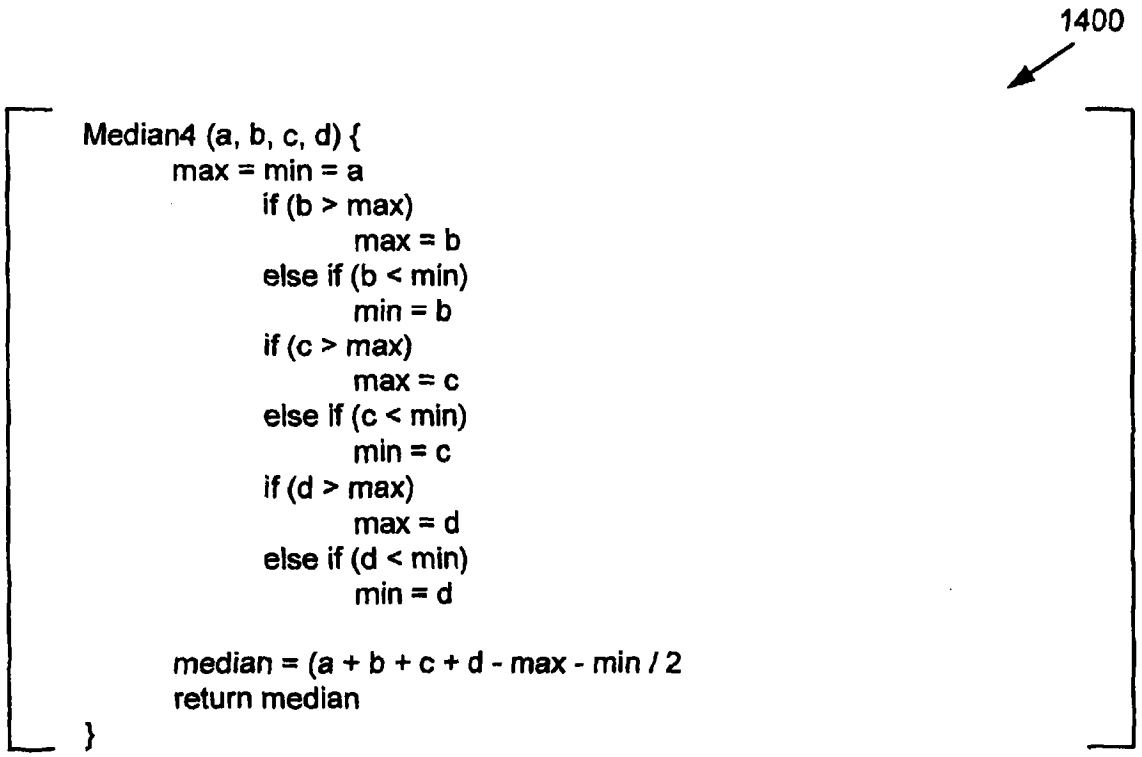


图 14

现有技术

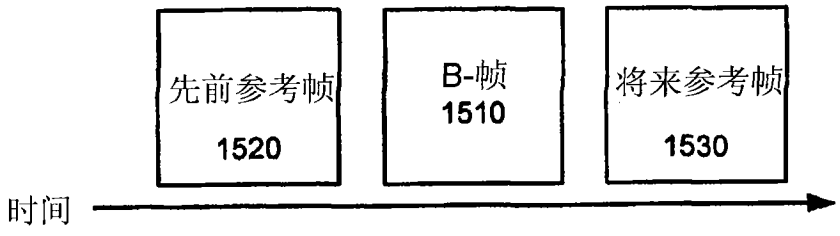


图 15

现有技术

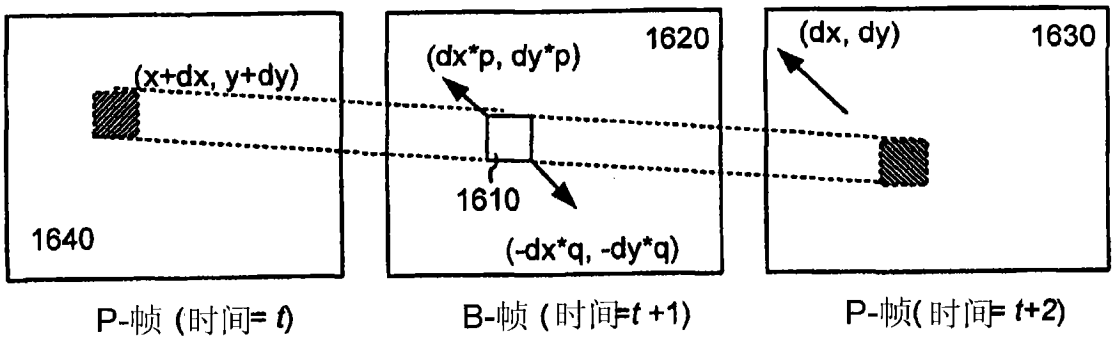


图 16

现有技术

1700

BFRACTION VLC	分数	BFRACTION VLC	分数	BFRACTION VLC	分数
000	1/2	1110000	3/5	1110111	4/7
001	1/3	1110001	4/5	1111000	5/7
010	2/3	1110010	1/6	1111001	6/7
011	1/4	1110011	5/6	1111010	1/8
100	3/4	1110100	1/7	1111011	3/8
101	1/5	1110101	2/7	1111100	5/8
110	2/5	1110110	3/7	1111101	7/8

图 17

现有技术

1800

```
Int NumShortVLC[] = {1, 1, 2, 1, 3, 1, 2};
Int DenShortVLC[] = {2, 3, 3, 4, 4, 5, 5};
Int NumLongVLC[] =  {3, 4, 1, 5, 1, 2, 3, 4, 5, 6, 1, 3, 5, 7};
Int DenLongVLC[] =  {5, 5, 6, 6, 7, 7, 7, 7, 7, 7, 8, 8, 8, 8};
Int Inverse[] = { 256, 128, 85, 64, 51, 43, 37, 32 };

Frame_Initialization(code word)
        if (long code word) {
                Numerator = NumLongVLC[code word - 112];
                Denominator = DenLongVLC[code word - 112];
        }
        else { /*短代码字*/
                Numerator = NumShortVLC[code word];
                Denominator = DenShortVLC[code word];
        }
        FrameReciprocal = Inverse[Denominator - 1];
        ScaleFactor = Numerator * FrameReciprocal;
End Frame_Initialization
```

图 18

现有技术

1900

```
Scale_Direct_MV (IN MV_X, IN MV_Y, OUT MV_XF , OUT MV_YF, OUT
MV_XB, OUT MV_YB)
        if (Half pel units) { //如果当前B图片使用1/2像素MV
                MV_XF = 2 * ((MV_X * ScaleFactor + 255) >> 9);
                MV_YF = 2 * ((MV_Y * ScaleFactor + 255) >> 9);
                MV_XB = 2 * ((MV_X * (ScaleFactor - 256) + 255) >> 9);
                MV_YB = 2 * ((MV_Y * (ScaleFactor - 256) + 255) >> 9);
        }
        else {/*1/4像素单元*/
                MV_XF = (MV_X * ScaleFactor + 128) >> 8;
                MV_YF = (MV_Y * ScaleFactor + 128) >> 8;
                MV_XB = (MV_X * (ScaleFactor - 256) + 128) >> 8;
                MV_YB = (MV_Y * (ScaleFactor - 256) + 128) >> 8;
        }
End Scale_Direct_MV
```

图 19

现有技术

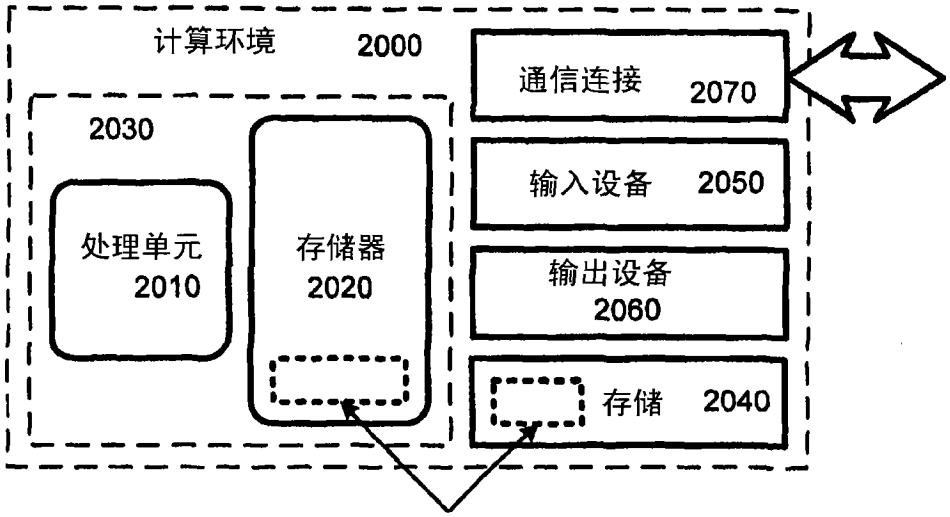

计算环境
2000
2030
处理单元
2010
存储器
2020
通信连接
2070
输入设备
2050
输出设备
2060
存储
2040
实现具有隔行扫描视频帧的双向预测的
视频编码器或解码器的软件 2080

图 20

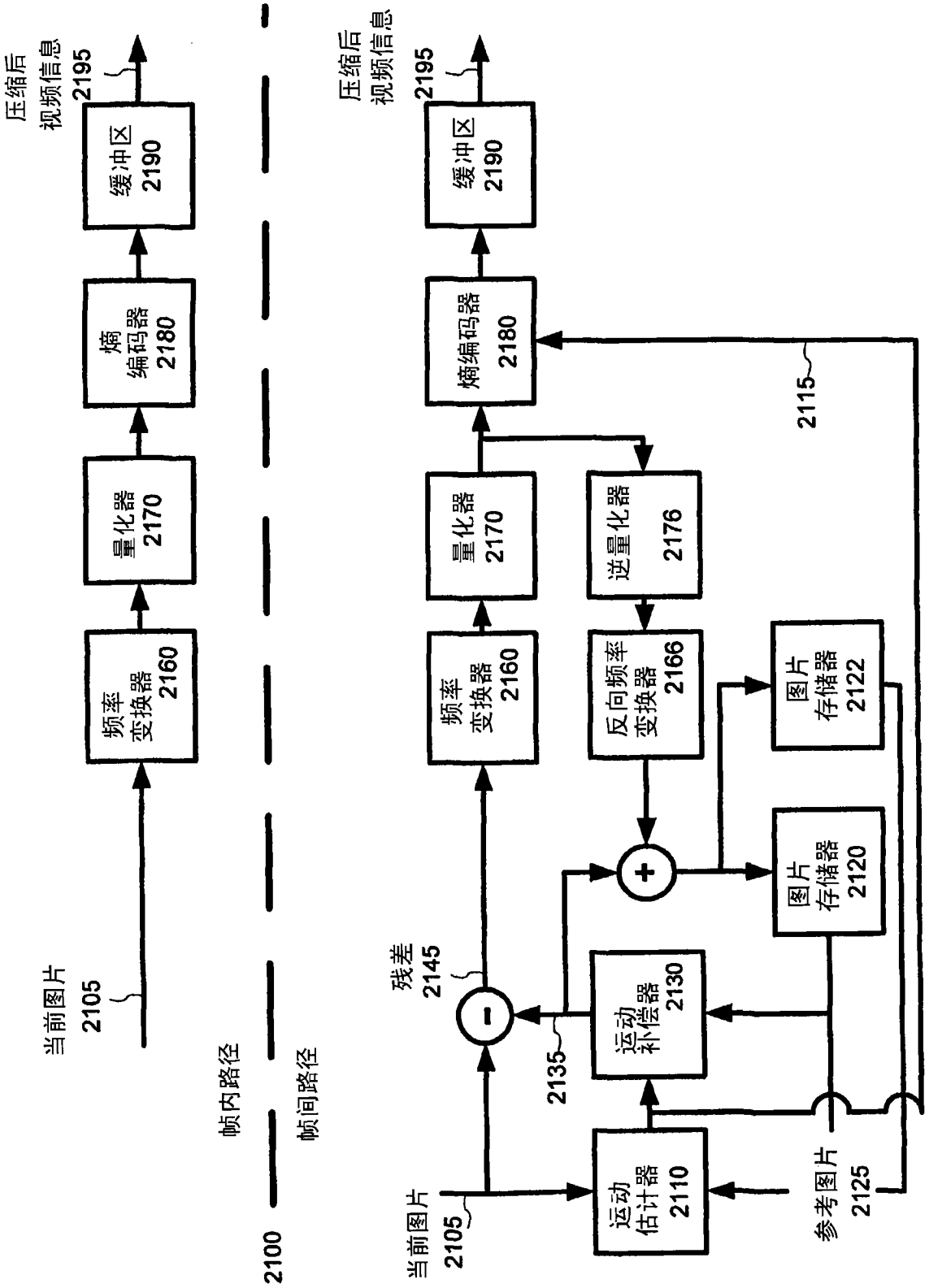

当前图片
2105
频率变换器
2160
量化器
2170
熵编码器
2180
缓冲区
2190
压缩后视频信息
2195
帧内路径
帧间路径
2100
当前图片
2105
残差
2145
频率变换器
2160
量化器
2170
熵编码器
2180
缓冲区
2190
压缩后视频信息
2195
2115
逆量化器
2176
反向频率变换器
2166
图片存储器
2122
图片存储器
2120
运动补偿器
2130
2135
运动估计器
2110
参考图片
2125

图 21

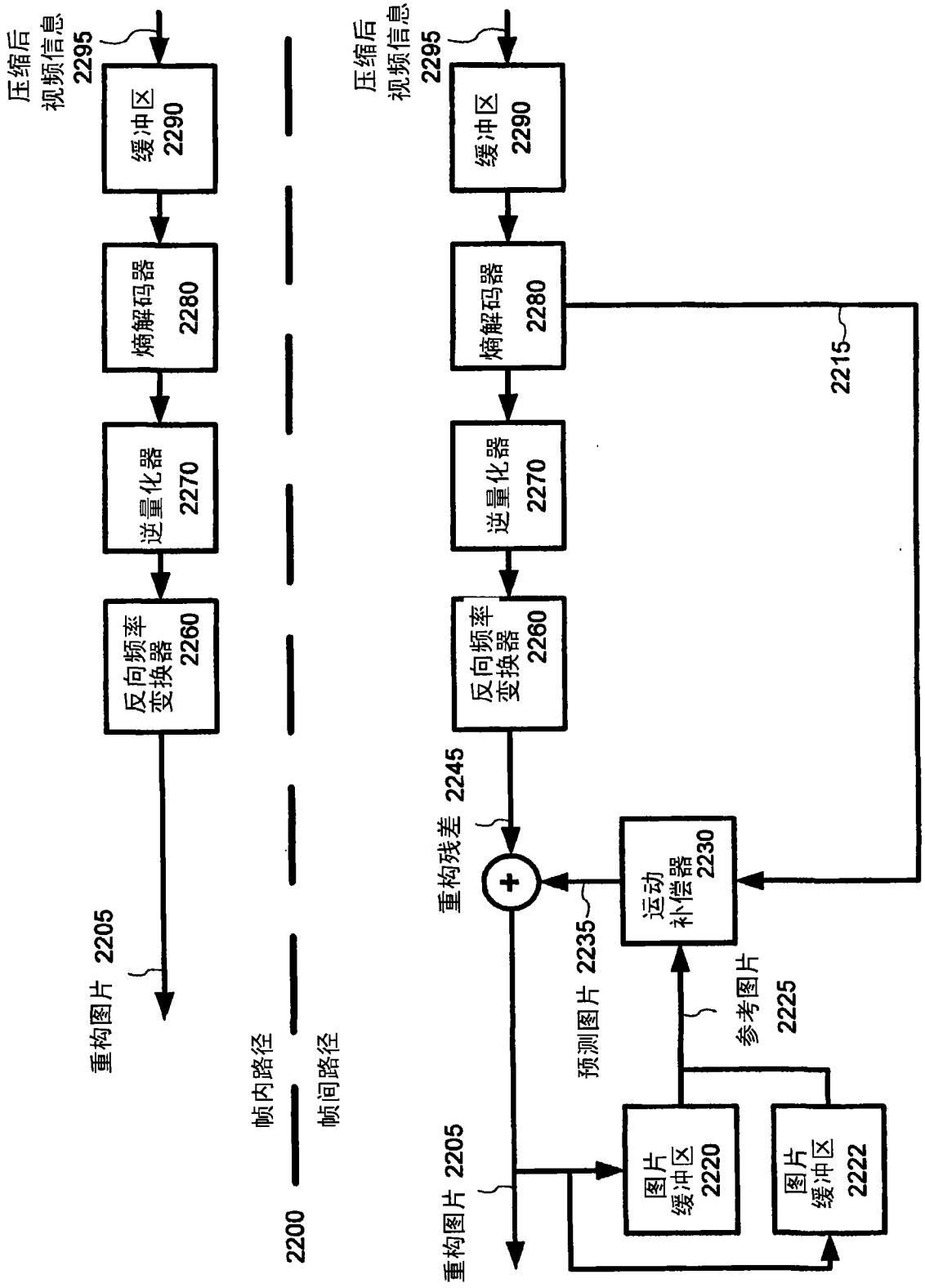

压缩后
视频信息
2295
缓冲区
2290
熵解码器
2280
逆量化器
2270
反向频率
变换器
2260
重构图片 2205
帧内路径
帧间路径
2200
重构残差 2245
运动
补偿器
2230
预测图片 2235
参考图片
2225
图片
缓冲区
2220
图片
缓冲区
2222
2215

