[19] 中华人民共和国国家知识产权局

[51] Int. Cl.
H04N 7/12 (2006.01)
H04N 11/04 (2006.01)
H04N 11/02 (2006.01)

[12] 发明专利申请公开说明书

[21] 申请号 200480025454.9

[43] 公开日 2006 年 10 月 11 日

[11] 公开号 CN 1846437A

[22] 申请日 2004.9.3
[21] 申请号 200480025454.9
[30] 优先权
[32] 2003. 9. 7 [33] US [31] 60/501,081
[32] 2004. 5.27 [33] US [31] 10/857,473
[32] 2004. 9. 2 [33] US [31] 10/933,882
[86] 国际申请 PCT/US2004/029035 2004.9.3
[87] 国际公布 WO2005/027497 英 2005.3.24
[85] 进入国家阶段日期 2006.3.6
[71] 申请人 微软公司
地址 美国华盛顿州
[72] 发明人 P·苏 T·W·赫尔科比 林志隆
S·斯里尼瓦杉 K·慕克吉

[74] 专利代理机构 上海专利商标事务所有限公司
代理人 张政权

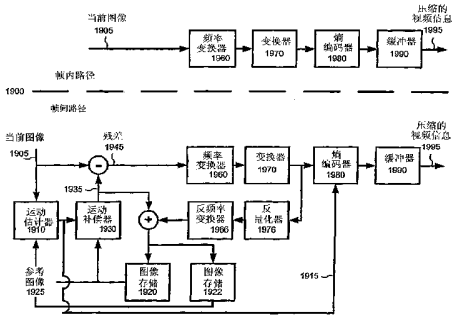
权利要求书 9 页 说明书 61 页 附图 40 页

[54] 发明名称

为隔行扫描和逐行扫描视频编码和解码宏块和运动信息中的革新

[57] 摘要

在一方面，编码器/解码器接收关于隔行扫描帧编码的前向预测图像中的宏块的四个半帧运动矢量的信息(1905)，并使用四个半帧运动矢量处理该宏块。 在另一方面，解码器解码隔行扫描帧的跳过宏块。 跳过宏块仅使用一个运动矢量，且没有运动矢量差分信息，且缺少残差信息。 跳过宏块信号指示 1 运动矢量编码。 在另一方面，解码器接收关于宏块的多个亮度运动矢量的亮度运动矢量信息，并通过在亮度运动矢量信息上执行至少一个计算来对每一亮度运动矢量导出色度运动矢量，从而维持了宏块的色度运动矢量与亮度运动矢量的 1：1 比。 例如，解码器接收宏块的四个亮度半帧运动矢量，并导出宏块的四个色度运动矢量。



1．一种方法，包括：

确定用于为隔行扫描帧编码的图像中的当前宏块预测半帧运动矢量的多个有效候选运动矢量；以及

至少部分地基于所述有效候选运动矢量，为所述半帧运动矢量计算运动矢量预测值，所述计算包括基于所述有效候选运动矢量的数目是否为三来确定是否对所述有效候选运动矢量执行中值运算。

2．如权利要求 1 所述的方法，其特征在于，还包括至少部分地基于所述运动矢量预测值和运动矢量差分信息来重构所述半帧运动矢量。

3．如权利要求 2 所述的方法，其特征在于，所述运动矢量差分信息指示对所述半帧运动矢量不存在运动矢量差分。

4．如权利要求 1 所述的方法，其特征在于，所述有效候选运动矢量的数目为三，且其中，所述计算还包括：

对所述有效候选运动矢量执行中值运算。

5．如权利要求 1 所述的方法，其特征在于，所述有效候选运动矢量的数目小于三，且其中，所述计算还包括：

选择所述有效候选运动矢量中的一个以用作所述运动矢量预测值，而无需在所述有效候选运动矢量上执行中值运算。

6．如权利要求 1 所述的方法，其特征在于，所述当前宏块是 2 半帧运动矢量宏块。

7．如权利要求 1 所述的方法，其特征在于，所述当前宏块是 4 半帧运动矢量宏块。

8．如权利要求 1 所述的方法，其特征在于，所述有效候选运动矢量中的至少一个是来自相邻宏块的帧运动矢量。

9．如权利要求 1 所述的方法，其特征在于，所述有效候选运动矢量中的至少一个是来自相邻宏块的半帧运动矢量。

10．如权利要求 1 所述的方法，其特征在于，还包括为所述当前宏块的一个或多个其它半帧运动矢量计算预测值。

11．如权利要求 1 所述的方法，其特征在于，所述隔行扫描帧编码的图像是

隔行扫描 P 帧。

12. 如权利要求 1 所述的方法，其特征在于，所述中值运算是分量级的中值运算。

13. 如权利要求 1 所述的方法，其特征在于，所述有效候选运动矢量中的每一个被表征为用于所述隔行扫描帧编码的图像中的块、宏块半帧、宏块半帧部分或宏块。

14. 如权利要求 1 所述的方法，其特征在于，所述有效候选运动矢量中的每一个被表征为用于与所述当前宏块相同的片内的块、宏块半帧、宏块半帧部分或宏块。

15. 如权利要求 1 所述的方法，其特征在于，所述有效候选运动矢量中的每一个被表征为实际的运动矢量值，而非用于帧内编码块、宏块半帧、宏块半帧部分或宏块。

16. 一种其上储存有用于在视频解码器中执行如权利要求 1 所述的方法的计算机可执行指令的计算机可读介质。

17. 一种其上储存有用于在视频编码器中执行如权利要求 1 所述的方法的计算机可执行指令的计算机可读介质。

18. 一种方法，包括：

接收关于隔行扫描帧编码的前向预测图像中的宏块的运动矢量信息，所述运动矢量信息包括关于四个半帧运动矢量的信息；以及

使用所述四个半帧运动矢量处理所述宏块；

其中，所述四个半帧运动矢量中的两个的第一集合表示所述宏块中的第一半帧中的运动，且其中，所述四个半帧运动矢量中的两个的第二集合表示所述宏块中的第二半帧中的运动。

19. 如权利要求 18 所述的方法，其特征在于，关于所述四个半帧运动矢量的信息包括关于所述四个半帧运动矢量的运动矢量差分。

20. 一种其上储存有用于在视频解码器中执行如权利要求 18 所述的方法的计算机可执行指令的计算机可读介质。

21. 一种其上储存有用于在视频编码器中执行如权利要求 18 所述的方法的计算机可执行指令的计算机可读介质。

22. 一种方法，包括：

确定用于为隔行扫描的帧编码的图像中的当前宏块的当前半帧预测半帧运动

矢量的候选运动矢量；

为一个或多个所述候选运动矢量中的每一个确定半帧极性；以及

至少部分地基于所述一个或多个半帧极性为所述半帧运动矢量计算运动矢量预测值。

23．如权利要求 22 所述的方法，其特征在于，所述候选运动矢量是以四分之一像素的增量来测量的。

24．如权利要求 22 所述的方法，其特征在于，所述确定半帧极性包括对候选运动矢量的 y 分量执行逐位 AND 运算。

25．如权利要求 22 所述的方法，其特征在于，所述当前宏块是 2 半帧运动矢量宏块。

26．如权利要求 22 所述的方法，其特征在于，所述当前宏块是 4 半帧运动矢量宏块。

27．如权利要求 22 所述的方法，其特征在于，所述候选运动矢量中的至少一个是来自相邻宏块的帧运动矢量。

28．如权利要求 22 所述的方法，其特征在于，所述候选运动矢量中的至少一个是来自相邻宏块的半帧运动矢量。

29．如权利要求 22 所述的方法，其特征在于，还包括为所述当前宏块的一个或多个半帧其它运动矢量中的每一个计算运动矢量预测值。

30．如权利要求 22 所述的方法，其特征在于，还包括至少部分地基于所述运动矢量预测值和运动矢量差分信息重构所述半帧运动矢量。

31．如权利要求 30 所述的方法，其特征在于，所述运动矢量差分信息指示对所述半帧运动矢量不存在运动矢量差分。

32．一种其上储存有用于在视频解码器中执行如权利要求 22 所述的方法的计算机可执行指令的计算机可读介质。

33．如权利要求 22 所述的方法，其特征在于，所述为候选运动矢量确定半帧极性包括确定所述候选运动矢量是否指示参考帧的上半帧或下半帧中的位移。

34．如权利要求 22 所述的方法，其特征在于，所述为候选运动矢量确定半帧极性包括确定所述候选运动矢量是否指示具有与所述当前半帧相同或相反极性的半帧中的位移。

35．一种其上储存有用于在视频编码器中执行如权利要求 22 所述的方法的计算机可执行指令的计算机可读介质。

36．一种方法，包括：

确定用于为隔行扫描帧编码的图像中的当前宏块的当前半帧预测半帧运动矢量的一个或多个有效候选运动矢量预测值；

为所述一个或多个有效候选运动矢量预测值中的每一个别有效候选运动矢量预测值确定半帧极性；

根据其半帧极性，将每一个别有效候选运动矢量预测值分配给两个集合中的一个；以及

至少部分地基于所述两个集合中的一个或多个为所述半帧运动矢量计算运动矢量预测值。

37．如权利要求36所述的方法，其特征在于，所述两个集合是由相反极性集合和相同极性集合构成的。

38．如权利要求37所述的方法，其特征在于，所述计算运动矢量预测值包括选择主要极性的有效候选运动矢量预测值。

39．如权利要求38所述的方法，其特征在于，所述选择主要极性的有效候选运动矢量预测值包括以指定的选择顺序选择第一有效候选运动矢量预测值。

40．如权利要求37所述的方法，其特征在于，所述分配包括仅将一个相同极性的有效候选运动矢量预测值分配给所述相同极性集合，以及仅将一个相反极性的有效候选运动矢量预测值分配给所述相反极性集合，且其中，所述计算运动矢量预测值包括选择所述相同极性的有效候选运动矢量预测值。

41．如权利要求36所述的方法，其特征在于，所述分配包括将三个有效候选运动矢量预测值分配给所述两个集合中的一个，以及不将任何有效候选运动矢量预测值分配给所述两个集合中的另一个。

42．如权利要求41所述的方法，其特征在于，所述计算运动矢量预测值包括对所述三个有效候选运动矢量预测值执行中值运算。

43．一种其上储存有用于在视频解码器中执行如权利要求36所述的方法的计算机可执行指令的计算机可读介质。

44．一种其上储存有用于在视频编码器中执行如权利要求36所述的方法的计算机可执行指令的计算机可读介质。

45．一种方法，包括：

解码隔行扫描帧的多个宏块中的一个或多个跳过宏块，其中，所述一个或多个跳过宏块中的每一个（1）是由比特流中的跳过宏块信号指示的，（2）仅使用一

个运动矢量且没有任何运动矢量差分信息，以及（3）缺少残差信息；

其中，对所述一个或多个跳过宏块中的每一个本身的跳过宏块信号指示对所述相应跳过宏块的 1 运动矢量经运动补偿的解码。

46．如权利要求 45 所述的方法，其特征在于，对所述一个或多个跳过宏块中的每一个的跳过宏块信号是在具有多个层的比特流中的帧层发送的压缩位平面的一部分。

47．如权利要求 46 所述的方法，其特征在于，所述多个层包括序列层、入口点层、帧层和宏块层。

48．如权利要求 45 所述的方法，其特征在于，对所述一个或多个跳过宏块中的每一个的跳过宏块信号是在具有多个层的比特流中的半帧层发送的压缩位平面的一部分。

49．如权利要求 45 所述的方法，其特征在于，对所述一个或多个跳过宏块中的每一个的跳过宏块信号是由在具有多个层的比特流中的宏块层发送的个别比特构成的。

50．如权利要求 45 所述的方法，其特征在于，所述隔行扫描帧是隔行扫描 P 帧。

51．如权利要求 45 所述的方法，其特征在于，所述隔行扫描帧是隔行扫描 B 帧。

52．如权利要求 45 所述的方法，其特征在于，所述隔行扫描帧是具有隔行扫描 P 半帧的帧。

53．如权利要求 45 所述的方法，其特征在于，所述隔行扫描帧是具有隔行扫描 B 半帧的帧。

54．如权利要求 45 所述的方法，其特征在于，所述仅一个运动矢量是帧运动矢量。

55．一种其上储存有用于执行如权利要求 45 所述的方法的计算机可执行指令的计算机可读介质。

56．一种方法，包括：

从一组多个可用编码模式中选择一个编码模式；以及

依照所选择的编码模式处理位平面，其中，所述位平面包括用信号表示隔行扫描帧中的宏块是被跳过还是未跳过的二进制信息，且其中，如果：（1）所述宏块只有一个运动矢量；（2）所述仅一个运动矢量等于为所述宏块预测的运动矢量；

以及（3）所述宏块没有残留误差，则所述隔行扫描帧中的宏块被跳过，但是如果所述宏块具有多个运动矢量，则所述宏块不被跳过。

57．一种其上储存有用于在视频编码器中执行如权利要求 56 所述的方法的计算机可执行指令的计算机可读介质。

58．一种其上储存有用于在视频解码器中执行如权利要求 56 所述的方法的计算机可执行指令的计算机可读介质。

59．一种方法，包括：

为隔行扫描 P 帧中的宏块选择运动补偿类型；

为所述宏块选择半帧/帧编码类型；以及

为所述宏块对所述运动补偿类型和所述半帧/帧编码类型进行联合编码。

60．如权利要求 59 所述的方法，其特征在于，还包括将所述宏块的其它信息与所述运动补偿类型和所述半帧/帧编码类型一起联合编码。

61．如权利要求 60 所述的方法，其特征在于，所述宏块的其它信息包括所述宏块的差分运动矢量的存在的指示符。

62．如权利要求 59 所述的方法，其特征在于，还包括将所述宏块的差分运动矢量的存在或缺乏的指示符与所述运动补偿类型和所述半帧/帧编码类型一起进行联合编码，其中所述宏块是 1 运动矢量宏块。

63．如权利要求 59 所述的方法，其特征在于，所述运动补偿类型是从由以下各项构成的组中选出的：1MV、4 帧 MV、2 半帧 MV 或 4 半帧 MV。

64．如权利要求 59 所述的方法，其特征在于，所述半帧/帧编码类型是从由以下各项构成的组中选出的：半帧变换、帧变换或无已编码块。

65．一种其上储存有用于执行如权利要求 59 所述的方法的计算机可执行指令的计算机可读介质。

66．一种方法，包括：

接收关于隔行扫描 P 帧中的宏块的宏块信息，所述宏块信息包括表示所述宏块的运动补偿类型和半帧/帧编码类型的联合码；以及

解码所述联合码以获得所述宏块的运动补偿类型信息和半帧/帧编码类型信息两者。

67．如权利要求 66 所述的方法，其特征在于，所述联合码是可变长度代码表中的可变长度码。

68．如权利要求 66 所述的方法，其特征在于，所述解码包括在可变长度代码

表中查找所述联合码。

69．如权利要求 66 所述的方法，其特征在于，所述联合码还表示关于所述宏块的其它信息。

70．如权利要求 69 所述的方法，其特征在于，所述宏块的其它信息包括所述宏块的差分运动矢量的存在的指示符。

71．如权利要求 66 所述的方法，其特征在于，所述联合码还表示所述宏块的差分运动矢量的存在的指示符，其中所述宏块是 1 运动矢量宏块。

72．如权利要求 66 所述的方法，其特征在于，所述运动补偿类型是从由以下各项构成的组中选出的：1MV、4 帧 MV、2 半帧 MV 或 4 半帧 MV。

73．如权利要求 66 所述的方法，其特征在于，所述半帧/帧编码类型是从由以下各项构成的组中选出的：半帧变换、帧变换或无已编码块。

74．一种其上储存有用于执行如权利要求 66 所述的方法的计算机可执行指令的计算机可读介质。

75．一种方法，包括：

接收关于一个宏块的两个以上亮度运动矢量的亮度运动矢量信息，所述两个以上亮度运动矢量中的每一个与所述宏块的至少一部分相关联；以及

对所述两个以上亮度运动矢量中的每一个导出与所述宏块的至少一部分相关联的色度运动矢量，所述导出包括执行涉及所述亮度运动矢量信息上的舍入表的至少一个计算。

76．如权利要求 75 所述的方法，其特征在于，所述宏块是半帧编码的宏块。

77．如权利要求 75 所述的方法，其特征在于，所述宏块是具有四个亮度半帧运动矢量的半帧编码的宏块，所述四个亮度半帧运动矢量中的每一个描述了所述宏块的亮度半帧的一部分中的运动。

78．如权利要求 77 所述的方法，其特征在于，每一色度运动矢量是描述所述宏块的色度半帧的一个不同部分中的运动的色度半帧运动矢量。

79．如权利要求 75 所述的方法，其特征在于，所述舍入表是基于半帧的舍入表，且其中，所述导出包括使用所述基于半帧的舍入表舍入所述亮度运动矢量信息的至少一部分。

80．如权利要求 75 所述的方法，其特征在于，所述宏块是具有四个亮度帧运动矢量的帧编码的宏块，所述四个亮度帧运动矢量中的每一个与所述宏块的一个不同亮度块相关联。

81．如权利要求 75 所述的方法，其特征在于，所述宏块是 4:2:0 宏块。

82．如权利要求 75 所述的方法，其特征在于，所述导出包括对所述亮度运动矢量信息的至少一部分进行舍入和二次采样。

83．如权利要求 82 所述的方法，其特征在于，所述舍入表是基于半帧的舍入查找表，且其中，所述舍入是使用所述基于半帧的舍入查找表来执行的。

84．如权利要求 75 所述的方法，其特征在于，所述两个以上亮度运动矢量是以四分之一像素为单位的。

85．一种其上储存有用于执行如权利要求 75 所述的方法的计算机可执行指令的计算机可读介质。

86．一种在视频解码器中的方法，包括：

接收关于一个或多个亮度运动矢量的亮度运动矢量信息，所述一个或多个亮度运动矢量中的每一个与隔行扫描 P 帧中的宏块的至少一部分相关联；以及

对所述一个或多个亮度运动矢量中的每一个导出与所述宏块的至少一部分相关联的色度运动矢量，所述导出至少部分地基于关于所述一个或多个亮度运动矢量的运动矢量信息；

其中，所述视频解码器可用于对使用隔行扫描 P 帧中的四个亮度半帧运动矢量预测的宏块进行解码。

87．如权利要求 86 所述的方法，其特征在于，所述解码器还可用于对具有四个亮度帧运动矢量的帧间编码的宏块、具有两个亮度半帧运动矢量的帧间编码的宏块、具有一个亮度帧运动矢量的帧间编码的宏块、以及帧内编码的宏块进行解码。

88．如权利要求 86 所述的方法，其特征在于，所述导出包括导出四个色度半帧运动矢量。

89．如权利要求 86 所述的方法，其特征在于，所述导出包括向所述亮度运动矢量信息的至少一部分应用基于半帧的舍入查找表。

90．一种包括用于使得计算机执行一种方法的计算机可执行指令的计算机可读介质，所述方法包括：

接收关于一个或多个亮度半帧运动矢量的亮度运动矢量信息，所述一个或多个亮度半帧运动矢量中的每一个与隔行扫描的帧编码的图像中的宏块的至少一部分相关联；以及

对所述一个或多个亮度半帧运动矢量中的每一个导出与所述宏块的至少一部分相关联的色度运动矢量，所述导出至少部分地基于关于所述一个或多个亮度半帧

运动矢量的运动矢量信息，所述导出色度运动矢量包括：

使用基于半帧的舍入表舍入亮度半帧运动矢量分量；以及

对所述亮度半帧运动矢量分量进行二次采样。

91．如权利要求 90 所述的方法，其特征在于，所述隔行扫描帧编码的图像是隔行扫描 P 帧。

92．如权利要求 90 所述的方法，其特征在于，所述基于半帧的舍入表是由以下值集合构成的整数数组：{0, 0, 1, 2, 4, 4, 5, 6, 2, 2, 3, 8, 6, 6, 7, 12}。

93．如权利要求 90 所述的方法，其特征在于，所述一个或多个亮度半帧运动矢量包括四个亮度半帧运动矢量。

94．如权利要求 90 所述的方法，其特征在于，所述宏块是 4:2:0 宏块，且其中，所述二次采样包括将所述亮度半帧运动矢量分量除以 2。

为隔行扫描和逐行扫描视频编码和解码宏块和运动信息中的革新

版权授权

技术领域

描述了用于逐行扫描和隔行扫描视频编码和解码的技术和工具。例如，编码器用信号表示隔行扫描帧编码图像中的宏块的宏块模式信息。作为另一示例，编码器/解码器编码和解码隔行扫描帧编码图像中的亮度和色度运动矢量。

背景

数字视频消耗了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 个图像。每一图像可包括几万或者几十万个像素（也称为 pel）。每一像素表示图像中的一个微小元素。在原始形式中，计算机通常用 24 个比特或更多来表示一个像素。由此，典型的原始数字视频序列的每秒的比特数，即比特率可以是 5 百万比特/秒或更多。

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，工程师使用压缩（也称为编码或写码）来降低数字视频的比特率。压缩可以是无损的，其中视频质量不受损害，但是比特率的降低受到视频复杂度的限制。或者，压缩可以是有损的，其中视频质量受到损害，但是比特率的降低更显著。解压反转了压缩。

一般而言，视频压缩技术包括“帧内”压缩和“帧间”或预测压缩。帧内压缩技术压缩个别的图像，通常称为 I 帧或关键帧。帧间压缩技术参考前导和/或后续帧来压缩各帧，且帧间压缩的帧通常被称为预测帧、P 帧或 B 帧。

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

微软公司的 Windows Media Video（Windows 媒体视频）版本 8[“WMV8”]

包括视频编码器和视频解码器。WMV8 编码器使用帧内和帧间压缩，而 WMV8 解码器使用帧内和帧间解压。Windows Media Video 版本 9[“WMV9”]对许多操作使用类似的体系结构。

WMV8 编码器中的帧间压缩使用基于块的经运动补偿的预测编码，之后是残留误差的变换编码。图 1 和 2 示出了用于 WMV8 编码器中的预测帧的基于块的帧间压缩。具体地，图 1 示出了用于预测帧 110 的运动估计，图 2 示出了用于预测帧的经运动补偿的块的预测残差的压缩。

例如，在图 1 中，WMV8 编码器为预测帧 110 中的宏块 115 计算运动矢量。为计算运动矢量，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 中，编码器将来自预测帧 110 的宏块 115 与各种候选宏块进行比较，以找出作为良好匹配的候选宏块。编码器输出指定匹配宏块的运动矢量（熵编码的）的信息。

由于运动矢量值通常与空间上的周围运动矢量的值相关，因此用于发送运动矢量信息的数据的压缩可通过基于相邻宏块的运动矢量选择运动矢量预测值，并使用运动矢量预测值为当前宏块预测运动矢量来实现。编码器可对运动矢量和预测值之间的差分进行编码。在通过将差分加到预测值重构运动矢量之后，解码器使用运动矢量，利用来自参考帧 130 的信息来为宏块 115 计算预测宏块，参考帧 130 是在编码器和解码器处可用的先前重构的帧。预测很少是完美的，因此编码器通常对预测宏块和宏块 115 本身之间的像素差块（也称为误差或残差块）进行编码。

图 2 示出了 WMV8 编码器中的误差块 235 的计算和编码的示例。误差块 235 是预测的块 215 和原始的当前块 225 之差。编码器向误差块 235 应用离散余弦变换[“DCT”]240，得到 8×8 的系数块 245。编码器然后量化（250）该 DCT 系数，得到 8×8 的经量化的 DCT 系数块 255。编码器将 8×8 的块 255 扫描（260）成一维数组 265，使得系数一般从最低频率到最高频率排序。编码器使用行程长度编码 270 的变更对扫描的系数进行熵编码。编码器从一个或多个“行程/等级/最后”表 275 中选择熵码，并输出该熵码。

图 3 示出了用于帧间编码块的对应解码过程 300 的示例。在图 3 的概述中，解码器使用可变长度解码 310，利用一个或多个“行程/级别/最后”表 315 和行程长度解码 320，对表示预测残差的熵编码的信息进行解码（310、320）。解码器将储存熵编码的信息的一维数组 325 反扫描（330）成二维块 335。解码器对该数据进行反量化和离散反余弦变换（共同在 340 处），得到重构的误差块 345。在独立的运动补偿路径中，解码器对从参考帧的偏移使用运动矢量信息 355 计算预测块

365。解码器将预测块 365 与重构的误差块 345 组合（370），以形成重构块 375。

原始的和重构的帧之间的改变量是失真，且编码该帧所需的比特数指示了帧速率。失真的量大致与速率成反比。

II. 隔行扫描视频和逐行扫描视频

视频帧包含视频信号的空间信息的行。对于逐行扫描视频，这些行包含从一个时刻开始继续通过连续的行直到帧底部的样值。逐行扫描的 I 帧是帧内编码的逐行扫描视频帧。逐行扫描的 P 帧是使用前向预测编码的逐行扫描视频帧，而逐行扫描的 B 帧是使用双向预测编码的逐行扫描视频帧。

典型的隔行扫描视频帧由两个半帧构成，这两个半帧是从不同的时刻开始扫描的。例如，参考图 4，隔行扫描的视频帧 400 包括上半帧 410 和下半帧 420。通常，偶数号的行（上半帧）是从一个时刻（例如，时刻 t）开始扫描的，而奇数号的行（下半帧）是从一个不同（通常稍晚）的时刻（例如，时刻 t+1）开始扫描的。这一时序可创建隔行扫描视频帧的存在运动的区域中看似锯齿状的特征，因为两个半帧是在不同的时刻开始扫描的。为此，可依照半帧结构对隔行扫描视频帧重新排列，使得奇数行被组合在一起成为一个半帧，而偶数行被组合在一起成为另一半帧。这一排列被称为半帧编码，它在高运动图像中对于降低这一锯齿状边缘的人为因素是非常有用的。另一方面，在静止区域中，隔行扫描视频帧中的图像细节可被更有效地保存，而无需这样的重新排列。因此，通常在静止或低运动的隔行扫描视频帧中使用帧编码，在这样的视频帧中，保存了原始的交错半帧行排列。

典型的逐行扫描视频帧由具有非交错行的内容的一帧构成。与隔行扫描视频相反，逐行扫描视频不将视频帧划分成独立的半帧，且从单个时刻开始，从左到右、从上到下扫描整个帧。

III. 早先的 WMV 编码器和解码器中的 P 帧编码和解码

编码器和解码器在 P 帧中使用逐行扫描和隔行扫描编码和解码。在隔行扫描和逐行扫描 P 帧中，在编码器中通过计算运动矢量和运动矢量预测值之间的差分来编码运动矢量，它是基于相邻运动矢量来计算的。在解码器中，通过将运动矢量差分加到运动矢量预测值来重构运动矢量，它同样也是基于相邻运动矢量来计算的（这次是在解码器中）。由此，当前宏块或当前宏块的半帧的运动矢量预测值是基于候选块来选择的，而运动矢量差分是基于运动矢量预测值来计算的。运动矢量可

通过在编码器或解码器侧将运动矢量差分加到所选择的运动矢量预测值来重构。通常，亮度运动矢量是从编码的运动信息中重构的，而色度运动矢量是从重构的亮度运动矢量中导出的。

A．逐行扫描 P 帧编码和解码

例如，在编码器和解码器中，逐行扫描 P 帧可包含在 1 运动矢量（1MV）模式或在 4 运动矢量（4MV）模式中编码的宏块，或跳过的宏块，其中决策一般是在逐个宏块的基础上作出的。仅具有 1MV 宏块（且可能有跳过宏块）的 P 帧被称为 1MV P 帧，而同时具有 1MV 和 4MV 宏块（可能具有跳过宏块）的 P 帧称成为混合 MV P 帧。一个亮度运动矢量与每一 1MV 宏块相关联，且多达四个亮度运动矢量与每一 4MV 宏块相关联（对每一块有一个亮度运动矢量）。

图 5A 和 5B 是示出对 1MV 逐行扫描 P 帧中的宏块的候选运动矢量预测值考虑的宏块的位置的图示。候选预测值取自左侧、顶部和右上角的宏块，除了在宏块是行中的最后一个宏块的情况之外。在这一情况下，预测值 B 取自左上角的宏块，而非右上角。对于其中该帧是一个宏块宽的特殊情况，预测值总是为预测值 A（顶部预测值）。当由于宏块在顶行中而使预测值 A 位于带外时，预测值为预测值 C。各种其它规则解决诸如帧内编码预测值等其它特殊情况。