图 22

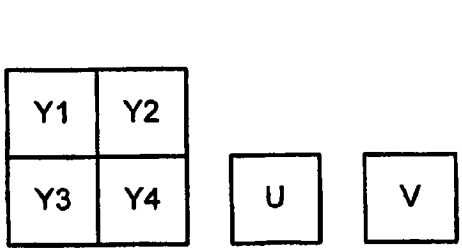


图 23

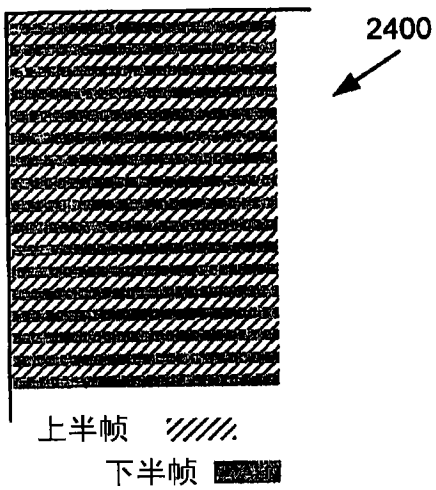


图 24A

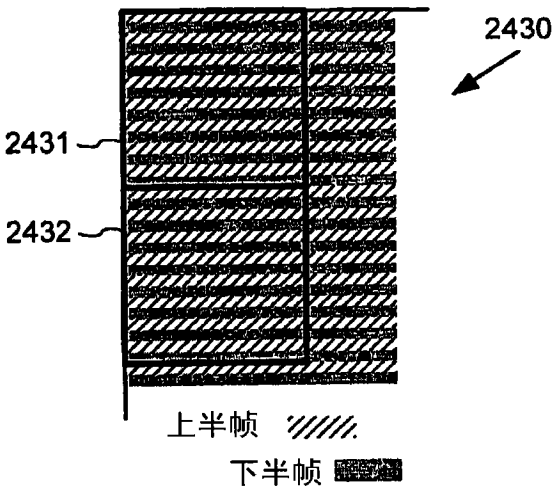


图 24B

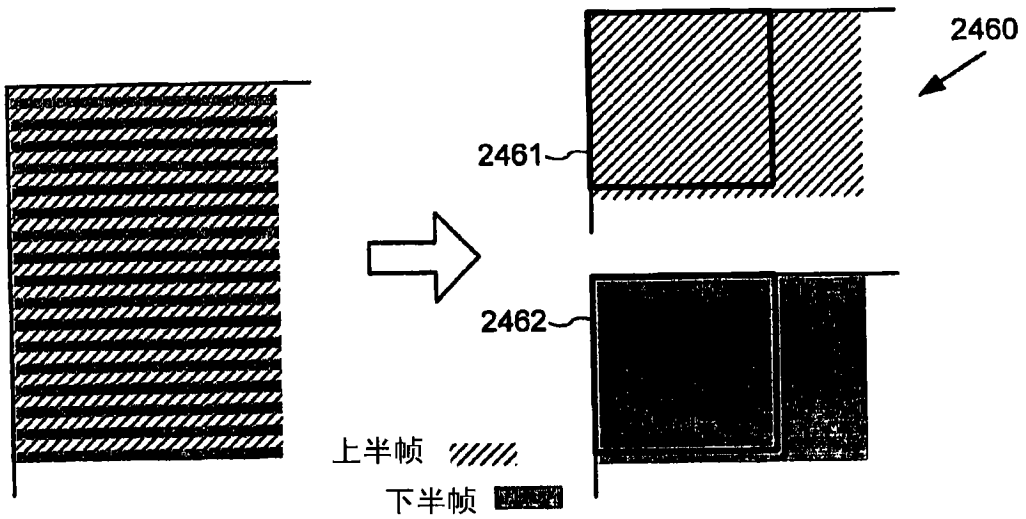

2460
2461
2462
上半帧
下半帧

图 24C

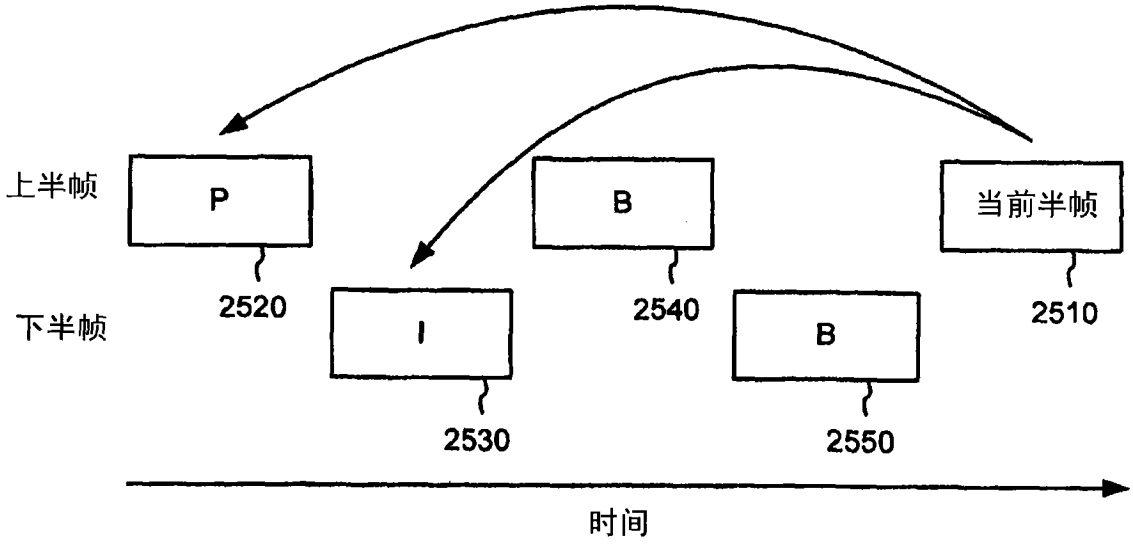

上半帧
下半帧
P
B
当前半帧
I
B
2520
2540
2510
2530
2550
时间

图 25

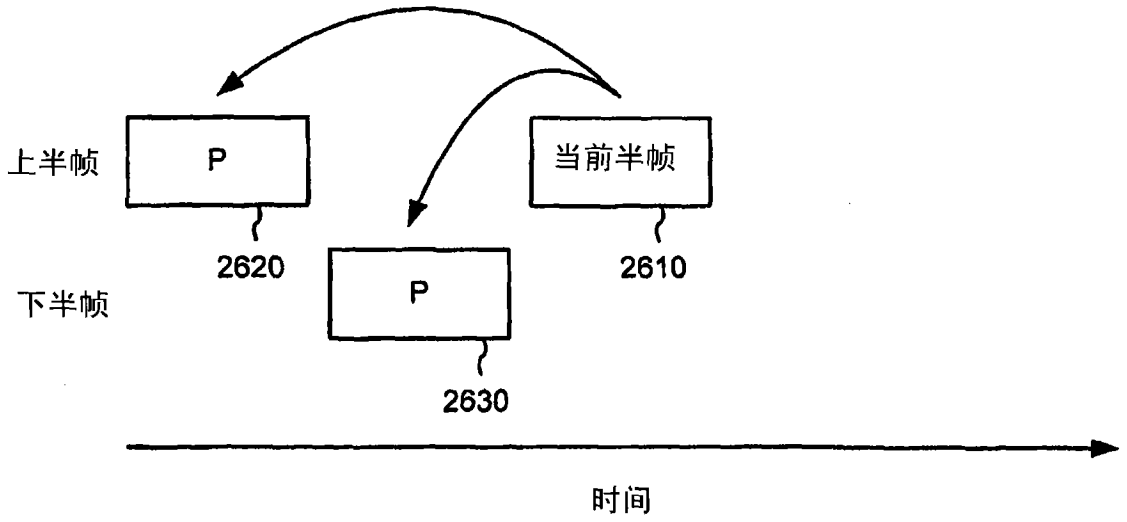

上半帧
P
当前半帧
2620
2610
下半帧
P
2630
时间

图 26

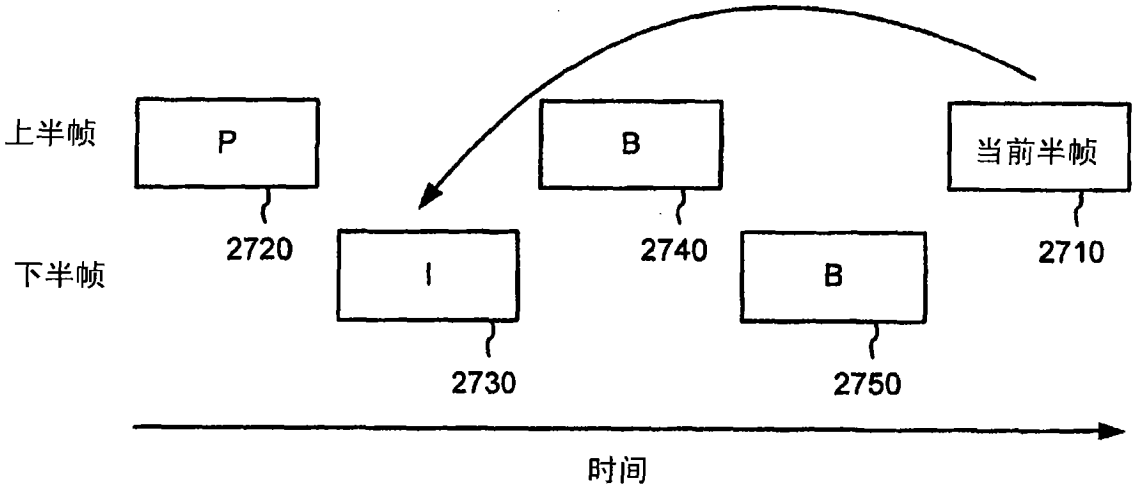

上半帧
P
B
当前半帧
2720
2740
2710
下半帧
I
B
2730
2750
时间

图 27

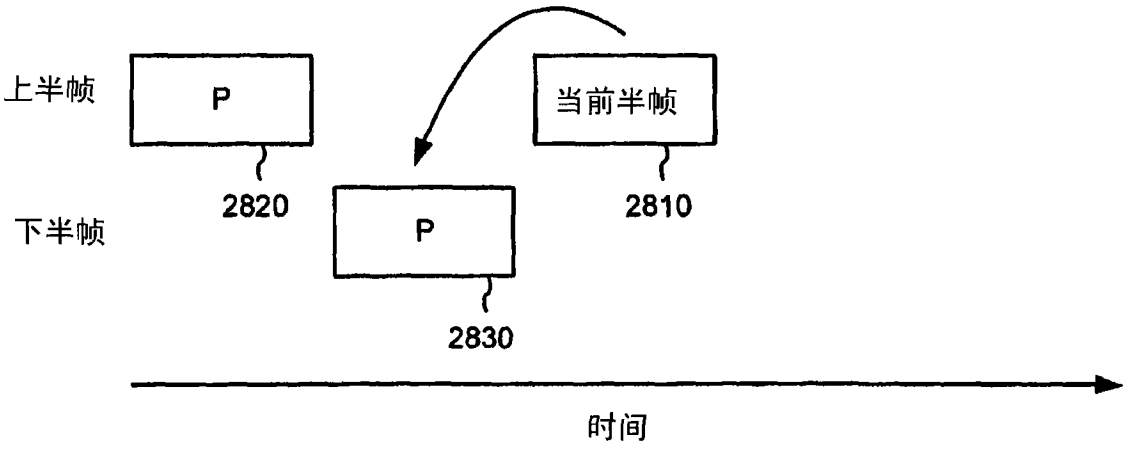

上半帧
P
当前半帧
2820
2810
下半帧
P
2830
时间

图 28

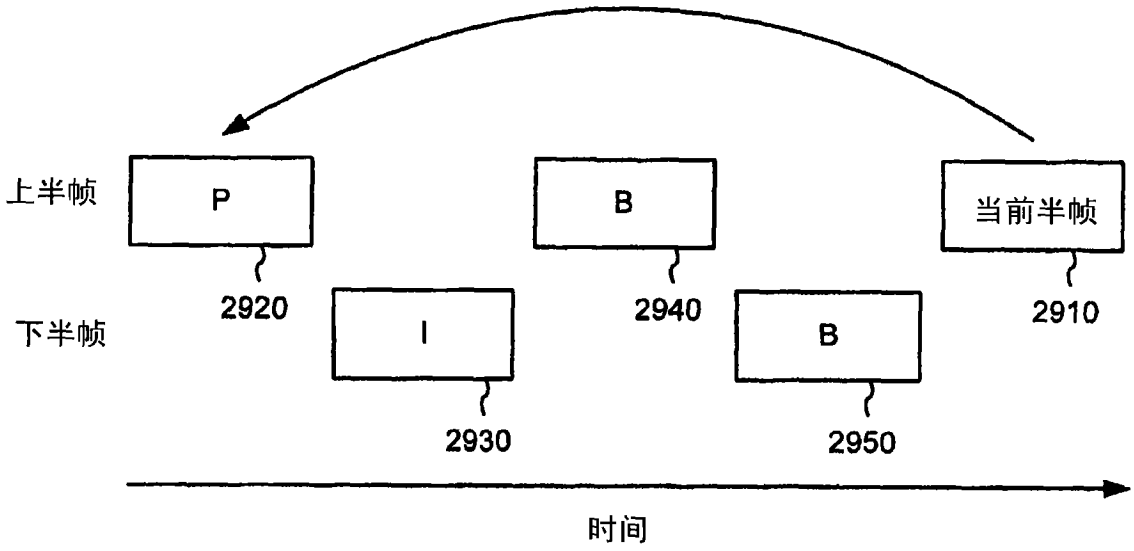

上半帧
下半帧
P
B
当前半帧
I
B
2920
2940
2910
2930
2950
时间

图 29

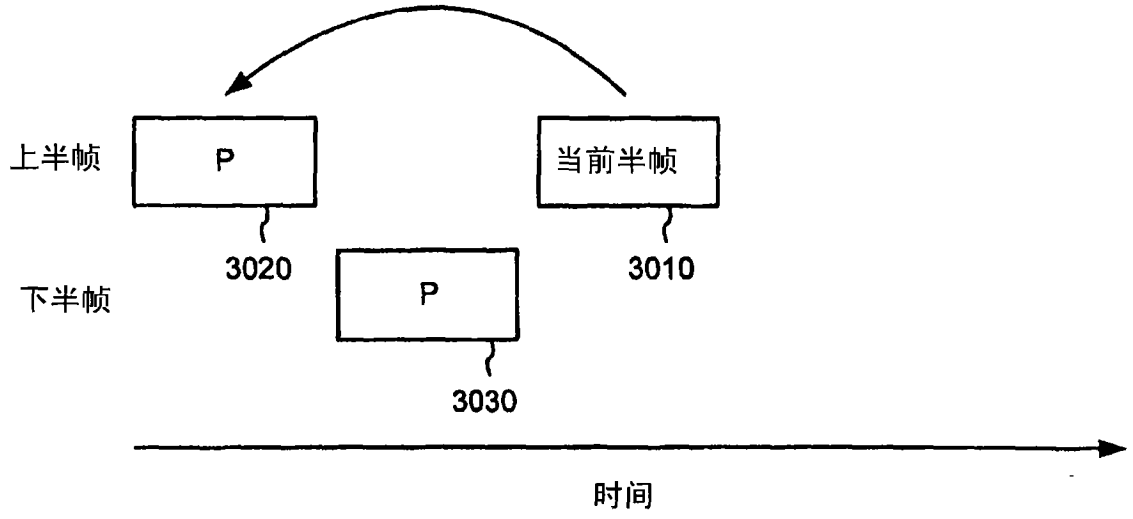

上半帧
下半帧
P
当前半帧
P
3020
3010
3030
时间

图 30

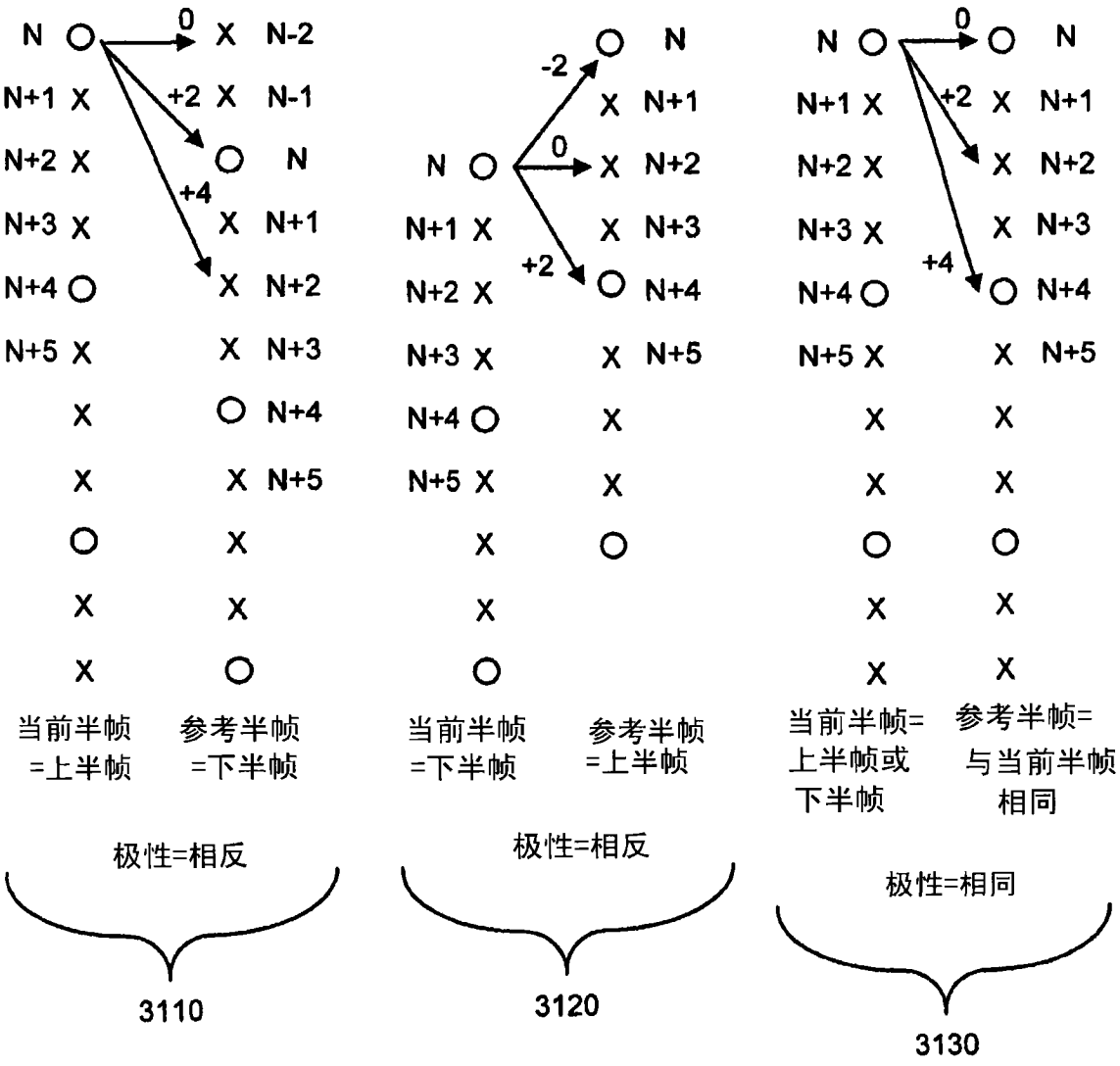

N
N+1
N+2
N+3
N+4
N+5
0
+2
+4
N-2
N-1
N
N+1
N+2
N+3
N+4
N+5
当前半帧
=上半帧
参考半帧
=下半帧
极性=相反
3110
-2
0
+2
N
N+1
N+2
N+3
N+4
N+5
当前半帧
=下半帧
参考半帧
=上半帧
极性=相反
3120
当前半帧=
上半帧或
下半帧
参考半帧=
与当前半帧
相同
极性=相同
3130

图 31

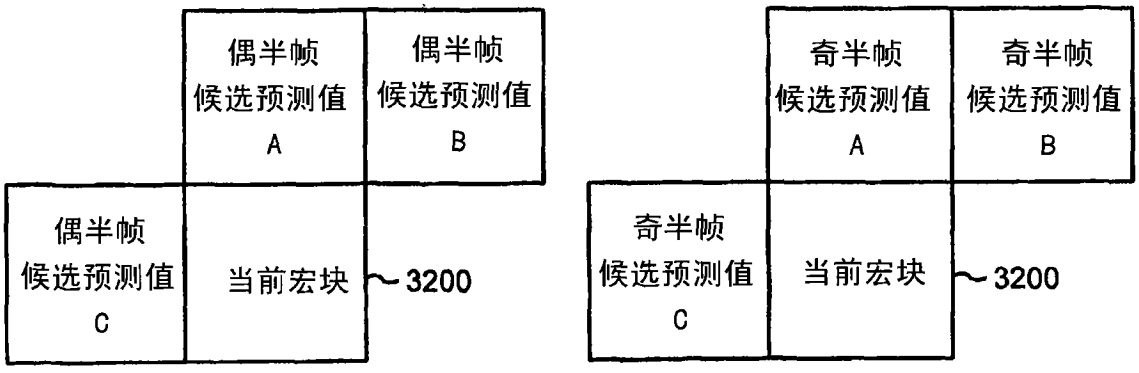

偶半帧
候选预测值
A
偶半帧
候选预测值
B
偶半帧
候选预测值
C
当前宏块
3200
奇半帧
候选预测值
A
奇半帧
候选预测值
B
奇半帧
候选预测值
C
当前宏块
3200

图 32

3300

```
if (predictorA is not out of bounds) {
   if (predictorC is not out of bounds) {
      if (predictorA is intra) {
          predictorA_x = 0
          predictorA_y = 0
      }
      if (predictorB is intra) {
         predictorB_x = 0
         predictorB_y = 0
      }
      if (predictorC is intra) {
         predictorC_x = 0
         predictorC_y = 0
      }
      if (predictorA is from same field) {
         samecount = samecount + 1
         samefieldpredA_x = predictorA_x
         samefieldpredA_y = predictorA_y
         oppositefieldpredA_x = scaleforopposite_x(predictorA_x)
         oppositefieldpredA_y = scaleforopposite_y(predictorA_y)
      }
      else {
         oppositecount = oppositecount + 1
         oppositefieldpredA_x = predictorA_x
         oppositefieldpredA_y = predictorA_y
         samefieldpredA_x = scaleforsame_x(predictorA_x)
         samefieldpredA_y = scaleforsame_y(predictorA_y)
      }
      if (predictorB is from same field) {
         samecount = samecount + 1
         samefieldpredB_x = predictorB_x
         samefieldpredB_y = predictorB_y
         oppositefieldpredB_x = scaleforopposite_x(predictorB_x)
         oppositefieldpredB_y = scaleforopposite_y(predictorB_y)
      }
      else {
         oppositecount  = oppositecount + 1
         oppositefieldpredB_x = predictorB_x
         oppositefieldpredB _y = predictorB_y
         samefieldpredB_x = scaleforsame_x(predictorB_x)
         samefieldpredB_y = scaleforsame_y(predictorB_y)
      }
```

图 33A

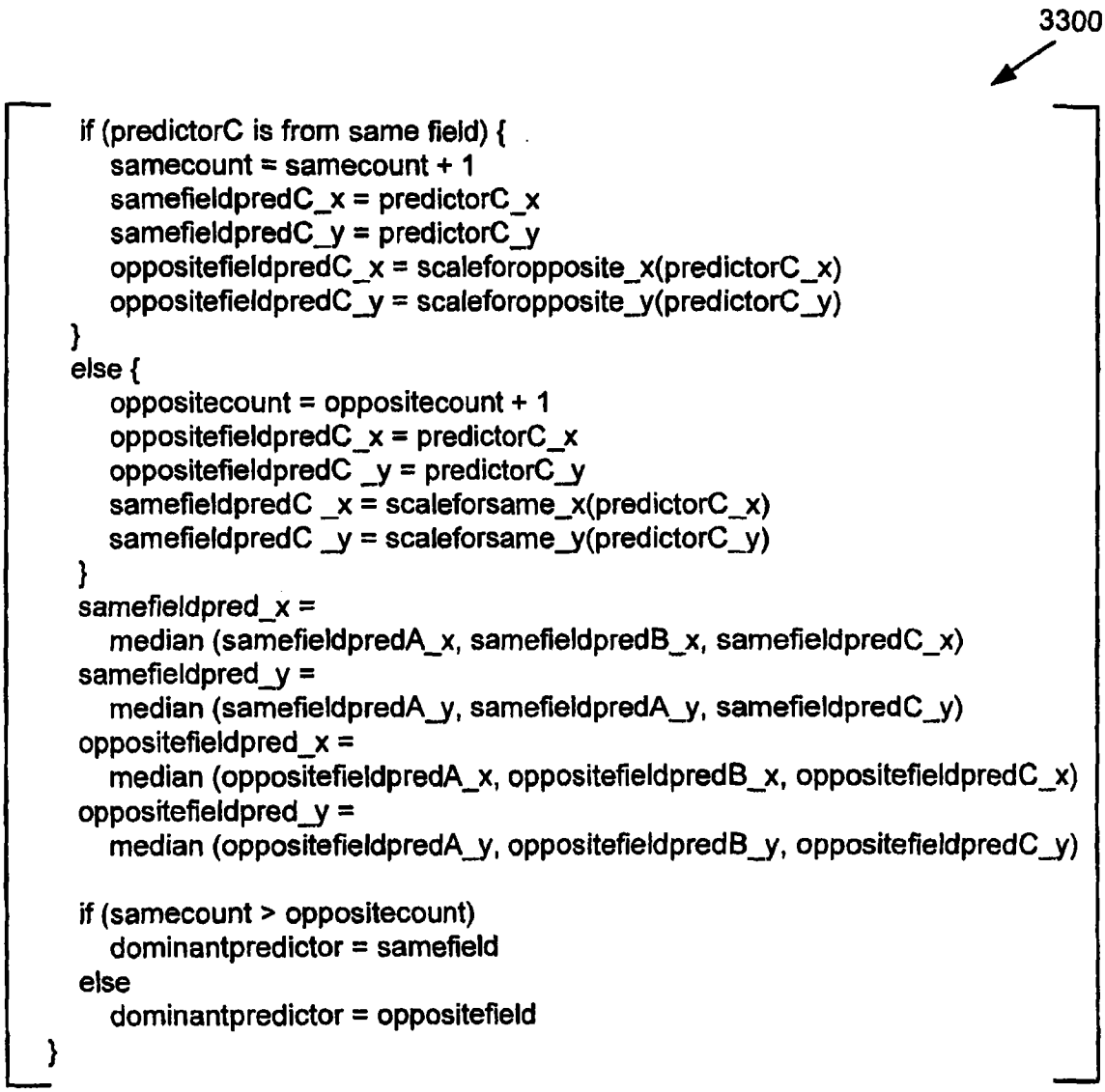

3300
if (predictorC is from same field) {
samecount = samecount + 1
samefieldpredC_x = predictorC_x
samefieldpredC_y = predictorC_y
oppositefieldpredC_x = scaleforopposite_x(predictorC_x)
oppositefieldpredC_y = scaleforopposite_y(predictorC_y)
}
else {
oppositecount = oppositecount + 1
oppositefieldpredC_x = predictorC_x
oppositefieldpredC _y = predictorC_y
samefieldpredC _x = scaleforsame_x(predictorC_x)
samefieldpredC _y = scaleforsame_y(predictorC_y)
}
samefieldpred_x =
median (samefieldpredA_x, samefieldpredB_x, samefieldpredC_x)
samefieldpred_y =
median (samefieldpredA_y, samefieldpredA_y, samefieldpredC_y)
oppositefieldpred_x =
median (oppositefieldpredA_x, oppositefieldpredB_x, oppositefieldpredC_x)
oppositefieldpred_y =
median (oppositefieldpredA_y, oppositefieldpredB_y, oppositefieldpredC_y)
if (samecount > oppositecount)
dominantpredictor = samefield
else
dominantpredictor = oppositefield
}

图 33B

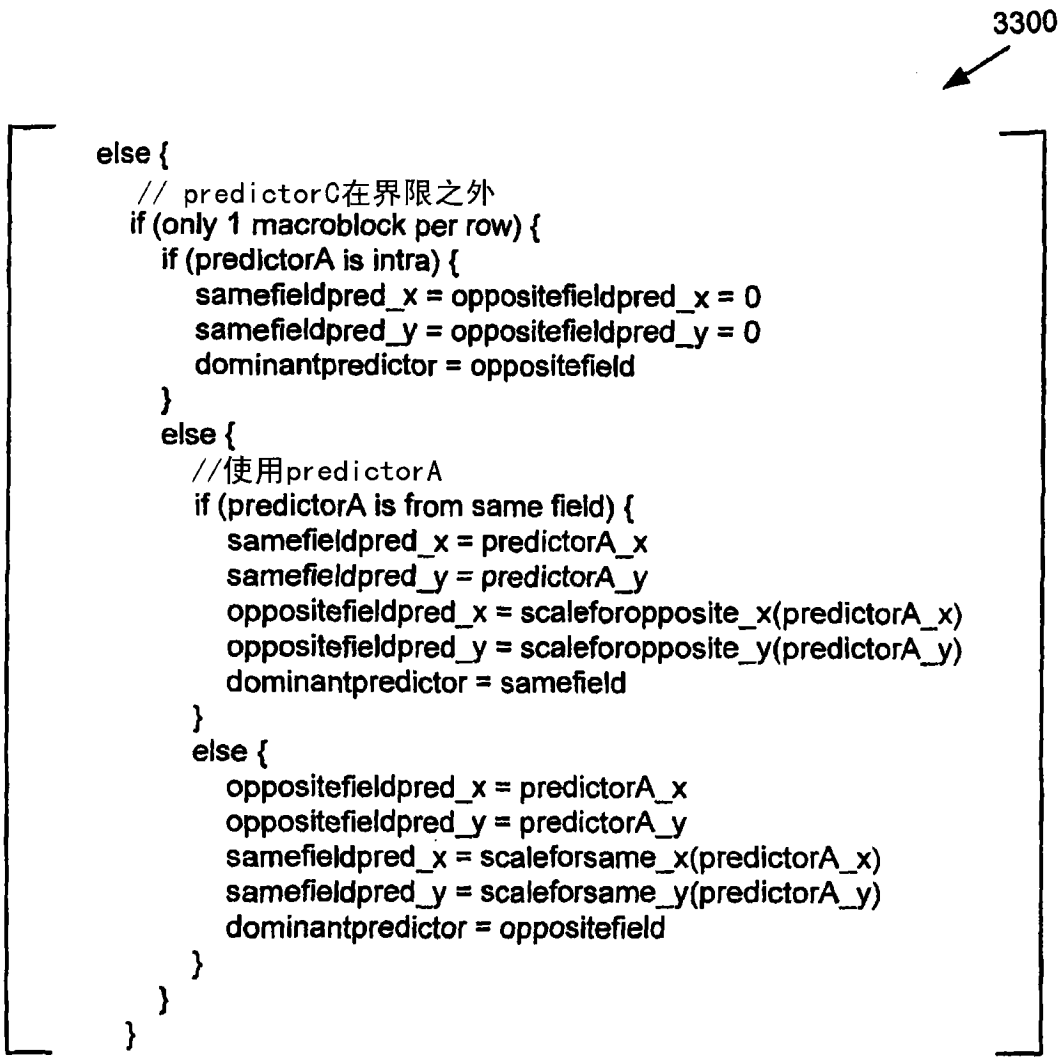

3300
else {
// predictorC在界限之外
if (only 1 macroblock per row) {
if (predictorA is intra) {
samefieldpred_x = oppositefieldpred_x = 0
samefieldpred_y = oppositefieldpred_y = 0
dominantpredictor = oppositefield
}
else {
//使用predictorA
if (predictorA is from same field) {
samefieldpred_x = predictorA_x
samefieldpred_y = predictorA_y
oppositefieldpred_x = scaleforopposite_x(predictorA_x)
oppositefieldpred_y = scaleforopposite_y(predictorA_y)
dominantpredictor = samefield
}
else {
oppositefieldpred_x = predictorA_x
oppositefieldpred_y = predictorA_y
samefieldpred_x = scaleforsame_x(predictorA_x)
samefieldpred_y = scaleforsame_y(predictorA_y)
dominantpredictor = oppositefield
}
}
}

图 33C

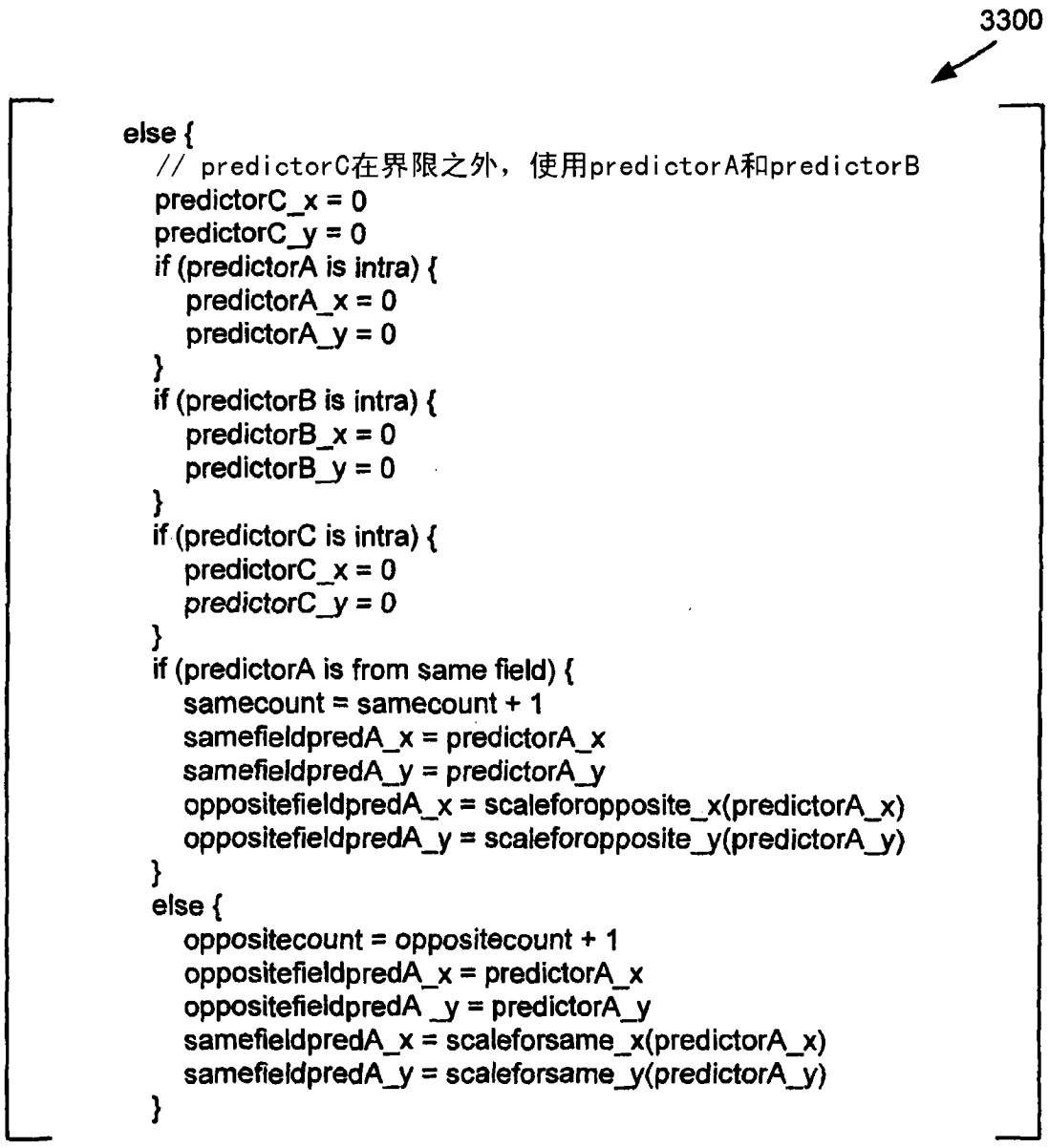

3300
else {
// predictorC在界限之外，使用predictorA和predictorB
predictorC_x = 0
predictorC_y = 0
if (predictorA is intra) {
predictorA_x = 0
predictorA_y = 0
}
if (predictorB is intra) {
predictorB_x = 0
predictorB_y = 0
}
if (predictorC is intra) {
predictorC_x = 0
predictorC_y = 0
}
if (predictorA is from same field) {
samecount = samecount + 1
samefieldpredA_x = predictorA_x
samefieldpredA_y = predictorA_y
oppositefieldpredA_x = scaleforopposite_x(predictorA_x)
oppositefieldpredA_y = scaleforopposite_y(predictorA_y)
}
else {
oppositecount = oppositecount + 1
oppositefieldpredA_x = predictorA_x
oppositefieldpredA_y = predictorA_y
samefieldpredA_x = scaleforsame_x(predictorA_x)
samefieldpredA_y = scaleforsame_y(predictorA_y)
}

图 33D

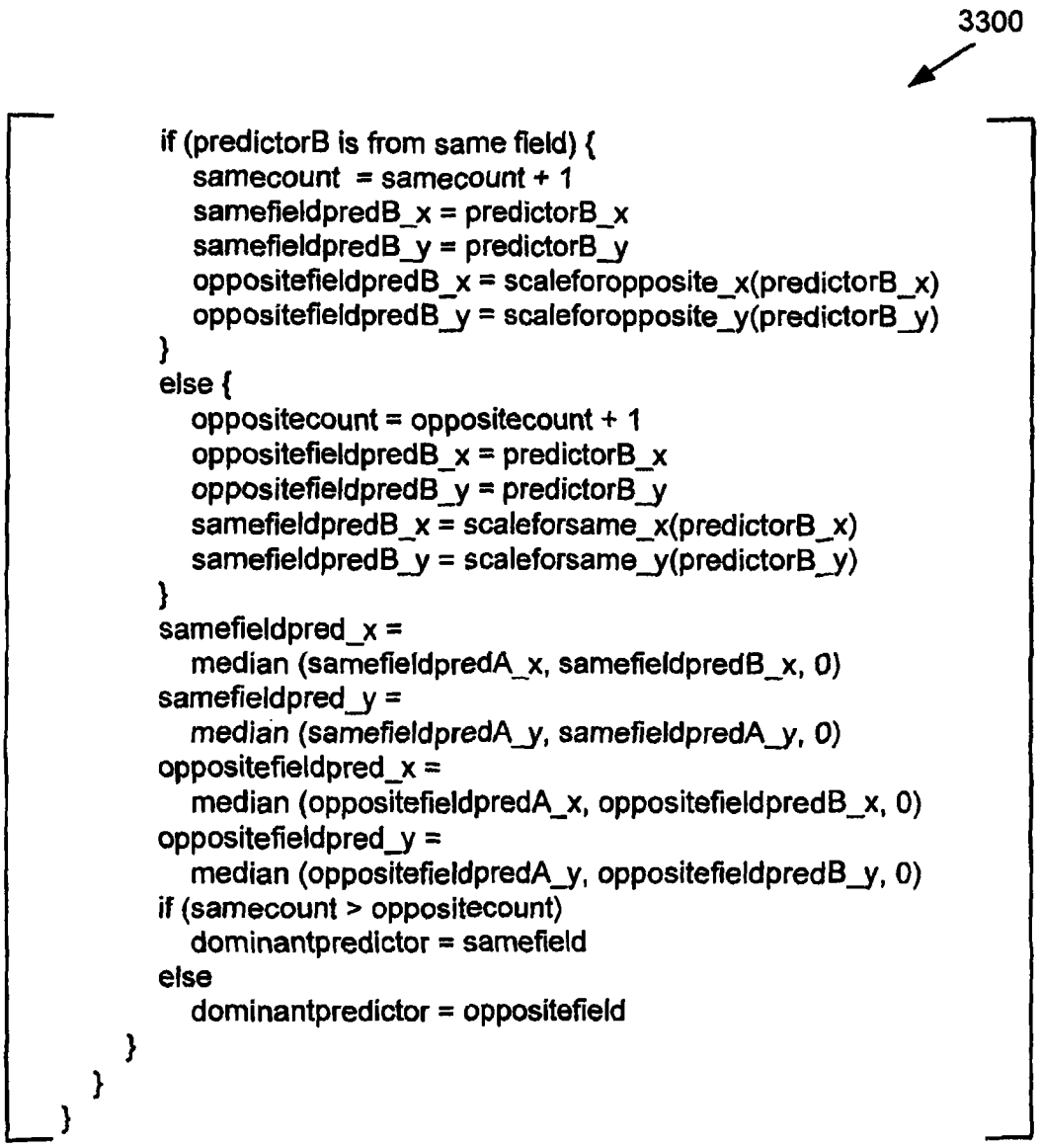

3300
if (predictorB is from same field) {
samecount = samecount + 1
samefieldpredB_x = predictorB_x
samefieldpredB_y = predictorB_y
oppositefieldpredB_x = scaleforopposite_x(predictorB_x)
oppositefieldpredB_y = scaleforopposite_y(predictorB_y)
}
else {
oppositecount = oppositecount + 1
oppositefieldpredB_x = predictorB_x
oppositefieldpredB_y = predictorB_y
samefieldpredB_x = scaleforsame_x(predictorB_x)
samefieldpredB_y = scaleforsame_y(predictorB_y)
}
samefieldpred_x =
median (samefieldpredA_x, samefieldpredB_x, 0)
samefieldpred_y =
median (samefieldpredA_y, samefieldpredA_y, 0)
oppositefieldpred_x =
median (oppositefieldpredA_x, oppositefieldpredB_x, 0)
oppositefieldpred_y =
median (oppositefieldpredA_y, oppositefieldpredB_y, 0)
if (samecount > oppositecount)
dominantpredictor = samefield
else
dominantpredictor = oppositefield
}
}
}

图 33E


3300
else {
// predictorA在界限之外
if (predictorC is out of bounds) {
samefieldpred_x = oppositefieldpred_x = 0
samefieldpred_y = oppositefieldpred_y = 0
dominantpredictor = oppositefield
}
else {
//使用predictorC
if (predictorC is from same field) {
samefieldpred_x = predictorC_x
samefieldpred_y = predictorC_y
oppositefieldpred_x = scaleforopposite_x(predictorC_x)
oppositefieldpred_y = scaleforopposite_y(predictorC_y)
dominantpredictor = samefield
}
else {
oppositefieldpred_x = predictorC_x
oppositefieldpred_y = predictorC_y
samefieldpred_x = scaleforsame_x(predictorC_x)
samefieldpred_y = scaleforsame_y(predictorC_y)
dominantpredictor = oppositefield
}
}
}

图 33F

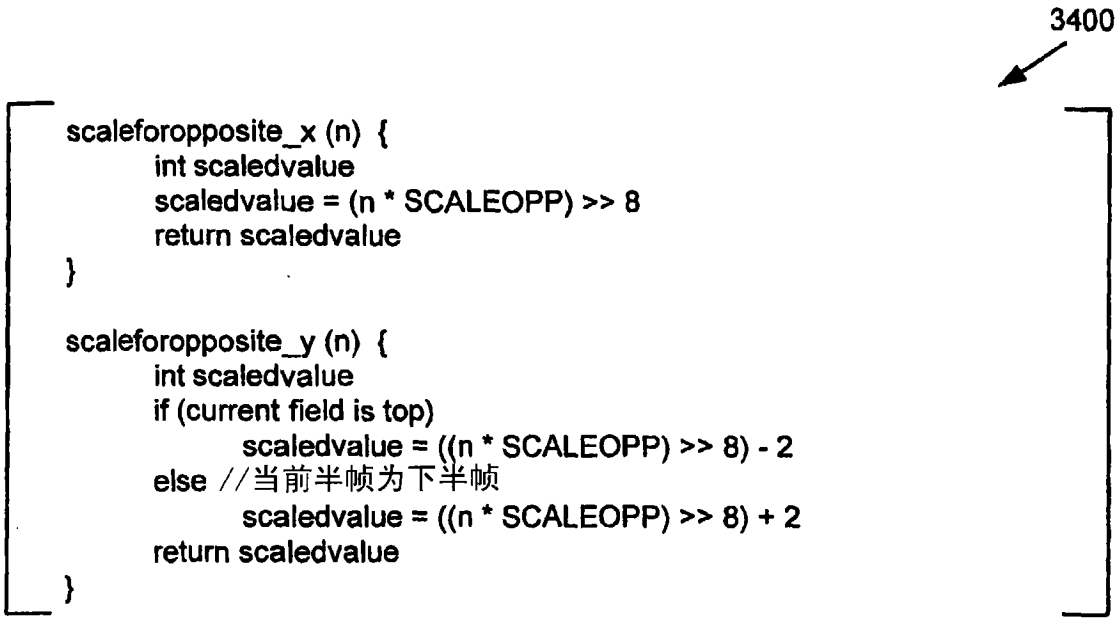

3400
scaleforopposite_x (n) {
int scaledvalue
scaledvalue = (n * SCALEOPP) >> 8
return scaledvalue
}
scaleforopposite_y (n) {
int scaledvalue
if (current field is top)
scaledvalue = ((n * SCALEOPP) >> 8) - 2
else //当前半帧为下半帧
scaledvalue = ((n * SCALEOPP) >> 8) + 2
return scaledvalue
}

图 34A

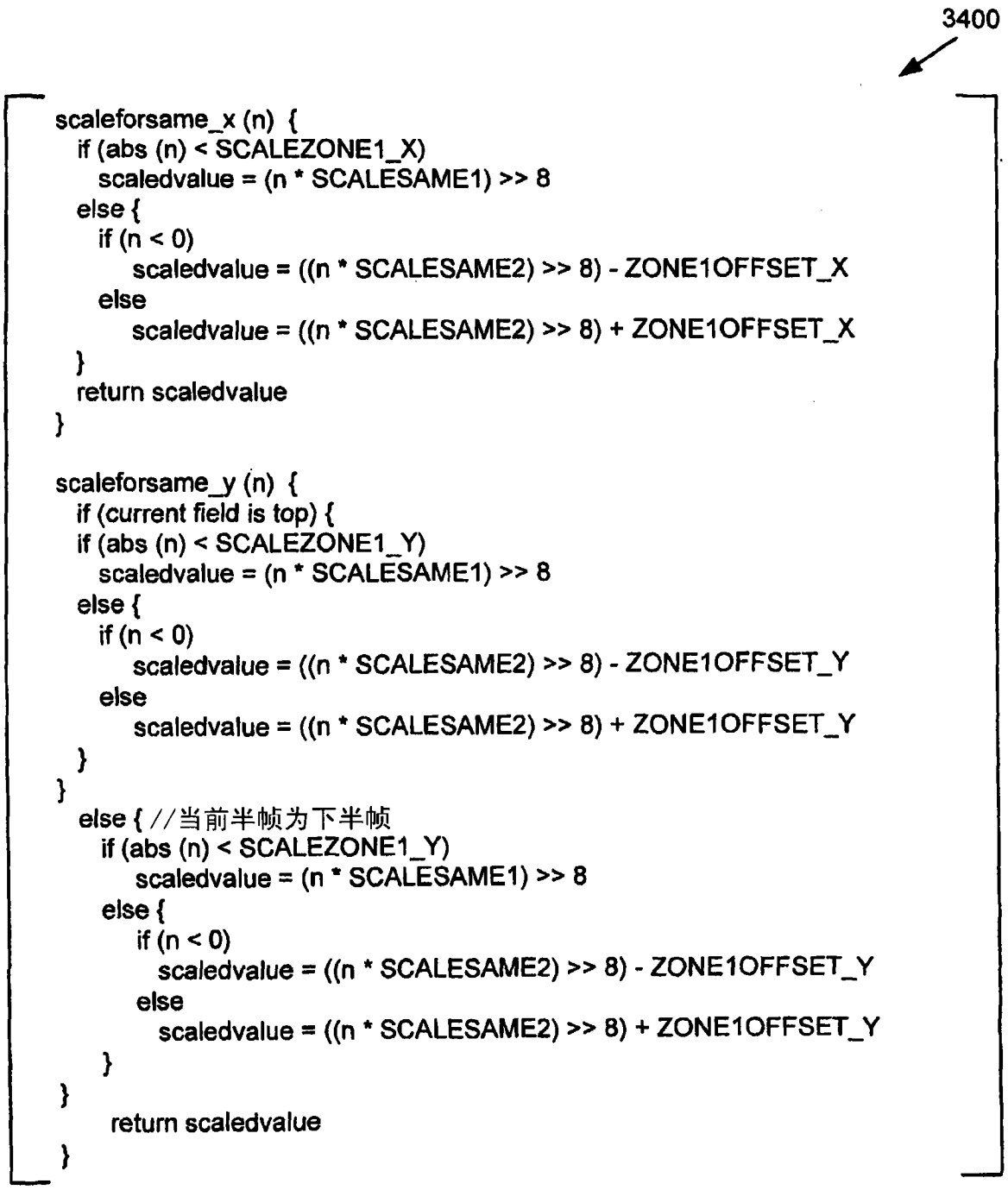

3400
scaleforsame_x (n) {
if (abs (n) < SCALEZONE1_X)
scaledvalue = (n * SCALESAME1) >> 8
else {
if (n < 0)
scaledvalue = ((n * SCALESAME2) >> 8) - ZONE1OFFSET_X
else
scaledvalue = ((n * SCALESAME2) >> 8) + ZONE1OFFSET_X
}
return scaledvalue
}
scaleforsame_y (n) {
if (current field is top) {
if (abs (n) < SCALEZONE1_Y)
scaledvalue = (n * SCALESAME1) >> 8
else {
if (n < 0)
scaledvalue = ((n * SCALESAME2) >> 8) - ZONE1OFFSET_Y
else
scaledvalue = ((n * SCALESAME2) >> 8) + ZONE1OFFSET_Y
}
}
else { //当前半帧为下半帧
if (abs (n) < SCALEZONE1_Y)
scaledvalue = (n * SCALESAME1) >> 8
else {
if (n < 0)
scaledvalue = ((n * SCALESAME2) >> 8) - ZONE1OFFSET_Y
else
scaledvalue = ((n * SCALESAME2) >> 8) + ZONE1OFFSET_Y
}
}
return scaledvalue
}

图 34B

3500

	参考帧距离			
	1	2	3	4或以上
SCALEOPP	128	192	213	224
SCALESAME1	512	341	307	293
SCALESAME2	219	236	242	245
SCALEZONE1_X	32 * N	48 * N	53 * N	56 * N
SCALEZONE1_Y	8 * N	12 * N	13 * N	14 * N
ZONE1OFFSET_X	37 * N	20 * N	14 * N	11 * N
ZONE1OFFSET_Y	10 * N	5 * N	4 * N	3 * N

图 35

3600

	参考帧距离			
	1	2	3	4或以上
SCALEOPP	128	64	43	32
SCALESAME1	512	1024	1536	2048
SCALESAME2	219	204	200	198
SCALEZONE1_X	32 * N	16 * N	11 * N	8 * N
SCALEZONE1_Y	8 * N	4 * N	3 * N	2 * N
ZONE1OFFSET_X	37 * N	52 * N	56 * N	11 * N
ZONE1OFFSET_Y	10 * N	5 * N	4 * N	3 * N