图 6A-10 示出了为混合 MV 帧中的 1MV 或 4MV 宏块的运动矢量的多达三个候选运动矢量考虑的块或宏块的位置。在以下附图中，较大的方块是宏块边界，而较小的方块是块边界。对于其中帧是一个宏块宽的特殊情况，预测值总是预测值 A（顶部预测值）。各种其它规则解决诸如对顶行的 4MV 宏块的顶行块、顶行的 1MV 宏块以及帧内编码预测值等其它特殊情况。

图 6A 和 6B 是示出为混合 MV 帧中的 1MV 当前宏块的候选运动矢量预测值考虑的块位置的图示。相邻宏块可以是 1MV 或 4MV 宏块。图 6A 和 6B 示出了假定邻块是 4MV 时候选运动矢量的位置（即，预测值 A 是对于在当前宏块上方的宏块中的块 2 的运动矢量，而预测值 C 是直接在当前宏块左边的宏块中的块 1 的运动矢量）。如果邻块中的任一块是 1MV 宏块，则图 5A 和 5B 所示的运动矢量预测值被带到整个宏块的运动矢量预测值。如图 6B 所示，如果宏块是行中的最后一个宏块，则预测值 B 来自左上角的宏块的块 3，而非其它情况下的右上角宏块中的块 2。

图 7A-10 示出了为 4MV 宏块中的 4 个亮度块的每一个的候选运动矢量预测值

考虑的块位置。图 7A 和 7B 是示出为位置 0 处的块的候选运动矢量预测值考虑的块位置的图示；图 8A 和 8B 是示出为位置 1 处的块的候选运动矢量预测值考虑的块位置的图示；图 9 是示出为位置 2 处的块的候选运动矢量预测值考虑的块位置的图示；图 10 是为位置 3 处的块的候选运动矢量预测值考虑的块位置的图示。再一次，如果邻块是 1MV 宏块，则该宏块的运动矢量预测值用于该宏块的各块。

对于其中宏块是行中的第一个宏块的情况，块 0 的预测值 B 与该行中其余宏块的块 0 不同地处理（见图 7A 和 7B）。在这一情况下，预测值 B 取自直接在当前宏块上方的宏块中的块 3，而非如其它情况下取自在当前宏块左上方的宏块中的块 3。类似地，对于其中宏块是行中的最后一个宏块的情况，块 1 的预测值 B 加以不同的处理（图 8A 和 8B）。在这一情况下，预测值取自直接在当前宏块上方的宏块中的块 2，而非如其它情况下取自当前宏块右上方的宏块中的块 2。一般而言，如果宏块是在第一个宏块列中，块 0 和 2 的预测值 C 被设为等于 0。

B．隔行扫描 P 帧编码和解码

编码器和解码器对隔行扫描的 P 帧使用 4:1:1 的宏块格式，它包含以半帧模式或帧模式编码的宏块，或跳过的宏块，其中决策一般是在逐个宏块的基础上作出的。两个运动矢量与每一半帧编码的帧间编码宏块相关联（对每一半帧有一个运动矢量），而一个运动矢量与每一帧编码的帧间编码宏块相关联。编码器联合编码运动信息，包括水平和垂直运动矢量差分分量，可能还有其它用信号表示的信息。

图 11、12 和 13 示出了在编码器和解码器的隔行扫描 P 帧中，分别用于帧编码的 4:1:1 宏块和半帧编码的 4:1:1 宏块的运动矢量预测的候选预测值的示例。图 11 示出了在隔行扫描的 P 帧中用于内部位置的当前帧编码的 4:1:1 宏块（不是宏块行中的第一个或最后一个宏块，不在顶行中）的候选预测值 A、B 和 C。预测值可以从除标记为 A、B 和 C 之外的不同候选方向上获得（例如，在诸如当前宏块是行中的第一个宏块或最后一个宏块，或在顶行中的特殊情况下，由于某些预测值对于这些情况是不可用的）。对于当前帧编码的宏块，候选预测值是根据相邻的宏块是半帧编码还是帧编码的来不同地计算的。对于相邻的帧编码的宏块，仅仅取运动矢量作为候选预测值。对于相邻的半帧编码的宏块，通过对上半帧和下半帧运动矢量求平均值来确定候选运动矢量。

图 12 和 13 示出了用于半帧中的内部位置的半帧编码的 4:1:1 宏块中的当前半帧的候选预测值 A、B 和 C。在图 12 中，当前半帧是下半帧，且相邻宏块中的下

半帧运动矢量用作候选预测值。在图 13 中，当前半帧是上半帧，且相邻宏块中的上半帧运动矢量用作候选预测值。由此，对于当前半帧编码的宏块中的每一半帧，每一半帧的运动矢量候选预测值的数目最多是 3，其中每一候选预测值来自与当前半帧相同的半帧类型（例如，上半帧或下半帧）。再一次，当当前宏块是行中的第一个宏块或最后一个宏块，或在顶行中时，由于某些预测值对于这些情况是不可用的，因此应用各种特殊情况（未示出）。

为从一组候选预测值中选出一个预测值，编码器和解码器使用不同的选择算法，诸如三者中的中值（median-of-three）算法。三者中的中值预测过程在图 14 的伪代码 14 中描述。

IV. 早先的 WMV 编码器和解码器中的 B 帧编码和解码

编码器和解码器使用逐行扫描和隔行扫描的 B 帧。B 帧使用来自源视频的两个帧作为参考（或定位）帧，而非 P 帧中使用的一个定位帧。在典型的 B 帧的定位帧中，一个定位帧来自时间上的过去，而另一定位帧来自时间上的未来。参考图 15，视频序列中的 B 帧 1510 具有时间上的先前参考帧 1520 以及时间上的未来参考帧 1530。B 帧的已编码比特流通常使用比非 B 帧的已编码比特流更少的比特数，而同时仍提供类似的视觉质量。解码器也可通过选择不解码或显示 B 帧来容纳空间和时间限制，因为 B 帧一般不被用作参考帧。

尽管前向预测帧（例如，P 帧）中的宏块只有一个预测方向模式（从前一 I 或 P 帧向前），但 B 帧中的宏块可使用五个不同的预测模式来预测：前向、后向、直接、内插和帧内编码。编码器选择比特流中的不同预测模式并用信号表示。前向模式类似于常规的 P 帧预测。在前向模式中，宏块是从时间上先前的定位帧中导出的。在后向模式中，宏块是从时间上后续的定位帧中导出的。直接或内插模式中预测的宏块同时使用前向和后向定位帧用于预测。

V. 早先的 WMV 编码器和解码器中用信号表示宏块信息

在编码器和解码器中，隔行扫描的 P 帧中的宏块可以是三种可能类型中的一种：帧编码的、半帧编码的和跳过的。宏块类型是由帧级和宏块级句法元素的多元素组合来表示的。

对于隔行扫描的 P 帧，帧级元素 INTRLCF 指示用于对该帧中的宏块编码的模式。如果 INTRLCF = 0，则该帧中的所有宏块都是帧编码的。如果 INTRLCF = 1，

则宏块可以是半帧编码或帧编码的。当 INTRLCF = 1 时，INTRLCMB 元素存在于帧层中。INTRLCMB 是位平面编码的数组，它指示了图像中的每一宏块的半帧/帧编码状态。解码的位平面将每一宏块的隔行扫描的状态表示为 1 比特值的数组。特定比特的 0 值指示对应的宏块是以帧模式编码的。1 值指示对应的宏块是以半帧模式编码的。

对于帧编码的宏块，宏块级 MVDATA 元素与宏块中的所有块相关联。MVDATA 用信号表示宏块中的块是帧内编码还是帧间编码的。如果它们是帧间编码的，则 MVDATA 也指示运动矢量差分。

对于半帧编码的宏块，TOPMVDATA 元素与该宏块中的上半帧块相关联，而 BOTMVDATA 元素与该宏块中的下半帧块相关联。TOPMVDATA 和 BOTMVDATA 在每一半帧的第一个块处发送，TOPMVDATA 指示上半帧块是帧内编码还是帧间编码的。同样，BOTMVDATA 指示下半帧块是帧内编码还是帧间编码的。对于帧间编码的块，TOPMVDATA 和 BOTMVDATA 也指示运动矢量差分信息。

CBPCY 元素指示用于宏块中的亮度和色度分量的已编码块模式（CBP）信息。CBPCY 元素也指示哪些半帧在比特流中具有运动矢量数据元素。CBPCY 和运动矢量数据元素用于指定块是否具有 AV 系数。如果从 MVDATA 解码的“最后一个”值指示在要解码的运动矢量之后有数据，则 CBPCY 对于隔行扫描的 P 帧的帧编码的宏块存在。如果 CBPCY 存在，则它解码到 6 比特字段，对四个 Y 块中的每一个有一个比特，对两个 U 块（上半帧和下半帧）有一个比特，且对两个 V 块（上半帧和下半帧）有一个比特。

CBPCY 对半帧编码的宏块总是存在。CBPCY 和两个半帧运动矢量数据元素用于确定宏块的块中 AC 系数的存在。CBPCY 的意义与对于比特 1、3、4 和 5 的帧编码的宏块相同。即，它们分别指示右上半帧 Y 块、右下半帧 Y 块、顶部/底部 U 块以及顶部/底部 V 块中 AC 系数的存在或不存在。对于比特位置 0 和 2，意义略有不同。比特位置 0 中的 0 指示 TOPMVDATA 不存在，且运动矢量预测值用作顶部的半帧块的运动矢量。它也指示左上半帧块不包含任何非零系数。比特位置 0 中的 1 指示存在 TOPMVDATA。TOPMVDATA 指示顶部的半帧块是帧间编码还是帧内编码的，如果是帧间编码，则还指示运动矢量差分。如果从 TOPMVDATA 中解码的“最后一个”值解码为 1，则对于左上半帧块不存在 AC 系数，否则，对于左上半帧块存在非零 AC 系数。类似地，上述规则应用于 BOTMVDATA 和左下

半帧块的比特位置 2。

VI. 早先的 WMV 编码器和解码器中跳过的宏块

编码器和解码器使用跳过的宏块来降低比特率。例如，编码器在比特流中用信号表示跳过的宏块。当解码器接收到比特流中指示该宏块被跳过的信息（例如，跳过的宏块标志）时，解码器对该宏块跳过残差块信息的解码。相反，解码器使用来自参考帧中共同定位或经运动补偿（用运动矢量预测值）的宏块的对应像素数据来重构宏块。编码器和解码器在多个编码/解码模式之间选择，以对跳过的宏块信息进行编码和解码。例如，跳过的宏块信息在比特流的帧级（例如，在压缩的位平面中）或在宏块级（例如，对每一宏块有一个“跳过”位）上用信号表示。对于位平面编码，编码器和解码器在不同的位平面编码模式之间选择。

一种早先的编码器和解码器将跳过的宏块定义为其运动等于其因果预测的运动且具有零残留误差的预测宏块。另一种早先的编码器和解码器将跳过的宏块定义为具有零运动和零残留误差的预测宏块。

对于跳过的宏块和位平面编码的更多信息，参见 2002 年 12 月 6 日提交的名为“Skip Macroblock Coding（跳过宏块编码）”的美国专利申请第 10/321,415 号。

VII. 早先的 WMV 编码器和解码器中的色度运动矢量

色度运动矢量导出是视频编码和解码的一个重要方面。因此，用于早先的 WMV 编码器和解码器的软件使用了舍入和二次采样来从亮度（或“luma”）运动矢量中导出色度（或“chroma”）运动矢量。

A. 亮度运动矢量重构

早先的 WMV 编码器和解码器对于逐行扫描帧中的 1MV 和 4MV 宏块，以及隔行扫描帧中的帧编码或半帧编码的宏块重构运动矢量。亮度运动矢量是通过将运动矢量差分加到运动矢量预测值来重构的。在逐行扫描帧的 1MV 宏块以及隔行扫描帧的帧编码宏块中，单个亮度运动矢量应用于构成该宏块的亮度分量的四个块。在逐行扫描帧的 4MV 宏块中，宏块中的每一帧间编码的亮度块具有其自己的亮度运动矢量。因此，对于每一 4MV 宏块有多达 4 个亮度运动矢量，取决于该宏块中帧间编码的块的数目。在隔行扫描帧的半帧编码宏块中，有两个亮度运动矢量，对每一半帧有一个。

B．色度运动矢量的导出和重构

编码器和解码器对逐行扫描帧使用 4:2:0 的宏块格式。帧数据包括亮度（“Y”）分量和色度分量（“U”和“V”）。每一宏块包括四个 8×8 的亮度块（有时候作为一个 16×16 的宏块来处理）以及两个 8×8 的色度块。图 16 示出了 4:2:0 YUV 采样网格。图 16 的 4:2:0 YUV 采样网格示出了宏块的亮度和色度样值之间的空间关系。色度样值的分辨率在水平（x）和垂直（y）方向上都是亮度样值分辨率的一半。由此，为从亮度网格上的对应距离中导出色度网格上的距离，早先的 WMV 编码器和解码器将亮度运动矢量分量除以 2。这是在逐行扫描帧中从亮度运动矢量中导出色度运动矢量时的下采样步骤的基础。

编码器和解码器用两个步骤对逐行扫描帧中的色度矢量进行重构。首先，通过适当地组合和比例缩放亮度运动矢量来获得名义色度运动矢量。其次，在比例缩放之后可任选地执行舍入来减少解码时间。

例如，在 1MV 宏块中，依照以下伪代码，通过比例缩放亮度分量从亮度运动矢量分量（lmv_x 和 lmv_y）中导出色度运动矢量分量（cmv_x 和 cmv_y）：

```
// s_RndTbl[0] = 0, s_RndTbl[1] = 0, s_RndTbl[2] = 0, s_RndTbl[3] = 1
cmv_x = (lmv_x + s_RndTbl[1mv_x &3]) >> 1
cmv_y = (lmv_y + s_RndTbl[1mv_y &3]) >> 1
```

比例缩放是用舍入表数组（s-RndTbl[ ]）来执行的，使得半像素偏移量优于四分之一像素偏移量。可以在比例缩放运算之后执行额外的舍入以减少解码时间。

在 4MV 宏块中，编码器和解码器依照图 17 的伪代码 1700 从四个亮度块内的运动信息中导出色度运动矢量。如图 17 所示，编码器和解码器使用中值预测或通过对宏块中的帧间编码块的亮度运动矢量求平均值。来导出色度运动矢量分量。在三个或更多块是帧内编码的特殊情况下，色度块也是帧内编码的。

如果序列级位 FASTUVMC = 1，则编码器和解码器执行额外的舍入。由 FASTUVMC 用信号表示的舍入促进色度运动矢量的半像素和整数像素位置，这可加速编码。

早先的 WMV 编码器和解码器对隔行扫描帧使用 4:1:1 的宏块格式。对于隔行扫描帧，帧数据也包括亮度（“Y”）分量和色度分量（“U”和“V”）。然而，在 4:1:1 的宏块格式中，色度样值的分辨率在水平方向上是亮度样值分辨率的四分之一，而在垂直方向上是全分辨率。由此，为从亮度网格上的对应距离中导出色度

网格上的距离，早先的 WMV 编码器和解码器将水平亮度运动矢量分量除以 4。

对于帧编码的宏块，导出对应于单个亮度运动矢量的一个色度运动矢量。对于半帧编码的宏块，导出对应于两个亮度运动矢量的两个色度运动矢量，一个对应于上半帧，另一个对应于下半帧。

在早先的 WMV 编码器和解码器中，用于在隔行扫描帧中导出色度运动矢量的规则与用于半帧编码的宏块和帧编码的宏块的规则相同。色度运动矢量的 x 分量按照 4 的因子比例缩放（下采样），而色度运动矢量的 y 分量与亮度运动矢量保持相同。经比例缩放的色度运动矢量的 x 分量也被舍入到相邻的四分之一像素位置。色度运动矢量依照以下伪代码来重构，其中 cmv_x 和 cmv_y 表示色度运动矢量分量，而 lmv_x 和 lmv_y 表示对应的亮度运动矢量分量。

```
frac_x4 = (lmv_x << 2) % 16;
int_x4 = (lmv_x << 2) - frac_x;
ChromaMvRound [16] = {0, 0, 0, .25, .25, .25, .5, .5, .5, .5, .5, .75, .75, .75, 1, 1};
cmv_y = lmv_y;
cmv_x = Sign (lmv_x) * (int_x4 >> 2) + ChromaMvRound [frac_x4];
```

VIII．用于视频压缩和解压的标准

若干国际标准涉及视频压缩和解压。这些标准包括来自国际电信联盟[“ITU”]的运动图像专家组[“MPEG”]1、2 和 4 标准以及 H.261、H.262（MPEG 2 的另一种名称）、H.263 和 H.264（也称为 JVT/AVC）标准。这些标准指定了用于压缩的视频信息的视频解码器和格式方面。它们直接或隐含地也指定了某些编码器细节，但是未指定其它编码器细节。这些标准使用帧内和帧间解压和压缩的不同组合（或支持其使用）。

A．标准中的运动估计/补偿

国际标准中用于实现数字视频序列的数据压缩的一种主要方法是降低图像之间的时间冗余度。这些常用的压缩模式（MPEG-1、MPEG-2、MPEG-4、H.261、H.263 等）使用运动估计和补偿。例如，当前帧被划分成均匀的正方形区域（例如块和/或宏块）。每一当前区域的匹配区域是通过发送该区域的运动矢量信息来指定的。运动矢量指示了要用作当前区域的预测值的先前编码（和重构）的参考帧中该区域的位置。导出当前区域和参考帧中的区域之间的逐像素差（称为误差信号）。

该误差信号通常具有比原始信号更低的熵。因此，可以按更低的速率来编码信息。如在 WMV 8 和 9 编码器和解码器中，用于表示运动矢量信息的数据的压缩可通过对当前运动矢量和基于先前编码的相邻运动矢量的运动矢量预测值之间的差分进行编码来实现。

某些国际标准描述了隔行扫描视频帧中的运动估计和补偿。例如，MPEG-2 标准的章节 7.6.1 描述了双主（dual-prime）编码模式。在双主编码中，在比特流中仅与差分运动矢量一起编码一个运动矢量（以其全格式）。在半帧图像的情况下，从该信息中导出两个运动矢量。两个导出的运动矢量用于形成来自两个参考半帧（一个上半帧和一个下半帧）的预测，对这两个半帧求平均值以形成最终的预测。在帧图像的情况下，对两个半帧重复该过程，使得能够作出总共四个半帧预测。

MPEG-4 标准的 1998 年 5 月 28 日的委员会草稿描述了用于隔行扫描和逐行扫描视频的运动补偿。章节 7.5.5 描述了逐行扫描视频对象平面（VOP）中的运动补偿。候选运动矢量是从相邻的宏块或块中收集的。来自 VOP 外部的候选运动矢量作为无效来对待。如果仅一个候选运动矢量无效，则它被设为 0。如果仅两个无效，则它们被设为等于第三个运动矢量。如果三个都无效，则它们被设为 0。对这三个候选运动矢量执行中值滤波以计算预测值。

MPEG-4 标准的委员会草稿的章节 7.6.2 描述了用于隔行扫描视频的运动补偿。例如，章节 7.6.2.1 描述了对每一半帧有一个运动矢量的半帧预测的宏块，以及对每一块有一个运动矢量或对每一宏块有一个运动矢量的帧预测的宏块。候选运动矢量是从相邻的宏块或块收集的，其中预测值是通过中值滤波来选择的。

JVT/AVC 标准的草稿 JVT-d157 的章节 8.4 也描述了运动补偿。当前块的运动矢量的分量是使用中值预测来预测的。（预测不跨不属于同一片的宏块的边界发生。）首先，确定三个候选相邻块的运动矢量值和参考图像。如果右上角的块在当前图像或片的外部或由于解码顺序而不可用，则认为右上角的块的运动矢量和参考图像等于左上角的块的运动矢量和参考图像。如果顶部、右上和左上块都在当前图像或片的外部，则其运动矢量和参考图像被认为等于左块的运动矢量和参考图像。在其它情况下，帧内编码的或位于当前图像或片外部的候选预测块的运动矢量值被认为是 0，且参考图像被认为与当前块不同。

一旦确定了候选预测值的运动矢量值和参考图像，如果左侧、顶部和右上块中只有一个块具有与当前块相同的参考图像，则当前块的预测运动矢量等于具有相同的参考图像的块的运动矢量值。否则，当前块的预测的运动矢量值的每一分量被

计算为左侧、顶部和右上块的对应的候选运动矢量分量值的中值。

章节 8.4 也描述了宏块自适应帧/半帧编码。在隔行扫描帧中，宏块被组合成宏块对（顶部和底部）。宏块对可以是半帧编码或帧编码的。在帧编码的宏块对中，宏块对被解码为两个帧编码的宏块。在半帧编码的宏块对中，顶部的宏块由宏块对中的上半帧行构成，底部的宏块由宏块对中的下半帧行构成。如果当前块在帧编码模式中，则相邻块的候选运动矢量也是基于帧的。如果当前块在半帧编码模式中，相邻块的候选运动矢量在相同的半帧奇偶性中也是基于半帧的。

B．标准中用信号表示半帧或帧编码的宏块

某些国际标准描述了对隔行扫描图像中的宏块用信号表示半帧/帧编码类型（例如，半帧编码或帧编码）。

JVT/AVC 标准的草稿 JVT-d157 描述了 mb_field_decoding_flag 句法元素，它用于用信号表示隔行扫描 P 帧中宏块对是以帧模式还是半帧模式解码的。章节 7.3.4 描述了一种比特流句法，其中在序列参数（mb_frame_field_adaptive_flag）指示宏块中的帧和半帧解码之间的切换并且片头部元素（pic_structure）将图像结构标识为逐行扫描图像或隔行扫描帧图像的情况下，mb_field_decoding_flag 作为片数据的元素发送。

MPEG-4 的 1998 年 5 月 28 日的委员会草稿描述了 dct_type 句法元素，该元素用于用信号表示宏块是帧 DCT 编码的还是半帧 DCT 编码的。依照章节 6.2.7.3 和 6.3.7.3，dct_type 是仅存在于其中宏块具有非零编码的块模式或是帧内编码的隔行扫描内容中的 MPEG-4 比特流中的宏块层元素。

在 MPEG-2 中，dct_type 元素指示宏块是帧 DCT 编码的还是半帧 DCT 编码的。MPEG-2 也描述了图像编码扩展元素 frame_pred_frame_dct。当 frame_pred_frame_dct 被设为“1”时，在隔行扫描的帧中仅使用帧 DCT 编码。当 frame_pred_frame_dct = 1 且 dct_type 元素在比特流中不存在时，条件 dct_type = 0 是“导出的”。

C．标准中的跳过的宏块

某些国际标准使用跳过的宏块。例如，JVT/AVC 标准的草稿 JVT-d157 将跳过的宏块定义为“其数据除关于该宏块要被解码为‘跳过’的指示之外不被编码的宏块”。类似地，MPEG-4 的委员会草稿规定，“跳过的宏块是其信息不被发送的

宏块”。

D．标准中的色度运动矢量

国际标准中用于实现数字视频序列的数据压缩的一种主要方法是降低图像之间的时间冗余度。这些常见压缩模式 MPEG-1、MPEG-2、MPEG-4、H.261、H.263 等）使用运动估计和补偿。例如，当前帧被划分成亮度和色度信息的均匀正方形区域（例如，块和/或宏块）。每一当前区域的匹配区域是通过发送该区域的运动矢量信息来指定的。例如，亮度运动矢量指示要用作当前区域的预测值的先前编码（和重构）的帧中的亮度样值区域的位置。导出当前区域和参考帧中的区域之间的逐像素差，称为误差信号。该误差信号通常具有比原始信号更低的熵。因此，信息可以按更低的速率来编码。如在早先的 WMV 编码器和解码器中，由于运动矢量值通常与空间上的周围运动矢量相关，因此用于表示运动矢量信息的数据的压缩可通过对当前运动矢量和基于先前编码的相邻运动矢量的运动矢量预测值之间的差分进行编码来实现。通常，色度运动矢量是从亮度运动矢量中导出的，以避免与单独地计算和编码色度运动矢量相关联的额外开销。

某些国际标准描述了从亮度运动矢量中导出色度运动矢量。MPEG-2 标准的章节 7.6.3.7 描述了通过将水平和垂直亮度运动矢量分量的每一个除以 2 来适当地按比例缩放色度运动矢量分量，以 4:2:0 宏块格式从亮度运动矢量中导出色度运动矢量。在 4:2:2 格式中，色度信息仅在水平方向上二次采样，因此垂直分量不除以 2。在 4:4:4 格式中，色度信息以与亮度信息相同的分辨率采样，因此两个分量都不除以 2。

H.263 标准的附录 F 描述了对每一宏块使用四个运动矢量用于预测的高级预测模式。在该高级预测模式中，四个运动矢量中的每一个用于宏块中四个亮度块中的一个中的所有像素。两个色度块（采用 4:2:0 格式）的运动矢量是通过在每一方向上计算四个亮度运动矢量分量的和然后除以 8 来导出的。类似地，MPEG-4 标准的 1998 年 5 月 28 日的委员会草稿的章节 7.5.5 描述了通过计算对应于 K 个 8×8 的块的 K 个运动矢量的和，然后将该和除以 2×K 来导出采用 4:2:0 格式的两个色度块的运动矢量 MVD_{CHR}。对于色度的预测是通过将运动矢量 MVD_{CHR} 应用于两个色度块中的所有像素来获得的。

JVT/AVC 标准的草稿 JVT-d157 的章节 8.4.1.4 也描述了通过将水平和垂直亮度运动矢量分量的每一个除以 2 来从采用 4:2:0 宏块格式的亮度运动矢量中导出色

度运动矢量。草稿 JVT-d157 的章节 7.4.5 描述了具有不同亮度块大小（例如，16×16、16×8、8×16 和 8×8）和相关联的色度块的宏块。例如，对于 P 片和 SP 片，“对每一 N×M 的亮度块和相关联的色度块提供运动矢量”。

E. 标准的限制

这些国际标准在若干重要方面都有限制。例如，JVT/AVC 标准的草稿 JVT-d157 以及 MPEG-4 标准的委员会草稿描述了使用中值预测来计算运动矢量预测值，即使当一个或多个候选运动矢量被设为 0 时也是如此。当候选运动矢量被设为 0 时使用中值预测通常会产生歪斜的运动矢量预测值。这些标准也没有描述用四个编码的半帧运动矢量来预测宏块，这对于运动估计和补偿的空间自适应性施加了限制性的局限。此外，草稿 JVT-d157 通过使用宏块对而非通过个别的隔行扫描的宏块来执行隔行扫描编码和解码，这限制了图像内的半帧编码/帧编码的自适应性。

作为另一示例，尽管这些标准能够用信号表示宏块类型，然而半帧/帧编码类型信息是与运动补偿类型（例如，半帧预测或帧预测、一个运动矢量或多个运动矢量等）分离地用信号表示的。

作为另一示例，尽管某些国际标准允许通过跳过某些宏块来节省比特率，但是这些标准中的跳过宏块的条件仅指示对该宏块不再编码任何其它信息，且无法提供关于宏块的其它可能有价值的信息。

作为另一示例，若干标准使用不足以表示色度运动中的局部变化的色度运动矢量。另一问题是色度运动矢量导出中，尤其是对于半帧编码的内容的无效率的舍入机制。

给定视频压缩和解压对于数字视频的关键重要性，视频压缩和解压是丰富开发的领域并不令人惊奇。然而，不论早先的视频压缩和解压技术的好处如何，它们都没有以下技术和工具的优点。

发明内容

概括而言，该详细描述针对用于编码和解码视频帧的各种技术和工具。所描述的实施例实现了包括但不限于以下所描述的技术和工具中的一个或多个：

在一个方面，一种编码器/解码器接收运动矢量信息，该信息包括对于隔行扫描帧编码的前向预测图像（例如，隔行扫描 P 帧）中的宏块的四个半帧运动矢量的信息。该编码器/解码器使用四个半帧运动矢量处理该宏块。这四个半帧运动矢

量中的两个表示宏块中的第一个半帧的运动，而这四个半帧运动矢量中的另外两个表示该宏块中的第二个半帧的运动。关于这四个半帧运动矢量的信息可包括关于这四个半帧运动矢量的运动矢量差分。

在另一方面，一种编码器/解码器确定用于预测隔行扫描帧编码的图像（例如，隔行扫描的 P 帧）中的当前宏块（例如，2 半帧运动矢量宏块或 4 半帧运动矢量宏块）中的半帧运动矢量的多个有效候选运动矢量（例如，来自相邻宏块的帧或半帧运动矢量）。该编码器/解码器至少部分地基于有效候选运动矢量为半帧运动矢量计算运动矢量预测值。

计算包括基于有效候选运动矢量的数目是否为 3 来确定是否在有效候选运动矢量上执行中值运算（例如，分量级中值运算）。如果有效候选运动矢量的数目为 3，则计算还可包括在有效候选运动矢量上执行中值运算。如果数目小于 3，则计算可包括选择一个有效候选运动矢量作为运动矢量预测值，而无需执行中值运算。编码器/解码器可至少部分地基于该运动矢量预测值和运动矢量差分信息（可指示不存在差分）来重构半帧运动矢量。

有效候选运动矢量可以通过用于隔行扫描帧编码的图像内或与当前宏块相同的片内的块、宏块半帧、宏块半帧部分或宏块来表征，并且可以是实际的运动矢量值，而不是用于帧内编码块、宏块半帧、宏块半帧部分或宏块。

在另一方面，一种编码器/解码器确定用于对隔行扫描帧编码图像中的当前宏块的当前半帧预测半帧运动矢量的候选运动矢量，确定一个或多个候选运动矢量的半帧极性，以及至少部分地基于半帧极性计算半帧运动矢量的运动矢量预测值。候选运动矢量可以按四分之一像素增量来测量。半帧极性可以通过在候选运动矢量的 y 分量上执行逐位 AND（与）运算来确定。

确定候选运动矢量的半帧极性可包括确定候选运动矢量是否指示参考帧的上半帧或下半帧中的位移，或者候选运动矢量是否指示与当前半帧具有相同极性或相反极性的半帧中的位移。

在另一方面，一种编码器/解码器确定用于预测半帧运动矢量的一个或多个有效候选运动矢量，为一个或多个有效候选运动矢量中的每一个别的有效候选运动矢量确定半帧极性，根据其半帧极性将每一个别的有效候选运动矢量分配给两个集合中的一个，以及至少部分地基于这两个集合为半帧运动矢量计算运动矢量预测值。这两个集合可以是相反极性集和相同极性集。计算运动矢量预测值可包括选择主要极性有效候选运动矢量（例如，来自具有最大数目的有效候选运动矢量的集合的候

选运动矢量）。候选运动矢量可以按指定的选择顺序来选择。如果仅向每一集合分配了一个有效候选运动矢量，则计算运动矢量预测值可包括选择相同极性的候选运动矢量而非相反极性的候选运动矢量。如果三个有效候选运动矢量被分配给两个集合中的一个，则计算运动矢量预测值可包括在三个有效候选运动矢量上执行中值运算。

在另一方面，解码器对隔行扫描帧（例如，隔行扫描 P 帧、隔行扫描 B 帧或具有隔行扫描 P 半帧和/或隔行扫描 B 半帧的帧）的多个宏块中的一个或多个跳过宏块进行解码。这一个或多个跳过宏块中的每一个：（1）是由比特流中的跳过宏块信号指示的，（2）仅使用一个预测的运动矢量（例如，帧运动矢量）且没有运动矢量差分信息，以及（3）缺少残差信息。一个或多个跳过宏块中的每一个的跳过宏块信号指示对相应的跳过宏块的 1 运动矢量的经运动补偿的解码。跳过宏块信号可以是在具有多个层的比特流中的帧层发送的压缩位平面的一部分。或者，跳过宏块信号可以是在宏块层发送的个别比特。

在另一方面，从一组多个可用编码模式中选择一个编码模式，且依照所选择的编码模式在编码器或解码器中处理位平面。位平面包括用信号表示隔行扫描帧中的宏块是被跳过还是不被跳过的二进制信息。如果宏块只有一个运动矢量，则隔行扫描帧中的宏块被跳过，唯一的运动矢量等于为该宏块预测的运动矢量，且该宏块没有残留误差。如果宏块具有多个运动矢量，则它不被跳过。

在另一方面，编码器对隔行扫描 P 帧中的宏块选择运动补偿类型（例如，1MV、4 帧 MV、2 半帧 MV 或 4 半帧 MV），并为该宏块选择半帧/帧编码类型（例如，半帧编码、帧编码或未编码块）。编码器对该宏块的运动补偿类型和半帧/帧编码类型进行联合编码。编码器也可对该宏块的其它信息（例如，差分运动矢量的存在的指示符，诸如对于 1 运动矢量的宏块）与运动补偿类型和半帧/帧编码类型一起进行联合编码。

在另一方面，解码器接收隔行扫描 P 帧中的宏块的宏块信息，包括表示该宏块的运动补偿类型和半帧/帧编码类型的联合码。解码器对该联合码（例如，可变长度编代码表中的可变长度码）进行解码，以获得该宏块的运动补偿类型信息和半帧/帧编码类型信息。

在另一方面，解码器接收一个宏块的两个以上亮度运动矢量的亮度运动矢量信息，两个以上亮度运动矢量中的每一个与宏块（例如，4:2:0 宏块）的至少一部分相关联。解码器通过执行涉及亮度运动矢量信息上的舍入表（例如，基于半帧的

舍入表）的至少一个运算、维护宏块的色度运动矢量与亮度运动矢量的 1:1 的比率，对两个以上亮度运动矢量中的每一个导出与宏块的至少一部分相关联的色度运动矢量。例如，解码器接收宏块的四个亮度（帧或半帧）运动矢量，并导出该宏块的四个色度运动矢量。导出可包括使用基于半帧的舍入表对亮度运动矢量信息的至少一部分的二次采样和/或舍入。

在另一方面，解码器至少部分地基于一个或多个亮度运动矢量的运动矢量信息，对一个或多个亮度运动矢量中的每一个导出与隔行扫描帧编码的图像（例如，隔行扫描 P 帧、隔行扫描 B 帧）中的宏块的至少一部分相关联的色度运动矢量。解码器可用于对使用四个亮度半帧运动矢量预测的宏块进行解码。例如，解码器诸如通过将基于半帧的舍入查找表应用于亮度运动矢量信息的至少一部分来导出四个色度半帧运动矢量。

在另一方面，解码器通过使用基于半帧的舍入表（例如，整数数组）舍入亮度半帧运动矢量分量，以及对亮度半帧运动矢量分量进行二次采样，对一个或多个亮度半帧运动矢量中的每一个导出与宏块的至少一部分相关联的色度运动矢量。

各种技术和工具可组合或单独使用。

当参考附图阅读以下不同实施例的详细描述时，可以清楚其它特征和优点。

附图说明

图 1 是示出依照现有技术的视频编码器中的运动估计的图示。

图 2 是示出依照现有技术的视频编码器中的 8×8 的预测残差块的基于块的压缩的图示。

图 3 是示出依照现有技术的视频编码器中的 8×8 的预测残差块的基于块的解压的图示。

图 4 是示出依照现有技术的隔行扫描帧的图示。

图 5A 和 5B 是示出依照现有技术，用于逐行扫描 P 帧中的 1MV 宏块的候选运动矢量预测值的宏块位置的图示。

图 6A 和 6B 是示出依照现有技术，用于混合的 1MV/4MV 逐行扫描 P 帧中的 1MV 宏块的候选运动矢量预测值的块位置的图示。

图 7A、7B、8A、8B、9 和 10 是示出依照现有技术，用于混合的 1MV/4MV 逐行扫描 P 帧中的 4MV 宏块中的各种位置处的块的候选运动数量预测值的块位置的图示。

图 11 是示出依照现有技术，用于隔行扫描 P 帧中的当前帧编码宏块的候选运动矢量预测值的图示。

图 12 和 13 是示出依照现有技术，用于隔行扫描 P 帧中的当前半帧编码宏块的候选运动矢量预测值的图示。

图 14 是示出依照现有技术用于执行 3 候选者中值的伪代码的代码图。

图 15 是示出依照现有技术具有过去和未来参考帧的 B 帧的图示。

图 16 是示出依照现有技术的 4:2:0 YUV 采样网格的图示。

图 17 是示出依照现有技术，用于从 4MV 宏块的四个亮度块中的运动信息导出色度运动矢量的伪代码的代码图。

图 18 是可结合其实现所描述的若干实施例的合适的计算环境的框图。

图 19 是可结合其实现所描述的若干实施例的广义视频编码器系统的框图。

图 20 是可结合其实现所描述的若干实施例的广义视频解码器系统的框图。

图 21 是在所描述的若干实施例中使用的宏块格式的图示。

图 22A 是隔行扫描视频帧的一部分的图示，示出了上半帧和下半帧的交错行。图 22B 是为编码/解码组织为帧的隔行扫描视频帧的图示，图 22C 是为编码/解码组织为半帧的隔行扫描视频帧的图示。

图 23 是示出隔行扫描 P 帧的 2 半帧 MV 宏块中用于亮度块的运动矢量和用于色度块的导出运动矢量的图示。

图 24 是示出隔行扫描 P 帧的 4 帧 MV 宏块中，用于四个亮度块的每一个的不同运动矢量以及用于四个色度子块的每一个的导出运动矢量的图示。

图 25 是示出隔行扫描 P 帧的 4 半帧 MV 宏块中，用于亮度块的运动矢量和用于色度块的导出运动矢量的图示。

图 26A-26B 是示出用于隔行扫描 P 帧的当前宏块的候选预测值的图示。

图 27 是示出用于处理隔行扫描 P 帧中具有四个半帧运动矢量的宏块的技术的流程图。

图 28 是示出用于基于候选运动矢量的极性为半帧编码的宏块计算运动矢量预测值的技术的流程图。

图 29 是示出用于在为半帧运动矢量计算运动矢量预测值时确定是否执行中值运算的技术的流程图。

图 30 是示出用于确定在隔行扫描预测帧中是否跳过对特定宏块的编码的技术的流程图。

图 31 是示出用于对隔行扫描 P 帧中的宏块解码联合编码的运动补偿类型信息和半帧/帧编码类型信息的技术的流程图。

图 32 是示出用于为多个亮度运动矢量中的每一个导出一个色度运动矢量的技术的流程图。

图 33 是示出用于使用基于半帧的舍入查找表来导出色度半帧运动矢量的技术的流程图。

图 34 是示出用于使用基于半帧的舍入查找表从亮度运动矢量分量中导出色度运动矢量分量的伪代码的代码图。

图 35 是示出用于基于半帧的舍入查找表中的值的半帧指定的图示。

图 36 是示出组合的实现中的入口点层比特流句法的图示。

图 37 是示出组合实现中用于隔行扫描 P 帧的帧层比特流句法的图示。

图 38 是示出组合实现中用于隔行扫描 B 帧的帧层比特流句法的图示。

图 39 是示出组合实现中用于隔行扫描 P 半帧或 B 半帧的帧层比特流句法的图示。

图 40 是示出组合实现中用于隔行扫描 P 帧的宏块的宏块层比特流句法的图示。

图 41 是示出组合实现中用于为隔行扫描 P 帧中的 1MV 宏块收集候选运动矢量的伪代码的代码清单。

图 42、43、44 和 45 是示出组合实现中用于为隔行扫描 P 帧中的 4 帧 MV 宏块收集候选运动矢量的伪代码的代码清单。

图 46 和 47 是示出组合实现中用于为隔行扫描 P 帧中的 2 半帧 MV 宏块收集候选运动矢量的伪代码的代码清单。

图 48、49、50 和 51 是示出组合实现中用于为隔行扫描 P 帧中的 4 半帧 MV 宏块收集候选运动矢量的伪代码的代码清单。

图 52 是示出组合实现中用于为隔行扫描 P 帧中的帧运动矢量计算运动矢量预测值的伪代码的代码清单。

图 53 是示出组合实现中用于为隔行扫描 P 帧中的半帧运动矢量计算运动矢量预测值的伪代码的代码清单。

图 54A 和 54B 是示出组合实现中用于为隔行扫描 P 帧解码运动矢量差分的伪代码的代码清单。

图 55A-55C 是示出组合实现中用于普通-6 和差分-6 位平面编码模式的平铺块

的图示。

具体实施方式

本发明涉及用于隔行扫描和逐行扫描视频的有效压缩和解码的技术和工具。在所描述的各实施例中，视频编码器和解码器结合了用于对隔行扫描和逐行扫描的视频进行编码和解码的技术，以及用于包括不同层或级（例如，序列级、帧级、半帧级、宏块级和/或块级）的比特流格式或句法的对应的信号表示技术。

对此处所描述的实现的各种替换是可能的。例如，参考流程图所描述的技术可以通过改变流程图中所示的阶段的排序，通过重复或省略某些阶段等来改变。作为另一示例，尽管某些实现此处是参考特定宏块格式描述的，但是也可使用其它格式。此外，此处参考前向预测所描述的技术和工具也可适用于其它类型的预测。

各种技术和工具可组合或单独使用。不同的实施例实现所描述的技术和工具的一个或多个。此处所描述的某些技术和工具可以在视频编码器或解码器中使用，或在不特别地限于视频编码或解码的某一其它系统中使用。

I. 计算环境

图 18 示出了适合在其中实现所描述的若干实施例的计算环境 1800 的一个广义示例。计算环境 1800 并非对使用范围或功能提出任何局限，因为这些技术和工具可以在完全不同的通用或专用计算环境中实现。

参考图 18，计算环境 1800 包括至少一个处理单元 1810 和存储器 1820。在图 18 中，这一最基本配置 1830 包括在虚线内。处理单元 1810 执行计算机可执行指令，且可以是真实或虚拟处理器。在多处理系统中，多个处理单元执行计算机可执行指令以提高处理能力。存储器 1820 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）或两者的某一组合。存储器 1820 储存用此处所描述的一个或多个技术或工具实现视频编码器或解码器的软件 1880。

计算环境可具有额外的特征。例如，计算环境 1800 包括存储 1840、一个或多个输入设备 1850、一个或多个输出设备 1860 以及一个或多个通信连接 1870。诸如总线、控制器或网络等互连机制（未示出）将计算环境 1800 的组件互连。通常，操作系统软件（未示出）为在计算环境 1800 中执行的其它软件提供了操作环境，并协调计算环境 1800 的组件的活动。

存储1840可以是可移动或不可移动的，且包括磁盘、磁带或磁带盒、CD-ROM、DVD或可用于储存信息并可在计算环境1800内访问的任何其它介质。存储1840储存用于软件1880实现视频编码器或解码器的指令。

输入设备1850可以是诸如键盘、鼠标、笔或跟踪球等触摸输入设备、语音输入设备、扫描设备或可向计算环境1800提供输入的另一设备。对于音频或视频编码，输入设备1850可以是声卡、显卡、TV调谐卡、或接受模拟或数字格式的音频或视频输入的类似的设备、或将音频或视频样值读入计算环境1800的CD-ROM或CD-RW。输出设备1860可以是显示器、打印机、扬声器、CD刻录机、或从计算环境1800提供输出的另一设备。

通信连接1870允许通过通信介质到另一计算实体的通信。通信介质传达诸如计算机可执行指令、音频或视频输入或输出、或已调制数据信号形式的其它数据等信息。已调制数据信号是其一个或多个特征以在信号中编码信息的方式设置或改变的信号。作为示例而非局限，通信介质包括以电、光、RF、红外、声学或其它载波实现的有线或无线技术。

各种技术和工具可以在计算机可读介质的一般上下文中描述。计算机可读介质可以是可在计算环境内访问的任何可用介质。作为示例而非局限，对于计算环境1800，计算机可读介质包括存储器1820、存储1840、通信介质以及上述任一个的组合。

各种技术和工具可以在诸如程序模块中所包括的在计算环境中的目标真实或虚拟处理器上执行的计算机可执行指令的一般上下文中描述。一般而言，程序模块包括例程、程序、库、对象、类、组件、数据结构等，它们执行特定任务或实现特定抽象数据类型。程序模块的功能可以如各实施例中所需的组合或在程序模块之间分离。用于程序模块的计算机可执行指令可以在本地或分布式计算环境中执行。

为演示起见，详细描述使用了如“估计”、“补偿”、“预测”和“应用”等术语，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级抽象，且不应与人类所执行的动作混淆。对应于这些术语的实际的计算机操作取决于实现而不同。

II. 广义的视频编码器和解码器

图19是可结合其实现所描述的某些实施例的广义的视频编码器1900的框图。图20是可结合其实现所描述的某些实施例的广义的视频解码器2000的框图。

编码器 1900 和解码器 2000 内的模块之间所示的关系指示了编码器和解码器中的一般信息流；为简明起见，未示出其它关系。具体地，图 19 和 20 一般不示出指示用于视频序列、图像、宏块、块等的编码器设置、模式、表等辅助信息。这一辅助信息通常在该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。

编码器 1900 和解码器 2000 处理视频图像，视频图像可以是视频帧、视频半帧或帧和半帧的组合。图像和宏块级的比特流句法和语法可取决于使用了帧还是半帧。也可以对宏块组织和总体时序有改变。编码器 1900 和解码器 2000 是基于块的，且对帧使用 4:2:0 的宏块格式，其中每一宏块包括四个 8×8 的亮度块（有时候作为一个 16×16 的宏块来对待）以及两个 8×8 的色度块。对于半帧，可使用相同或不同的宏块组织和格式。8×8 的块还可在不同的级细分，例如在频率变换和熵编码级。示例性视频帧组织在以下更详细描述。或者，编码器 1900 和解码器 2000 可以是基于对象的，使用不同的宏块或块格式，或在与 8×8 的块和 16×16 的宏块不同的大小或配置的像素集上执行操作。

取决于所需的实现和压缩类型，编码器或解码器的模块可被添加、省略、分成多个模块、与其它模块组合、和/或用相似的模块来替代。在替换实施例中，具有不同模块和/或其它模块配置的编码器或解码器执行一个或多个所描述的技术。

A．视频帧组织

在某些实现中，编码器 1900 和解码器 2000 处理如下组织的视频帧。帧包含视频信号的空间信息行。对于逐行扫描视频，这些行包含从一个时刻开始并继续通过连续的行直到帧底部的样值。逐行扫描视频帧被划分成诸如图 21 所示的宏块 2100 等宏块。宏块 2100 包括四个 8×8 的亮度块（Y1 到 Y4）以及两个 8×8 的色度块，这些色度块与四个亮度块共同定位，但是水平和垂直分辨率都是一半，遵循常规的 4:2:0 的宏块格式。8×8 的块还可以在不同的级上细分，例如在频率变换级（例如，8×4、4×8 或 4×4 DCT）和熵编码级。逐行扫描 I 帧是帧内编码的逐行扫描视频帧。逐行扫描 P 帧是使用前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 帧和 B 帧可包括帧内编码的宏块以及不同类型的预测宏块。

隔行扫描视频帧由一帧的两次扫描构成——一次包括帧的偶数行（上半帧），另一次包括帧的奇数行（下半帧）。这两个半帧可表示两个不同的时间段，或者它

们可以来自同一时间段。图 22A 示出了隔行扫描视频帧 2200 的一部分，包括位于隔行扫描视频帧 2200 的左上部分的上半帧和下半帧的交替行。

图 22B 示出了为编码/解码组织为帧 2230 的图 22A 的隔行扫描视频帧 2200。隔行扫描视频帧 2200 被划分成诸如宏块 2231 和 2232 等宏块，它们使用如图 21 所示的 4:2:0 的格式。在亮度平面中，每一宏块 2231、2232 包括来自上半帧的 8 行，这 8 行与来自下半帧的 8 行交替，总共有 16 行，且每一行是 16 个像素长。（宏块 2231、2232 内亮度块和色度块的实际组织和布置未示出，且实际上可以对不同的编码决策不同。）在给定宏块内，上半帧信息和下半帧信息可以联合编码或在各种阶段的任一个单独编码。隔行扫描 I 帧是隔行扫描视频帧的两个帧内编码的半帧，其中宏块包括关于这两个半帧的信息。隔行扫描 P 帧是使用前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，其中宏块包括关于这两个半帧的信息。隔行扫描 P 帧和 B 帧可包括帧内编码宏块以及不同类型的预测宏块。隔行扫描 BI 帧是隔行扫描 I 帧和隔行扫描 B 帧的混合，它们是帧内编码的，但是不用作其它帧的定位帧。

图 22C 示出了为编码/解码被组织成半帧 2260 的图 22A 的隔行扫描视频帧 2200。隔行扫描视频帧 2200 的两个半帧中的每一个被划分成宏块。上半帧被划分成诸如宏块 2261 等宏块，下半帧被划分成诸如宏块 2262 等宏块。（这些宏块也使用如图 21 所示的 4:2:0 格式，且宏块内亮度块和色度块的组织和布置未示出）。在亮度平面中，宏块 2261 包括来自上半帧的 16 行，且宏块 2262 包括来自下半帧的 16 行，且每一行是 16 个像素长。隔行扫描 I 半帧是隔行扫描视频帧的单个单独表示的半帧。隔行扫描 P 半帧是使用前向预测编码的隔行扫描视频帧的单个单独表示的半帧，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单独表示的半帧。隔行扫描 P 半帧和 B 半帧可包括帧内编码宏块以及不同类型的预测宏块。隔行扫描 BI 半帧是隔行扫描 I 半帧和隔行扫描 B 半帧的混合；它们是帧内编码的，但是不用作其它半帧的定位帧。

为编码/解码组织为半帧的隔行扫描视频帧可包括不同半帧类型的各种组合。例如，这样的帧可在上半帧和下半帧中具有相同的半帧类型，或者在每一半帧中具有不同的半帧类型。在一个实现中，半帧类型的可能组合包括 I/I、I/P、P/I、P/P、B/B、B/BI、BI/B 以及 BI/BI。

术语图像一般指的是源、已编码的或已重构的图像数据。对于逐行扫描视频，图像是逐行扫描视频帧。对于隔行扫描视频，图像可以指的是隔行扫描视频帧、帧

的上半帧、或帧的下半帧，取决于上下文。

或者，编码器 1900 和解码器 2000 是基于对象的，使用不同的宏块或块格式，或对与 8×8 的块和 16×16 的宏块不同大小或配置的像素集执行操作。

B．视频编码器

图 19 是广义的视频编码器系统 1900 的框图。编码器系统 1900 接收包括当前图像 1905（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）的视频图像序列，并产生压缩的视频信息 1995 作为输出。视频编码器的具体实施例通常使用广义编码器 1900 的变化或补充版本。

编码器系统 1900 压缩预测图像和关键图像。为演示起见，图 19 示出了关键图像通过编码器系统 1900 的路径以及用于预测图像的路径。编码器系统 1900 的许多组件用于同时压缩关键图像和预测图像两者。由这些组件执行的确切操作可以取决于所压缩的信息类型而变化。

预测图像（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）按照来自一个或多个其它图像（通常被称为参考图像或定位帧）的预测（或差）来表示。预测残差是所预测的和原始图像之差。相反，关键图像（例如，逐行扫描 I 帧、隔行扫描 I 半帧或隔行扫描 I 帧）不参考其它图像来压缩。

如果当前图像 1905 是前向预测图像，则运动估计器 1910 估计当前图像 1905 的宏块或其它像素集相对于一个或多个参考图像（例如，缓冲在图像存储 1920 中的重构的前一图像 1925）的运动。如果当前图像 1905 是双向预测图像，则运动估计器 1910 估计当前图像 1905 中相对于多达四个重构的参考图像（例如，对于隔行扫描 B 半帧）的运动。通常，运动估计器估计 B 图像中相对于一个或多个在时间上先前的参考图像以及一个或多个在时间上未来的参考图像的运动。因此，编码器系统 1900 可使用单独的存储 1920 和 1922 用于多个参考图像。对于关于逐行扫描 B 帧和隔行扫描 B 帧和 B 半帧的更多信息，参见 2003 年 7 月 18 日提交的名为“Advanced Bi-Directional Predictive Coding of Video Frames（视频帧的高级双向预测编码）”的美国专利申请第 10/622,378 号，以及 2004 年 6 月 29 日提交的名为“Advanced Bi-Directional Predictive Coding of Interlaced Video（隔行扫描视频的高级双向预测编码）”的美国专利申请第 10/882,135 号。

运动估计器 1910 可按照像素、1/2 像素、1/4 像素或其它增量来估计运动，并可在逐图像的基础或其它基础上切换运动估计的分辨率。运动估计器 1910（和补

偿器 1930）也可在每一帧或其它基础上在参考图像像素内插的类型之间切换（例如，在双三次内插和双线性内插之间）。运动估计的分辨率可以在水平和垂直上相同或不同。运动估计器 1910 输出运动信息 1915，诸如差分运动矢量信息作为辅助信息。编码器 1900 通过例如为运动矢量计算一个或多个预测值，计算运动矢量和预测值之间的差分，以及对差分进行熵编码，来对运动信息 1915 进行编码。为重构运动矢量，运动补偿器 1930 将预测值与差分运动矢量信息组合。以下描述用于计算运动矢量预测值、计算差分运动矢量、以及重构运动矢量的各种技术。

运动补偿器 1930 将重构的运动矢量应用于重构的图像 1925，以形成经运动补偿的当前图像 1935。然而，预测很少是完美的，且经运动补偿的当前图像 1935 和原始的当前图像 1905 之间的差异是预测残差 1945。在稍后的图像重构期间，将预测残差 1945 加到经运动补偿的当前图像 1935，以获得更接近于原始的当前图像 1905 的经重构的图像。然而，在有损压缩中，某些信息仍从原始当前图像 1905 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。

频率变换器 1960 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像，频率变换器 1960 向像素数据或预测残差数据的块应用 DCT、DCT 的变体或其它块变换，从而产生频率变换系数块。或者，频率变换器 1960 应用诸如傅立叶变换等另一常规频率变换或使用小波或子带分析。频率变换器 1960 可应用 8×8、8×4、4×8、4×4 或其它大小的频率变换。

量化器 1970 然后量化频谱数据系数块。量化器向频谱数据应用均匀的标量量化，其步长在逐图像的基础或其它基础上变化。或者，量化器向频谱数据系数应用另一类型的量化，例如非均匀的、矢量或非自适应量化，或直接在不使用频率变换的编码器系统中量化空间域数据。除自适应量化之外，编码器 1900 可使用帧丢弃、自适应滤波或其它技术用于速率控制。

编码器 1900 可对跳过宏块（它是没有某些类型的信息的宏块）使用特殊的信号表示。跳过宏块在以下更详细描述。

当需要重构的当前图像用于后续的运动估计/补偿时，反量化器 1976 在量化的频谱数据系数上执行反量化。反频率变换器 1966 然后执行频率变换器 1960 的逆运算，从而产生重构的预测残差（对于预测图像）或重构的关键图像。如果当前图像 1905 是关键图像，则重构的关键图像用作重构的当前图像（未示出）。如果当前图像 1905 是预测图像，则重构的预测残差被加到经运动补偿的当前图像 1935，以形成重构的当前图像。图像存储 1920、1922 之一或两者缓冲重构的当前图像，以

在经运动补偿的预测中使用。在某些实施例中，编码器向重构的帧应用分块滤波器，以自适应地平滑图像中的不连续性和其它人为因素。

熵编码器 1980 压缩量化器 1970 的输出以及某些辅助信息（例如，运动信息 1915、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、行程长度编码、LZ 编码、字典式编码以及上述的组合。熵编码器 1980 通常对不同种类的信息（例如，DC 系数、AC 系数、不同种类的辅助信息）使用不同的编码技术，并可从特定编码技术内的多个代码表中进行选择。

熵编码器 1980 向多路复用器[“MUX”]1990 提供压缩的视频信息 1995。MUX 1990 可包括缓冲器，且可将缓冲器级别指示符反馈给比特率自适应模块用于速率控制。在 MUX 1990 之前或之后，压缩的视频信息 1995 可被信道编码用于通过网络发送。信道编码可向压缩的视频信息 1995 应用检错和纠错数据。