图 36

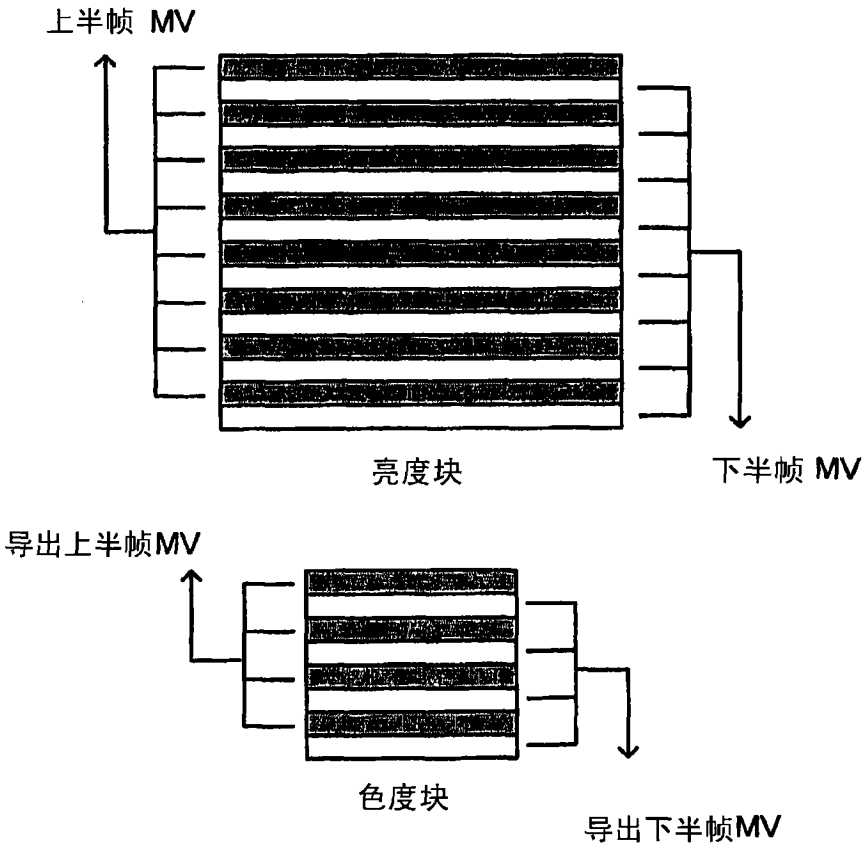

上半帧 MV
亮度块
下半帧 MV
导出上半帧MV
色度块
导出下半帧MV

图 37

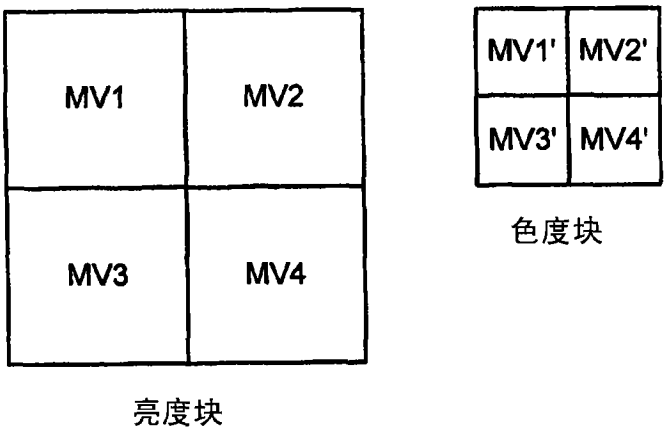

MV1
MV2
MV3
MV4
亮度块
MV1'
MV2'
MV3'
MV4'
色度块

图 38

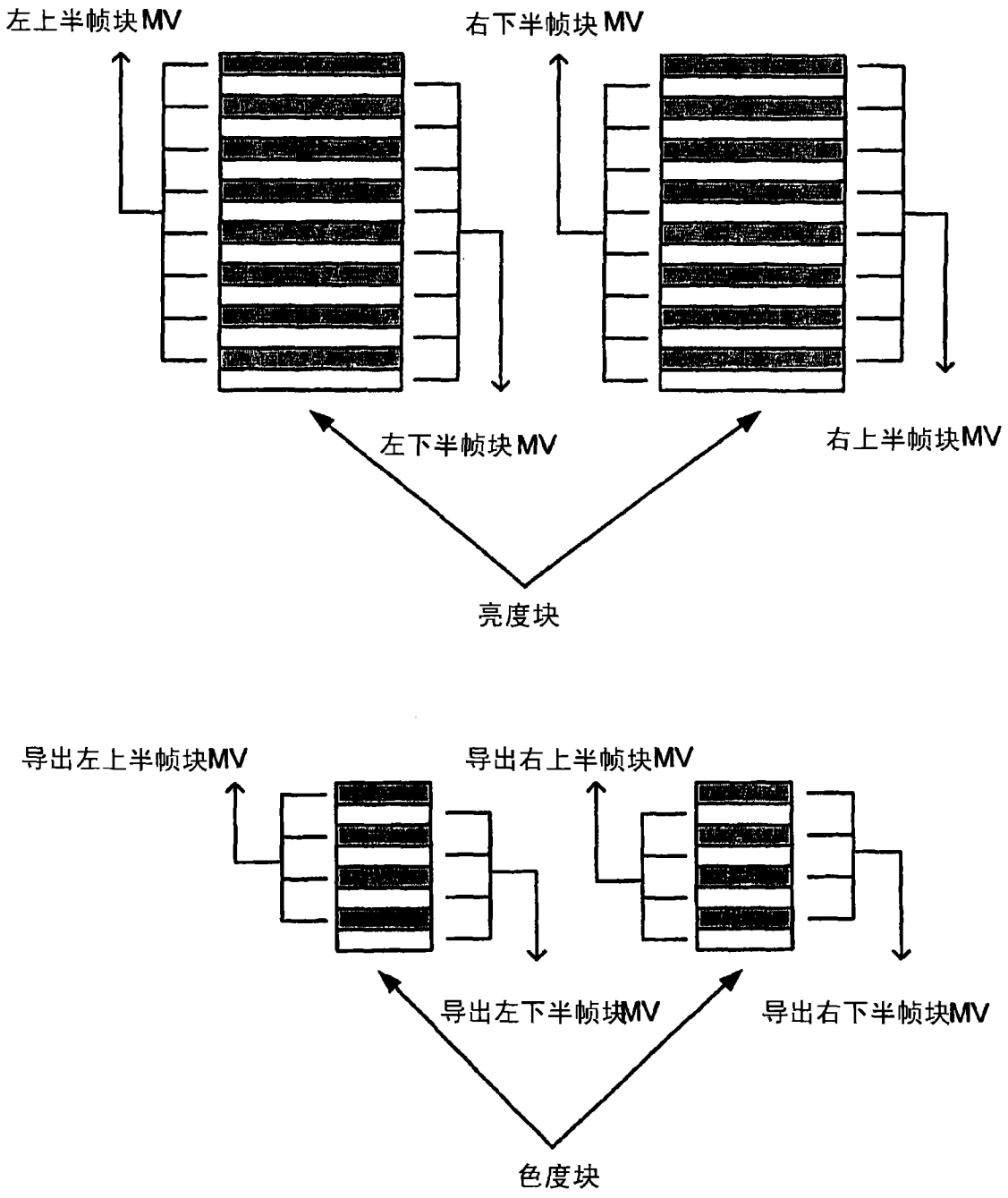

左上半帧块MV
右下半帧块MV
左下半帧块MV
右上半帧块MV
亮度块
导出左上半帧块MV
导出右上半帧块MV
导出左下半帧块MV
导出右下半帧块MV
色度块

图 39

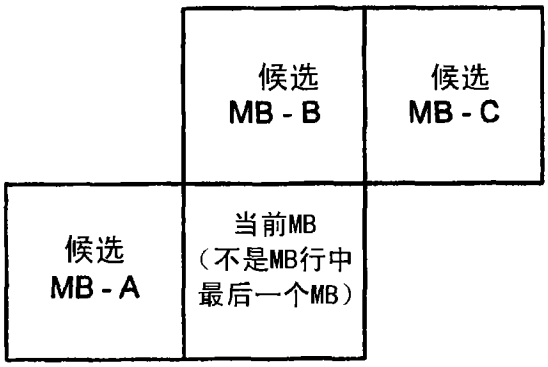


图 40A

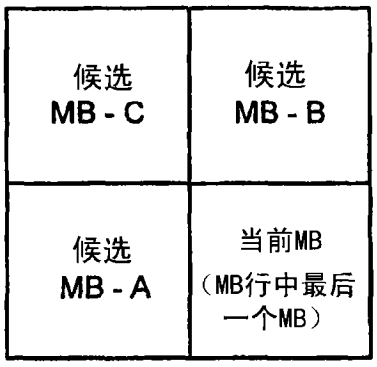


图 40B

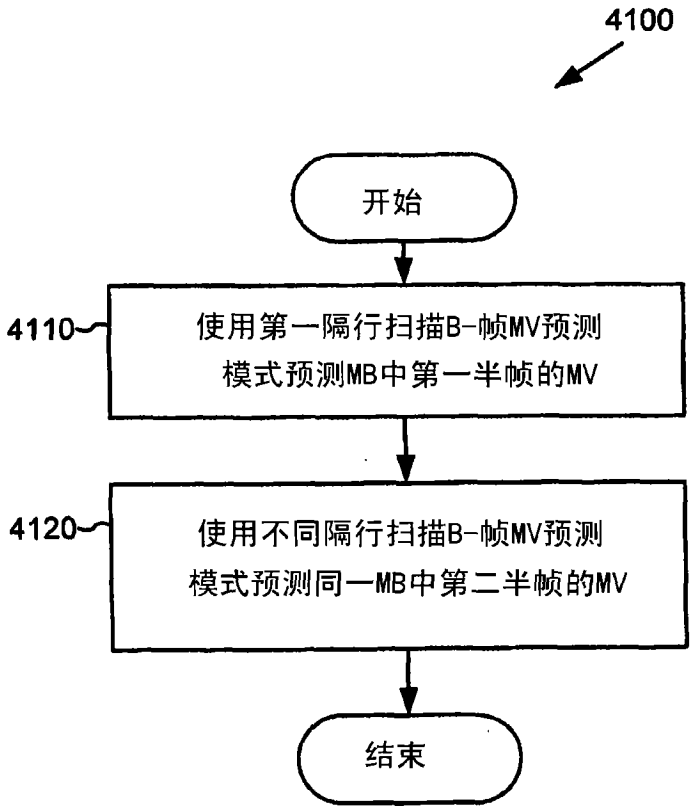


图 41

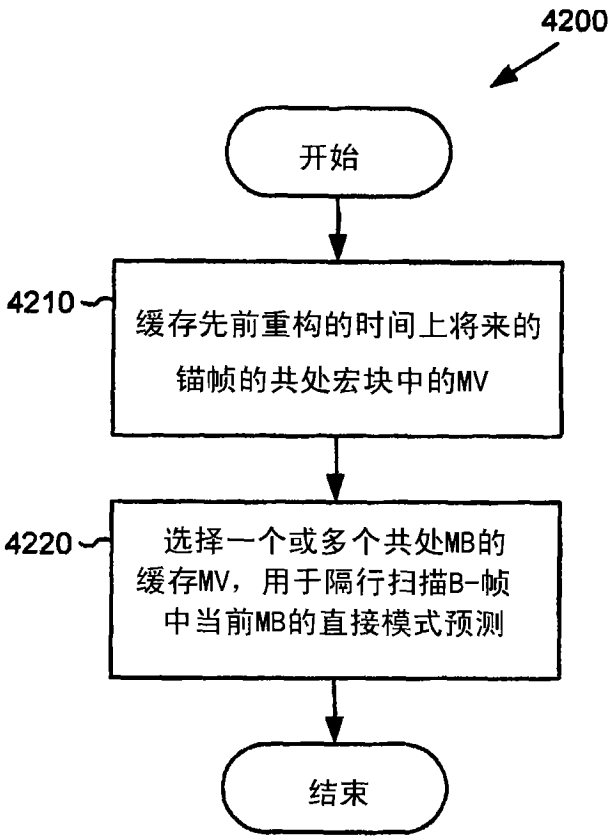

4200
开始
4210
缓存先前重构的时间上将来的锚帧的共处宏块中的MV
4220
选择一个或多个共处MB的缓存MV，用于隔行扫描B-帧中当前MB的直接模式预测
结束

图 42

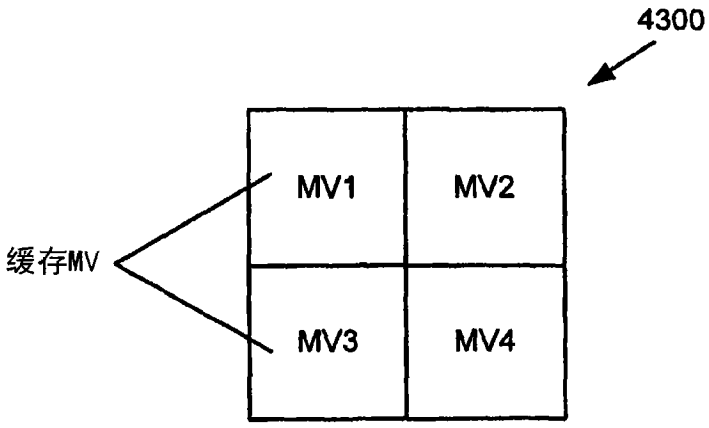

4300
MV1
MV2
MV3
MV4
缓存MV

图 43

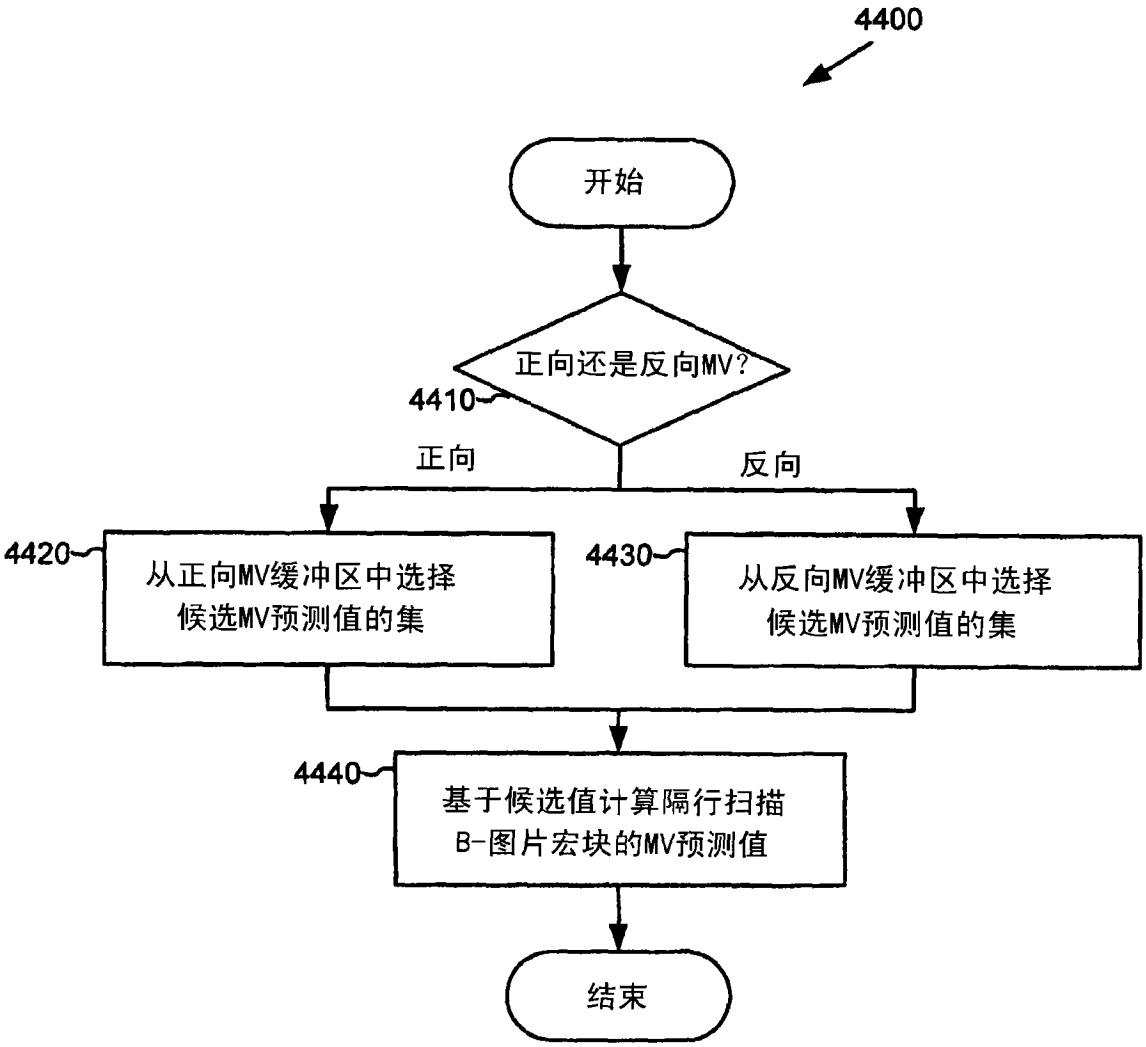

4400
开始
正向还是反向MV?
4410
正向
反向
4420
从正向MV缓冲区中选择
候选MV预测值的集
4430
从反向MV缓冲区中选择
候选MV预测值的集
4440
基于候选值计算隔行扫描
B-图片宏块的MV预测值
结束

图 44

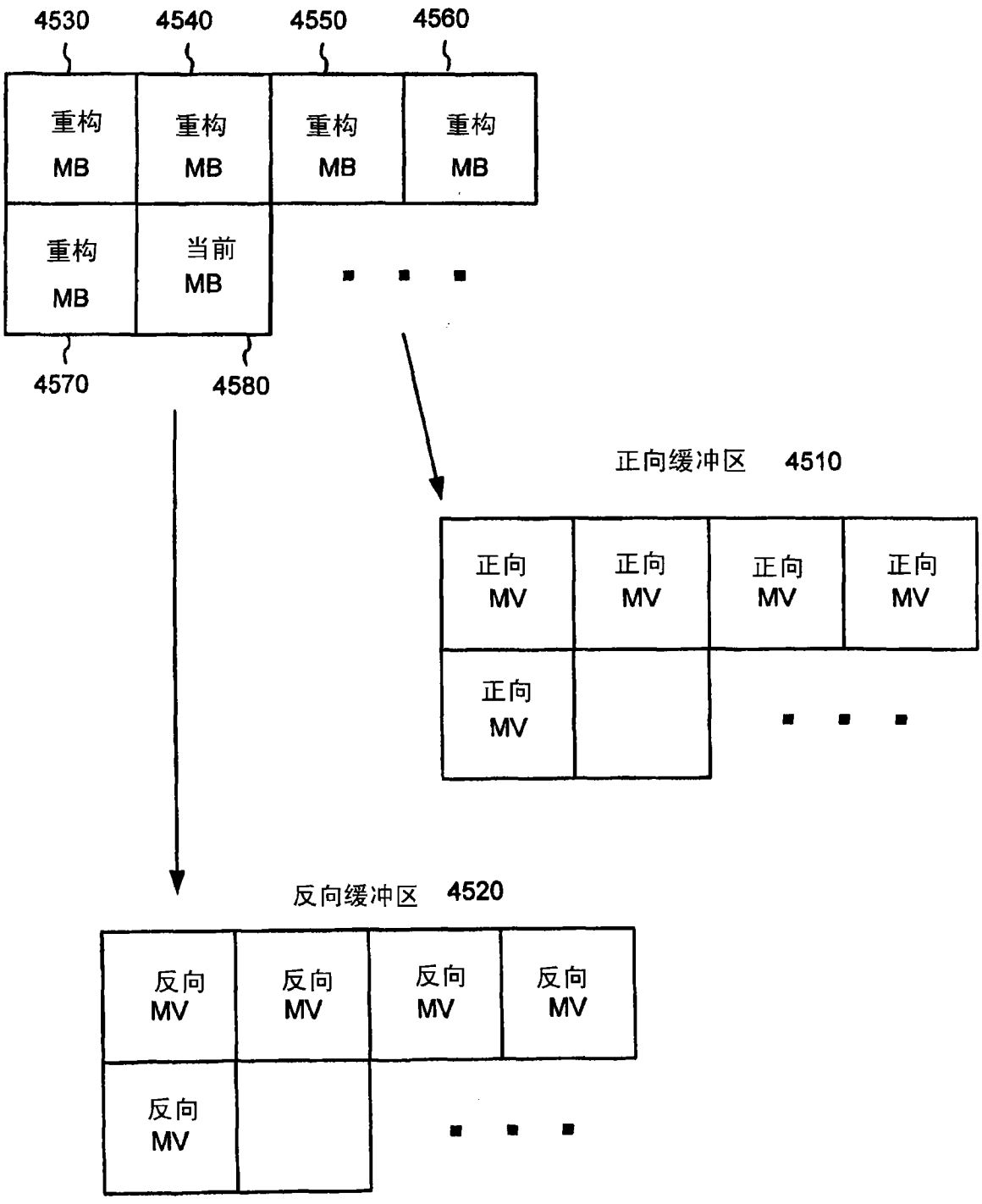

4530
4540
4550
4560
重构 MB
重构 MB
重构 MB
重构 MB
重构 MB
当前 MB
4570
4580
正向缓冲区 4510
正向 MV
正向 MV
正向 MV
正向 MV
正向 MV
反向缓冲区 4520
反向 MV
反向 MV
反向 MV
反向 MV
反向 MV

图 45

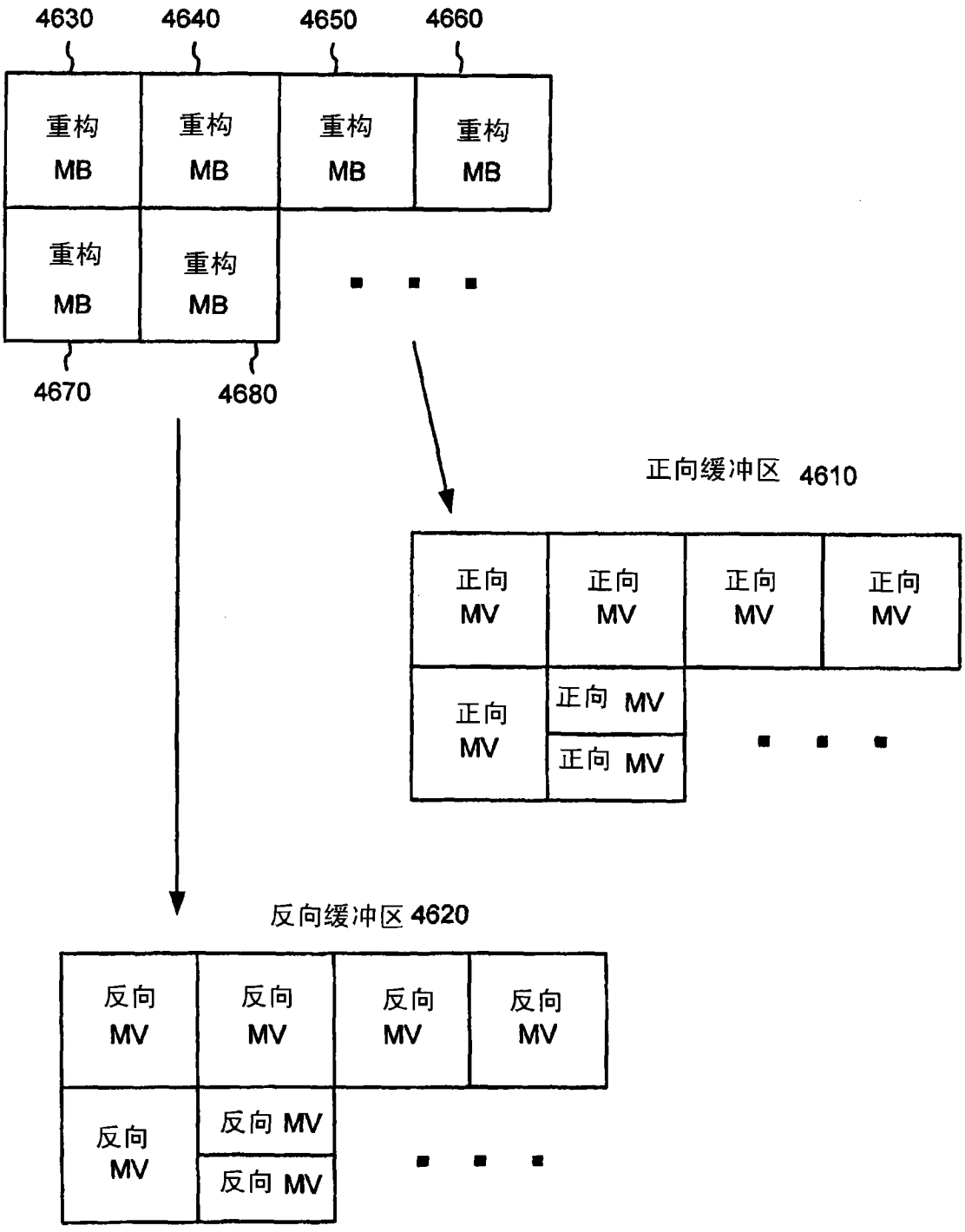

4630
4640
4650
4660
重构 MB
重构 MB
重构 MB
重构 MB
重构 MB
重构 MB
4670
4680
正向缓冲区 4610
正向 MV
正向 MV
正向 MV
正向 MV
正向 MV
正向 MV
正向 MV
反向缓冲区 4620
反向 MV
反向 MV
反向 MV
反向 MV
反向 MV
反向 MV
反向 MV

图 46

4700

```
PredictedMV PredictMVFieldBPicture (Type of pred) // 实际或"空穴填充"
{

   if (Type of prediction = Real) {
        // 在该情形中我们接收该类实际MV
        then use the normal field MV prediction described above with the
              polarity of the real MV received
   }
   else {

        // 空穴填充预测—当我们未接收到该类实际MV

        use the field MV prediction, and pick the predicted MV of the most
              dominant field
   }
   return the predicted MV
}
```

图 47

4800

```
scaleforsame_x (n) {
        int scaledvalue
        scaledvalue = (n * SCALESAME) >> 8
        return scaledvalue
}

scaleforsame_y (n) {
        int scaledvalue
        if (current field is top)
              scaledvalue = ((n * SCALESAME) >> 8) - 2
        else  //当前半帧为下半帧
              scaledvalue = ((n * SCALESAME) >> 8) + 2
        return scaledvalue
}
```

图 48A

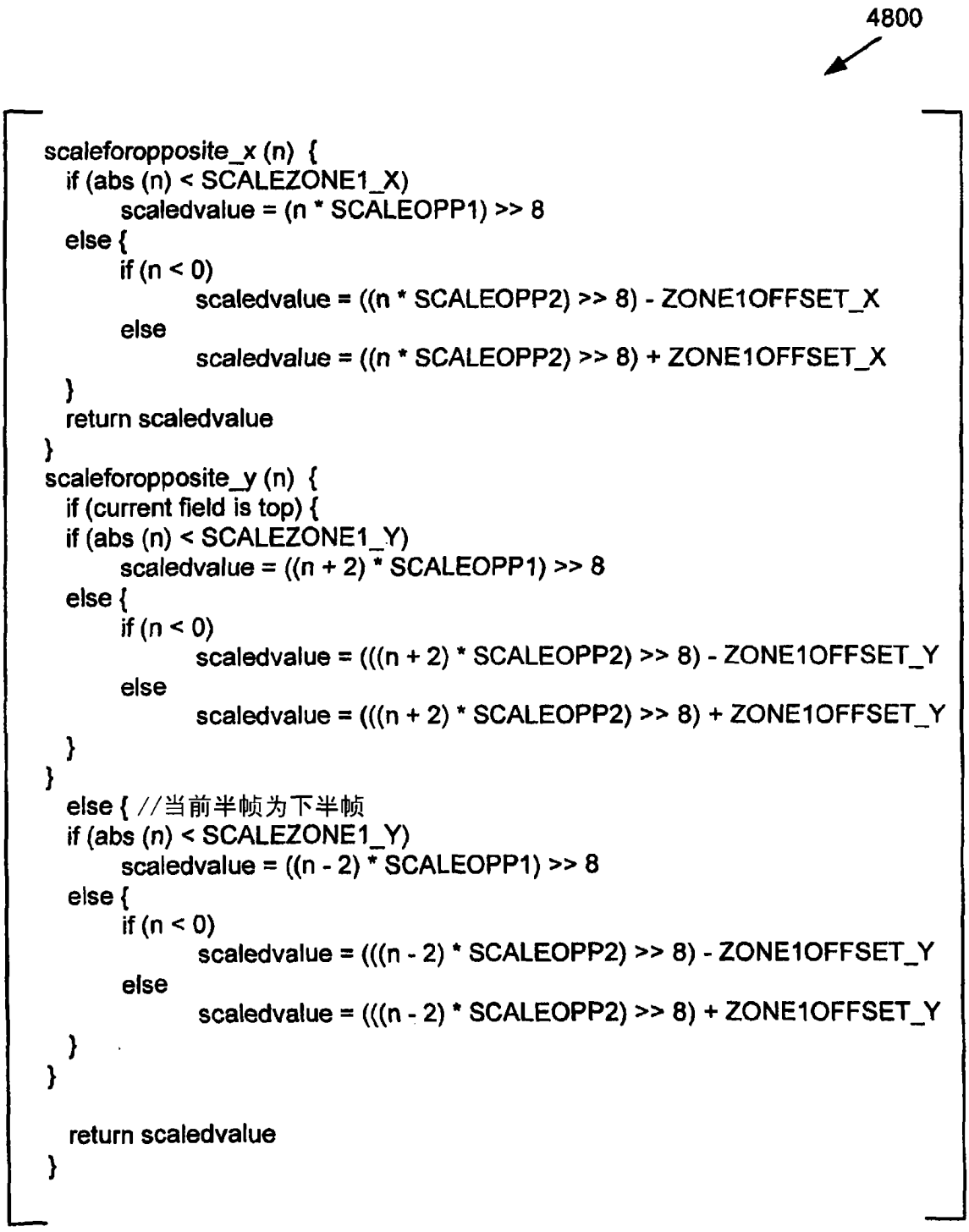

4800
scaleforopposite_x (n) {
if (abs (n) < SCALEZONE1_X)
scaledvalue = (n * SCALEOPP1) >> 8
else {
if (n < 0)
scaledvalue = ((n * SCALEOPP2) >> 8) - ZONE1OFFSET_X
else
scaledvalue = ((n * SCALEOPP2) >> 8) + ZONE1OFFSET_X
}
return scaledvalue
}
scaleforopposite_y (n) {
if (current field is top) {
if (abs (n) < SCALEZONE1_Y)
scaledvalue = ((n + 2) * SCALEOPP1) >> 8
else {
if (n < 0)
scaledvalue = (((n + 2) * SCALEOPP2) >> 8) - ZONE1OFFSET_Y
else
scaledvalue = (((n + 2) * SCALEOPP2) >> 8) + ZONE1OFFSET_Y
}
}
else { //当前半帧为下半帧
if (abs (n) < SCALEZONE1_Y)
scaledvalue = ((n - 2) * SCALEOPP1) >> 8
else {
if (n < 0)
scaledvalue = (((n - 2) * SCALEOPP2) >> 8) - ZONE1OFFSET_Y
else
scaledvalue = (((n - 2) * SCALEOPP2) >> 8) + ZONE1OFFSET_Y
}
}
return scaledvalue
}

图 48B

4900

	参考图片距离			
	1	2	3	4或以上
SCALESAME	171	205	219	228
SCALEOPP1	384	320	299	288
SCALEOPP2	230	239	244	246
SCALEZONE1_X	32 * N	48 * N	53 * N	56 * N
SCALEZONE1_Y	8 * N	12 * N	13 * N	14 * N
ZONE1OFFSET_X	37 * N	20 * N	14 * N	11 * N
ZONE1OFFSET_Y	10 * N	5 * N	4 * N	3 * N

图 49

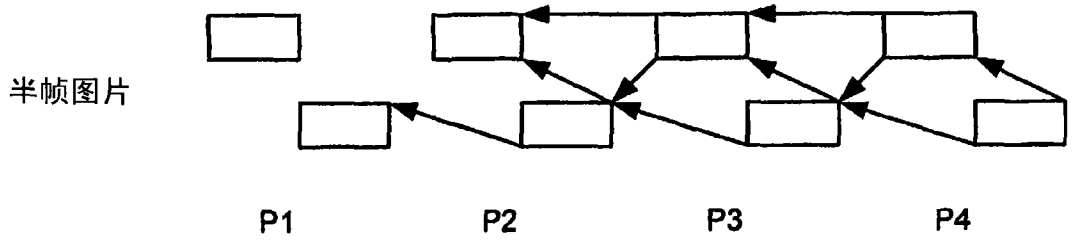


图 50A

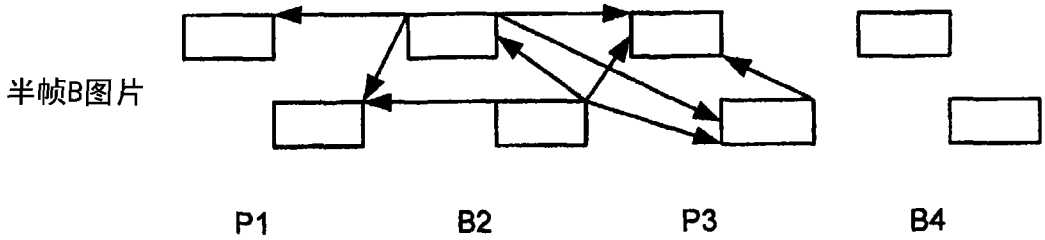


图 50B

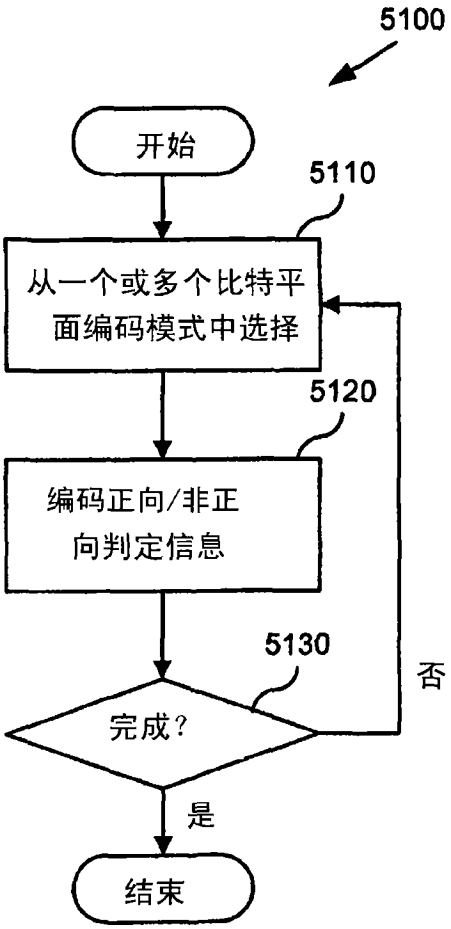

5100
开始
5110
从一个或多个比特平面编码模式中选择
5120
编码正向/非正向判定信息
5130
完成?
否
是
结束

图 51

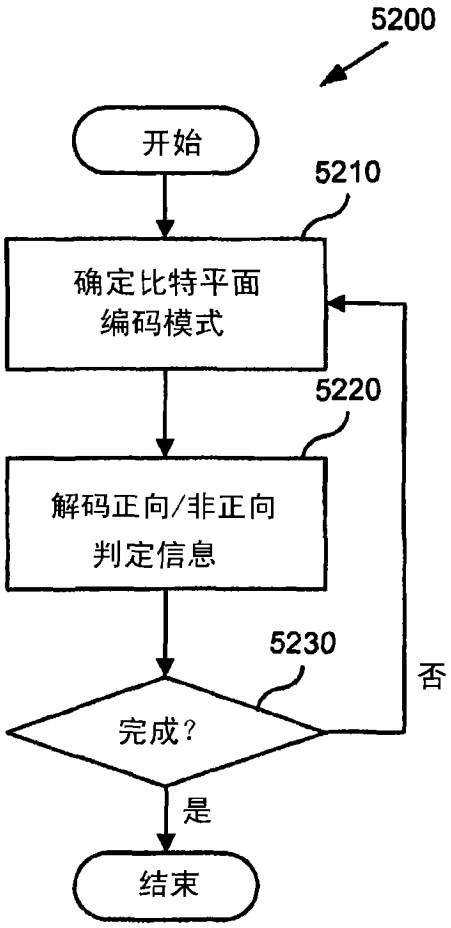

5200
开始
5210
确定比特平面编码模式
5220
解码正向/非正向判定信息
5230
完成?
否
是
结束

图 52

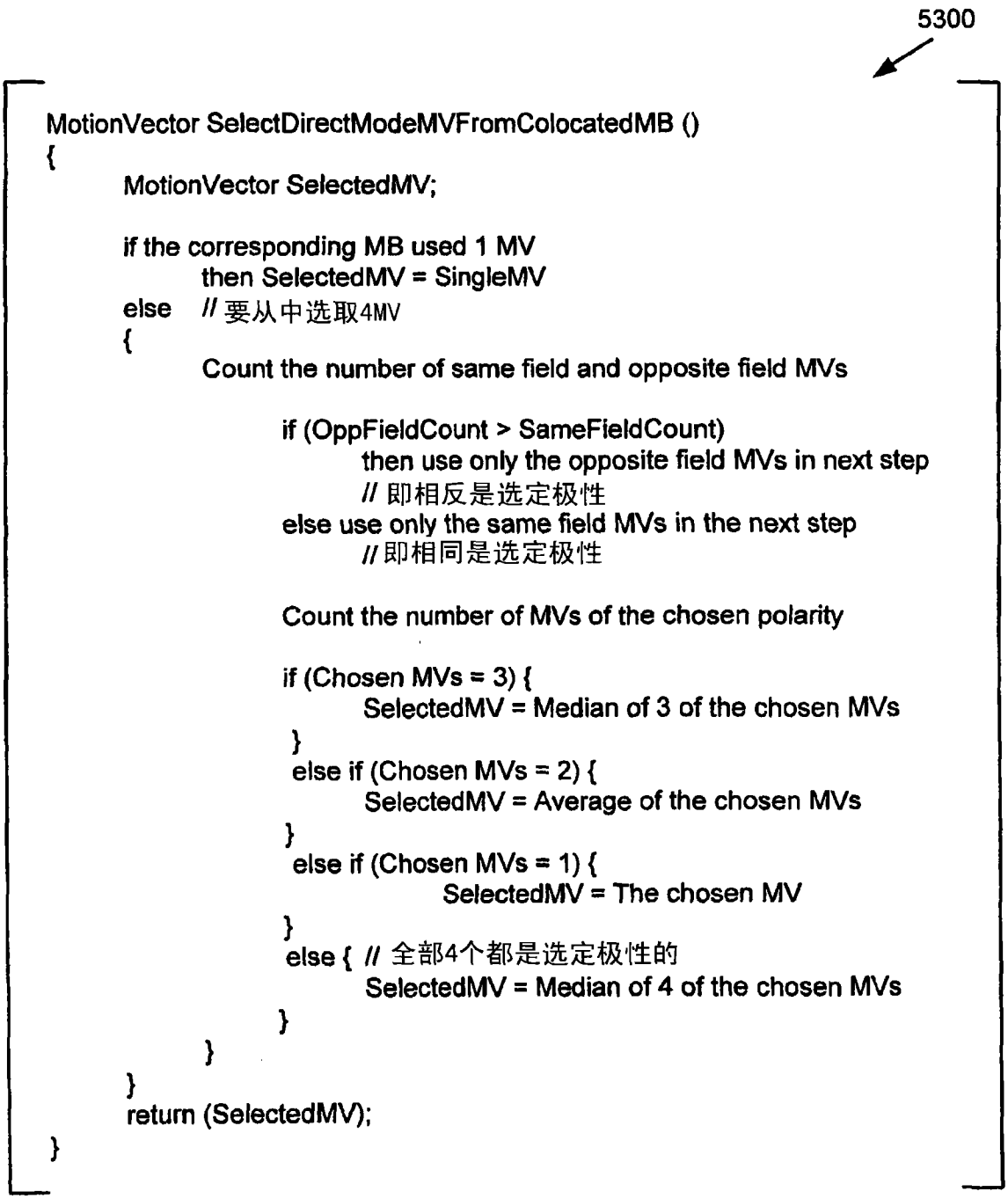

5300
MotionVector SelectDirectModeMVFromColocatedMB ()
{
MotionVector SelectedMV;
if the corresponding MB used 1 MV
then SelectedMV = SingleMV
else // 要从中选取4MV
{
Count the number of same field and opposite field MVs
if (OppFieldCount > SameFieldCount)
then use only the opposite field MVs in next step
// 即相反是选定极性
else use only the same field MVs in the next step
//即相同是选定极性
Count the number of MVs of the chosen polarity
if (Chosen MVs = 3) {
SelectedMV = Median of 3 of the chosen MVs
}
else if (Chosen MVs = 2) {
SelectedMV = Average of the chosen MVs
}
else if (Chosen MVs = 1) {
SelectedMV = The chosen MV
}
else { // 全部4个都是选定极性的
SelectedMV = Median of 4 of the chosen MVs
}
}
}
return (SelectedMV);
}

图 53

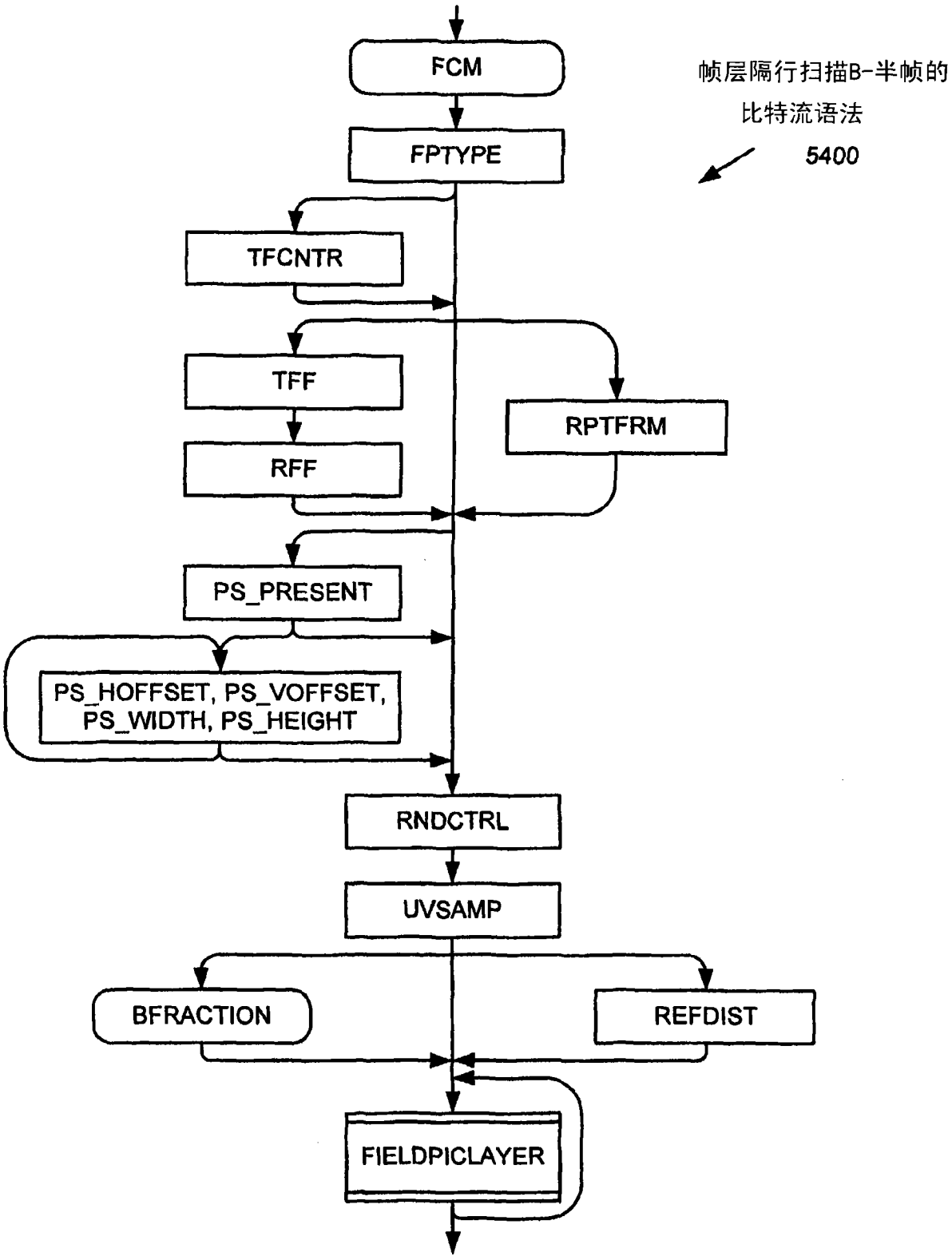

帧层隔行扫描B-半帧的
比特流语法
5400
FCM
FPTYPE
TFCNTR
TFF
RFF
RPTFRM
PS_PRESENT
PS_HOFFSET, PS_VOFFSET,
PS_WIDTH, PS_HEIGHT
RNDCTRL
UVSAMP
BFRACTION
REFDIST
FIELDPICLAYER

图 54

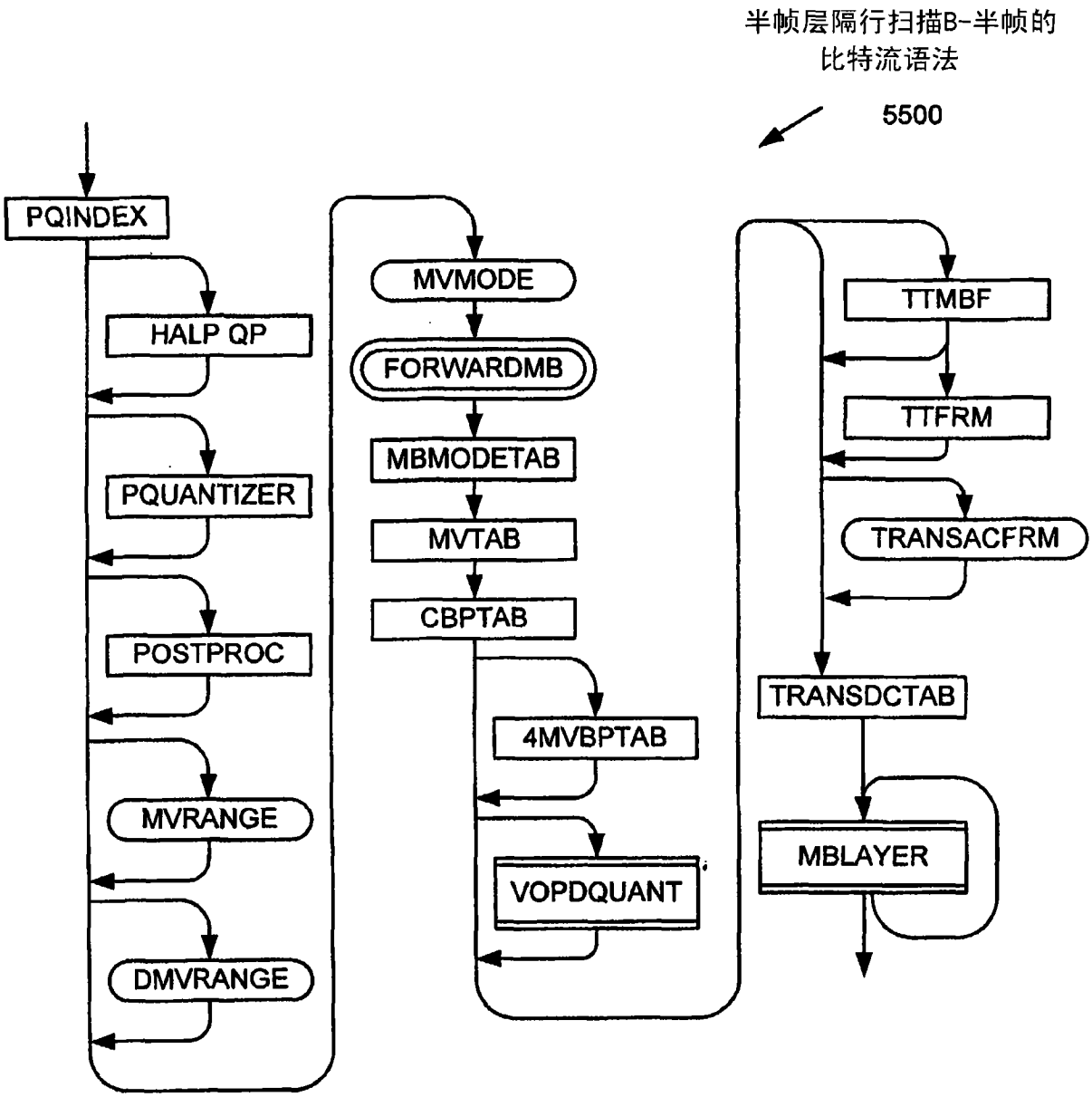

半帧层隔行扫描B-半帧的
比特流语法
5500
PQINDEX
HALP QP
PQUANTIZER
POSTPROC
MVRANGE
DMVRANGE
MVMODE
FORWARDMB
MBMODETAB
MVTAB
CBPTAB
4MVBPTAB
VOPDQUANT
TTMBF
TTFRM
TRANSACFRM
TRANSDCTAB
MBLAYER

图 55

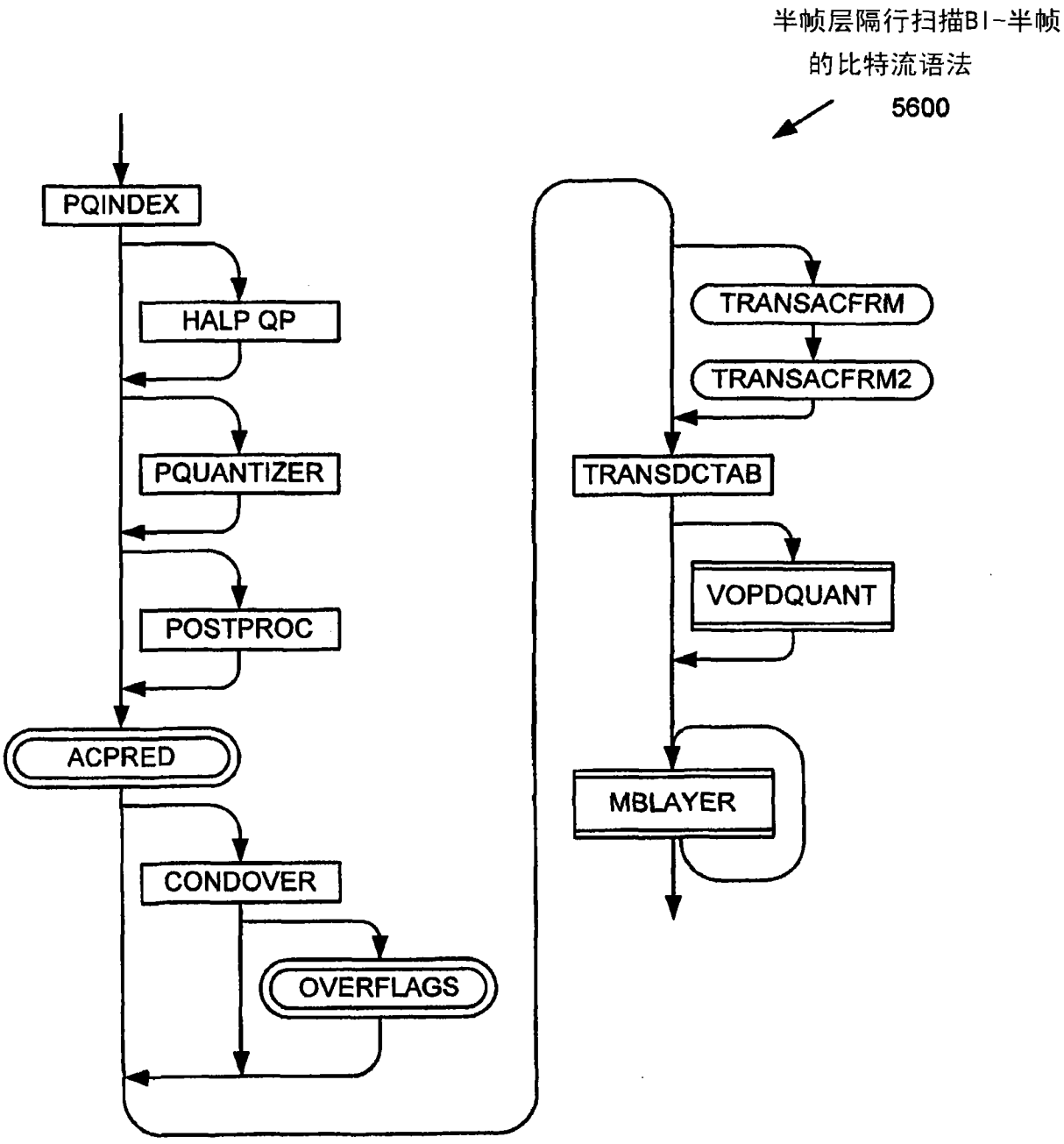

半帧层隔行扫描BI-半帧
的比特流语法
5600
PQINDEX
HALP QP
PQUANTIZER
POSTPROC
ACPRED
CONDOVER
OVERFLAGS
TRANSACFRM
TRANSACFRM2
TRANSDCTAB
VOPDQUANT
MBLAYER