C．视频解码器

图 20 是通用视频解码器系统 2000 的框图。解码器系统 2000 接收关于压缩的视频图像序列的信息 2095，并产生包括重构的图像 2005（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）的输出。视频解码器的具体实施例通常使用广义解码器 2000 的变体或补充版本。

解码器系统 2000 解压预测图像和关键图像。为演示起见，图 20 示出了关键图像通过解码器系统 2000 的路径以及用于前向预测图像的路径。解码器系统 2000 的许多组件用于解压关键图像和预测图像。由这些组件执行的确切操作可以取决于所解压的信息类型而变化。

DEMUX 2090 接收关于压缩的视频序列的信息 2095，并使得所接收的信息对熵解码器 2080 可用。DEMUX 2090 可包括抖动缓冲器以及其它缓冲器。在 DEMUX 2090 之前或之后，压缩的视频信息可以被信道解码，并被处理用于检错和纠错。

熵解码器 2080 对熵编码的量化数据以及熵编码的辅助信息（例如，运动信息 2015、量化步长）进行解码，通常应用编码器中执行的熵编码的逆运算。熵解码技术包括算术解码、差分解码、哈夫曼解码、行程长度解码、LZ 解码、字典式解码以及上述的组合。熵解码器 2080 通常对不同种类的信息（例如，DC 系数、AC 系数、不同种类的辅助信息）使用不同的解码技术，并可从特定的解码技术中的多个代码表之中进行选择。

解码器 2000 通过例如为运动矢量计算一个或多个预测值、对差分运动矢量进

行熵解码、以及将解码的差分运动矢量与预测值组合以重构运动矢量，来对运动信息 2015 进行解码。

运动补偿器 2030 向一个或多个参考图像 2025 应用运动信息 2015，以形成所重构的图像 2005 的预测 2035。例如，运动补偿器 2030 使用一个或多个宏块运动矢量以找出参考图像 2025 中的宏块。一个或多个图像存储（例如，图像存储 2020、2022）储存先前重构的图像以用作参考图像。通常，B 图像具有一个以上参考图像（例如，至少一个时间上先前的参考图像以及至少一个时间上未来的参考图像）。因此，解码器系统 2000 可对多个参考图像使用单独的图像存储 2020 和 2022。运动补偿器 2030 可以按像素、1/2 像素、1/4 像素或其它增量来补偿运动，并可在逐图像的基础或其它基础上切换运动补偿的分辨率。运动补偿器 2030 也可在每一帧或其它基础上在参考图像像素内插的类型之间（例如，在双三次内插和双线性内插之间）切换。运动补偿的分辨率可以在水平和垂直上相同或不同。或者，运动补偿器应用另一类型的运动补偿。运动补偿器的预测很少是完美的，因此解码器 2000 也重构预测残差。

反量化器 2070 对熵解码的数据进行反量化。一般而言，反量化器向熵解码的数据应用均匀的标量反量化，其中步长在逐图像的基础或其它基础上变化。或者，反量化器向数据应用另一类型的反量化，例如用于在非均匀矢量量化或非自适应量化之后重构，或直接在不使用反频率变换的解码器系统中对空间域数据进行反量化。

反频率变换器 2060 将量化的频域数据转换成空间域视频信息。对于基于块的视频图像，反频率变换器 2060 向频率变换系数块应用反 DCT[“IDCT”]、IDCT 的变体或其它反块变换，从而分别对关键图像或预测图像产生像素数据或预测残差数据。或者，反频率变换器 2060 应用另一常规的反频率变换，诸如傅立叶反变换或使用小波或子带合成。反频率变换器 2060 可应用 8×8、8×4、4×8、4×4 或其它大小的反频率变换。

对于预测图像，解码器 2000 将重构的预测残差 2045 与经运动补偿的预测 2035 组合，以形成重构的图像 2005。当解码器需要重构的图像 2005 用于后续的运动补偿时，图像存储（例如，图像存储 2020）之一或两者缓冲重构的图像 2005 以供预测下一图像时使用。在某些实施例中，解码器 2000 向重构的图像应用分块滤波器，以自适应地平滑图像中的不连续性以及其它人为因素。

III. 隔行扫描 P 帧

典型的隔行扫描视频帧由在不同的时刻扫描的两个半帧（例如，上半帧和下半帧）构成。一般而言，通过将两个半帧一起编码（“帧模式”编码）来对隔行扫描视频帧的静止区域进行编码是更有效的。另一方面，通过单独对各半帧编码（“半帧模式”编码）来对隔行扫描视频帧的运动区域进行编码通常是更有效的，因为两个半帧往往具有不同的运动。前向预测的隔行扫描视频帧可以被编码为两个单独的前向预测的半帧－隔行扫描 P 半帧。对前向预测的隔行扫描视频帧单独地编码半帧在例如贯穿该隔行扫描视频帧中有高运动，且因此在半帧之间有较多差异时可能是有效的。隔行扫描的 P 半帧参考一个或多个先前解码的半帧。例如，在某些实现中，隔行扫描的 P 半帧参考一个或两个先前解码的半帧。对于关于隔行扫描 P 半帧的更多信息，参见 2003 年 9 月 7 日提交的名为“Video Encoding and Decoding Tools and Techniques（视频编码和解码工具和技术）”的美国临时专利申请第 60/501,081 号，以及 2004 年 5 月 27 日提交的名为“Predicting Motion Vectors for Fields of Forward-predicted Interlaced Video Frames（为前向预测的隔行扫描视频帧的半帧预测运动矢量）”的美国专利申请第 10/857,473 号。

或者，前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，作为隔行扫描的 P 帧。对于隔行扫描 P 帧的宏块，宏块包括上半帧和下半帧的像素行，且这些行可以在帧编码模式中共同编码或者在半帧编码模式中单独编码。

A. 隔行扫描 P 帧中的宏块类型。

在某些实现中，隔行扫描 P 帧中的宏块可以是以下五种类型之一：1MV、2 半帧 MV、4 帧 MV、4 半帧 MV 以及帧内编码。

在 1MV 宏块中，宏块中四个亮度块的位移由单个运动矢量来表示。对应的色度运动矢量可以从亮度运动矢量中导出以表示该运动矢量的两个 8×8 的色度块中的每一个的位移。例如，再次参考图 21 中所示的宏块的排列，1MV 宏块 2100 包括四个 8×8 的亮度块以及两个 8×8 的色度块。亮度块（Y1 到 Y4）的位移是由单个运动矢量来表示的，而对应的色度运动矢量可以从亮度运动矢量中导出，以表示两个色度块（U 和 V）中的每一个的位移。

在 2 半帧 MV 宏块中，宏块中 16×16 的亮度分量的每一半帧的位移是由不同的运动矢量来描述的。例如，图 23 示出了上半帧运动矢量描述了亮度分量的偶数行的位移，而下半帧运动矢量描述了亮度分量的奇数行的位移。使用上半帧运动矢

量，编码器可导出描述色度快的偶数行的位移的对应的上半帧色度运动矢量。类似地，编码器可导出描述色度块的奇数行的位移的下半帧色度运动矢量。

参考图 24，在 4 帧 MV 宏块中，四个亮度块中的每一个的位移是由不同的运动矢量（MV1、MV2、MV3 和 MV4）来描述的。每一色度块可以通过使用描述四个 4×4 色度子块的位移的四个导出的色度运动矢量（MV1'、MV2'、MV3'和 MV4'）来运动补偿。用于每一 4×4 色度子块的运动矢量可以从用于空间上对应的亮度块的运动矢量中导出。

参考图 25，在 4 半帧 MV 宏块中，16×16 亮度分量中的每一半帧的位移是由两个不同的运动矢量来描述的。亮度分量的行被垂直地细分以形成两个 8×16 的区域，其每一个由与 8×8 的奇数行区域交错的 8×8 的偶数行区域构成。对于偶数行，左侧的 8×8 区域的位移是由左上半帧块运动矢量来描述的，而右侧的 8×8 区域的位移是由右上半帧块运动矢量来描述的。对于奇数行，左侧的 8×8 区域的位移是由左下半帧块运动矢量来描述的，而右侧的 8×8 区域的位移是由右下半帧块运动矢量来描述的。每一色度块也可被划分成四个区域，而每一色度块区域可以使用导出的运动矢量来运动补偿。

对于帧内编码宏块，运动被假定为零。

B．计算隔行扫描 P 帧中的运动矢量预测值

一般而言，为隔行扫描 P 帧中的当前宏块计算运动矢量预测值的过程由两个步骤构成。首先，从其相邻宏块中收集当前宏块的多达三个候选运动矢量。例如，在一个实现中，基于图 26A-26B 所示的排列（以及对于顶行宏块的各种特殊情况等）收集候选运动矢量。或者，候选运动矢量可以按某一其它顺序或排列收集。其次，从候选运动矢量集中计算当前宏块的运动矢量预测值。例如，预测值可使用 3 预测中值或某一其它方法来计算。

IV．隔行扫描 P 帧中的运动矢量预测的革新

运动矢量预测的过程可以在概念上分成两个步骤。首先，从相邻块中收集候选运动矢量集，且如果适当，根据用于当前运动矢量的预测类型将其转换成适当的类型。然后，从候选运动矢量集中导出运动矢量预测值。如下更详细描述的，当前运动矢量预测值可以用不同的方式从候选运动矢量中导出，诸如通过在一组完全的有效候选运动矢量上执行中值运算，通过当一组完全的有效候选运动矢量不可用时

仅选择一个候选运动矢量，或通过某一其它方法。

如上文在第 III 节中所解释的，隔行扫描 P 帧中的每一宏块可以使用 1 帧运动矢量、4 帧运动矢量、2 半帧运动矢量或 4 半帧运动矢量来运动补偿。每一半帧运动矢量可以指的是参考帧的上半帧或下半帧，而与对方应用哪一半帧无关。在某些实现中，运动矢量预测考虑若干因素，诸如相邻的宏块或块的运动矢量和运动预测类型、当前运动矢量类型等，以为当前宏块（或其块或半帧）的当前运动矢量导出运动矢量预测值。

所描述的实施例实现了所描述的一个或多个技术和工具，用于预测、编码和解码隔行扫描帧编码的图像中的运动矢量，包括但不限于：

1. 使用 4 半帧运动矢量预测隔行扫描 P 帧中的宏块（例如，与其它诸如 1MV、4 帧 MV、2 半帧 MV 和/或帧内编码等其它宏块类型相结合。）
2. 通过将中值运算的使用限于所有三个候选相邻运动矢量都可用的情况，并在其它情况下以预先指定的顺序取相邻运动矢量之一，来改进运动矢量预测。
3. 对于半帧运动矢量，考虑合法候选运动矢量集合中的半帧极性。

所描述的技术和工具可彼此结合或与其它技术和工具结合使用，或可单独使用。

A．隔行扫描 P 帧中的 4 半帧 MV 宏块

在某些实现中，编码器/解码器处理隔行扫描 P 帧中具有四个半帧运动矢量的宏块（例如，4 半帧 MV 宏块）。4 半帧 MV 宏块提供了对运动的改进的空间自适应性，用于隔行扫描 P 帧中的半帧编码宏块的运动估计和补偿。

例如，图 27 示出了用于处理隔行扫描 P 帧中具有四个半帧运动矢量的宏块的技术 2700。在 2710，编码器/解码器接收隔行扫描 P 帧中的宏块的四个半帧运动矢量的运动矢量信息（例如，运动矢量差分、运动矢量值）。然后，在 2720，编码器/解码器使用四个半帧运动矢量来处理这些宏块（例如，通过重构宏块）。

或者，编码器/解码器在没有 4 半帧 MV 宏块的隔行扫描 P 帧中执行运动估计/补偿。

B．收集候选运动矢量

在某些实现中，候选运动矢量的收集顺序是重要的。例如，隔行扫描 P 帧中

当前宏块的三个候选运动矢量是从其相邻宏块中收集的，如图 26A-26B 所示。在一个实现中，收集顺序从相邻宏块 A 开始，前进到宏块 B，然后在宏块 C 结束。

图 41-51 中的伪代码描述了在一个实现中如何收集候选运动矢量。编码器/解码器检查相邻宏块以确定它们是否“存在”（即，有效），用于为当前宏块（包括为块、宏块的半帧或宏块的半个半帧）预测运动矢量。如果对应的宏块/块位于帧边界之外，或者如果对应的宏块/块是不同片的一部分，则候选预测值被认为是不存在的（即，无效）。由此，不跨片边界来执行运动矢量预测。

如图 41-51 所示，如果相邻宏块有效且不是帧内编码的，则将来自相邻宏块的运动矢量添加到候选运动矢量集。添加到候选运动矢量集的实际运动矢量取决于相邻宏块相对于当前宏块的位置（例如，位置 A、B 或 C），且取决于相邻宏块和当前宏块的类型（例如，1MV、4 帧 MV、2 半帧 MV 或 4 半帧 MV）。

图 41 中的伪代码 4100 示出在一个实现中如何为 1MV 宏块中的当前运动矢量收集多达三个候选运动矢量。

图 42、43、44 和 45 中的伪代码 4200、4300、4400 和 4500 分别示出在一个实现中，如何为当前 4 帧 MV 宏块中的四个帧块运动矢量中的每一个收集来自相邻宏块/块的候选运动矢量。在这一实现中，编码器/解码器使用伪代码 4200、4300、4400 和 4500 中所示的算法，以分别为左上帧块运动矢量、右上帧块运动矢量、左下帧块运动矢量和右下帧块运动矢量收集多达三个候选运动矢量。

图 46 和 47 中的伪代码 4600 和 4700 分别示出在一个实现中如何为当前 2 半帧 MV 宏块中的两个半帧运动矢量中的每一个收集来自相邻宏块的候选运动矢量。在这一实现中，编码器/解码器适用伪代码 4600 和 4700 中所示的算法来分别为上半帧运动矢量和下半帧运动矢量收集多达三个候选运动矢量。

图 48、49、50 和 51 中的伪代码 4800、4900、5000 和 5100 分别示出在一个实现中，如何为当前 4 半帧 MV 宏块中的四个半帧块运动矢量中的每一个收集来自相邻宏块/块的候选运动矢量。在这一实现中，编码器/解码器使用伪代码 4800、4900、5000 和 5100 中所示的算法来分别为左上半帧块运动矢量、右上半帧块运动矢量、左下半帧块运动矢量和右下半帧块运动矢量收集多达三个候选运动矢量。

在某些情况下，所选择的候选运动矢量实际上是相邻宏块中的运动矢量的平均值。在一个实现中，给定两个半帧运动矢量(MVX_1, MVY_1)和(MVX_2, MVY_2)，则用于形成候选帧运动矢量(MVX_A, MVY_A)的求平均运算为：

$$MVX_A = (MVX_1 + MVX_2 + 1) >> 1;$$

$MVY_A = (MVY_1 + MVY_2 + 1) >> 1;$

或者，候选运动矢量可以依照与图 41-51 中所示的不同的收集规则来收集。例如，从具有一个以上运动矢量的相邻宏块（例如，4 帧 MV、2 半帧 MV 或 4 半帧 MV 宏块）中选择为候选运动矢量的特定运动矢量可以变化。作为另一示例，候选运动矢量集中的运动矢量的顺序可以从所描述的 A-B-C 顺序调整为某一其它顺序。

C．从候选运动矢量中计算帧 MV 预测值

图 52 中的伪代码 5200 描述了一个实现中如何为帧运动矢量计算运动矢量预测值(PMV_x, PMV_y)。在伪代码 5200 中，TotalValidMV（总有效 MV）表示候选运动收矢量集中的有效运动矢量的总数（TotalValidMV = 0、1、2 或 3），而 ValidMV（有效 MV）数组包括候选运动矢量集中的有效运动矢量。

在一个实现中，伪代码 5200 用于为 1MV 宏块中的运动矢量以及 4 帧 MV 宏块中四个帧块运动矢量中的每一个从候选运动矢量集中计算预测值。

D．从候选运动矢量中计算半帧 MV 预测值

本节描述了用于在给定候选运动矢量集时为半帧运动矢量计算运动矢量预测值的技术和工具。本节中所描述的技术和工具可用于为两个半帧运动矢量（在 2 半帧 MV 宏块中）中的每一个以及四个半帧运动矢量（在 4 半帧 MV 宏块中）中的每一个计算预测值。

在某些实现中，编码器/解码器在计算预测值时考虑由候选运动矢量参考的半帧的极性。图 28 示出了用于基于候选运动矢量的极性为半帧编码的宏块计算运动矢量预测值的技术 2800。在 2810，编码器/解码器确定用于为当前宏块的当前半帧预测半帧运动矢量的候选运动矢量。在 2820，编码器/解码器确定一个或多个有效候选运动矢量的半帧极性。然后，在 2830，编码器/解码器至少部分地基于一个或多个有效候选运动矢量的半帧极性为半帧运动矢量计算运动矢量预测值。例如，在一个实现中，候选运动矢量被分成两个集合，其中一个集合仅包含指向与当前半帧相同的半帧的运动矢量，而另一集合包含指向相反的半帧的运动矢量。然后，编码器/解码器使用这两个集合之一来计算运动矢量预测值。尽管未在图 28 中示出，但为当前宏块计算运动矢量预测值的过程可对宏块的每一运动矢量以及帧中的其它宏块重复。

假定运动矢量是以四分之一像素的单位来表示的，编码器/解码器可通过对其

y 分量执行以下检查来检查候选运动矢量是指向相同的半帧还是相反的半帧（相对于当前半帧的极性）：

if($ValidMV_y$ &4) {

ValidMV 指向相反的半帧。

} else{

ValidMV 指向相同的半帧。

}

在以上伪代码中，$ValidMV_y$ 是以四分之一像素的增量所测得的候选运动矢量的 y 分量。因此，在二进制中，$ValidMV_y$ 的最低位是四分之一像素位，第二最低位是半像素位，第三最低位是全像素位。因此，逐位 AND 运算($ValidMV_y$ & 4)确定全像素位是 1（指示奇数）还是 0（指示偶数）。如果整数偏移量为奇数，则候选运动矢量参考了参考帧中相反极性的半帧。如果整数偏移量是偶数，则候选运动矢量参考了参考帧中相同极性的半帧。

或者，候选运动矢量的极性可以按某一其它方式来确定，或者在计算运动矢量预测值的过程中可以不考虑候选运动矢量的极性。

图 53 中的伪代码 5300 描述了一个实现中如何为半帧运动矢量计算运动矢量预测值(PMV_x, PMV_y)。在伪代码 5300 中，SameFieldMV[ ]（相同半帧 MV）和 OppFieldMV[ ]（相反半帧 MV）表示两个候选运动矢量集，而 NumSameFieldMV（相同半帧 MV 数）和 NumOppFieldMV（相反半帧 MV 数）表示属于每一集合的候选运动矢量的数目。在伪代码 5300 所示的示例中，这些集合中可用候选运动矢量的数目确定了是否使用中值运算来计算运动矢量预测值。

图 29 示出了用于确定在为半帧运动矢量计算运动矢量预测值时是否执行中值运算的技术 2900。在 2910，编码器/解码器确定用于预测当前宏块中的半帧运动矢量的有效候选运动矢量的数目。在 2920，如果有三个有效候选运动矢量，则在 2930，可在计算预测值过程中使用中值运算。在 2940，如果没有三个有效候选运动矢量（即，有两个或更少的有效候选运动矢量），则使用另一方法从可用有效候选运动矢量中选择一个运动矢量预测值，而不使用中值运算。

在伪代码 5300 的示例中，编码器/解码器通过在所有三个候选运动矢量有效时，以及在所有三个有效候选运动矢量都是相同极性时，对候选运动矢量的 x 分量或 y 分量执行中值运算（例如，median3），来导出运动矢量预测值。如果所有三个候选运动矢量都有效，但是并非它们全部都是相同的极性，则编码器/解码器选

择具有最多候选运动矢量的集合来导出运动矢量预测值。在两个集合具有相同数目的候选运动矢量的情况下，编码器/解码器使用集合 SameFieldMV[ ]。对于有三个以下候选运动矢量的情况，编码器/解码器以预先指定的顺序从所选择的集合中选择第一个候选运动矢量。在该示例中，每一集合中的候选运动矢量的顺序从候选运动矢量 A（如果存在）开始，接着是候选运动矢量 B（如果存在），然后是候选运动矢量 C（如果存在）。例如，如果编码器/解码器使用集合 SameFieldMV[ ]，且如果集合 SameFieldMV[ ]仅包含候选运动矢量 B 和候选运动矢量 C，则编码器/解码器使用候选运动矢量 B 作为运动矢量预测值。如果有效候选运动矢量的数目为零，则编码器/解码器将预测值设置为(0, 0)。

或者，运动矢量预测值可以按除上述以外的方式来计算。例如，尽管伪代码 5300 包括对于选择相同半帧候选运动矢量的偏差，然而可以调整预测值的计算以去除该偏差或使用相反半帧的偏差。作为另一示例，可以在更多情况下使用中值运算（例如，当存在两个有效候选半帧运动矢量时），或完全不使用中值运算，或者对于两个有效候选运动矢量的非中值运算可以是求平均运算。作为又一示例，集合中候选运动矢量的顺序可被调整，或者候选运动矢量可被全部储存在一个集合中。

V. 用于隔行扫描帧编码图像的宏块信息信号表示的革新

所描述的实施例包括用于用信号表示隔行扫描帧编码图像（例如，隔行扫描 P 帧、隔行扫描 B 帧等）的宏块信息的技术和工具。例如，所描述的技术和工具包括用于用信号表示隔行扫描 P 帧的宏块信息的技术和工具，以及用于使用和用信号表示隔行扫描 P 帧和其它隔行扫描图像（例如，隔行扫描 B 帧、隔行扫描 P 半帧、隔行扫描 B 半帧等）中的跳过宏块的技术和工具。所描述的实施例实现了包括但不限于以下的所描述的技术和工具中的一个或多个：

1. 用对于隔行扫描 P 帧的半帧/帧编码类型信息（例如，使用宏块级句法元素 MBMODE），对运动补偿类型（例如，1 帧 MV、4 帧 MV、2 半帧 MV、4 半帧 MV 等）以及可能的其它信息进行联合编码。
2. 用信号表示宏块跳过条件。信号表示可以与诸如 MBMODE 等其它句法元素分离地执行。跳过条件指示宏块是 1MV 宏块，具有零差分运动矢量，以及没有编码的块。跳过信息可以在压缩位平面中编码。

所描述的技术和工具可彼此结合或与其它技术和工具结合使用，或可单独使用。

A．跳过宏块的信号表示

在某些实现中，编码器用信号表示跳过的宏块。例如，当宏块用一个运动矢量来编码，具有零运动矢量差分以及没有编码的块（即，没有任何块的残差）时，编码器用信号表示隔行扫描帧中跳过的宏块。跳过信息可被编码为压缩位平面（例如，在帧级），或者可在每一宏块的基础上在一个比特上用信号表示（例如，在宏块级）。对宏块的跳过条件的信号表示与对宏块的宏块模式的信号表示分离。解码器执行对应的解码。

对跳过宏块的这一定义利用了这样一个观察结果：当使用一个以上运动矢量来编码宏块时，由于不可能所有的运动矢量差分都为零且所有的块都不被编码，因此该宏块很少被跳过。由此，当宏块用信号表示为被跳过时，从跳过条件中隐含了宏块模式（1MV），则无需为该宏块发送该模式。在隔行扫描 P 帧中，1MV 宏块是用一个帧运动矢量来运动补偿的。

图 30 示出了用于确定是否跳过隔行扫描预测帧（例如，隔行扫描 P 帧、隔行扫描 B 帧、或包括隔行扫描 P 半帧和/或隔行扫描 B 半帧的帧）中特定宏块的编码的技术 3000。对于给定的宏块，编码器在 3010 检查该宏块是否为 1MV 宏块。在 3020，如果宏块不是 1MV 宏块，则编码器不跳过该宏块。否则，在 3030，编码器检查该宏块的一个运动矢量是否等于其因果预测的运动矢量（例如，该宏块的差分运动矢量是否等于零）。在 3040，如果宏块的运动矢量不等于因果预测的运动，则编码器不跳过该宏块。否则，在 3050，编码器检查对该宏块的块是否还存在任何残差要编码。在 3060，如果存在残差要编码，则编码器不跳过该宏块。在 3070，如果对该宏块的块没有残差，则编码器跳过该宏块。在 3080，编码器可继续编码或跳过宏块，直到编码完成。

在一个实现中，宏块级 SKIPMBBIT 字段（也可被标记为 SKIPMB 等）指示宏块的跳过条件。如果 SKIPMBBIT 字段为 1，则当前宏块被跳过，且在 SKIPMBBIT 字段之后不发送任何其它信息。另一方面，如果 SKIPMBBIT 字段不是 1，则解码 MBMODE 字段以指示宏块的类型以及关于当前宏块的其它信息，诸如以下在第 V.B 节中描述的信息。

在帧级，SKIPMB 字段指示帧中宏块的跳过信息。在一个实现中，跳过信息可以在若干中模式之一中编码。例如，在原始编码模式中，SKIPMB 字段指示在宏块级 SKIPMBBIT 的存在。在位平面编码模式中，SKIPMB 字段将跳过信息储存在压缩的位平面中。可用的位平面编码模式包括普通-2 模式、差分-2 模式、普通-6

模式、差分-6 模式、行跳过模式和列跳过模式。位平面编码模式在以下第 VII 节中更详细描述。解码的 SKIPMB 位平面对每一宏块包含一个比特，且指示每一相应宏块的跳过条件。

或者，跳过的宏块以某一其它方式或在比特流中的某一其它级上用信号表示。例如，在半帧级发送压缩的位平面。作为另一替换，跳过条件可被定义成隐含与上述信息不同和/或除上述信息之外的关于跳过宏块的信息。

B．宏块模式的信号表示

在某些实现中，编码器用宏块的半帧/帧编码类型信息，对关于宏块的运动补偿类型和可能的其它信息进行联合编码。例如，编码器使用一个或多个可变长度编代码表，对五种运动补偿类型（1MV、4 帧 MV、2 半帧 MV、4 半帧 MV 和帧内编码）之一与半帧变换/帧变换/没有编码的块事件一起进行联合编码。解码器执行对应的解码。

关于宏块的联合编码运动补偿类型和半帧/帧编码类型信息利用了这样一个观察结果：某些半帧/帧编码类型更有可能出现在给定运动补偿类型的宏块的某些上下文中。然后可使用可变长度编码来向运动补偿类型和半帧/帧编码类型的更可能的组合分配较短的码。对于更进一步的灵活性，可使用多个可变长度编代码表，并且编码器可根据情况在表之间切换。由此，对宏块的运动补偿类型和半帧/帧编码类型信息进行联合编码可提供编码额外开销中的节省，否则这些开销将用于对每一宏块单独地用信号表示半帧/帧编码类型。

例如，在某些实现中，编码器为宏块选择一个运动补偿类型（例如，1MV、4 帧 MV、2 半帧 MV 或 4 半帧 MV）以及半帧/帧编码类型（例如，半帧、帧或没有编码的块）。编码器对该宏块的运动补偿类型和半帧/帧编码类型进行联合编码。编码器也可将其它信息与运动补偿类型和半帧/帧编码类型一起进行联合编码。例如，编码器可对指示宏块的差分运动矢量的存在或缺乏（例如，对于具有一个运动矢量的宏块）的信息进行联合编码。

解码器执行对应的解码。例如，图 31 示出了在某些实现中，用于对隔行扫描 P 帧中的宏块的联合编码的运动补偿类型信息和半帧/帧编码类型信息进行解码的技术 3100。在 3110，解码器接收包括表示宏块的运动补偿类型和半帧/帧编码类型的联合码（例如，来自可变编代码表的可变长度码）的宏块信息。在 3120，解码器对该联合码进行解码（例如通过在可变长度编代码表中查找该联合码），以获得

该宏块的运动补偿类型信息和半帧/帧编码类型信息。

在一个实现中，宏块级比特流元素 MBMODE 联合地指定宏块的类型（1MV、4 帧 MV、2 半帧 MV、4 半帧 MV 或帧内编码）、帧间编码宏块的半帧/帧编码类型（半帧、帧或没有编码的块）、以及对 1MV 宏块是否存在差分运动矢量。在该示例中，MBMODE 可取 15 个可能值中的一个。设<MVP>表示对于是存在还是缺少非零 1MV 差分运动矢量的信号表示。设<Fiele/Frame transform>表示对该宏块的残差是（1）帧编码的；（2）半帧编码的；还是（3）零个编码块（即 CBP = 0）的信号表示。MBMODE 用信号联合表示以下信息：