图 56

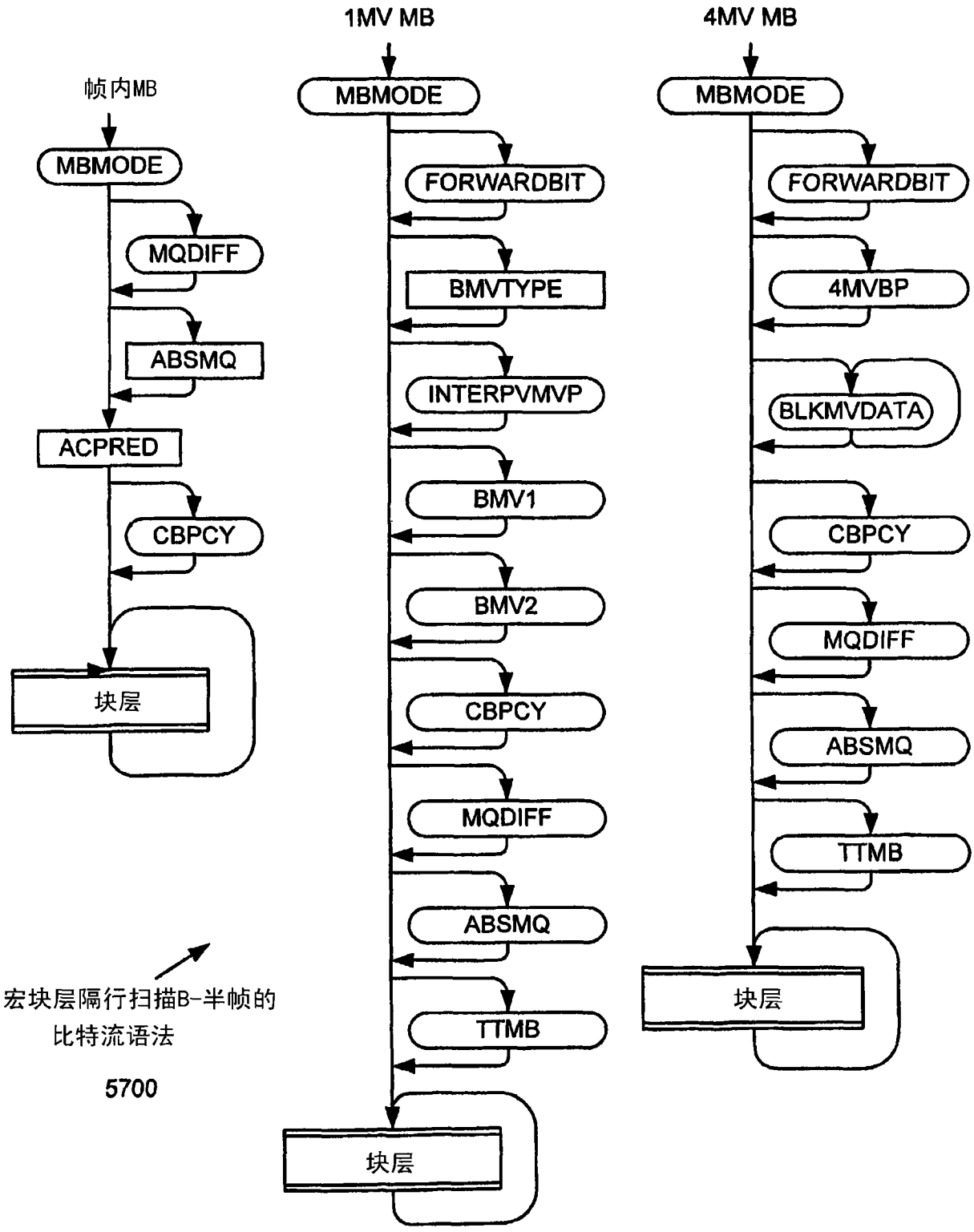

帧内MB
MBMODE
MQDIFF
ABSMQ
ACPRED
CBPCY
块层
1MV MB
MBMODE
FORWARDBIT
BMVTYPE
INTERPVMVP
BMV1
BMV2
CBPCY
MQDIFF
ABSMQ
TTMB
块层
4MV MB
MBMODE
FORWARDBIT
4MVBP
BLKMVDATA
CBPCY
MQDIFF
ABSMQ
TTMB
块层
宏块层隔行扫描B-半帧的
比特流语法
5700

图 57

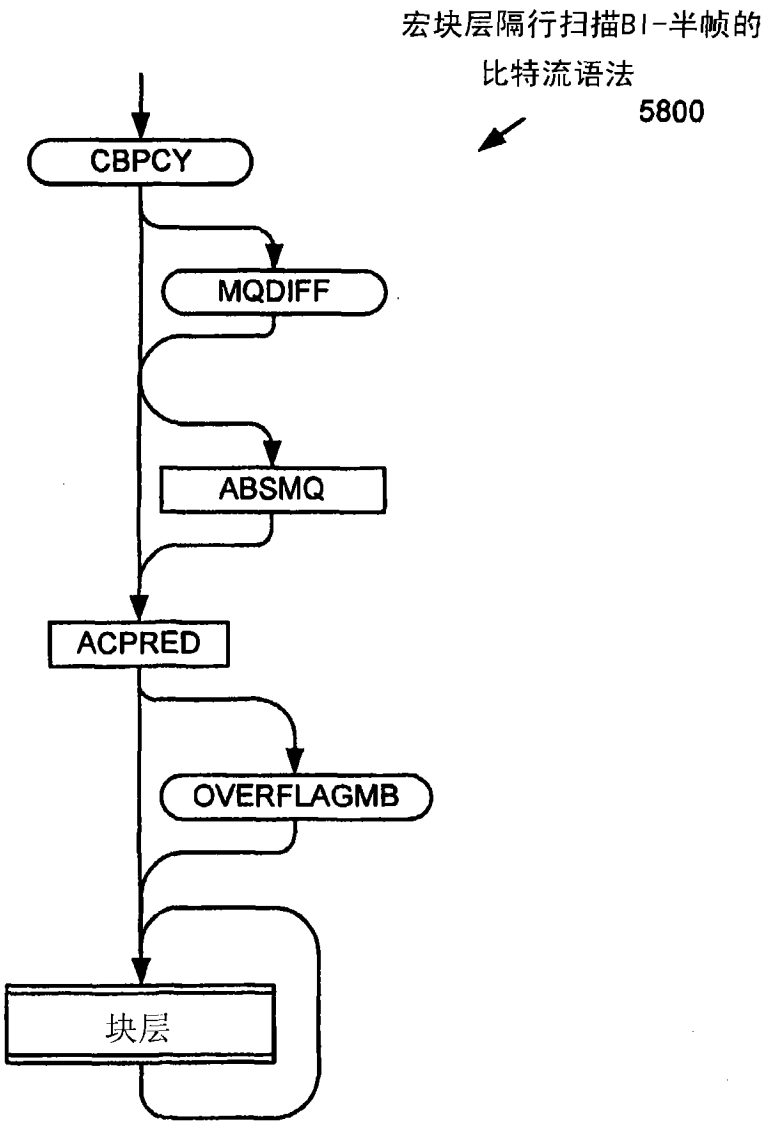

宏块层隔行扫描BI-半帧的
比特流语法
5800
CBPCY
MQDIFF
ABSMQ
ACPRED
OVERFLAGMB
块层

图 58

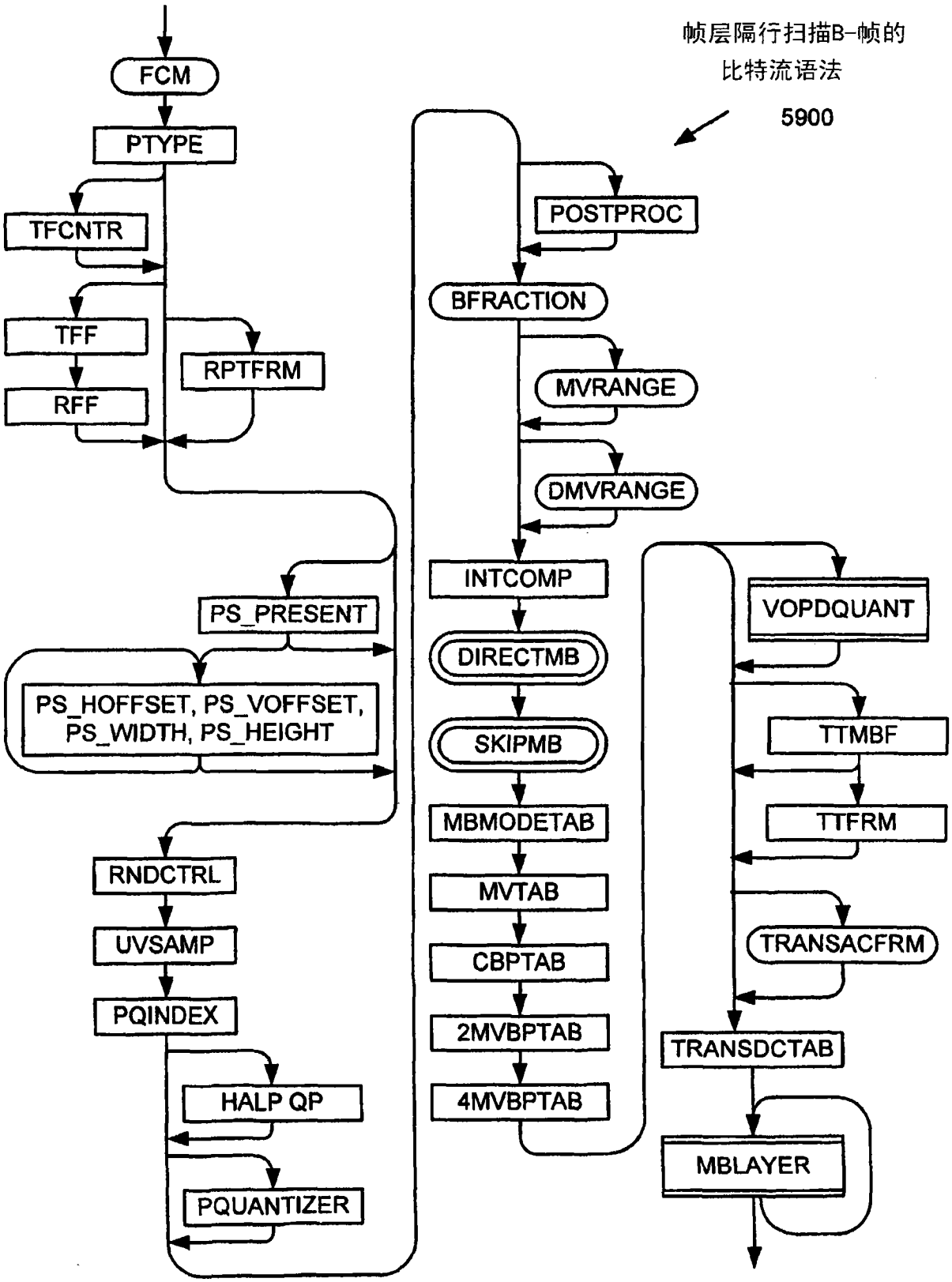

帧层隔行扫描B-帧的
比特流语法
5900
FCM
PTYPE
TFCNTR
TFF
RFF
RPTFRM
PS_PRESENT
PS_HOFFSET, PS_VOFFSET,
PS_WIDTH, PS_HEIGHT
RNDCTRL
UVSAMP
PQINDEX
HALP QP
PQUANTIZER
POSTPROC
BFRACTION
MVRANGE
DMVRANGE
INTCOMP
DIRECTMB
SKIPMB
MBMODETAB
MVTAB
CBPTAB
2MVBPTAB
4MVBPTAB
VOPDQUANT
TTMBF
TTFRM
TRANSACFRM
TRANSDCTAB
MBLAYER

图 59

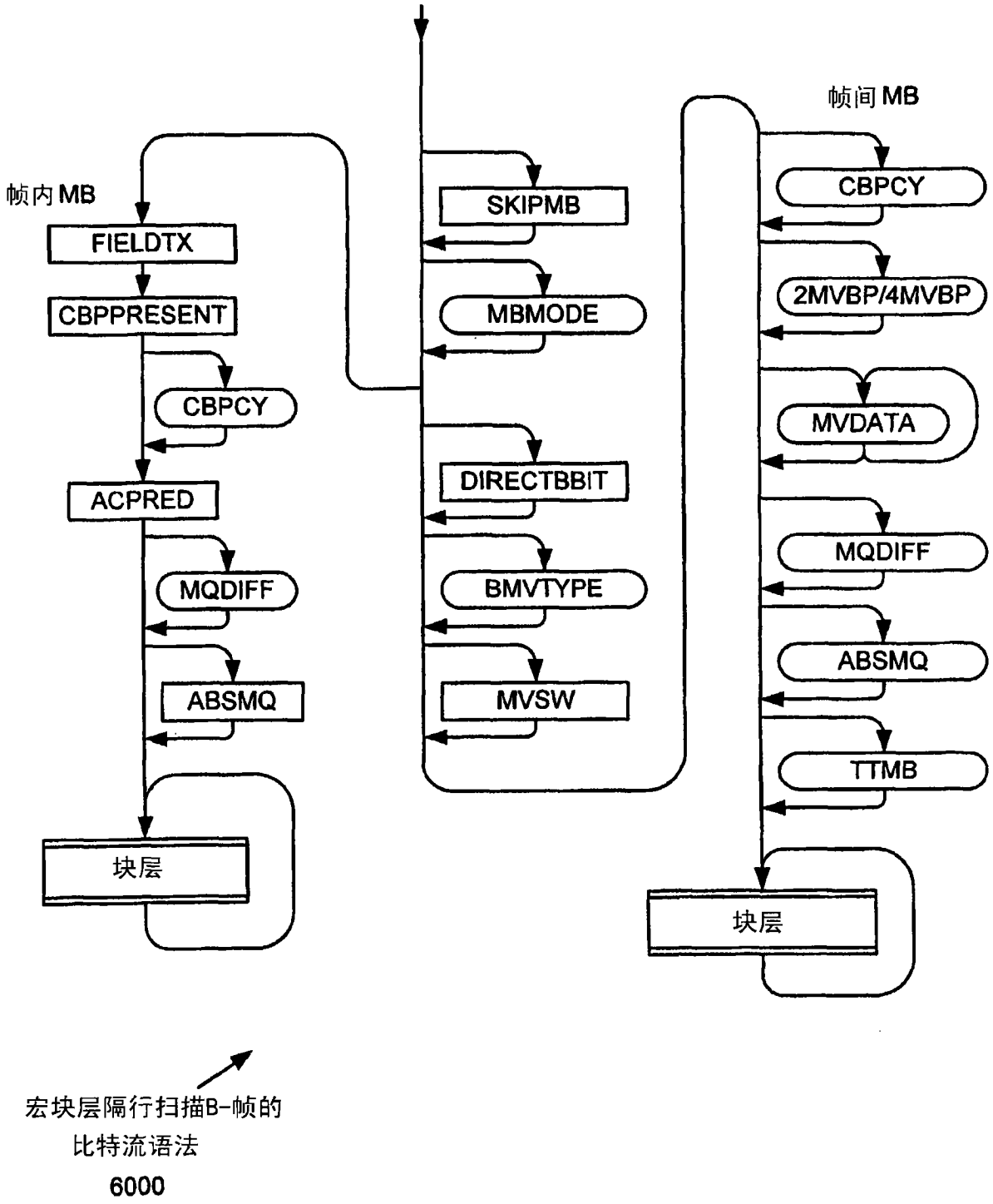

帧内 MB
FIELDTX
CBPPRESENT
CBPCY
ACPRED
MQDIFF
ABSMQ
块层
SKIPMB
MBMODE
DIRECTBBIT
BMVTYPE
MVSW
帧间 MB
CBPCY
2MVBP/4MVBP
MVDATA
MQDIFF
ABSMQ
TTMB
块层
宏块层隔行扫描B-帧的
比特流语法
6000

图 60

6100

```
size_table[16] = {0, 0, 1, 1, 2, 2, 3, 3, 4, 4, 5, 5, 6, 6, 7, 7}
offset_table1[9] = {0, 1, 2, 4, 8, 16, 32, 64, 128}
offset_table2[9] = {0, 1, 3, 7, 15, 31, 63, 127, 255}
index = vlc_decode() //使用MVTAB表示的哈夫曼表格
if (index == 125)
{
        dmv_x = get_bits(k_x)
        dmv_y = get_bits(k_y)
        predictor_flag = dmv_y & 1
        dmv_y = dmv_y >> 1
}
else
{
    if (extend_x == 1)
      offset_table = offset_table2
    else
      offset_table = offset_table1
    index1 = (index + 1) % 9
    if (index1 != 0)
    {
      val = get_bits (index1 + extend_x)
      sign = 0 - (val & 1)
      dmv_x = sign ^ ((val >> 1) + offset_table[index1])
      dmv_x = dmv_x - sign
    }
    else
      dmv_x = 0
    if (extend_y == 1)
      offset_table = offset_table2
    else
      offset_table = offset_table1
    index1 = (index + 1) / 9
    if (index1 != 0)
    {
      val = get_bits (size_table[index1 + 2 * extend_y])
      sign = 0 - (val & 1)
      dmv_y = sign ^ ((val >> 1) + offset_table[index1 >> 1])
      dmv_y = dmv_y - sign
      predictor_flag = index1 & 1
    }
    else
    {
      dmv_y = 0
      predictor_flag = 0
    }
}
```

图 61A

6110

```
size_table[14] = {0, 0, 1, 1, 2, 2, 3, 3, 4, 4, 5, 5, 6, 6}
offset_table[9] = {0, 1, 2, 4, 8, 16, 32, 64, 128}
index = vlc_decode()  //使用由图片层中MVTAB表示的哈夫曼表

if (index == 0) {
        dmv_x = 1 - 2 * get_bits(1)
        dmv_y = 0
        predictor_flag = 0
}
if (index == 125)
{
        dmv_x = get_bits(k_x - halfpel_flag)
        dmv_y = get_bits(k_y - halfpel_flag)
        predictor_flag = dmv_y & 1
        dmv_y = dmv_y >> 1
}
else
{
index1 = (index + 1) % 9
      val = get_bits (index1)
      sign = 0 - (val & 1)
      dmv_x = sign ^ ((val >> 1) + offset_table[index1])
      dmv_x = dmv_x - sign

index1 = (index + 1) / 9
      val = get_bits (size_table[index1])
      sign = 0 - (val & 1)
      dmv_y = sign ^ ((val >> 1) + offset_table[index1])
      dmv_y = dmv_y - sign
      predictor_flag = index1 & 1
}
```

图 61B

6200

```
samecount = 0;
oppositecount = 0;
if (predictorA is not out of bounds) {
  if (predictorC is not out of bounds) {
    if (predictorA is intra) {
      samefieldpred_x = oppositefieldpred_x = samefieldpredA_x = oppositefieldpredA_x = 0
      samefieldpred_y = oppositefieldpred_y = samefieldpredA_y = oppositefieldpredA_y = 0
    }
    if (predictorB is intra) {
      samefieldpred_x = oppositefieldpred_x = samefieldpredB_x = oppositefieldpredB_x = 0
      samefieldpred_y = oppositefieldpred_y = samefieldpredB_y = oppositefieldpredB_y = 0
    }
    if (predictorC is intra) {
      samefieldpred_x = oppositefieldpred_x = samefieldpredC_x = oppositefieldpredC_x = 0
      samefieldpred_x = oppositefieldpred_x = samefieldpredC_y = oppositefieldpredC_y = 0
    }
    if (predictorA is not intra) {
      if (predictorA is from same field) {
        samecount = samecount + 1
        samefieldpred_x = samefieldpredA_x = predictorA_x
        samefieldpred_y = samefieldpredA_y = predictorA_y
        oppositefieldpred_x = oppositefieldpredA_x = scaleforopposite_x(predictorA_x)
        oppositefieldpred_y = oppositefieldpredA_y = scaleforopposite_y(predictorA_y)
      }
      else {
        oppositecount = oppositecount + 1
        oppositefieldpred_x = oppositefieldpredA_x = predictorA_x
        oppositefieldpred_y = oppositefieldpredA_y = predictorA_y
        samefieldpred_x = samefieldpredA_x = scaleforsame_x(predictorA_x)
        samefieldpred_y = samefieldpredA_y = scaleforsame_y(predictorA_y)
      }
    }
    if (predictorB is not intra) {
      if (predictorB is from same field) {
        samecount = samecount + 1
        samefieldpred_x = samefieldpredB_x = predictorB_x
        samefieldpred_y = samefieldpredB_y = predictorB_y
        oppositefieldpred_x = oppositefieldpredB_x = scaleforopposite_x(predictorB_x)
        oppositefieldpred_y = oppositefieldpredB_y = scaleforopposite_y(predictorB_y)
      }
      else {
        oppositecount = oppositecount + 1
        oppositefieldpred_x = oppositefieldpredB_x = predictorB_x
        oppositefieldpred_y = oppositefieldpredB_y = predictorB_y
        samefieldpred_x = samefieldpredB_x = scaleforsame_x(predictorB_x)
        samefieldpred_y = samefieldpredB_y = scaleforsame_y(predictorB_y)
      }
    }
```

图 62A

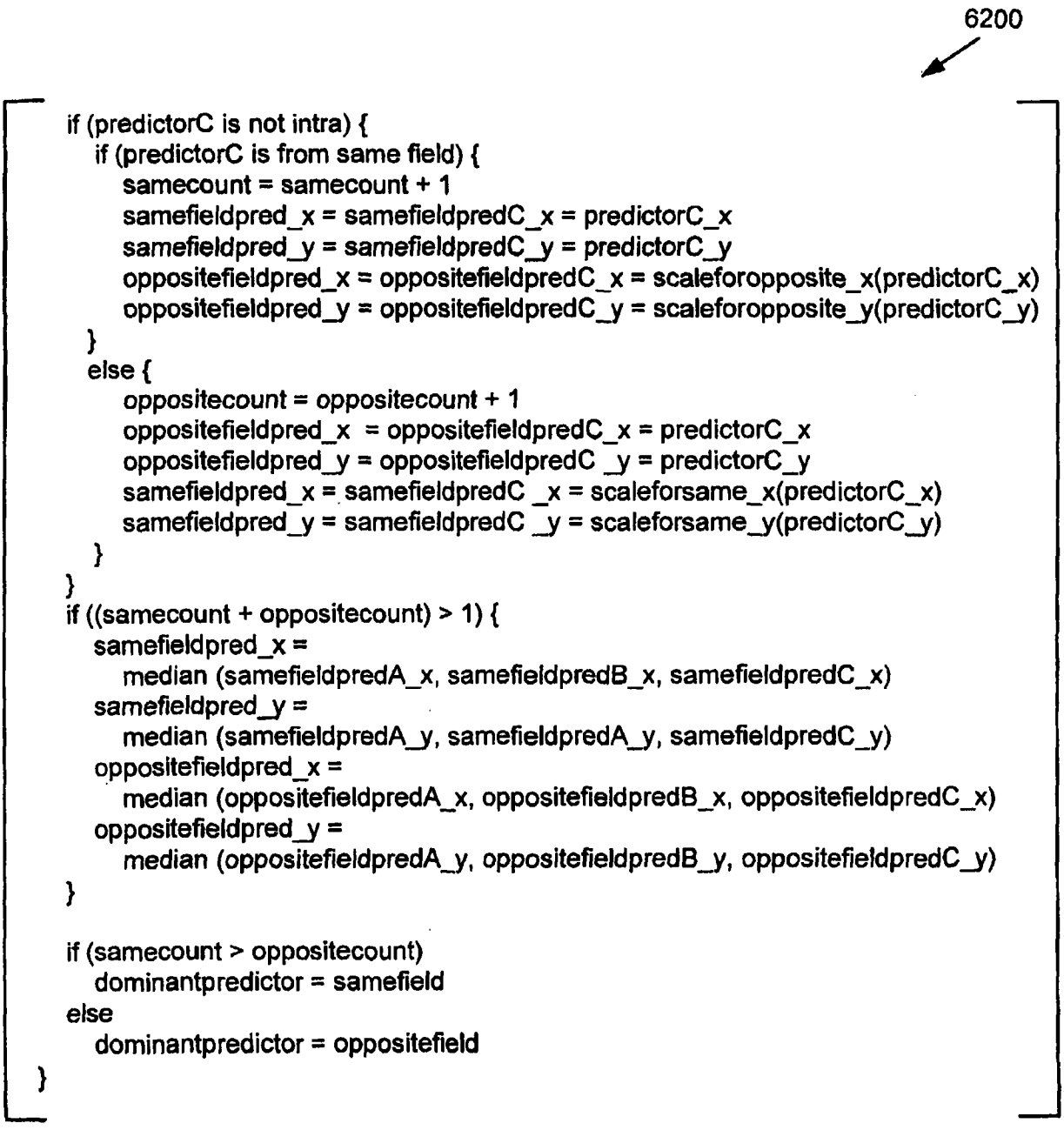

6200
if (predictorC is not intra) {
if (predictorC is from same field) {
samecount = samecount + 1
samefieldpred_x = samefieldpredC_x = predictorC_x
samefieldpred_y = samefieldpredC_y = predictorC_y
oppositefieldpred_x = oppositefieldpredC_x = scaleforopposite_x(predictorC_x)
oppositefieldpred_y = oppositefieldpredC_y = scaleforopposite_y(predictorC_y)
}
else {
oppositecount = oppositecount + 1
oppositefieldpred_x = oppositefieldpredC_x = predictorC_x
oppositefieldpred_y = oppositefieldpredC _y = predictorC_y
samefieldpred_x = samefieldpredC _x = scaleforsame_x(predictorC_x)
samefieldpred_y = samefieldpredC _y = scaleforsame_y(predictorC_y)
}
}
if ((samecount + oppositecount) > 1) {
samefieldpred_x =
median (samefieldpredA_x, samefieldpredB_x, samefieldpredC_x)
samefieldpred_y =
median (samefieldpredA_y, samefieldpredA_y, samefieldpredC_y)
oppositefieldpred_x =
median (oppositefieldpredA_x, oppositefieldpredB_x, oppositefieldpredC_x)
oppositefieldpred_y =
median (oppositefieldpredA_y, oppositefieldpredB_y, oppositefieldpredC_y)
}
if (samecount > oppositecount)
dominantpredictor = samefield
else
dominantpredictor = oppositefield
}

图 62B

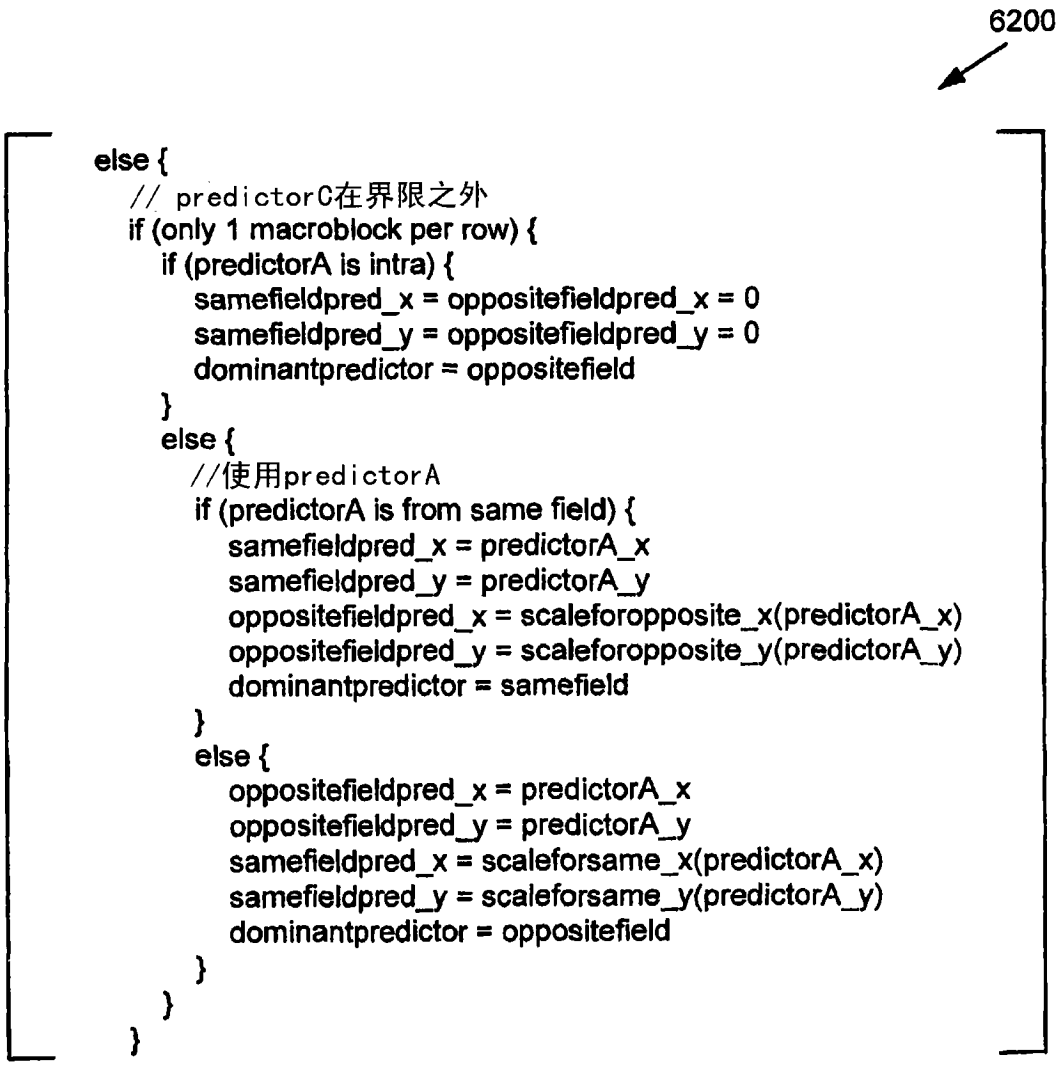

6200
else {
// predictorC在界限之外
if (only 1 macroblock per row) {
if (predictorA is intra) {
samefieldpred_x = oppositefieldpred_x = 0
samefieldpred_y = oppositefieldpred_y = 0
dominantpredictor = oppositefield
}
else {
//使用predictorA
if (predictorA is from same field) {
samefieldpred_x = predictorA_x
samefieldpred_y = predictorA_y
oppositefieldpred_x = scaleforopposite_x(predictorA_x)
oppositefieldpred_y = scaleforopposite_y(predictorA_y)
dominantpredictor = samefield
}
else {
oppositefieldpred_x = predictorA_x
oppositefieldpred_y = predictorA_y
samefieldpred_x = scaleforsame_x(predictorA_x)
samefieldpred_y = scaleforsame_y(predictorA_y)
dominantpredictor = oppositefield
}
}
}

图 62C

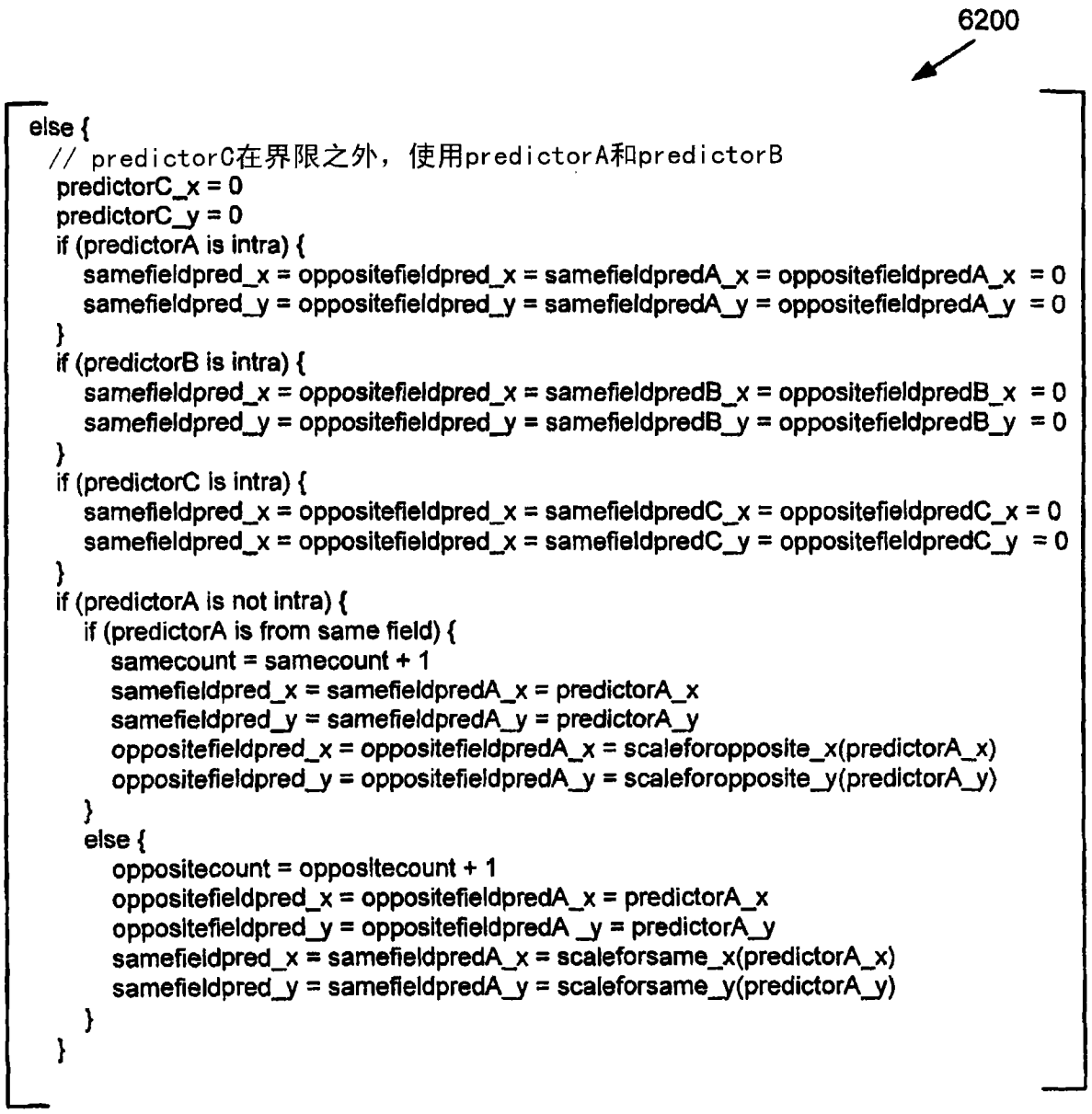

6200
else {
// predictorC在界限之外，使用predictorA和predictorB
predictorC_x = 0
predictorC_y = 0
if (predictorA is intra) {
samefieldpred_x = oppositefieldpred_x = samefieldpredA_x = oppositefieldpredA_x = 0
samefieldpred_y = oppositefieldpred_y = samefieldpredA_y = oppositefieldpredA_y = 0
}
if (predictorB is intra) {
samefieldpred_x = oppositefieldpred_x = samefieldpredB_x = oppositefieldpredB_x = 0
samefieldpred_y = oppositefieldpred_y = samefieldpredB_y = oppositefieldpredB_y = 0
}
if (predictorC is intra) {
samefieldpred_x = oppositefieldpred_x = samefieldpredC_x = oppositefieldpredC_x = 0
samefieldpred_x = oppositefieldpred_x = samefieldpredC_y = oppositefieldpredC_y = 0
}
if (predictorA is not intra) {
if (predictorA is from same field) {
samecount = samecount + 1
samefieldpred_x = samefieldpredA_x = predictorA_x
samefieldpred_y = samefieldpredA_y = predictorA_y
oppositefieldpred_x = oppositefieldpredA_x = scaleforopposite_x(predictorA_x)
oppositefieldpred_y = oppositefieldpredA_y = scaleforopposite_y(predictorA_y)
}
else {
oppositecount = oppositecount + 1
oppositefieldpred_x = oppositefieldpredA_x = predictorA_x
oppositefieldpred_y = oppositefieldpredA _y = predictorA_y
samefieldpred_x = samefieldpredA_x = scaleforsame_x(predictorA_x)
samefieldpred_y = samefieldpredA_y = scaleforsame_y(predictorA_y)
}
}

图 62D

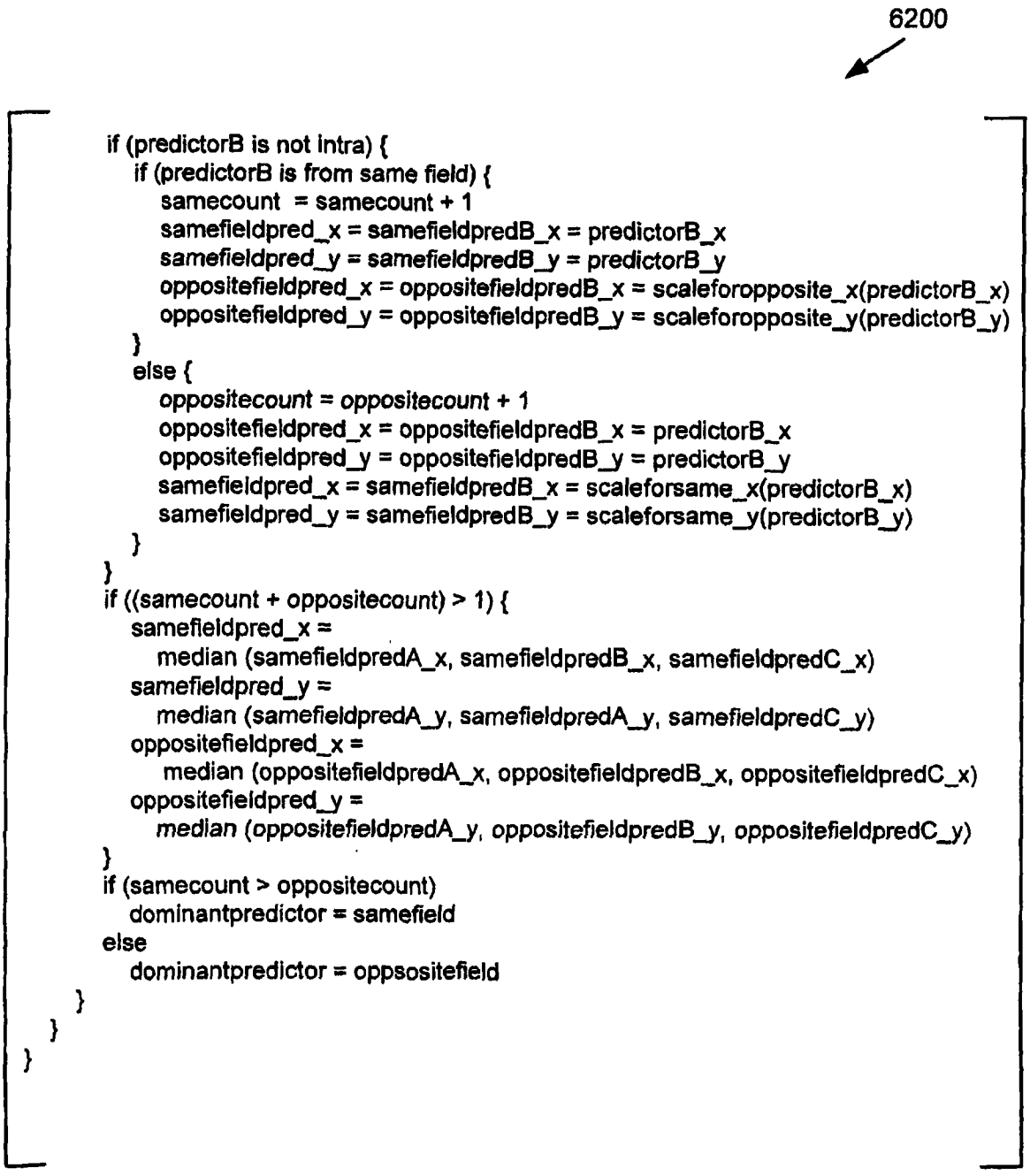

6200
if (predictorB is not intra) {
if (predictorB is from same field) {
samecount = samecount + 1
samefieldpred_x = samefieldpredB_x = predictorB_x
samefieldpred_y = samefieldpredB_y = predictorB_y
oppositefieldpred_x = oppositefieldpredB_x = scaleforopposite_x(predictorB_x)
oppositefieldpred_y = oppositefieldpredB_y = scaleforopposite_y(predictorB_y)
}
else {
oppositecount = oppositecount + 1
oppositefieldpred_x = oppositefieldpredB_x = predictorB_x
oppositefieldpred_y = oppositefieldpredB_y = predictorB_y
samefieldpred_x = samefieldpredB_x = scaleforsame_x(predictorB_x)
samefieldpred_y = samefieldpredB_y = scaleforsame_y(predictorB_y)
}
}
if ((samecount + oppositecount) > 1) {
samefieldpred_x =
median (samefieldpredA_x, samefieldpredB_x, samefieldpredC_x)
samefieldpred_y =
median (samefieldpredA_y, samefieldpredA_y, samefieldpredC_y)
oppositefieldpred_x =
median (oppositefieldpredA_x, oppositefieldpredB_x, oppositefieldpredC_x)
oppositefieldpred_y =
median (oppositefieldpredA_y, oppositefieldpredB_y, oppositefieldpredC_y)
}
if (samecount > oppositecount)
dominantpredictor = samefield
else
dominantpredictor = oppsositefield
}
}
}

图 62E


6200
else {
// predictorA在界限之外
if (predictorC is out of bounds) {
samefieldpred_x = oppositefieldpred_x = 0
samefieldpred_y = oppositefieldpred_y = 0
dominantpredictor = oppositefield
}
else {
//使用predictorC
if (predictorC is from same field) {
samefieldpred_x = predictorC_x
samefieldpred_y = predictorC_y
oppositefieldpred_x = scaleforopposite_x(predictorC_x)
oppositefieldpred_y = scaleforopposite_y(predictorC_y)
dominantpredictor = samefield
}
else {
oppositefieldpred_x = predictorC_x
oppositefieldpred_y = predictorC_y
samefieldpred_x = scaleforsame_x(predictorC_x)
samefieldpred_y = scaleforsame_y(predictorC_y)
dominantpredictor = oppositefield
}
}
}

图 62F

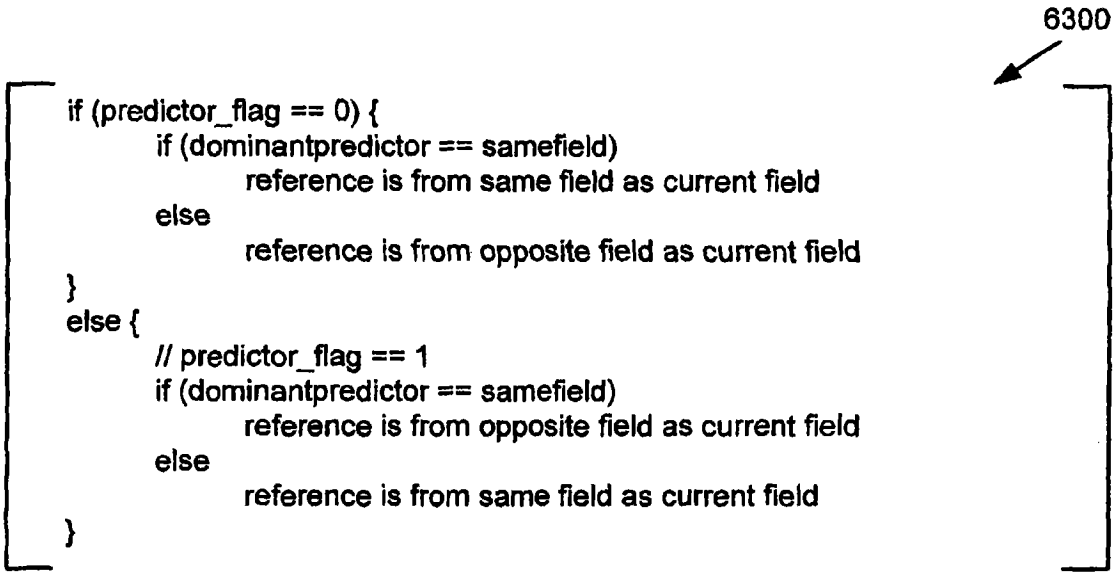

6300
if (predictor_flag == 0) {
if (dominantpredictor == samefield)
reference is from same field as current field
else
reference is from opposite field as current field
}
else {
// predictor_flag == 1
if (dominantpredictor == samefield)
reference is from opposite field as current field
else
reference is from same field as current field
}

图 63

6400

```
if (A exists and A is not intra coded) {
   if (A is 1 MV) {
      Add MV of A to the set of candidate MVs.
   } else if (A is 4 Frame MV) {
      Add the top right block MV of A to the set of candidate motion vectors.
   } else if (A is 2 Field MV) {
      Average the two field MVs of A and add the resulting MV to the set of candidate MVs.
   } else if (A is 4 Field MV) {
      Average the top right block field MV and bottom right block field MV of A and add the
                  resulting MV to the set of candidate MVs.
   }
}

if (B exists and B is not intra coded) {
   if (B is 1 MV) {
      Add MV of B to the set of candidate MVs.
   } else if (B is 4 Frame MV) {
      Add the bottom left block MV of B to the set of candidate MVs.
   } else if (B is 2 Field MV) {
      Average the two field MVs of B and add the resulting MV to the set of candidate MVs.
   } else if (B is 4 Field MV) {
      Average the top left block field MV and bottom left block field MV of B and add the resulting
            MV to the set of candidate MVs.
   }
}

if (C exists and C is not intra coded) {
   if (C is 1 MV) {
      Add MV of C to the set of candidate MVs.
   } else if (C is 4 Frame MV) {
      if (C is top right MB) {
         Add the bottom left block MV of C to the set of candidate MVs.
      } else { //C是左上MB
         Add the bottom right block MV of C to the set of candidate MVs.
      }
   } else if (C is 2 Field MV) {
      Average the two field MVs of C and add the resulting MV to the set of candidate MVs.
   } else if (C is 4 Field MV) {
      if (C is top right MB) {
         Average the top left block field MV and bottom left block field MV of C and add the
               resulting MV to the set of candidate MVs.
      } else {  //C是左上MB
         Average the top right block field MV and bottom right block field MV of C and add the
               resulting MV to the set of candidate MVs.
      }
   }
}
```