MBMODE = {<1MV, MVP, Field/Frame transform>, <2 Field MV, Field/Frame transform>, <4 Frame MV, Field/Frame transform>, <4 Field MV, Field/Frame transform>, <INTRA>};

情况<1MV, MVP=0, CBP=0>不是由 MBMODE 来用信号表示的，而是由跳过条件来用信号表示的。（用信号表示该跳过条件的示例在以上第 V.A 节中提供。）

在该示例中，对于帧间编码的宏块，当 MBMODE 中的<Field/frame Transform>指示没有编码的块时，CBPCY 句法元素不被解码。另一方面，如果 MBMODE 中的<Field/frame Transform>指示半帧或帧变换，则 CBPCY 被解码。对于非 1MV 帧间编码的宏块，发送一附加字段以指示差分运动矢量中哪一个是非零的。在 2 半帧 MV 宏块的情况下，发送 2MVBP 字段以指示两个运动矢量中的哪一个包含非零差分运动矢量。类似地，发送 4MVBP 字段以指示四个运动矢量中的哪一个包含非零差分运动矢量。对于帧内编码的宏块，半帧/帧编码类型和零编码块在单独的字段中编码。

或者，编码器/解码器对运动补偿类型和半帧/帧编码类型的不同组合进行联合编码。作为另一替换，编码器/解码器对除运动矢量差分的存在以外的其它信息进行联合编码/解码。

在某些实现中，编码器/解码器使用若干可变长度代码表之一来编码 MBMODE，并且可自适应地在代码表之间切换。例如，在某些实现中，帧级句法元素 MBODETAB 是 2 位字段，它指示用于对该帧中的宏块的 MBMODE 解码的表。在该示例中，表被组合成四个表的集合，且表的集合取决于是否对该帧启用了 4 运动矢量编码来使用。

示例性 MBMODE 可变长度代码表（例如，对每一集合的表 0-3－混合 MV 或 1MV）在以下表 1-8 中提供：

表 1：隔行扫描 P 帧混合 MV MB 模式表 0

MB 类型	MV 存在	变换	VLC 码字	VLC 大小	VLC（二进制）
1MV	1	帧	22	5	10110
1MV	1	半帧	17	5	10001
1MV	1	无 CBP	0	2	00
1MV	0	帧	47	6	101111
1MV	0	半帧	32	6	100000
2 半帧 MV	N/A	帧	10	4	1010
2 半帧 MV	N/A	半帧	1	2	01
2 半帧 MV	N/A	无 CBP	3	2	11
4 帧 MV	N/A	帧	67	7	1000011
4 帧 MV	N/A	半帧	133	8	10000101
4 帧 MV	N/A	无 CBP	132	8	10000100
4 半帧 MV	N/A	帧	92	7	1011100
4 半帧 MV	N/A	半帧	19	5	10011
4 半帧 MV	N/A	无 CBP	93	7	1011101
帧内编码	N/A	N/A	18	5	10010

表 2：隔行扫描帧混合 MV MB 模式表 1

MB 类型	MV 存在	变换	VLC 码字	VLC 大小	VLC（二进制）
1MV	1	帧	3	3	011
1MV	1	半帧	45	6	101101
1MV	1	无 CBP	0	3	000
1MV	0	帧	7	3	111
1MV	0	半帧	23	5	10111
2 半帧 MV	N/A	帧	6	3	110
2 半帧 MV	N/A	半帧	1	3	001
2 半帧 MV	N/A	无 CBP	2	3	010
4 帧 MV	N/A	帧	10	4	1010
4 帧 MV	N/A	半帧	39	6	100111

4 帧 MV	N/A	无 CBP	44	6	101100
4 半帧 MV	N/A	帧	8	4	1000
4 半帧 MV	N/A	半帧	18	5	10010
4 半帧 MV	N/A	无 CBP	77	7	1001101
帧内编码	N/A	N/A	76	7	1001100

表 3：隔行扫描帧混合 MV MB 模式表 2

MB 类型	MV 存在	变换	VLC 码字	VLC 大小	VLC（二进制）
1MV	1	帧	15	4	1111
1MV	1	半帧	6	3	110
1MV	1	无 CBP	28	5	11100
1MV	0	帧	9	5	01001
1MV	0	半帧	41	7	0101001
2 半帧 MV	N/A	帧	6	4	0110
2 半帧 MV	N/A	半帧	2	2	10
2 半帧 MV	N/A	无 CBP	15	5	01111
4 帧 MV	N/A	帧	14	5	01110
4 帧 MV	N/A	半帧	8	5	01000
4 帧 MV	N/A	无 CBP	40	7	0101000
4 半帧 MV	N/A	帧	29	5	11101
4 半帧 MV	N/A	半帧	0	2	00
4 半帧 MV	N/A	无 CBP	21	6	010101
帧内编码	N/A	N/A	11	5	01011

表 4：隔行扫描帧混合 MV MB 模式表 3

MB 类型	MV 存在	变换	VLC 码字	VLC 大小	VLC（二进制）
1MV	1	帧	7	4	0111
1MV	1	半帧	198	9	011000110
1MV	1	无 CBP	1	1	1
1MV	0	帧	2	3	010

1MV	0	半帧	193	9	011000001
2 半帧 MV	N/A	帧	13	5	01101
2 半帧 MV	N/A	半帧	25	6	011001
2 半帧 MV	N/A	无 CBP	0	2	00
4 帧 MV	N/A	帧	97	8	01100001
4 帧 MV	N/A	半帧	1599	12	011000111111
4 帧 MV	N/A	无 CBP	98	8	01100010
4 半帧 MV	N/A	帧	398	10	0110001100
4 半帧 MV	N/A	半帧	798	11	01100011110
4 半帧 MV	N/A	无 CBP	192	9	011000000
帧内编码	N/A	N/A	1598	12	011000111110

表 5：隔行扫描帧 1 MV MB 模式表 0

MB 类型	MV 存在	变换	VLC 码字	VLC 大小	VLC（二进制）
1MV	1	帧	9	4	1001
1MV	1	半帧	22	5	10110
1MV	1	无 CBP	0	2	00
1MV	0	帧	17	5	10001
1MV	0	半帧	16	5	10000
2 半帧 MV	N/A	帧	10	4	1010
2 半帧 MV	N/A	半帧	1	2	01
2 半帧 MV	N/A	无 CBP	3	2	11
帧内编码	N/A	N/A	23	5	10111

表 6：隔行扫描帧 1MV MB 模式表 1

MB 类型	MV 存在	变换	VLC 码字	VLC 大小	VLC（二进制）
1MV	1	帧	7	3	111
1MV	1	半帧	0	4	0000
1MV	1	无 CBP	5	6	000101
1MV	0	帧	2	2	10

1MV	0	半帧	1	3	001
2 半帧 MV	N/A	帧	1	2	01
2 半帧 MV	N/A	半帧	6	3	110
2 半帧 MV	N/A	无 CBP	3	5	00011
帧内编码	N/A	N/A	4	6	000100

表 7：隔行扫描帧 1MV MB 模式表 2

MB 类型	MV 存在	变换	VLC 码字	VLC 大小	VLC（二进制）
1MV	1	帧	1	2	01
1MV	1	半帧	0	2	00
1MV	1	无 CBP	10	4	1010
1MV	0	帧	23	5	10111
1MV	0	半帧	44	6	101100
2 半帧 MV	N/A	帧	8	4	1000
2 半帧 MV	N/A	半帧	3	2	11
2 半帧 MV	N/A	无 CBP	9	4	1001
帧内编码	N/A	N/A	45	6	101101

表 8：隔行扫描帧 1MV MB 模式表 3

MB 类型	MV 存在	变换	VLC 码字	VLC 大小	VLC（二进制）
1MV	1	帧	7	4	0111
1MV	1	半帧	97	8	01100001
1MV	1	无 CBP	1	1	1
1MV	0	帧	2	3	010
1MV	0	半帧	49	7	0110001
2 半帧 MV	N/A	帧	13	5	01101
2 半帧 MV	N/A	半帧	25	6	011001
2 半帧 MV	N/A	无 CBP	0	2	00
帧内编码	N/A	N/A	96	8	01100000

VI. 色度运动矢量导出中的革新

所描述的实施例包括用于从亮度（或“luma”）运动矢量中导出色度（或“chroma”）运动矢量的技术和工具。某些所描述的技术和工具涉及导出帧编码的隔行扫描图像（例如，隔行扫描 P 帧、隔行扫描 B 帧等）中的色度运动矢量以及改进隔行扫描帧编码图像的速率/失真性能。在所描述的技术和工具中，色度运动矢量不在比特流中显式地发送。相反，它们从对帧的宏块或块编码和发送的亮度运动矢量中导出。

所描述的实施例实现了包括但不限于以下描述的技术和工具中的一个或多个：

1. 编码器/解码器通过对宏块中的每一亮度运动矢量导出色度运动矢量，获得隔行扫描帧编码图像（例如，隔行扫描 P 帧、隔行扫描 B 帧）中的亮度和色度运动矢量之间的一对一对应性。色度运动矢量然后用于对相应的色度块或半帧进行运动补偿。
2. 当对应的宏块通过在二次采样之后将可变偏移量（例如，使用查找表）加到色度运动矢量来进行半帧编码时，编码器/解码器维护亮度和色度运动矢量之间的相干性。

尽管所描述的技术应用于隔行扫描视频中的 4:2:0 宏块格式，然而所描述的技术可应用于其它宏块格式（例如，4:2:2、4:4:4 等）以及其它种类的视频。所描述的技术和工具可彼此结合或与其它技术和工具结合使用，或可单独使用。

A. 一对一色度运动矢量对应性

在某些实现中，编码器/解码器导出并使用与用于预测宏块的亮度运动矢量相同数目的色度运动矢量来预测宏块。例如，当编码器/解码器对给定宏块使用一个、两个或四个亮度半帧或帧类型的运动矢量时，编码器/解码器对给定宏块分别导出一个、两个或四个色度运动矢量。这一技术与早先的编码器和解码器（例如，在逐行扫描帧或隔行扫描 P 帧上下文中）不同，其中早先的编码器或解码器总是对每一宏块中任何数量的亮度运动矢量（例如，一个或四个）导出单个色度运动矢量。

图 32 是示出用于对宏块中的多个亮度运动矢量中的每一个导出色度运动矢量的技术 3200 的流程图。在 3210，编码器/解码器接收宏块的多个亮度运动矢量。在 3220，编码器/解码器对多个亮度运动矢量中的每一个导出色度运动矢量。导出的色度运动矢量的数目取决于用于预测当前宏块的亮度运动矢量的数目而变化。

在某些实现中，编码器/解码器对 1MV 宏块导出一个色度运动矢量，对 2 半帧 MV 宏块导出两个半帧色度运动矢量，对 4 帧 MV 宏块导出四个帧色度运动矢量，并对 4 半帧 MV 宏块导出四个半帧色度运动矢量。

例如，再次参考图 23-25，图 23 示出了 2 半帧 MV 宏块中从亮度运动矢量中导出的对应的上半帧和下半帧色度运动矢量。所导出的上半帧和下半帧色度运动矢量分别描述了色度块的偶数行和奇数行的位移。

图 24 示出了 4 帧 MV 宏块中对四个块中的每一个从帧亮度运动矢量中导出的对应的帧色度运动矢量（MV1'、MV2'、MV3'和 MV4'）。四个导出的色度运动矢量描述了四个 4×4 色度子块各自的位移。

图 25 示出了 4 半帧 MV 宏块中从半帧亮度运动矢量中导出的对应的半帧色度运动矢量。两个半帧色度运动矢量描述了色度块中每一半帧的位移。色度块的行被垂直细分以形成两个 4×8 的区域，其每一个具有与下半帧行的 4×4 区域交错的上半帧行的 4×4 区域。对于上半帧行，左侧 4×4 区域的位移是由左上半帧色度块运动矢量描述的，而右侧 4×4 区域的位移是由右上半帧色度块运动矢量描述的。对于下半帧行，左侧 4×4 区域的位移是由左下半帧色度块运动矢量描述的，而右侧 4×4 区域的位移是由右下半帧色度块运动矢量描述的。

每一色度块区域可以使用导出的运动矢量来进行运动补偿。这允许比早先的编码器和解码器中更大的色度运动补偿分辨率，早先的编码器和解码器通常通过二次采样（也称为“下采样”）和/或求平均来从任何数目的亮度运动矢量中导出单个色度运动矢量（例如，其中从宏块中的四个亮度运动矢量中导出一个色度运动矢量）。

或者，编码器/解码器可从不同数目和/或类型的亮度运动矢量中导出色度运动矢量（例如，从用两个帧亮度运动矢量编码的宏块中导出两个帧色度运动矢量，从用四个以上亮度运动矢量编码的宏块中导出四个以上色度运动矢量等等）。或者，色度运动矢量可以按某一其它方式导出，而同时维持与宏块的亮度运动矢量数目的 1:1 对应性。

B．半帧编码宏块中基于半帧的舍入

在某些实现中，编码器/解码器在对半帧编码的宏块导出色度运动矢量的过程中使用基于半帧的舍入来维持亮度运动矢量和色度运动矢量之间的相干性。

给定亮度帧运动矢量或半帧运动矢量，编码器/解码器导出对应的色度帧运动

矢量或半帧运动矢量，以对色度（Cb/Cr）块的一部分（以及可能所有）执行运动补偿。在某些实现中，对隔行扫描帧编码图像（例如，隔行扫描 P 帧、隔行扫描 B 帧等）的色度运动矢量导出包括舍入和二次采样。例如，当从亮度半帧运动矢量中导出色度运动矢量时，编码器/解码器在二次采样之后将可变偏移量（例如，使用查找表）添加到色度运动矢量。

图 33 是示出用于使用基于半帧的舍入查找表导出宏块中的色度运动矢量的技术 3300 的流程图。在 3310，编码器/解码器对亮度半帧运动矢量分量进行二次采样（例如，通过以 4:2:0 宏块格式将 y 分量值除以 2）。在 3320，编码器/解码器然后使用基于半帧的舍入查找表执行舍入。

图 34 中的伪代码 3400 描述了在一个实现中如何在 4:2:0 宏块中从亮度运动矢量分量(LMV_x, LMV_y)中导出色度运动矢量分量(CMV_x, CMV_y)。如在伪代码 3400 中示出的，编码器/解码器在水平二次采样之前使用简单的舍入策略（使用舍入查找表 s_RndTbl[ ]）来上舍入运动矢量的 x 分量的 3/4 像素位置。如果宏块是帧编码的，则编码器/解码器在对 y 分量进行垂直二次采样之前使用相同的舍入查找表。然而，如果宏块是半帧编码的，则编码器/解码器不同地对待色度运动矢量的 y 分量。在半帧编码的宏块中，色度运动矢量对应于上半帧或下半帧。上半帧和下半帧各自包括色度块的交替的水平行。因此，在这一情况下，编码器/解码器使用伪代码 3400 中所示的基于半帧的舍入查找表 s_RndTblField[ ]。基于半帧的舍入查找表允许编码器/解码器在舍入时维持正确的半帧偏移量，使得亮度和色度运动矢量映射到一致的半帧偏移量。例如，参考图 35，s_RndTblField[ ] 中的值 0, 0, 1, 2 和 2, 2, 3, 8（图 35 中的上半帧值 3510）应用于一个半帧（例如，上半帧），而值 4, 4, 5, 6 和 6, 6, 7, 12（图 35 中的下半帧值 3520）应用于宏块中的另一半帧（例如，下半帧）。

或者，编码器/解码器可使用不同的基于半帧的舍入查找表或以某一其它方式来执行舍入和/或二次采样。例如，处理不同格式的宏块的编码器/解码器可使用不同的二次采样因子和/或查找表值。

VII. 组合实现

现在描述对于比特流句法、语义和解码器的详细的组合实现，以及具有来自主要的组合实现的微小差别的替换组合实现。

A．比特流句法

在各种组合实现中，用于隔行扫描图像的数据以具有多个层（例如，序列、入口点、帧、半帧、宏块、块和/或子块层）的比特流的形式存在。

在句法图中，箭头路径示出句法元素的可能流程。所示的带有正方形边框的句法元素指示固定长度的句法元素；带有圆形边框示出的句法元素指示可变长度句法元素，而带有外部的圆形边框内的圆形边框的句法元素指示由更简单句法元素构成的句法元素（例如，位平面）。固定长度句法元素被定义为其句法元素长度不依赖于句法元素本身中的数据的句法元素；固定长度句法元素的长度是常量，或由句法流中先前的数据来确定。分层图中的较低层（例如，帧层图中的宏块层）是由矩形内的矩形来指示的。

入口点级比特流元素在图 36 中示出。一般而言，入口点标记了比特流中解码器可开始解码的位置（例如，I 帧或其它关键帧）。换言之，不需要比特流中在入口点之前的任何图像来解码入口点之后的图像。入口点头部可用于用信号表示编码控制参数中的改变（例如，对入口点之后的帧启用或禁用压缩工具（例如，内循环分块滤波））。

对于隔行扫描的 P 帧和 B 帧，帧级比特流元素分别在图 37 和 38 中示出。对于每一帧的数据由帧头部及之后的宏块层数据（不论是用于帧内编码还是各种帧间编码类型的宏块）构成。构成隔行扫描 P 帧的宏块层（无论是用于帧内编码还是各种帧间编码类型宏块）的比特流元素在图 40 中示出。用于隔行扫描 P 帧的宏块层中的比特流元素可对其它隔行扫描图像（例如，隔行扫描 B 帧、隔行扫描 P 半帧、隔行扫描 B 半帧等）中的宏块存在。

对于具有隔行扫描 P 半帧和/或 B 半帧的隔行扫描视频帧，帧级比特流元素在图 39 中示出。对于每一帧的数据由帧头部以及之后的用于半帧层的数据（被示为每一半帧的重复的“FieldPicLayer”元素）以及用于宏块层的数据（无论是用于帧内编码、1MV 还是 4MV 宏块）构成。

以下章节描述了帧和宏块层内涉及对隔行扫描图像的信号表示的所选择的比特流元素。尽管所选择的比特流元素是在特定层的上下文中描述的，然而某些比特流元素可在一个以上层中使用。

1．选择的入口点层元素

循环滤波器（LOOPFILTER）（1 比特）

LOOPFILTER 是指示是否对入口点段启用循环滤波的布尔标志。如果 LOOPFILTER = 0，则不启用循环滤波。如果 LOOPFILTER = 1，则启用循环滤波。在替换的组合实现中，LOOPFILTER 是序列级元素。

扩展的运动矢量（EXTENDED_MV）（1 比特）

EXTENDED_MV 是指示扩展运动矢量是打开（值 1）还是关闭（值 0）的 1 比特句法元素。EXTENDED_MV 指示 P 帧和 B 帧中扩展的运动矢量（在帧级用句法元素 MVRANGE 来用信号表示）的可能性。

扩展的差分运动矢量范围（EXTENDED_DMV）（1 比特）

EXTENDED_DMV 是如果 EXTENDED_MV = 1 则存在的 1 比特句法元素。如果 EXTENDED_DMV 为 1，则应当对入口点段中的 P 帧和 B 帧在帧层上用信号表示扩展的差分运动矢量范围（DMVRANGE）。如果 EXTENDED_DMV 为 0，则不应用信号表示 DMVRANGE。

快速 UV 运动比较（FASTUVMC）（1 比特）

FASTUVMC 是控制色度运动矢量的子像素内插和舍入的布尔标志。如果 FASTUVMC = 1，则在四分之一像素偏移量处的色度运动矢量被舍入到最近的半或全像素位置。如果 FASTUVMC = 0，则不对色度执行任何特殊的舍入或滤波。FASTUVMC 句法元素在隔行扫描 P 帧和隔行扫描 B 帧中忽略。

可变大小变换（VSTRANSFORM）（1 比特）

VSTRANSFORM 是指示对序列是否启用可变大小变换编码的布尔标志。如果 VSTRANSFORM = 0，则不启用可变大小变换编码。如果 VSTRANSFORM = 1，则启用可变大小变换编码。

2．选择的帧层元素

图 37 和 38 是分别示出用于隔行扫描 P 帧和隔行扫描 B 帧的帧级比特流句法的图示。图 39 是示出用于包含隔行扫描 P 半帧和/或 B 半帧（或可能其它种类的隔行扫描的半帧）的帧的帧层比特流句法。具体的比特流元素描述如下。

帧编码模式（FCM）（可变大小）

FCM 是用于指示图像编码类型的可变长度码字[“VLC”]。FCM 采取以下表 9 中示出的用于帧编码模式的值：

表 9：帧编码模式 VLC

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

半帧图像类型（FPTYPE）（3 比特）

FPTYPE 是存在于包括隔行扫描 P 半帧和/或隔行扫描 B 半帧以及可能的其它种类的半帧的帧的帧头部中的 3 比特句法元素。FPTYPE 依照以下表 10，采取隔行扫描视频帧中半帧类型的不同组合的值。

表 10：半帧图像类型 FLC

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

图像类型（PTYPE）（可变大小）

PTYPE 是存在于隔行扫描 P 帧和隔行扫描 B 帧（或其它种类的隔行扫描帧，诸如隔行扫描 I 帧）的帧头部中的可变大小句法元素。PTYPE 采取依照以下表 11 的用于不同帧类型的值。

表 11：图像类型 VLC

PTYPE VLC	图像类型

110	I
0	P
10	B
1110	BI
1111	跳过

如果 PTYPE 指示帧被跳过，则该帧作为与其参考帧相同的 P 帧来对待。跳过的帧的重构在概念上等效于复制参考帧。跳过的帧意味着对该帧无需再发送任何数据。

UV 采样格式（UVSAMP）（1 比特）

UVSAMP 是当序列级字段 INTERLACE = 1 时存在的 1 比特句法元素。UVSAMP 指示用于当前帧的色度二次采样的类型。如果 UVSAMP = 1，则使用色度的逐行扫描二次采样，否则，使用色度的隔行扫描二次采样。该句法元素不影响比特流的解码。

扩展的 MV 范围（MVRANGE）（可变大小）

MVRANGE 是当入口点层 EXTENDED_MV 位被设为 1 时存在的可变大小句法元素。MVRANGE VLC 表示运动矢量范围。

扩展的差分 MV 范围（DMVRANGE）（可变大小）

DMVRANGE 是如果入口点层句法元素 EXNTEDED_DMV = 1 时存在的可变大小句法元素。DMVRANGE VLC 表示运动矢量差分范围。

4 运动矢量切换（4MVSWITCH）（可变大小或 1 比特）

对于隔行扫描 P 帧，4MVSWITCH 句法元素是 1 比特标志。如果 4MVSWITCH 被设为零，则图像中的宏块只有一个运动矢量或有两个运动矢量，分别取决于该宏块是帧编码的还是半帧编码的。如果 4MVSWITCH 被设为 1，则对每一宏块可以有一个、两个或四个运动矢量。

跳过宏块解码（SKIPMB）（可变大小）

对于隔行扫描 P 帧，SKIPMB 句法元素是包含指示图像中每一宏块的跳过/未跳过状态的信息的压缩的位平面。解码的位平面用 1 比特的值表示每一宏块的跳过/未跳过状态。0 值指示该宏块未被跳过。1 值指示该宏块被编码为跳过。隔行扫描 P 帧中的宏块的跳过状态意味着解码器将该宏块作为具有零运动矢量差分和零编码块模式的 1MV 来对待。不期望对跳过宏块跟任何其它信息。

宏块模式表（MBMODETAB）（2 或 3 比特）

MBMODETAB 句法元素是固定长度字段。对于隔行扫描 P 帧，MBMODETAB 是 2 比特值，它指示四个代码表中的哪一个用于解码宏块层中的宏块模式句法元素（MBMODE）。有两组四个代码表，且使用的组取决于如 4MVSWITCH 标志所指示的是否使用了 4MV。

运动矢量表（MVTAB）（2 或 3 比特）

MVTAB 句法元素是固定长度字段。对于隔行扫描 P 帧，MVTAB 是指示四个逐行扫描(或一个参考)运动矢量代码表中的哪一个用于编码宏块层中的 MVDATA 句法元素。

2MV 块模式表（2MVBPTAB）（2 比特）

2MVBPTAB 句法元素是用信号表示四个代码表中的哪一个用于解码 2 半帧 MV 宏块中的 2MV 块模式（2MVBP）句法元素的 2 比特值。

4MV 块模式表（4MVBPTAB）（2 比特）

4MVBPTAB 句法元素是用信号表示四个代码表中的哪一个用于解码 4MV 宏块中的 4MV 块模式（4MVBP）句法元素的 2 比特值。对于隔行扫描 P 帧，如果 4MVSWITCH 句法元素被设为 1，则它存在。

宏块级变换类型标志（TTMBF）（1 比特）

如果序列级句法元素 VSTRANSFORM = 1，则该句法元素存在于 P 帧和 B 帧中。TTMBF 是用信号表示是否在帧或宏块级启用变换类型编码的 1 比特句法元素。如果 TTMBF = 1，则对帧中的所有块使用相同的变换类型。在这一情况下，在之后的帧级变换类型（TTFRM）句法元素中用信号表示变换类型。如果 TTMBF = 0，

则变换类型可贯穿帧而变化，且在宏块或块级用信号表示。

帧级变换类型（TTFRM）（2 比特）

如果 VSTRANSFORM = 1 且 TTMBF = 1，则该句法元素存在于 P 帧和 B 帧中。TTFRM 用信号表示用于变换预测块中的 8×8 的像素误差信号的变换类型。8×8 的误差块可使用 8×8 变换、两个 8×4 变换、两个 4×8 变换或四个 4×4 变换来变换。

3．选择的宏块层元素

图 40 是示出组合实现中用于隔行扫描 P 帧中的宏块的宏块级比特流句法。具体的比特流元素描述如下。宏块数据由宏块头部及之后的块层数据构成。用于隔行扫描 P 帧的宏块层中的比特流元素（例如，SKIPMBBIT）可能对于其它隔行扫描图像（例如，隔行扫描 B 帧等）中的宏块存在。

跳过 MB 位（SKIPMBBIT）（1 比特）

SKIPMBBIT 是当帧级句法元素 SKIPMB 指示使用原始模式时存在于隔行扫描 P 帧宏块和隔行扫描 B 帧宏块中的 1 比特句法元素。如果 SKIPMBBIT = 1，则跳过该宏块。SKIPMBBIT 也可被标记为宏块级的 SKIPMB。

宏块模式（MBMODE）（可变大小）

MBMODE 是联合地指定宏块类型（例如，1MV、2 半帧 MV、4 半帧 MV、4 帧 MV 或帧内编码）、半帧/帧编码类型（例如，半帧、帧或无编码块）以及 1MV 宏块的差分运动矢量数据的存在的可变大小句法元素。MBMODE 在以下和以上第 V 节中详细解释。

2MV 块模式（2MVBP）（可变大小）

2MVBP 是存在于隔行扫描 P 帧和隔行扫描 B 帧宏块中的可变大小句法元素。在隔行扫描 P 帧宏块中，如果 MBMODE 指示宏块具有两个半帧运动矢量，则 2MVBP 存在。在这一情况下，2MVBP 指示两个亮度块中的哪一个包含非零运动矢量差分。

4MV 块模式（4MVBP）（可变大小）

4MVBP 是存在于隔行扫描 P 半帧、隔行扫描 B 半帧、隔行扫描 P 帧和隔行扫描 B 帧宏块中的可变大小句法元素。在隔行扫描 P 帧中，如果 MBMODE 指示宏块具有四个运动矢量，则 4MVBP 存在。在这一情况下，4MVBP 指示四个亮度块中的哪一个包含非零运动矢量差分。

半帧变换标志（FIELDTX）（1 比特）

FIELDTX 是存在于隔行扫描 B 帧帧内编码宏块中的 1 比特句法。该句法元素指示宏块是帧还是半帧编码的（基本上是宏块的内部组织）。FIELDTX = 1 指示宏块是半帧编码的。否则，宏块是帧编码的。在帧间编码的宏块中，该句法元素可从如以下以及在以上第 V 节中详细解释的 MBMODE 中推导出来。