图 64

6500

```
//左上块MV
if (A exists and A is not intra coded) {
   if (A is 1 MV) {
      Add MV of A to the set of candidate MVs.
   } else if (A is 4 Frame MV) {
      Add the top right block MV of A to the set of candidate MVs.
   } else if (A is 2 Field MV) {
      Average the two field MVs of A and add the resulting MV to the set of candidate MVs.
   } else if (A is 4 Field MV) {
      Average the top right block field MV and bottom right block field MV of A and add the
            resulting MV to the set of candidate MVs.
   }
}
if (B exists and B is not intra coded) {
   if (B is 1 MV) {
      Add MV of B to the set of candidate MVs.
   } else if (B is 4 Frame MV) {
      Add the bottom left block MV of B to the set of candidate MVs.
   } else if (B is 2 Field MV) {
      Average the two field MVs of B and add the resulting MV to the set of candidate MVs.
   } else if (B is 4 Field MV) {
      Average the top left block field MV and bottom left block field MV of B and add the resulting
            MV to the set of candidate MVs.
   }
}
if (C exists and C is not intra coded) {
   if (C is 1 MV) {
      Add MV of C to the set of candidate MVs.
   } else if (C is 4 Frame MV) {
      if (C is top right MB) {
         Add the bottom left block MV of C to the set of candidate MVs.
      } else { //C是左上MB
         Add the bottom right block MV of C to the set of candidate MVs.
      }
   } else if (C is 2 Field MV) {
      Average the two field MVs of C and add the resulting MV to the set of candidate MVs.
   } else if (C is 4 Field MV) {
      If (C is top right MB) {
         Average the top left block field MV and bottom left block field MV of C and add the
               resulting MV to the set of candidate MVs.
      } else { //C是左上MB
         Average the top right block field MV and bottom right block field MV of C and add the
               resulting MV to the set of candidate MVs.
      }
   }
}
```

图 65

6600

```
//右上块MV
Add the top left block MV of the current MB to the set of candidate MVs.

if (B exists and B is not intra coded) {
   if (B is 1 MV) {
      Add MV of B to the set of candidate MVs.
   } else if (B is 4 Frame MV) {
      Add the bottom right block MV of B to the set of candidate MVs.
   } else if (B is 2 Field MV) {
      Average the two field MVs of B and add the resulting MV to the set of candidate MVs.
   } else if (B is 4 Field MV) {
      Average the top right block field MV and bottom right
      block field MV of B and add the resulting MV to the set of candidate MVs.
   }
}

if (C exists and C is not intra coded) {
   if (C is 1 MV) {
      Add MV of C to the set of candidate MVs.
   } else if (C is 4 Frame MV) {
      if (C is top right MB) {
         Add the bottom left block MV of C to the set of candidate MVs.
      } else {//C是左上MB
         Add the bottom right block MV of C to the set of candidate MVs.
      }
   } else if (C is 2 Field MV) {
      Average the two field MVs of C and add the resulting MV to the set of candidate MVs.
   } else if (C is 4 Field MV) {
      if (C is top right MB) {
         Average the top left block field MV and bottom left block field MV of C and add the
               resulting MV to the set of candidate MVs.
      } else { //C是左上MB
         Average the top right block field MV and bottom right block field MV of C and add the
               resulting MV to the set of candidate MVs.
      }
   }
}
```

图 66

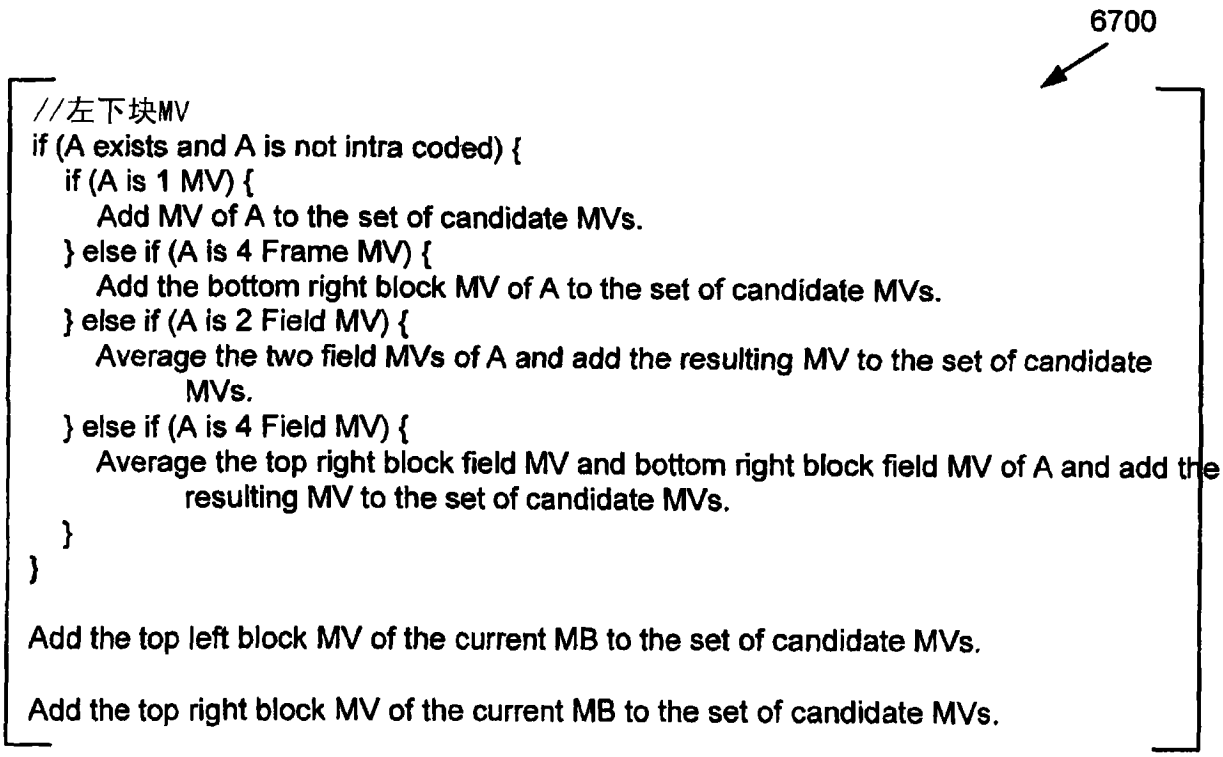

6700
//左下块MV
if (A exists and A is not intra coded) {
if (A is 1 MV) {
Add MV of A to the set of candidate MVs.
} else if (A is 4 Frame MV) {
Add the bottom right block MV of A to the set of candidate MVs.
} else if (A is 2 Field MV) {
Average the two field MVs of A and add the resulting MV to the set of candidate MVs.
} else if (A is 4 Field MV) {
Average the top right block field MV and bottom right block field MV of A and add the resulting MV to the set of candidate MVs.
}
}
Add the top left block MV of the current MB to the set of candidate MVs.
Add the top right block MV of the current MB to the set of candidate MVs.

图 67

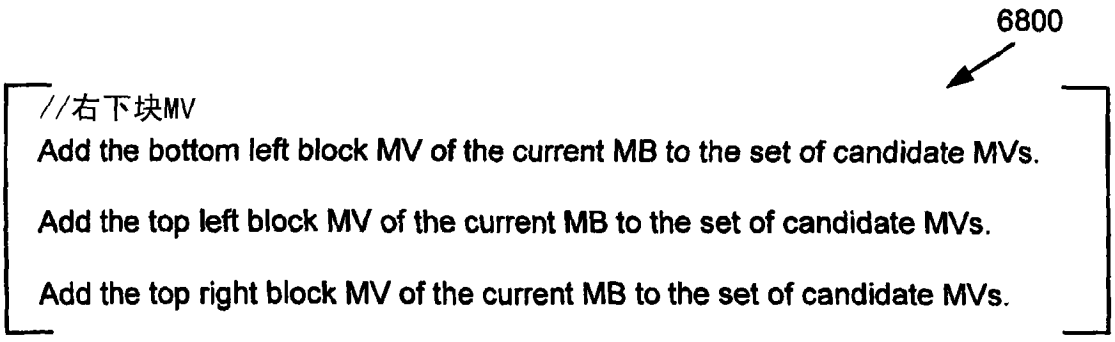

6800
//右下块MV
Add the bottom left block MV of the current MB to the set of candidate MVs.
Add the top left block MV of the current MB to the set of candidate MVs.
Add the top right block MV of the current MB to the set of candidate MVs.

图 68

6900

```
//上半帧MV
if (A exists and A is not intra coded) {
   if (A is 1 MV) {
      Add MV of A to the set of candidate MVs.
   } else if (A is 4 Frame MV) {
      Add the top right block MV of A to the set of candidate MVs.
   } else if (A is 2 Field MV) {
      Add the top field MV of A to the set of candidate MVs.
   } else if (A is 4 Field MV) {
      Add the top right field block MV of A to the set of candidate MVs.
   }
}
if (B exists and B is not intra coded) {
   if (B is 1 MV) {
      Add MV of B to the set of candidate MVs.
   } else if (B is 4 Frame MV) {
      Add the bottom left block MV of B to the set of candidate MVs.
   } else if (B is 2 Field MV) {
      Add the top field MV of B to the set of candidate MVs.
   } else if (B is 4 Field MV) {
      Add the top left field block MV of B to the set of candidate MVs.
   }
}
if (C exists and C is not intra coded) {
   if (C is 1 MV) {
      Add MV of C to the set of candidate MVs.
   } else if (C is 4 Frame MV) {
      if (C is top right MB) {
         Add the bottom left block MV of C to the set of candidate MVs.
      } else { //C是左上MB
         Add the bottom right block MV of C to the set of candidate MVs.
      }
   } else if (C is 2 Field MV) {
      Add the top field MV of C to the set of candidate MVs.
   } else if (C is 4 Field MV) {
      if (C is top right MB) {
         Add the top left field block MV of C to the set of candidate MVs.
      } else { //C是左上MB
         Add the top right field block MV of C to the set of candidate MVs.
      }
   }
}
```

图 69

7000

```
//下半帧MV
if (A exists and A is not intra coded) {
   if (A is 1 MV) {
      Add MV of A to the set of candidate MVs.
   } else if (A is 4 Frame MV) {
      Add the bottom right block MV of A to the set of candidate MVs.
   } else if (A is 2 Field MV) {
      Add the bottom field MV of A to the set of candidate MVs.
   } else if (A is 4 Field MV) {
      Add the bottom right field block MV of A to the set of candidate MVs.
   }
}
if (B exists and B is not intra coded) {
   if (B is 1 MV) {
      Add MV of B to the set of candidate MVs.
   } else if (B is 4 Frame MV) {
      Add the bottom left block MV of B to the set of candidate MVs.
   } else if (B is 2 Field MV) {
      Add the bottom field MV of B to the set of candidate MVs.
   } else if (B is 4 Field MV) {
      Add the bottom left field block MV of B to the set of candidate MVs.
   }
}
if (C exists and C is not intra coded) {
   if (C is 1 MV) {
      Add MV of C to the set of candidate MVs.
   } else if (C is 4 Frame MV) {
      if (C is top right MB) {
         Add the bottom left block MV of C to the set of candidate MVs.
      } else { //C是左上MB
         Add the bottom right block MV of C to the set of candidate MVs.
      }
   } else if (C is 2 Field MV) {
      Add the bottom field MV of C to the set of candidate MVs.
   } else if (C is 4 Field MV) {
      if (C is top right MB) {
         Add the bottom left field block MV of C to the set of candidate MVs.
      } else {//C是左上MB
         Add the bottom right field block MV of C to the set of candidate MVs.
      }
   }
}
```

图 70

7100

```
//左上半帧块MV
if (A exists and A is not intra coded) {
   if (A is 1 MV) {
      Add MV of A to the set of candidate MVs.
   } else if (A is 4 Frame MV) {
      Add the top right block MV of A to the set of candidate MVs.
   } else if (A is 2 Field MV) {
      Add the top field MV of A to the set of candidate MVs.
   } else if (A is 4 Field MV) {
      Add the top right field block MV of A to the set of candidate MVs.
   }
}
if (B exists and B is not intra coded) {
   if (B is 1 MV) {
      Add MV of B to the set of candidate MVs.
   } else if (B is 4 Frame MV) {
      Add the bottom left block MV of B to the set of candidate MVs.
   } else if (B is 2 Field MV) {
      Add the top field MV of B to the set of candidate MVs.
   } else if (B is 4 Field MV) {
      Add the top left field block MV of B to the set of candidate MVs.
   }
}
if (C exists and C is not intra coded) {
   if (C is 1 MV) {
      Add MV of C to the set of candidate MVs.
   } else if (C is 4 Frame MV) {
      if (C is top right MB) {
         Add the bottom left block MV of C to the set of candidate MVs.
      } else { //C是左上MB
         Add the bottom right block MV of C to the set of candidate MVs.
      }
   } else if (C is 2 Field MV) {
      Add the top field MV of C to the set of candidate MVs.
   } else if (C is 4 Field MV) {
      if (C is top right MB) {
         Add the top left field block MV of C to the set of candidate MVs.
      } else { //C是左上MB
         Add the top right field block MV of C to the set of candidate MVs.
      }
   }
}
```

图 71

7200

```
//右上半帧块MV
Add the top left field block MV of the current MB to the set of candidate MVs.

if (B exists and B is not intra coded) {
   if (B is 1 MV) {
      Add MV of B to the set of candidate MVs.
   } else if (B is 4 Frame MV) {
      Add the bottom right block MV of B to the set of candidate MVs.
   } else if (B is 2 Field MV) {
      Add the top field MV of B to the set of candidate MVs.
   } else if (B is 4 Field MV) {
      Add the top right field block MV of B to the set of candidate MVs.
   }
}

if (C exists and C is not intra coded) {
   if (C is 1 MV) {
      Add MV of C to the set of candidate MVs.
   } else if (C is 4 Frame MV) {
      if (C is top right MB) {
         Add the bottom left block MV of C to the set of candidate MVs.
      } else { //C是左上MB
         Add the bottom right block MV of C to the set of candidate MVs.
      }
   } else if (C is 2 Field MV) {
      Add the top field MV of C to the set of candidate MVs.
   } else if (C is 4 Field MV) {
      if (C is top right MB) {
         Add the top left field block MV of C to the set of candidate MVs.
      } else { //C是左上MB
         Add the top right field block MV of C to the set of candidate MVs.
      }
   }
}
```

图 72

7300

```
//左下半帧块MV
if (A exists and A is not intra coded) {
   if (A is 1 MV) {
      Add MV of A to the set of candidate MVs.
   } else if (A is 4 Frame MV) {
      Add the bottom right block MV of A to the set of candidate MVs.
   } else if (A is 2 Field MV) {
      Add the bottom field MV of A to the set of candidate MVs.
   } else if (A is 4 Field MV) {
      Add the bottom right field block MV of A to the set of candidate MVs.
   }
}
if (B exists and B is not intra coded) {
   if (B is 1 MV) {
      Add MV of B to the set of candidate MVs.
   } else if (B is 4 Frame MV) {
      Add the bottom left block MV of B to the set of candidate MVs.
   } else if (B is 2 Field MV) {
      Add the bottom field MV of B to the set of candidate MVs.
   } else if (B is 4 Field MV) {
      Add the bottom left field block MV of B to the set of candidate MVs.
   }
}
if (C exists and C is not intra coded) {
   if (C is 1 MV) {
      Add MV of C to the set of candidate MVs.
   } else if (C is 4 Frame MV) {
      if (C is top right MB) {
         Add the bottom left block MV of C to the set of candidate MVs.
      } else {//C是左上MB
         Add the bottom right block MV of C to the set of candidate MVs.
      }
   } else if (C is 2 Field MV) {
      Add the bottom field MV of C to the set of candidate MVs.
   } else if (C is 4 Field MV) {
      if (C is top right MB) {
         Add the bottom left field block MV of C to the set of candidate MVs.
      } else { //C是左上MB
         Add the bottom right field block MV of C to the set of candidate MVs.
      }
   }
}
```

图 73

7400

```
//右下半帧块MV
Add the bottom left field block MV of the current MB to the set of candidate MVs.

if (B exists and B is not intra coded) {
   if (B is 1 MV) {
      Add MV of B to the set of candidate MVs.
   } else if (B is 4 Frame MV) {
      Add the bottom right block MV of B to the set of candidate MVs.
   } else if (B is 2 Field MV) {
      Add the bottom field MV of B to the set of candidate MVs.
   } else if (B is 4 Field MV) {
      Add the bottom right field block MV of B to the set of candidate MVs.
   }
}

if (C exists and C is not intra coded) {
   if (C is 1 MV) {
      Add MV of C to the set of candidate MVs.
   } else if (C is 4 Frame MV) {
      if (C is top right MB) {
         Add the bottom left block MV of C to the set of candidate MVs.
      } else { //C是左上MB
         Add the bottom right block MV of C to the set of candidate MVs.
      }
   } else if (C is 2 Field MV) {
      Add the bottom field MV of C to the set of candidate MVs.
   } else if (C is 4 Field MV) {
      if (C is top right MB) {
         Add the bottom left field block MV of C to the set of candidate MVs.
      } else { //C是左上MB
         Add the bottom right field block MV of C to the set of candidate MVs.
      }
   }
}
```

图 74

7500

```
if (TotalValidMV >= 2) {
   //注意，如果只有两个有效MV，则第三个ValidMV被设置成（0,0）
    PMVx = median3 (ValidMVx [0], ValidMVx [1], ValidMVx [2]);
    PMVy = median3 (ValidMVy [0], ValidMVy [1], ValidMVy [2]);
} else if (TotalValidMV is 1) {
    PMVx = ValidMVx [0];
    PMVy = ValidMVy [0];
} else {
    PMVx = 0;
    PMVy = 0;
}
```

图 75

7600

```
if (TotalValidMV == 3) {
    if (NumSameFieldMV == 3 || NumOppFieldMV == 3) {
        PMVx = median3 (ValidMVx [0], ValidMVx [1], ValidMVx [2]);
        PMVy = median3 (ValidMVy [0], ValidMVy [1], ValidMVy [2]);
    } else if (NumSameFieldMV >= NumOppFieldMV) {
        PMVx = SameFieldMVx [0];
        PMVy = SameFieldMVy [0];
    } else {
        PMVx = OppFieldMVx [0];
        PMVy = OppFieldMVy [0];
    }
} else if (TotalValidMV == 2) {
    if (NumSameFieldMV >= NumOppFieldMV) {
        PMVx = SameFieldMVx [0];
        PMVy = SameFieldMVy [0];
    } else {
        PMVx = OppFieldMVx [0];
        PMVy = OppFieldMVy [0];
    }
} else if (TotalValidMV == 1) {
    PMVx = ValidMVx [0];
    PMVy = ValidMVy [0];
} else {
    PMVx = 0;
    PMVy = 0;
}
```

图 76

7700

```
offset_table1[9] = {0, 1, 2, 4, 8, 16, 32, 64, 128,}
offset_table2[9] = {0, 1, 3, 7, 15, 31, 63, 127, 255}
index = vlc_decode()
//使用由图片层中MVTAB表示的哈夫曼表
if (index == 71)
{
        dmv_x = get_bits(k_x)
        dmv_y = get_bits(k_y)
}
else
{
      if (extend_x == 1)
        offset_table = offset_table2
      else
        offset_table = offset_table1
 index1 = (index + 1) % 9
      if (index1 != 0)
      {
         val = get_bits (index1 + extend_x)
         sign = 0 - (val & 1)
         dmv_x = sign ^ ((val >> 1) + offset_table[index1])
         dmv_x = dmv_x - sign
      }
      else
         dmv_x = 0
      if (extend_y == 1)
        offset_table = offset_table2
      else
        offset_table = offset_table1
 index1 = (index + 1) / 9
      if (index1 != 0)
     {
         val = get_bits (index1 + extend_y)
         sign = 0 - (val & 1)
         dmv_y = sign ^ ((val >> 1) + offset_table[index1])
         dmv_y = dmv_y - sign
      }
      else
         dmv_y = 0
}
```

图 77A

7710

```
offset_table[9] = {0, 1, 2, 4, 8, 16, 32, 64, 128}
index = vlc_decode()
//使用由图片层中MVTAB表示的哈夫曼表
if (index == 0) {
        dmv_x = 1 - 2 * get_bits(1)
        dmv_y = 0
}
if (index == 125)
{
        dmv_x = get_bits(k_x - halfpel_flag)
        dmv_y = get_bits(k_y - halfpel_flag)
}
else
{
index1 = (index + 1) % 9
      val = get_bits (index1)
      sign = 0 - (val & 1)
      dmv_x = sign ^ ((val >> 1) + offset_table[index1])
      dmv_x = dmv_x - sign

index1 = (index + 1) / 9
      val = get_bits (index1)
      sign = 0 - (val & 1)
      dmv_y = sign ^ ((val >> 1) + offset_table[index1])
      dmv_y = dmv_y - sign
}
```

图 77B

7800

```
Int s_RndTbl [] = {0, 0, 0, 1};
Int s_RndTblField [] = {0, 0, 1, 2, 4, 4, 5, 6, 2, 2, 3, 8, 6, 6, 7, 12};
CMVX = (LMVX + s_RndTbl[LMVX & 3]) >> 1;
if (LMV is a field motion vector) {
CMVY = (LMVY >> 4)*8 + s_RndTblField [LMVY & 0xF];
      } else {
      CMVY = (LMVY + s_RndTbl[LMVY & 3]) >> 1;
      }
```

图 78

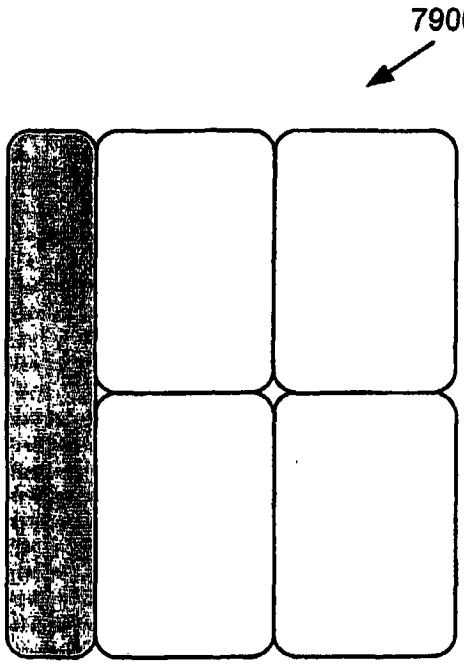


图 79A

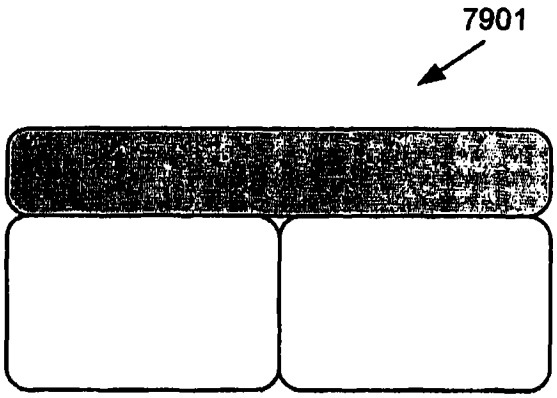


图 79B

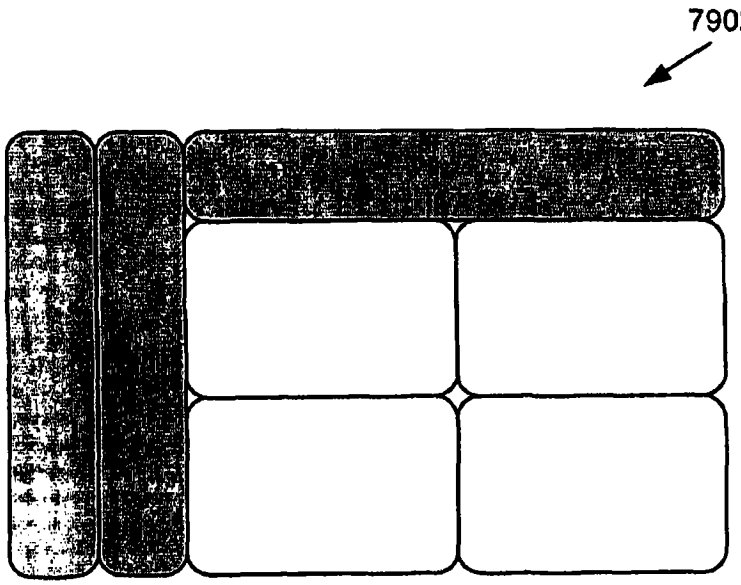


图 79C