CBP 存在标志（CBPPRESENT）（1 比特）

CBPPRESENT 是存在于隔行扫描 P 帧和隔行扫描 B 帧中的帧内编码的宏块中的 1 比特句法。如果 CBPPRESENT 为 1，则对该宏块存在 CBPCY 句法元素并解码该元素。如果 CBPPRESENT 为 0，则不存在 CPBCY 句法元素，且应被设为零。

已编码块模式（CBPCY）（可变大小）

CBPCY 是指示对于宏块中的每一块的变换系数状态的可变长度句法元素。CBPCY 解码成指示对于对应的块是否存在系数的 6 比特字段。对于帧内编码的宏块，特定位位置中的零值指示对应的块不包含任何非零 AC 系数。1 值指示存在至少一个非零 AC 系数。在所有情况下，对于每一块仍存在 DC 系数。对于帧间编码的宏块，特定位位置中的 0 值指示对应的块不包含任何非零系数。1 值指示存在至少一个非零系数。对于该位为 0 的情况，不对该块编码任何数据。

运动矢量数据（MVDATA）（可变大小）

MVDATA 是对宏块的运动矢量的差分编码的可变大小句法元素，其解码在以下详细描述。

MB 级变换类型（TTMB）（可变大小）

TTMB 是当图像层句法元素 TTMBF = 0 时 P 图像和 B 图像宏块中的可变大小

句法元素。TTMB 指定了变换类型、变换类型信号水平以及子块模式。

B．解码隔行扫描 P 帧

在一个组合实现中，用于解码隔行扫描 P 帧的过程描述如下。

1．隔行扫描 P 帧的宏块层解码

在隔行扫描 P 帧中，每一宏块可以在帧模式中使用一个或四个运动矢量，或在半帧模式中使用两个或四个运动矢量来进行运动补偿。帧间编码的宏块不包含任何帧内编码的块。另外，运动补偿之后的残差可以在帧变换模式或半帧变换模式中编码。更具体地，如果以半帧变换模式编码，则残差的亮度分量依照半帧来重新排列，但是在帧变换模式中保持不变，而色度分量保持相同。宏块也可作为帧内编码来编码。

运动补偿可被限于不包括四个（半帧/帧两者）运动矢量，且这通过 4MVSWITCH 用信号表示。运动补偿和残差编码的类型通过 MBMODE 和 SKIPMB 对每一宏块联合指示。MBMODE 采用了依照 4MVSWITCH 的一组不同的表。

隔行扫描 P 帧中的宏块被分类为五个类型：1MV、2 半帧 MV、4 帧 MV、4 半帧 MV 和帧内编码。这五个类型已在上文第 III 节中详细描述。前四个宏块类型是帧间编码的，而最后一个类型指示宏块是帧内编码的。宏块类型是由宏块层中的 MBMODE 句法元素连同跳过位一起用信号表示的。（宏块的跳过条件也可在压缩位平面中的帧级处用信号表示。）MBMODE 对不同类型的宏块，对宏块类型以及关于宏块的各个信息进行联合编码。

跳过宏块的信号表示

宏块级 SKIPMBBIT 字段指示宏块的跳过条件。（关于跳过条件和对应的信号表示的其它细节在以上第 V 节中提供。）如果 SKIPMBBIT 字段为 1，则当前宏块被认为是跳过，且在 SKIPMBBIT 字段之后不发送任何其它信息。（在帧级，SKIPMB 字段指示在宏块级 SKIPMBBIT 的存在（采用原始模式），或在压缩位平面中储存跳过信息。解码的位平面对每一宏块包含一个比特，且指示每一相应宏块的跳过条件。）跳过条件暗示当前宏块是具有零差分运动矢量的 1MV（即，该宏块使用其 1MV 运动预测值来运动补偿），且没有已编码的块（CBP = 0）。在替换的组合实现中，残差被假定为为循环滤波目的而进行帧编码。

另一方面，如果 SKIPMB 字段不是 1，则 MBMODE 字段被解码为指示宏块类型以及关于当前宏块的其它信息，诸如以下章节中所描述的信息。

宏块模式的信号表示

MBMODE 联合地指定了宏块的类型（1MV、4 帧 MV、2 半帧 MV、4 半帧 MV 或帧内编码）、帧间编码宏块的变换类型（即，半帧或帧或无已编码块）、以及对 1MV 宏块是否存在差分运动矢量。（关于宏块信息的信号表示的其它细节在以上第 V 节中提供。）MBMODE 可采取 15 个可能的值中的一个：

设<MVP>表示存在还是缺少非零 1MV 差分运动矢量的信号表示。设<Field/Frame transform>表示宏块的残差是（1）帧变换编码的；（2）半帧变换编码的；还是（3）零编码的块（即，CBP = 0）的信号表示。MBMODE 用信号联合表示以下信息：

MBMODE = {<1MV, MVP, Field/Frame transform>, <2 Field MV, Field/Frame transform>, <4 Frame MV, Field/Frame transform>, <4 Field MV, Field/Frame transform>, <INTRA>};

情况<1MV, MVP=0, CBP=0>不是由 MBMODE 来用信号表示的，而是由跳过条件来用信号表示的。

对于帧间编码的宏块，当 MBMODE 中的<Field/frame Transform>指示无已编码块时，不解码 CBPCY 句法元素。另一方面，如果 MBMODE 中的<Field/frame Transform>指示半帧或帧变换，则解码 CBPCY。解码的<Field/frame Transform>用于设置标志 FIELDTX。如果它指示宏块是半帧变换编码的，则 FIELDTX 被设为 1。如果它指示宏块是帧变换编码的，则 FIELDTX 被设为 0。如果它指示零编码的块，则 FIELDTX 被设为与运动矢量相同的类型，即如果它是半帧运动矢量，则 FIELDTX 被设为 1，如果是帧运动矢量，则设为 0。

对于非 1MV 帧间编码的宏块，发送一附加字段以指示差分运动矢量中的哪一个是非零的。在 2 半帧 MV 宏块的情况下，发送 2MVBP 字段以指示两个运动矢量中的哪一个包含非零差分运动矢量。类似地，发送 4MVBP 字段以指示四个运动矢量中的哪一个包含非零差分运动矢量。

对于帧内编码的宏块，在单独的字段中编码半帧/帧变换和零编码块。

2．用于隔行扫描 P 帧的运动矢量解码

用于隔行扫描P帧的运动矢量预测值

为当前宏块计算运动矢量预测值的过程由两个步骤构成。首先，从其相邻宏块收集当前宏块的三个候选运动矢量。其次，从该候选运动矢量集计算当前宏块的运动矢量预测值。图 26A-26B 示出了从其中收集候选运动矢量的相邻宏块。候选运动矢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，前进到 B，且在 C 结束。如果对应的块位于帧边界之外，或者如果对应的块是不同片的一部分，则候选预测值被认为不存在。由此，运动矢量预测不跨片边界执行。

以下章节描述了如何为不同类型的宏块收集候选运动矢量以及如何计算运动矢量预测值。

1MV 候选运动矢量

在该组合实现中，图 41 中的伪代码 4100 用于为运动矢量收集多达三个候选运动矢量。

4 帧 MV 候选运动矢量

对于 4 帧 MV 宏块，对于当前宏块中四个帧块运动矢量中的每一个，收集来自相邻块的候选运动矢量。在该组合实现中，图 42 中的伪代码 4200 用于为左上帧运动矢量收集多达三个候选运动矢量。图 43 中的伪代码 4300 用于为右上帧块运动矢量收集多达三个候选运动矢量。图 44 中的伪代码 4400 用于为左下帧块运动矢量收集多达三个候选运动矢量。图 45 中的伪代码 4500 用于为右下帧块运动矢量收集多达三个候选运动矢量。

2 半帧 MV 候选运动矢量

对于 2 半帧 MV 宏块，对于当前宏块中的两个半帧运动矢量中的每一个，收集来自相邻块的候选运动矢量。图 46 中的伪代码 4600 用于为上半帧运动矢量收集多达三个候选运动矢量。图 47 中的伪代码 4700 用于为下半帧运动矢量收集多达三个候选运动矢量。

4 半帧 MV 候选运动矢量

对于 4 半帧 MV 宏块，对于当前宏块中四个半帧块中的每一个，收集来自相邻块的候选运动矢量。图 48 中的伪代码 4800 用于为左上半帧块运动矢量收集多达

三个候选运动矢量。图 49 中的伪代码 4900 用于为右上半帧块运动矢量收集多达三个候选运动矢量。图 50 中的伪代码 5000 用于为左下半帧块运动矢量收集多达三个候选运动矢量。图 51 中的伪代码 5100 用于为右下半帧块运动矢量收集多达三个候选运动矢量。

平均半帧运动矢量

给定两个半帧运动矢量(MVX_1, MVY_1)和(MVX_2, MVY_2)，用于形成候选运动矢量(MVX_A, MVY_A)的求平均运算为：

$MVX_A = (MVX_1 + MVX_2 + 1) >> 1;$

$MVY_A = (MVY_1 + MVY_2 + 1) >> 1;$

从候选运动矢量计算帧 MV 预测值

本节描述了如何在给定一组候选运动矢量时为帧运动矢量计算运动矢量预测值。在该组合实现中，对于为 1MV 计算预测值或对 4 帧 MV 宏块中四个帧块运动矢量中的每一个计算预测值的运算是相同的。

图 52 中的伪代码 5200 描述了如何为帧运动矢量计算运动矢量预测值(PMV_x, PMV_y)。在伪代码 5200 中，TotalValidMV 表示候选运动矢量集中运动矢量的总数（TotalValidMV = 0、1、2 或 3），而 ValidMV 数组表示候选运动矢量集中的运动矢量。

从候选运动矢量计算半帧 MV 预测值

本节描述了在组合实现中，如何在给定候选运动矢量集的情况下为半帧运动矢量计算运动矢量预测值。在该组合实现中，对于为 2 半帧 MV 宏块中两个半帧运动矢量中的每一个计算预测值或对于为 4 半帧 MV 宏块中四个半帧块运动矢量中的每一个计算预测值的运算是相同的。

首先，将候选运动矢量分为两个集合，其中一个集合仅包含指向与当前半帧相同的半帧的候选运动矢量，而另一集合包含指向相反半帧的候选运动矢量。假定候选运动矢量是以四分之一像素单位表示的，以下对其 y 分量的校验验证了候选运动矢量是否指向相同的半帧：

```
if(ValidMVy & 4) {
    ValidMV 指向相反半帧。
```

} else {

ValidMV 指向相同半帧。

}

图 53 中的伪代码 5300 描述了如何为半帧运动矢量计算运动矢量预测值(PMV_x, PMV_y)。在伪代码 5300 中，SameFieldMV 和 OppFieldMV 表示两个候选运动矢量集，NumSameFieldMV 和 NumOppFieldMV 表示属于每一集合的候选运动矢量的数目。每一集合中候选运动矢量的顺序以候选运动矢量 A（如果存在）开始，之后是候选运动矢量 B（如果存在），之后是候选运动矢量 C（如果存在）。例如，如果集合 SameFieldMV 仅包含候选运动矢量 B 和候选运动矢量 C，则 SameFieldMV[0] 是候选运动矢量 B。

解码运动矢量差分

MVDATA 句法元素包含宏块的运动矢量差分信息。取决于每一宏块处用信号表示的运动补偿类型和运动矢量块模式，对每一宏块可以有从 0 到 4 个 MVDATA 句法元素。更具体地，

- 对于 1MV 宏块，取决于 MBMODE 中的 MVP 字段，可以存在 0 个或 1 个 MVDATA 句法元素。
- 对于 2 半帧 MV 宏块，取决于 2MVBP，可以存在 0、1 或 2 个 MVDATA 句法元素。
- 对于 4 帧/半帧 MV 宏块，取决于 4MVBP，可以存在 0、1、2、3 或 4 个 MVDATA 句法元素。

在该组合实现中，以与隔行扫描 P 半帧的 1 参考半帧运动矢量差分相同的方式解码运动矢量差分，而无需半像素模式。（图 54A 中的伪代码 5400 示出了如何为一个参考半帧解码运动矢量差分。图 54B 中的伪代码 5410 示出了替换的组合实现中如何为一个参考半帧解码运动矢量差分。伪代码 5410 以不同的方式解码运动矢量差分。例如，伪代码 5410 省略了对扩展的运动矢量差分范围的处理。）

重构运动矢量

给定运动矢量差分 dmv，通过将该差分加到预测值来重构亮度运动矢量，如下：

mv_x = (dmv_x + predictor_x) smod range_x

mv_y = (dmv_y + predictor_y) smod range_y

smod运算确保重构的矢量是有效的。(A smod b)处于于-b和b – 1以内。range_x和 range_y 取决于 MVRANGE。

给定亮度帧或半帧运动矢量，导出对应的色度帧或半帧运动矢量以补偿色度（C_b/C_r）块的一部分（或可能全部）。在隔行扫描 P 帧和隔行扫描 B 帧中忽略 FASTUVMC 句法元素。图 34 中的伪代码 3400 描述了在隔行扫描 P 帧中如何从亮度运动矢量 LMV 中导出色度运动矢量 CMV。

C．位平面编码

诸如跳过位等宏块专用二进制信息可以对每一宏块在一个二进制码元中编码。例如，宏块是否被跳过可以用一个比特的信号来表示。在这些情况下，半帧或帧中的所有宏块的状态可以被编码为一个位平面，且在半帧或帧头部中发送。对于该规则的一个例外是如果位平面编码模式被设为原始模式，在这一情况下，每一宏块的状态被编码为每一码元一个比特，且在宏块级连同其它宏块级句法元素一起发送。

半帧/帧级位平面编码用于对二维二进制数组进行编码。每一数组的大小是 *rowMB*×*colMB*，其中 *rowMB* 和 *colMB* 分别是所讨论的半帧或帧中宏块行和列的数目。在比特流中，每一数组被编码为一组接连的比特。使用七种模式之一来编码每一数组。这七种模式是：

1. 原始模式－信息被编码为每一码元一个比特，并作为 MB 级句法的一部分发送；
2. 普通-2 模式－两个码元联合编码；
3. 差分-2 模式－位平面的差分编码，之后对两个残差码元联合编码；
4. 普通-6 模式－6 个码元联合编码；
5. 差分-6 模式－位平面的差分编码，之后对 6 个残差码元联合编码；
6. 行跳过模式－跳过一个比特以用信号表示没有置位的行；以及
7. 列跳过模式－跳过一个比特以用信号表示没有置位的列。

半帧级或帧级对于位平面的句法元素在以下序列中：INVERT、IMODE 和 DATABITS。

反转标志（INVERT）

INVERT 句法元素是 1 比特的值，如果被置位，则指示位平面具有比零个位更多的位被置位。取决于 INVERT 和模式，解码器应当反转所解释的位平面以重新创建原始信号。注意，当使用原始模式时，该位的值应当被忽略。如何在解码位平面时使用 INVERT 值的描述在下文提供。

编码模式（IMODE）

IMODE 句法元素是指示用于对位平面编码的编码模式的可变长度值。表 12 示出了用于对 IMODE 句法元素编码的代码表。对 IMODE 值如何用于解码位平面的描述在下文提供。

表 12：IMODE VLC 代码表

IMODE VLC	编码模式
10	普通-2
11	普通-6
010	行跳过
011	列跳过
001	差分-2
0001	差分-6
0000	原始

位平面编码位（DATABITS）

DATABITS 句法元素是对位平面的码元流进行编码的可变大小句法元素。用于编码位平面的方法是由IMODE的值来确定的。以下章节中描述了七种编码模式。

原始模式

在该模式中，位平面被编码为对宏块中以光栅扫描顺序扫描的每一码元一个比特，并作为宏块层的一部分发送。或者，原始模式中在半帧或帧级编码的信息和 DATABITS 的长度是 rowMB×colMB 比特。

普通-2 模式

如果 rowMB×colMB 是奇数，则第一码元是原始编码的。后续的码元以自然

扫描顺序成对地编码。表 13 中的二进制 VLC 表用于编码码元对。

表 13：普通-2/差分-2 代码表

码元 2n	码元 2n + 1	码字
0	0	0
1	0	100
0	1	101
1	1	11

差分-2 模式

普通-2 方法用于产生如上所述的位平面，然后如下所述向位平面应用 $Diff^1$ 运算。

普通-6 模式

在普通-6 和差分-6 模式中，位平面按照 6 个像素的组来编码。这些像素被组合成 2×3 或 3×2 的平铺块。位平面使用一组规则最大化地平铺，且剩余的像素使用行跳过和列跳过模式的变体来编码。当且仅当 rowMB 是 3 的倍数且 colMB 不是时，使用 2×3 的“垂直”平铺块。否则，使用 3×2 的“水平”平铺块。图 55A 示出了 2×3 的“垂直”平铺块的一个简化示例。图 55B 和 55C 示出了 3×2 的“水平”平铺块的简化示例，对于该平铺块，拉长的深色矩形是 1 个像素宽，且使用行跳过和列跳过编码来编码。对于如图 55C 所示地平铺的平面，其中线性平铺块沿着图像的上边和左边，平铺块的编码顺序遵循以下模式。首先编码 6 个元素的平铺块，然后是列跳过和行跳过编码的线性平铺块。如果数组大小是 2×3 或 3×2 的倍数，则后一线性平铺块不存在，且位平面是完美地平铺的。

6 元素矩形平铺块使用不完整的哈夫曼码来编码，即不使用所有的最终节点用于编码的哈夫曼码。设 N 是平铺块中置位的数目，即 $0 \le N \le 6$。对于 $N<3$，使用 VLC 来编码该平铺块。对于 $N=3$，固定长度换码之后跟随 5 比特的固定长度码，且对于 $N>3$，固定长度换码之后跟随该平铺块的补码的代码。

矩形平铺块包含 6 比特的信息。设 k 是与平铺块相关联的代码，其中 $k = b_i 2^j$，其中 b_i 是平铺块中自然扫描顺序的第 i 个比特的二进制值。因此 $0 \le k \le 64$。使用 VLC 和满足固定长度码的换码的组合来用信号表示 k。

差分-6 模式

普通-6 方法用于产生如上所述的位平面，然后如下所述向该位平面应用 $Diff^1$ 运算。

行跳过模式

在行跳过编码模式中，以一个比特的额外开销跳过所有的零行。句法如下：对于每一行，单个 ROWSKIP 位指示该行是否被跳过；如果该行被跳过，则接着是下一行的 ROWSKIP 位；否则（该行不被跳过），则接着是 ROWBITS 位（对行中的每一宏块有一个比特）。由此，如果整个行为零，则发送一个 0 比特作为 ROWSKIP 码元，且跳过 ROWBITS。如果在行中有一个置位，则 ROWSKIP 被设为 1，且整个行原始地发送（ROWBITS）。行是在半帧或帧中从上到下扫描的。

列跳过模式

列跳过是行跳过的转置。列是在半帧或帧中从左到右扫描的。

$Diff^1$：：反差分解码

如果使用了任一差分模式（差分-2 或差分-6），则首先使用对应的普通模式（分别为普通-2 或普通-6）来解码“差分位”的位平面。差分位用于重新生成原始的位平面。重新生成过程是二进制字母表上的 2-D DPCM。为重新生成位置(i, j)处的位，如下（从位置(i, j)处的位 b(i, j)）生成预测值 $b_p(i, j)$：

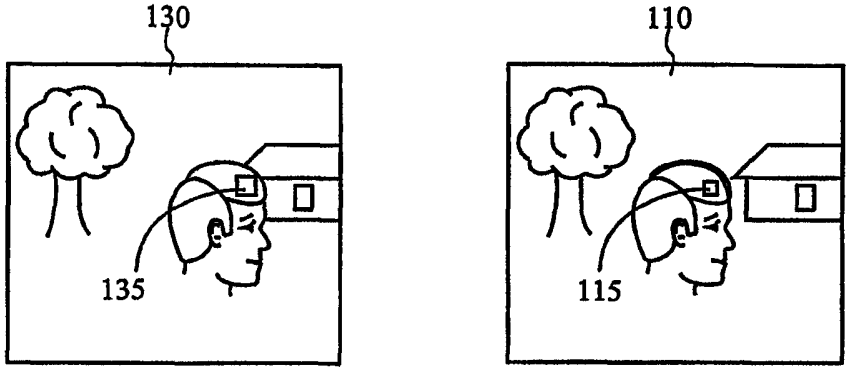
$$b_p(i,j)=\begin{cases} A & i=j=0, \text{或} b(i,j-1)\neq b(i-1,j) \\ b(0,j-1) & i==0 \\ b(i-1,j) & \text{其它} \end{cases}$$

对于差分编码模式，不执行基于 INVERT 的逐位反转过程。然而，在不同的容量中使用 INVERT 标志以指示用于上述预测值的导出的码元 A 的值。更具体地，如果 INVERT 等于 0，则 A 等于 0，如果 INVERT 等于 1，则 A 等于 1。位平面的实际值是通过将预测值与解码的差分位值进行 xor（异或）运算来获得的。在以上公式中，b(i, j)是在最终的解码之后（即，在完成了普通-2/普通-6，接着完成了与其

预测值的差分 xor 之后），第 i,j 位置处的位。

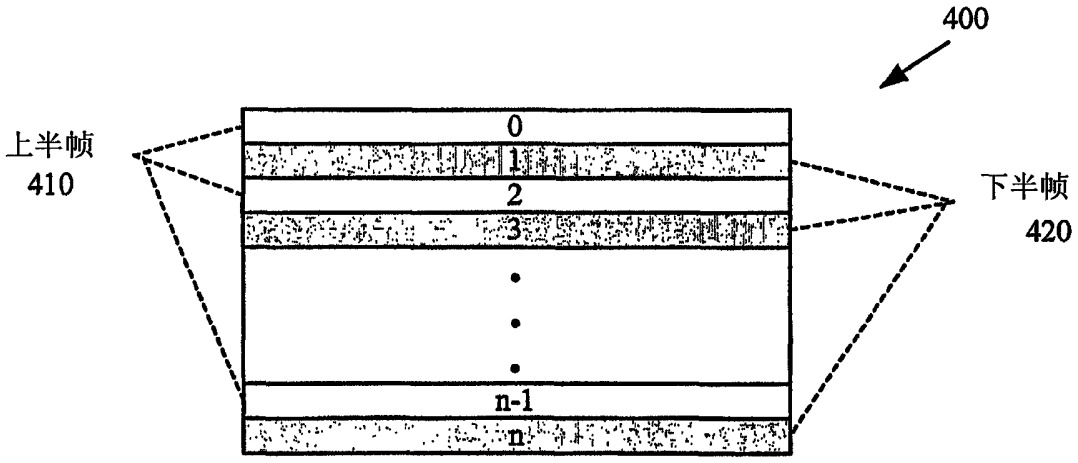
参考各实施例描述和示出了本发明的原理之后，可以认识到，可以在排列和细节上修改各实施例，而不脱离这些原理。应当理解，此处所描述的程序、过程或方法不相关于或不限于任何特定类型的计算环境，除非另外指明。可依照此处所描述的教导来使用各种类型的通用或专用计算环境或执行操作。以软件示出的实施例的元素可以用硬件来实现，反之亦然。

鉴于可应用本发明的原理的许多可能的实施例，要求保护落入所附权利要求书及其等效技术方案的范围和精神之内的所有这样的实施例作为本发明。



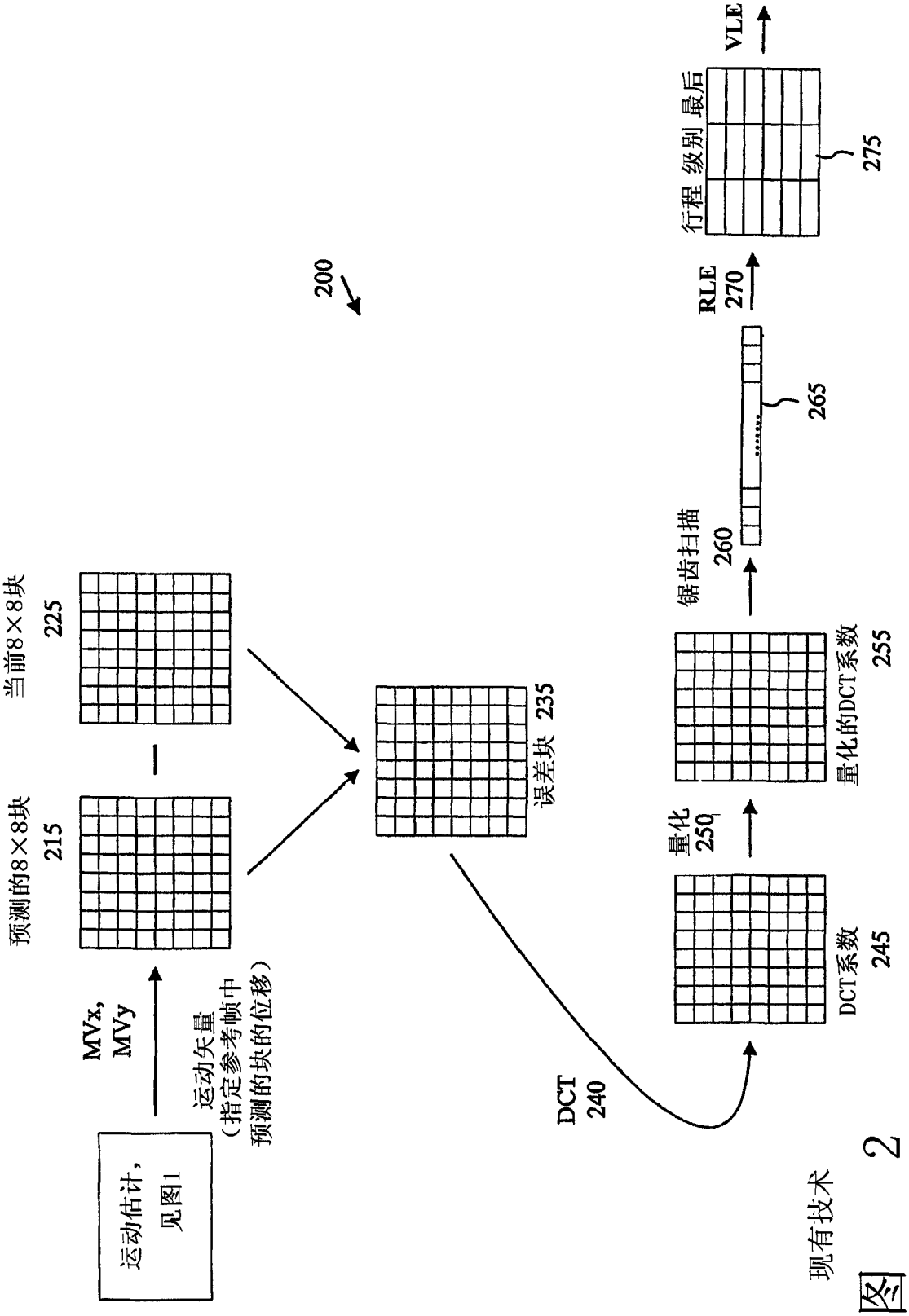
现有技术

图 1

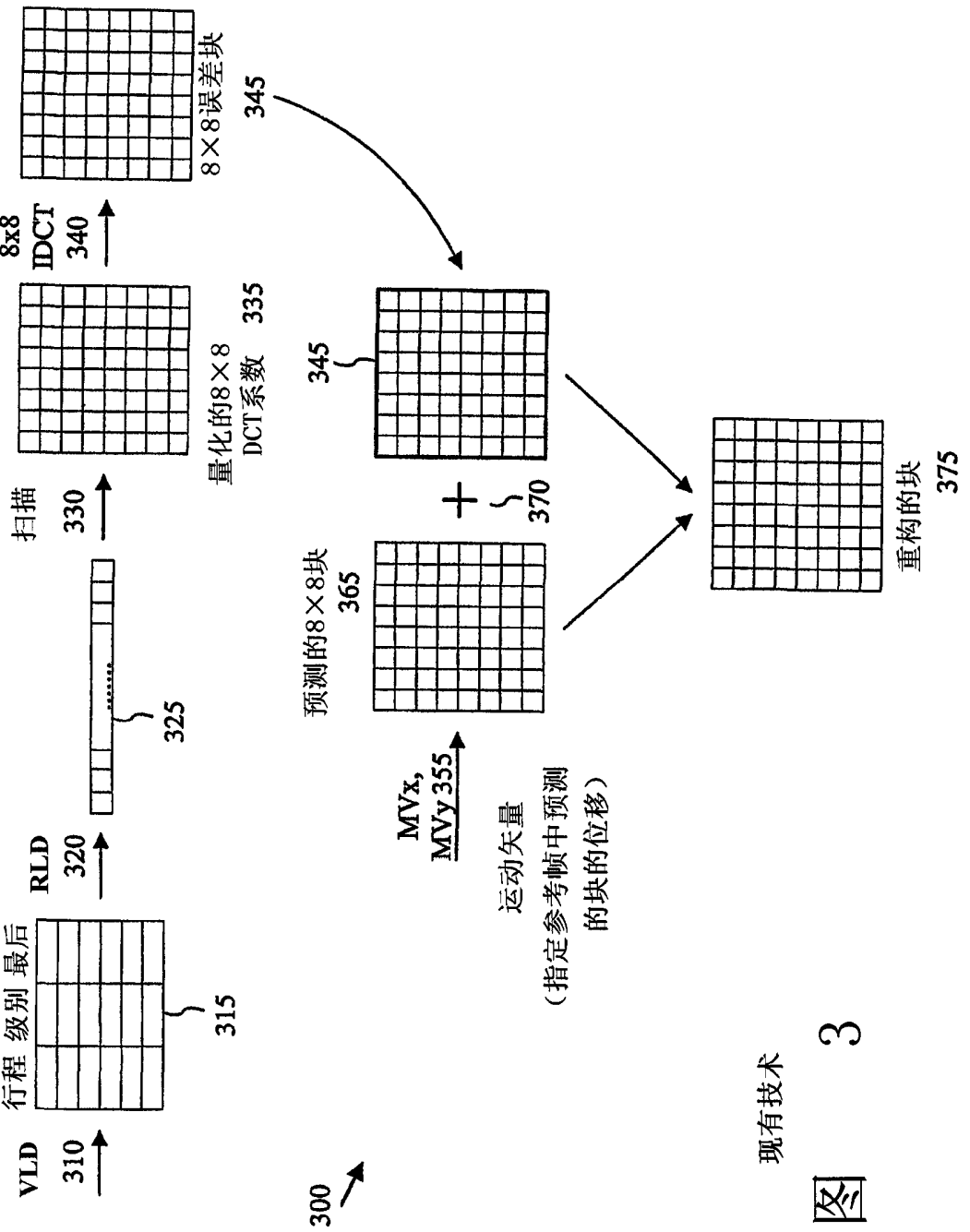


现有技术

图 4

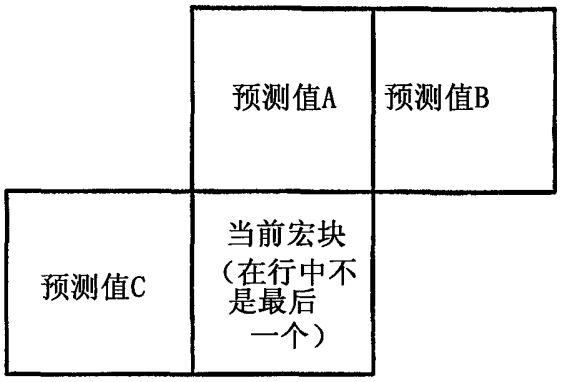

200
运动估计，
见图1
MVx,
MVy
运动矢量
（指定参考帧中
预测的块的位移）
预测的8×8块
215
当前8×8块
225
误差块 235
DCT
240
DCT系数
245
量化
250
量化的DCT系数
255
锯齿扫描
260
265
RLE
270
行程 级别 最后
275
VLE

现有技术 图 2


300
VLD
310
行程 级别 最后
315
RLD
320
325
反锯齿
扫描
330
量化的8×8
DCT系数
335
反量化/
8x8
IDCT
340
8×8误差块
345
预测的8×8块
365
MVx,
MVy 355
运动矢量
(指定参考帧中预测
的块的位移)
345
370
重构的块
375

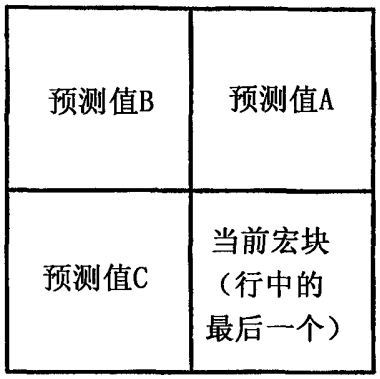
现有技术

图 3



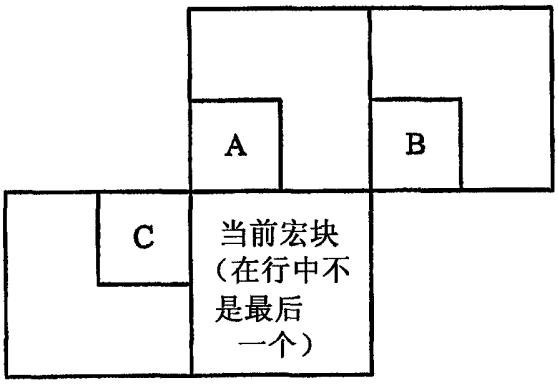
现有技术

图 5A



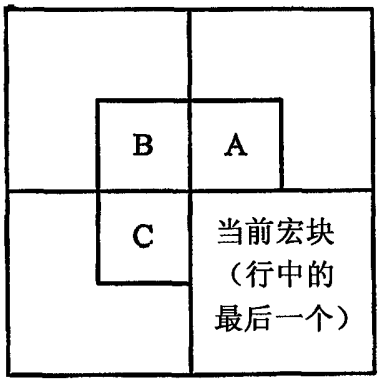
现有技术

图 5B



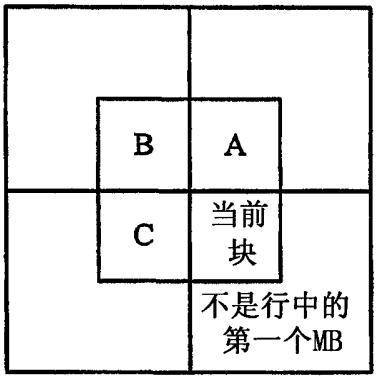
现有技术

图 6A



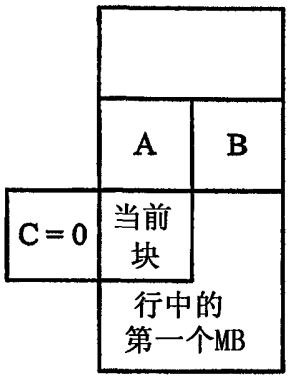
现有技术

图 6B



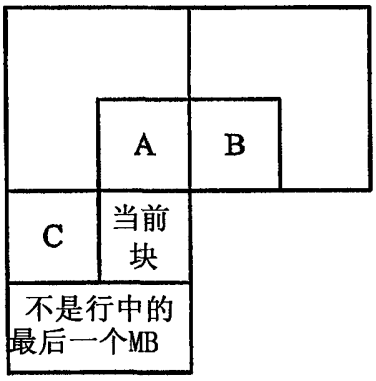
现有技术

图 7A



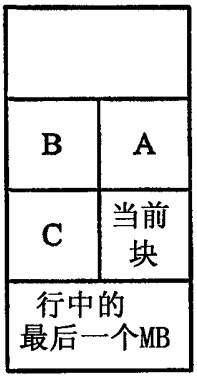
现有技术

图 7B



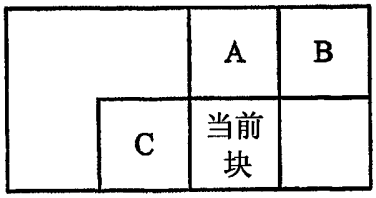
现有技术

图 8A



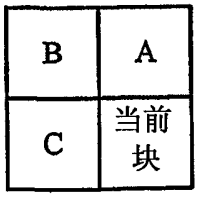
现有技术

图 8B



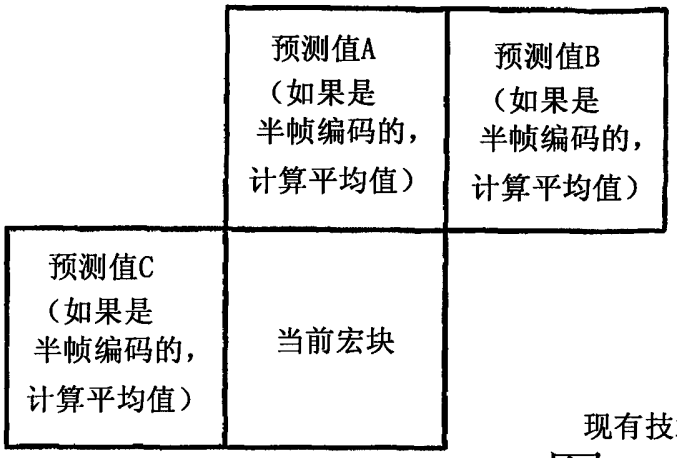
现有技术

图 9



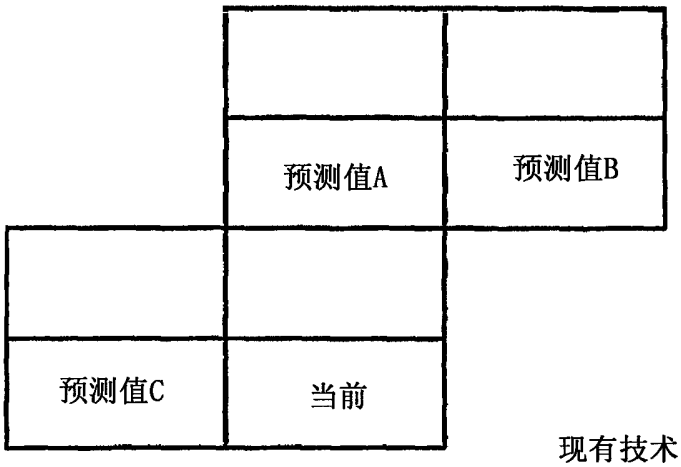
现有技术

图 10



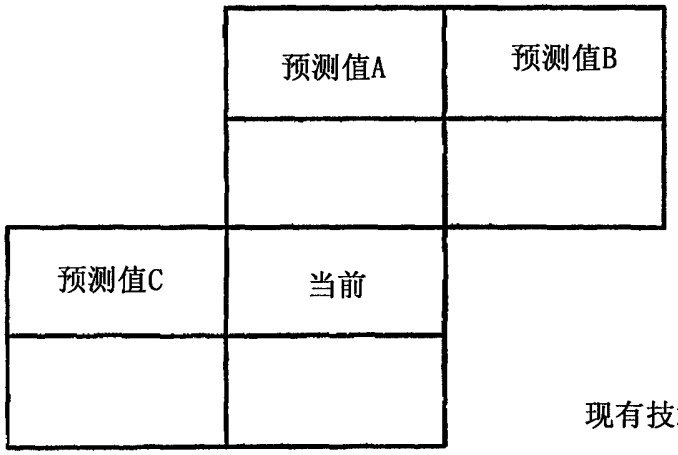
现有技术

图 11



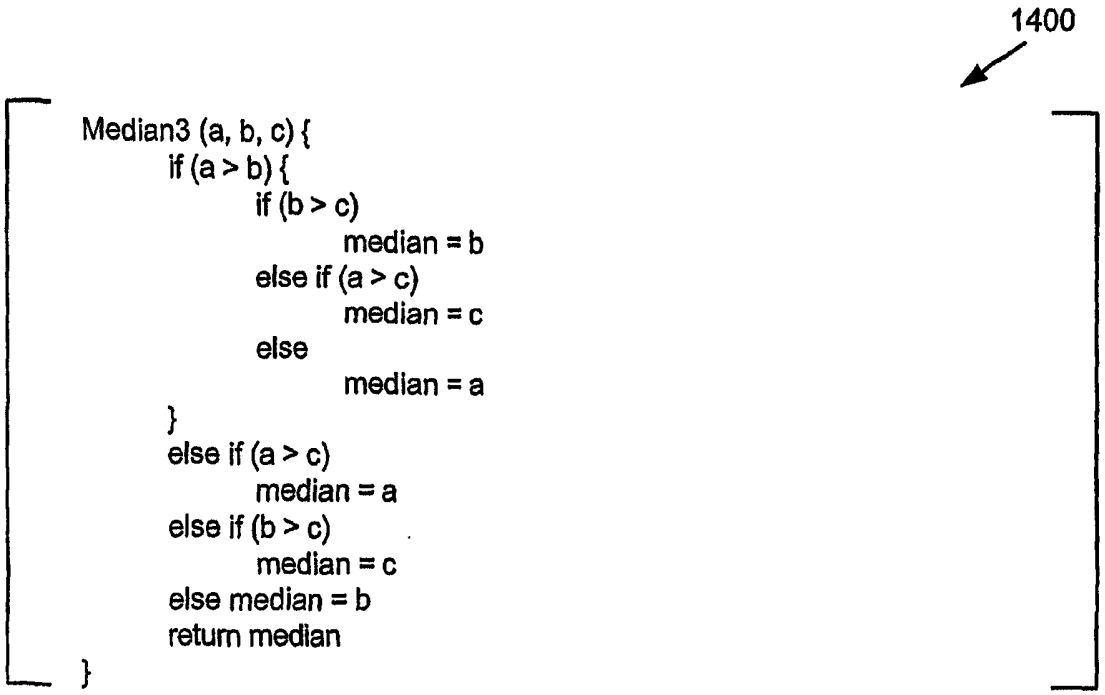
现有技术

图 12



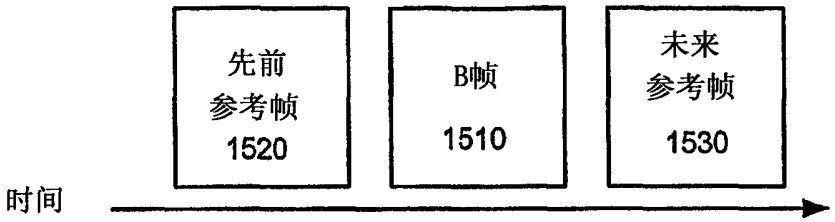
现有技术

图 13


1400
Median3 (a, b, c) {
if (a > b) {
if (b > c)
median = b
else if (a > c)
median = c
else
median = a
}
else if (a > c)
median = a
else if (b > c)
median = c
else median = b
return median
}

现有技术

图 14



现有技术

图 15

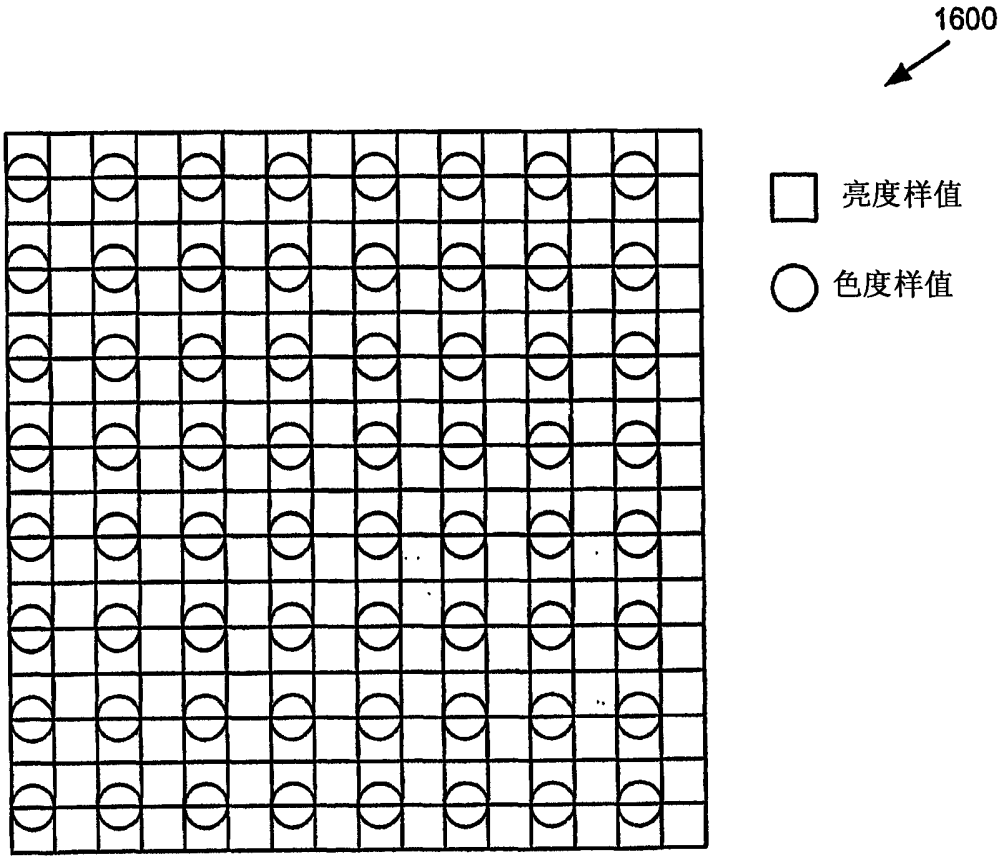


图 16

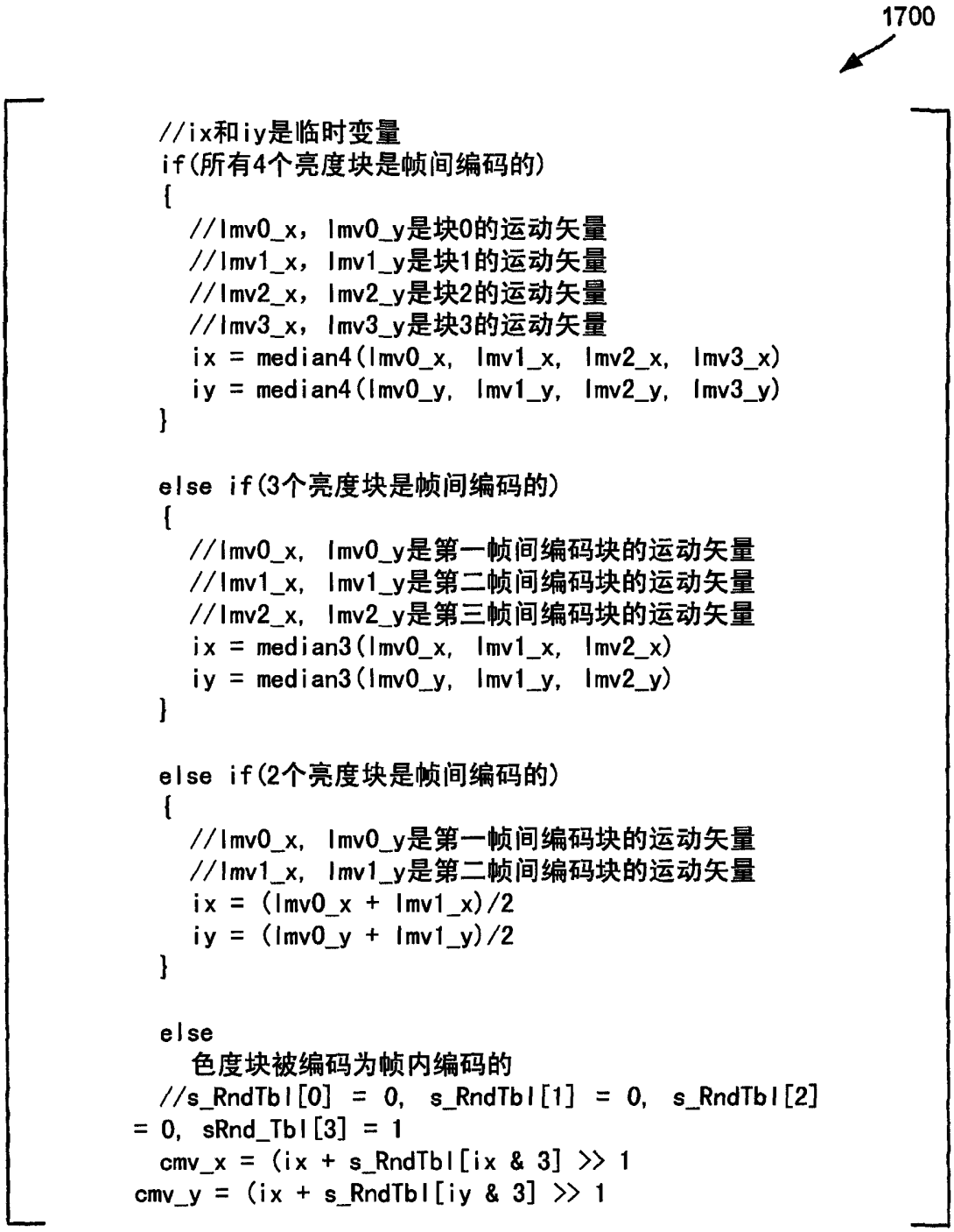

1700
//ix和iy是临时变量
if(所有4个亮度块是帧间编码的)
{
//lmv0_x, lmv0_y是块0的运动矢量
//lmv1_x, lmv1_y是块1的运动矢量
//lmv2_x, lmv2_y是块2的运动矢量
//lmv3_x, lmv3_y是块3的运动矢量
ix = median4(lmv0_x, lmv1_x, lmv2_x, lmv3_x)
iy = median4(lmv0_y, lmv1_y, lmv2_y, lmv3_y)
}
else if(3个亮度块是帧间编码的)
{
//lmv0_x, lmv0_y是第一帧间编码块的运动矢量
//lmv1_x, lmv1_y是第二帧间编码块的运动矢量
//lmv2_x, lmv2_y是第三帧间编码块的运动矢量
ix = median3(lmv0_x, lmv1_x, lmv2_x)
iy = median3(lmv0_y, lmv1_y, lmv2_y)
}
else if(2个亮度块是帧间编码的)
{
//lmv0_x, lmv0_y是第一帧间编码块的运动矢量
//lmv1_x, lmv1_y是第二帧间编码块的运动矢量
ix = (lmv0_x + lmv1_x)/2
iy = (lmv0_y + lmv1_y)/2
}
else
色度块被编码为帧内编码的
//s_RndTbl[0] = 0, s_RndTbl[1] = 0, s_RndTbl[2] = 0, sRnd_Tbl[3] = 1
cmv_x = (ix + s_RndTbl[ix & 3] >> 1
cmv_y = (ix + s_RndTbl[iy & 3] >> 1

图 17

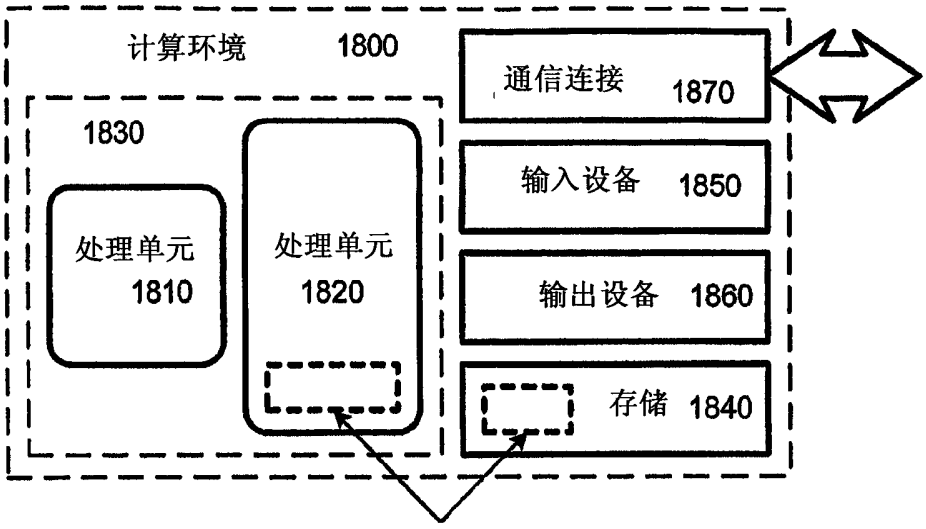

计算环境
1800
1830
处理单元
1810
处理单元
1820
通信连接
1870
输入设备
1850
输出设备
1860
存储
1840
用所描述的技术和工具实现
视频编码器或解码器的软件1880

图 18

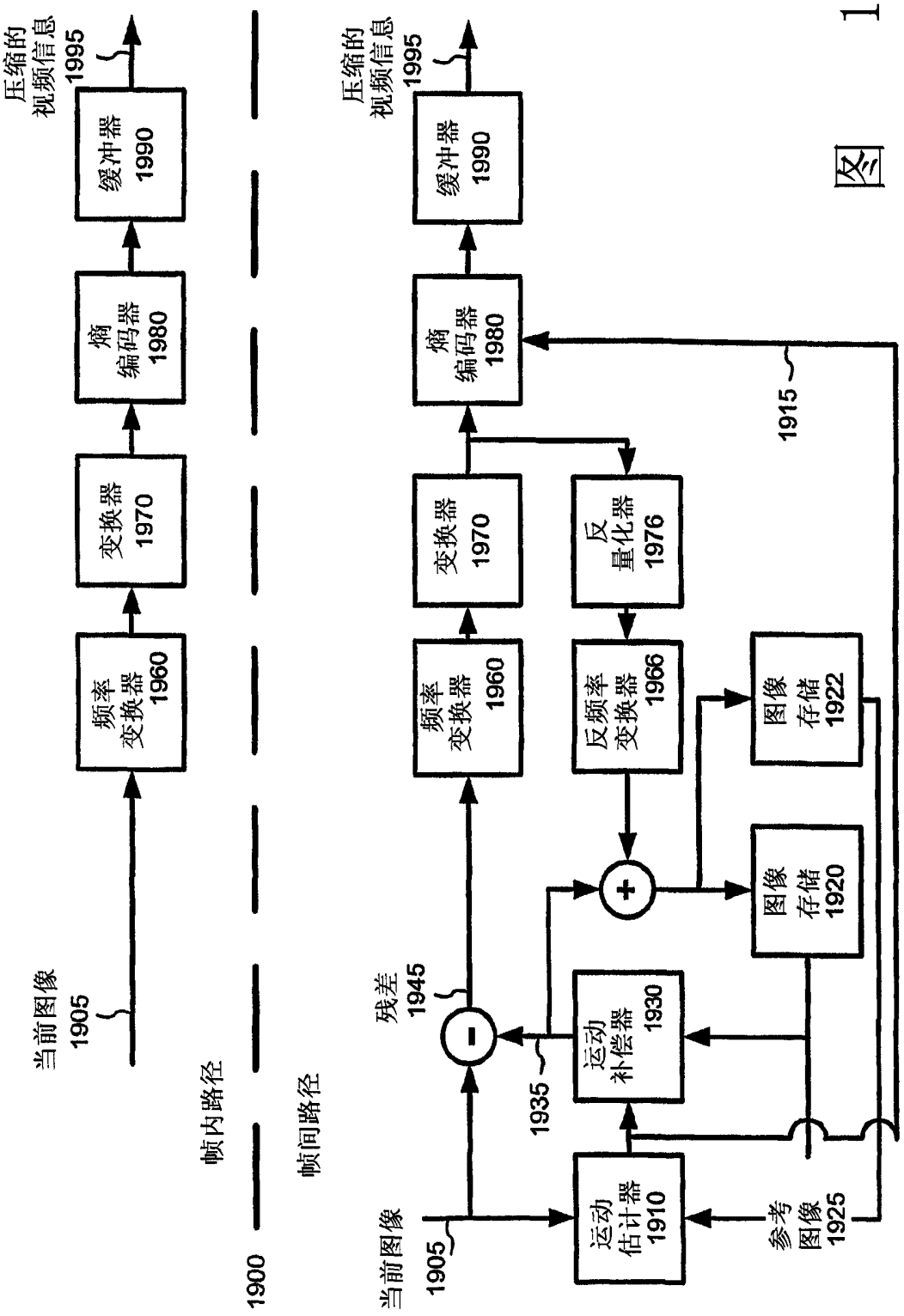

当前图像
1905
频率变换器
1960
变换器
1970
熵编码器
1980
缓冲器
1990
压缩的视频信息
1995
帧内路径
帧间路径
1900
当前图像
1905
残差
1945
频率变换器
1960
变换器
1970
熵编码器
1980
缓冲器
1990
压缩的视频信息
1995
反量化器
1976
反频率变换器
1966
1915
图像存储
1922
图像存储
1920
运动补偿器
1930
1935
运动估计器
1910
参考图像
1925

图 19

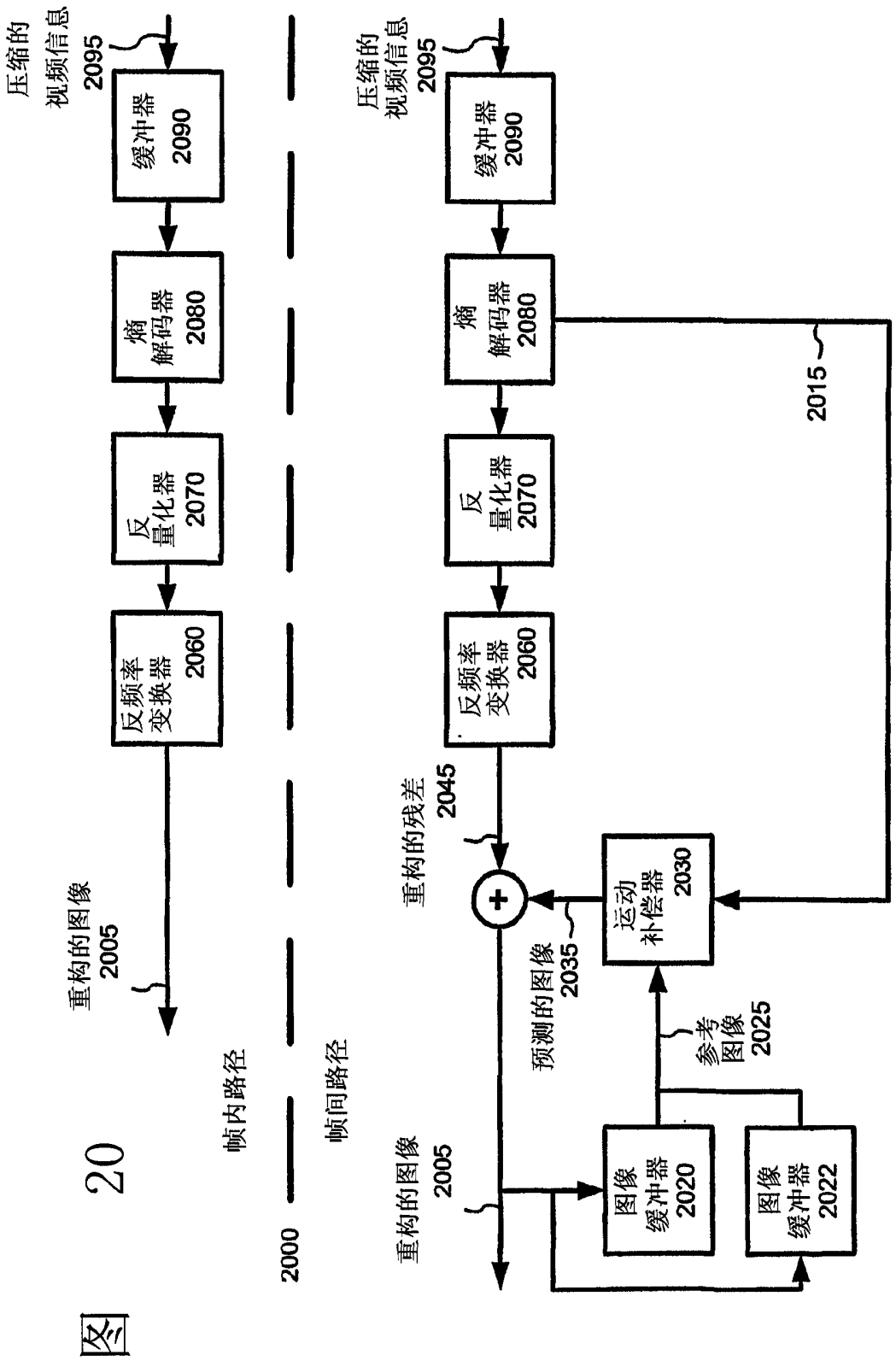

压缩的视频信息 2095
缓冲器 2090
熵解码器 2080
反量化器 2070
反频率变换器 2060
重构的图像 2005
帧内路径
帧间路径
2000
重构的残差 2045
预测的图像 2035
运动补偿器 2030
参考图像 2025
图像缓冲器 2020
图像缓冲器 2022
2015

图 20

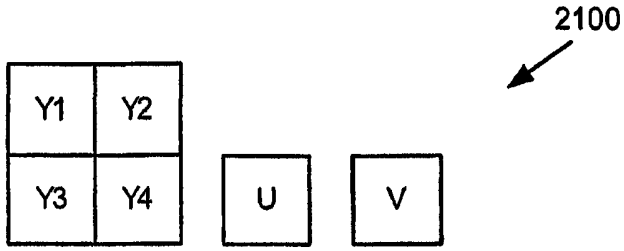


图 21

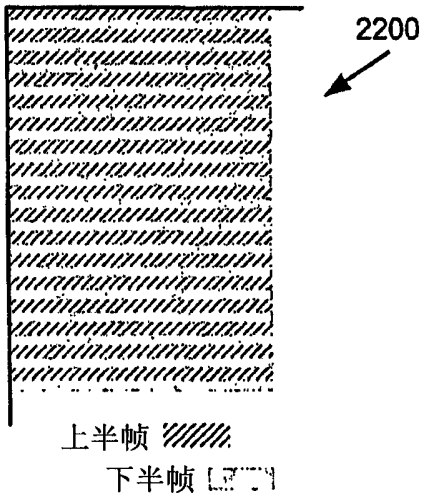


图 22A

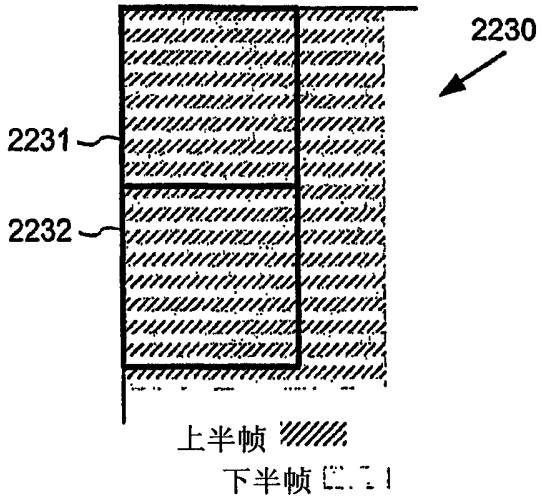


图 22B

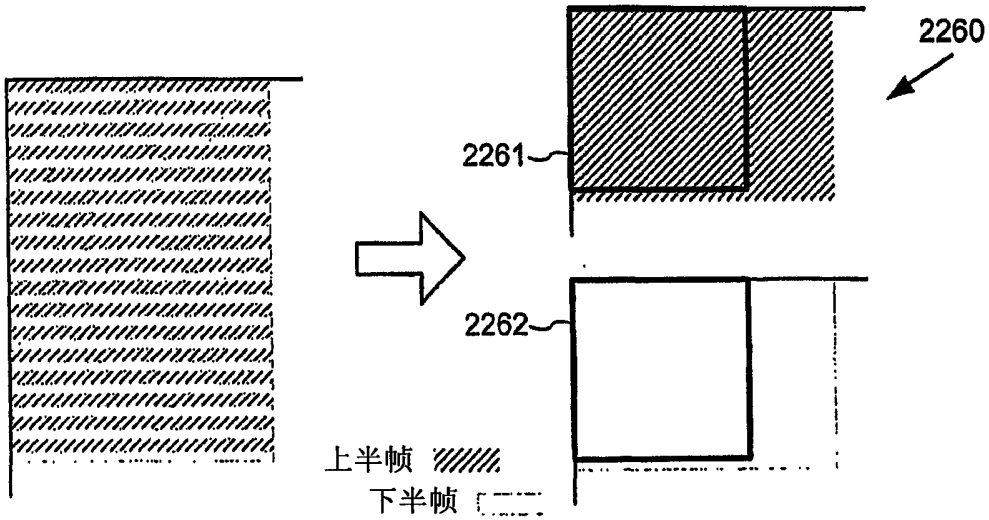


图 22C

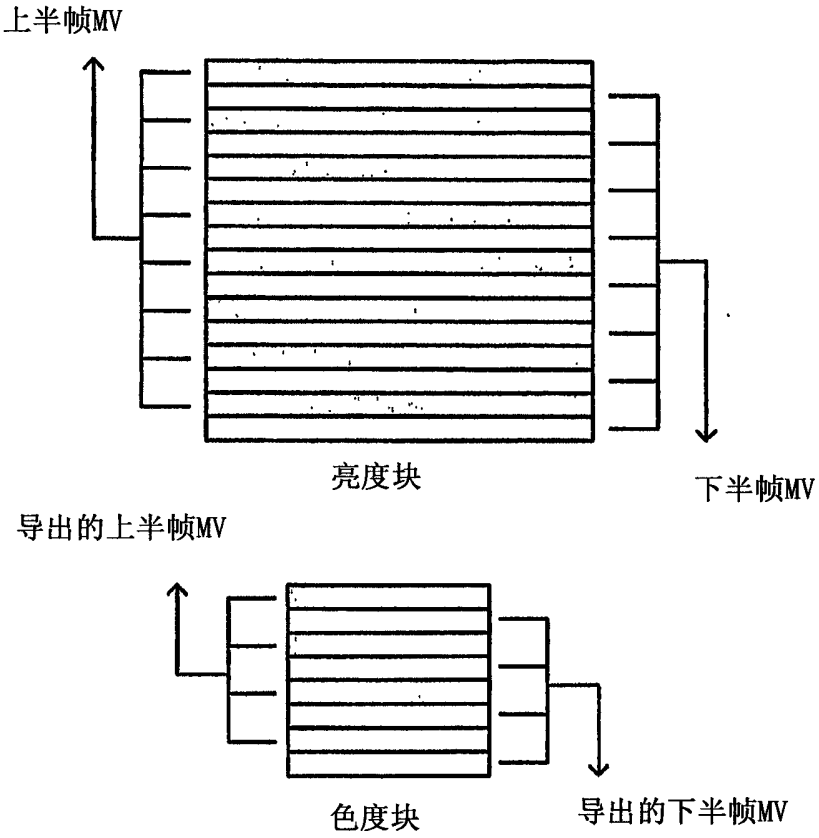

上半帧MV
亮度块
下半帧MV
导出的上半帧MV
色度块
导出的下半帧MV

图 23

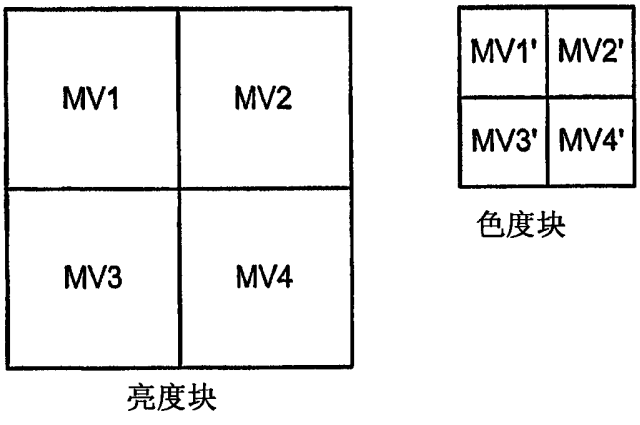
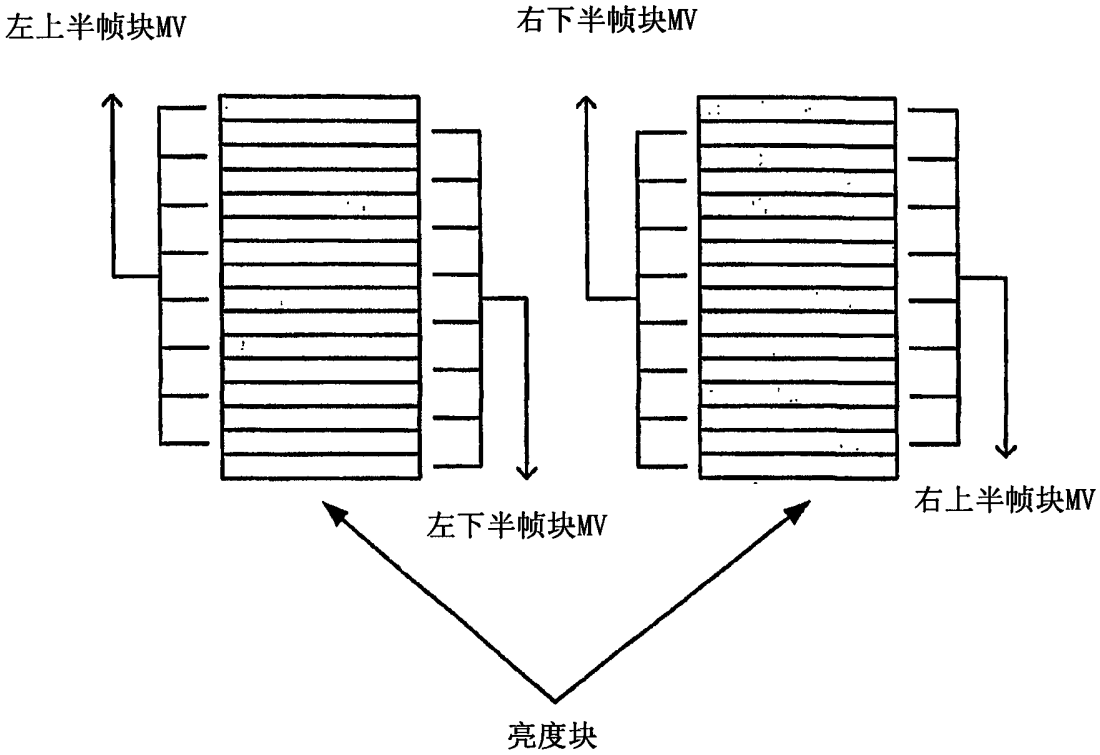

MV1
MV2
MV3
MV4
亮度块
MV1'
MV2'
MV3'
MV4'
色度块

图 24


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

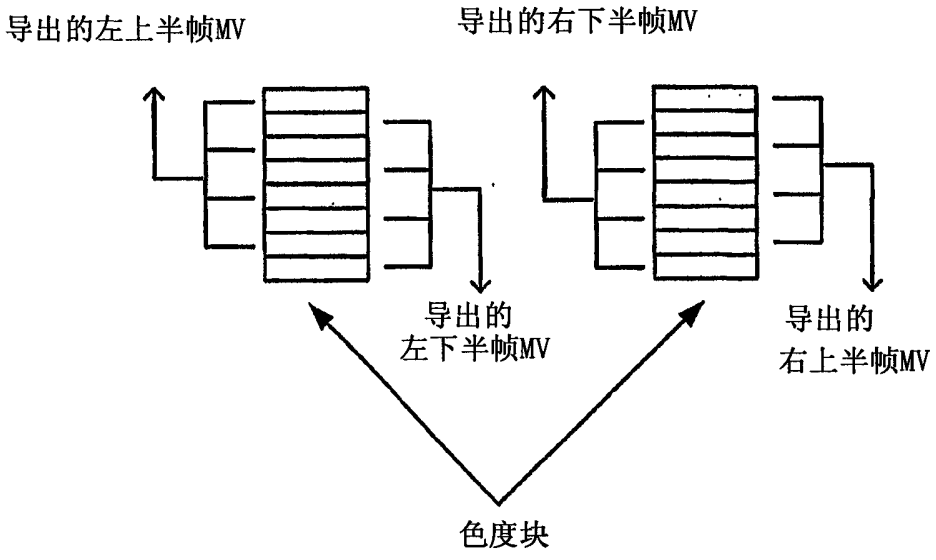

导出的左上半帧MV
导出的右下半帧MV
导出的
左下半帧MV
导出的
右上半帧MV
色度块

图 25

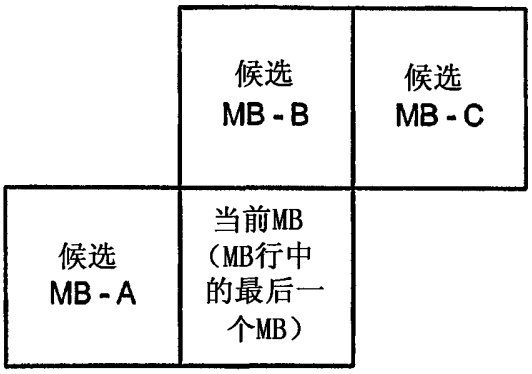


图 26A

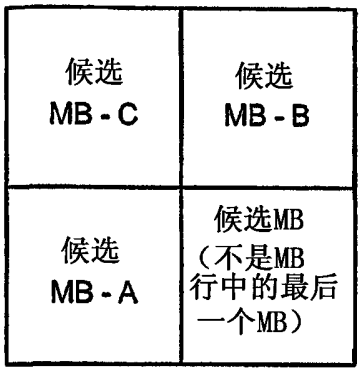


图 26B

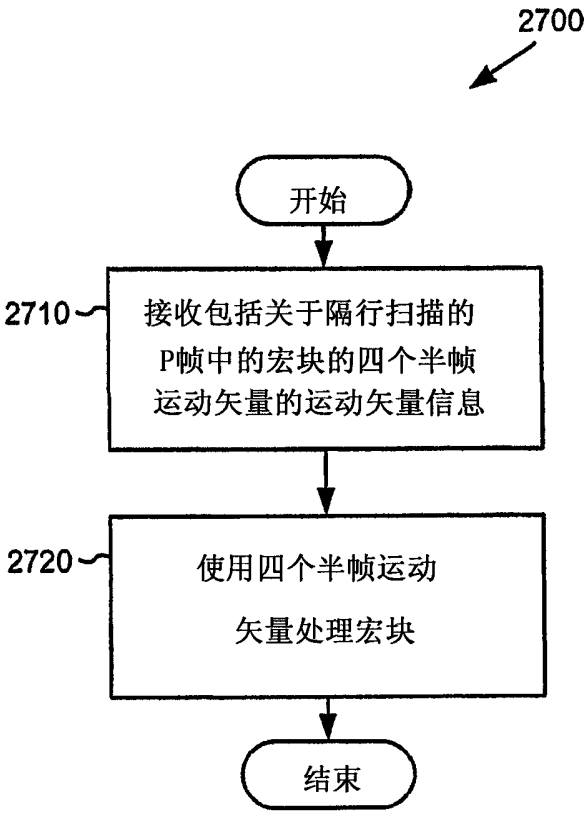


图 27

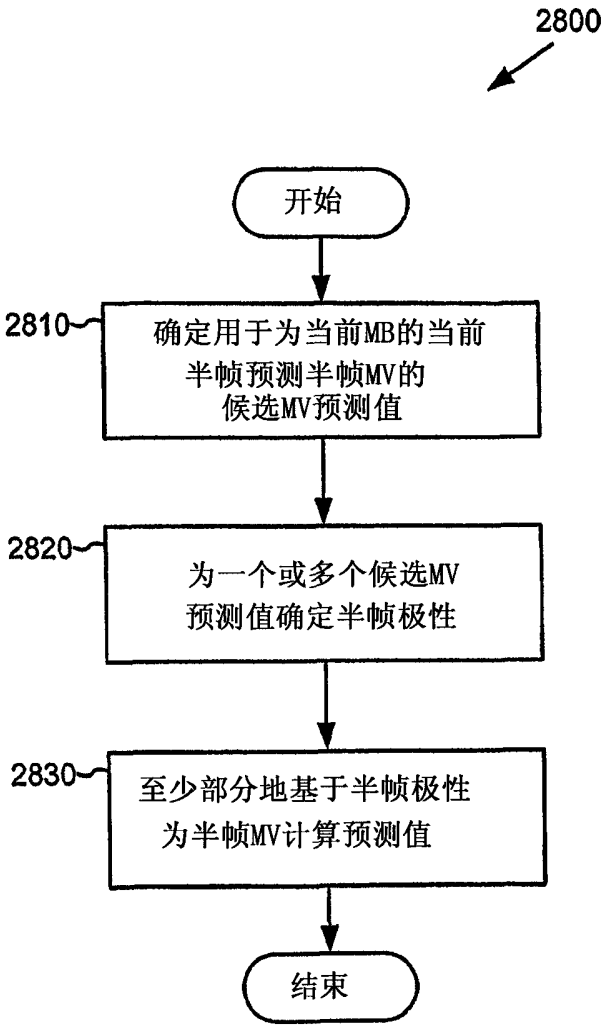

2800
开始
2810
确定用于为当前MB的当前半帧预测半帧MV的候选MV预测值
2820
为一个或多个候选MV预测值确定半帧极性
2830
至少部分地基于半帧极性为半帧MV计算预测值
结束

图 28

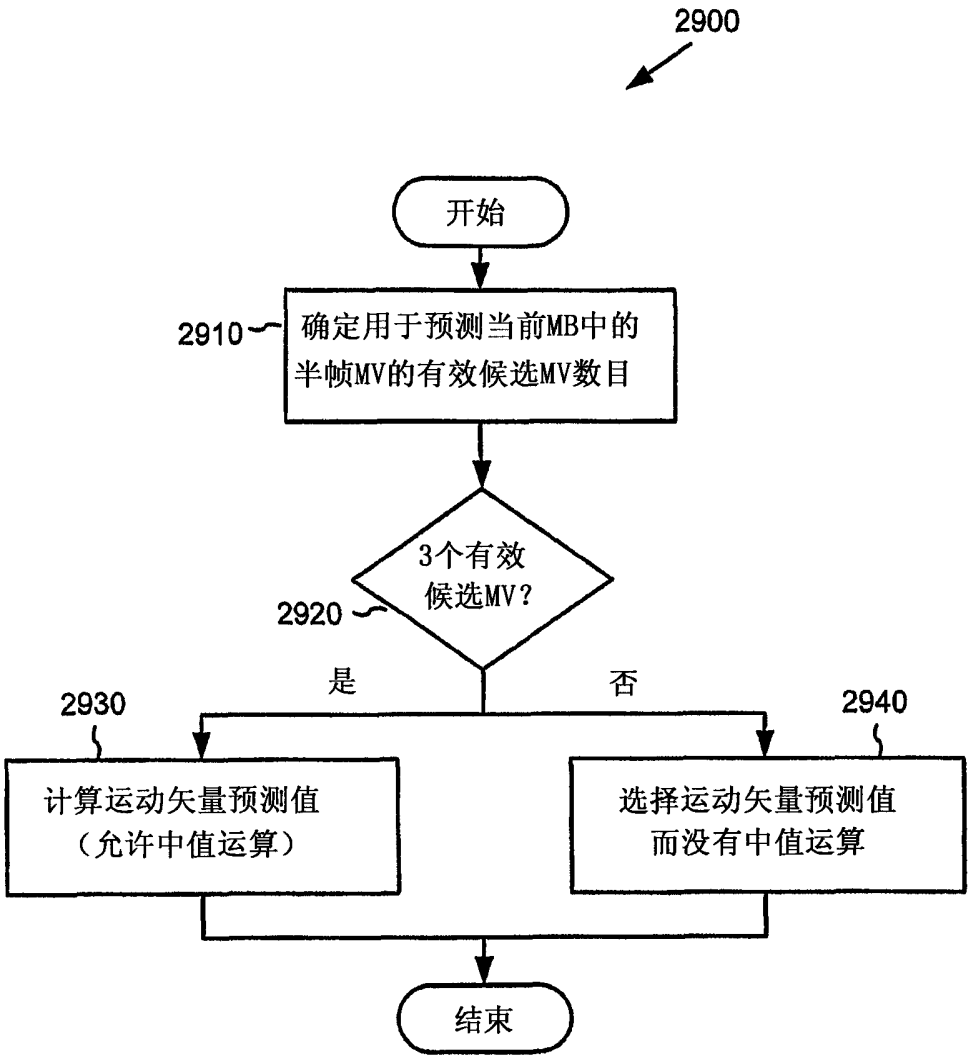

2900
开始
2910
确定用于预测当前MB中的半帧MV的有效候选MV数目
3个有效候选MV?
2920
是
否
2930
2940
计算运动矢量预测值（允许中值运算）
选择运动矢量预测值而没有中值运算
结束

图 29

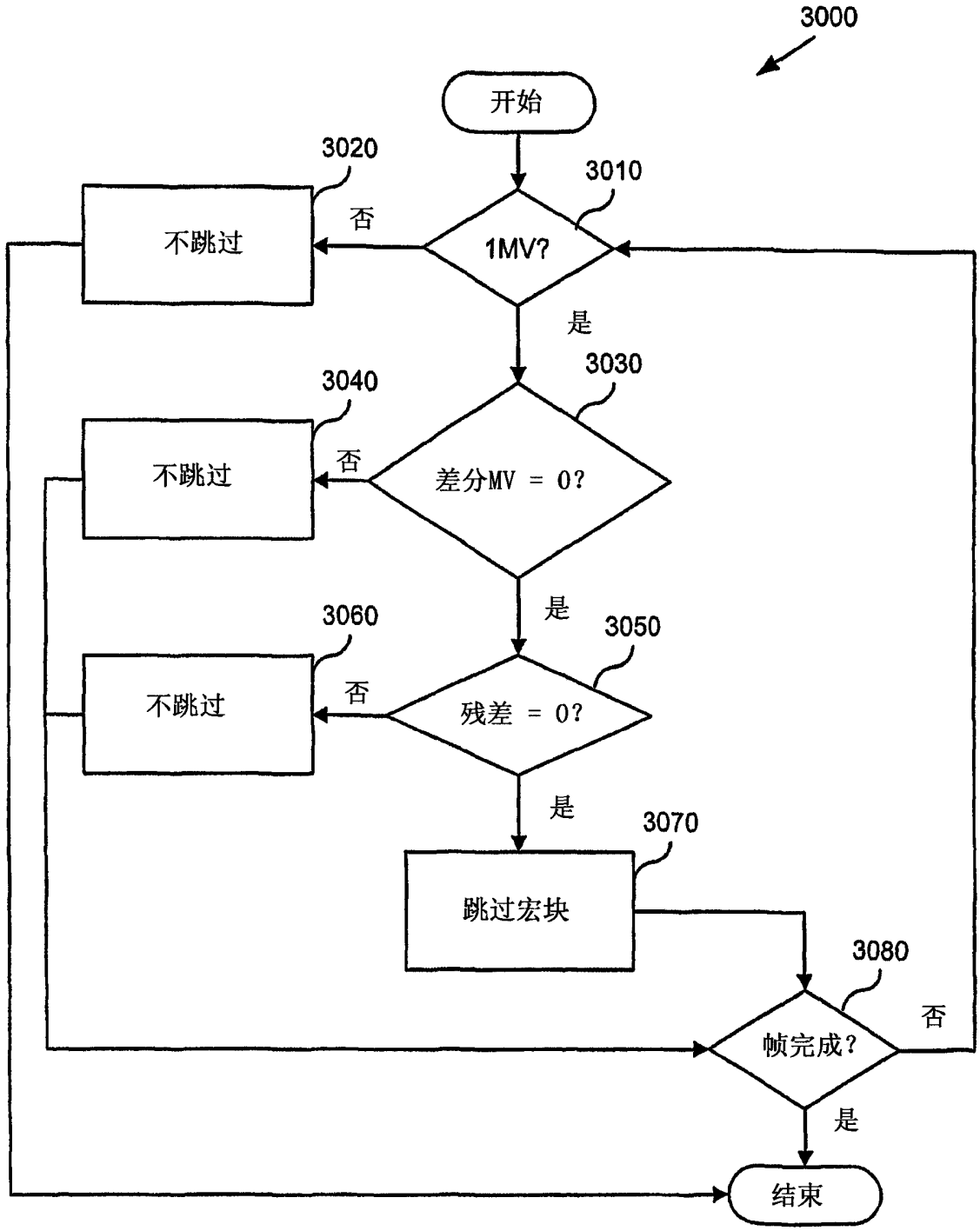

3000
开始
3010
1MV?
否
3020
不跳过
是
3030
差分MV = 0?
否
3040
不跳过
是
3050
残差 = 0?
否
3060
不跳过
是
3070
跳过宏块
3080
帧完成?
否
是
结束

图 30

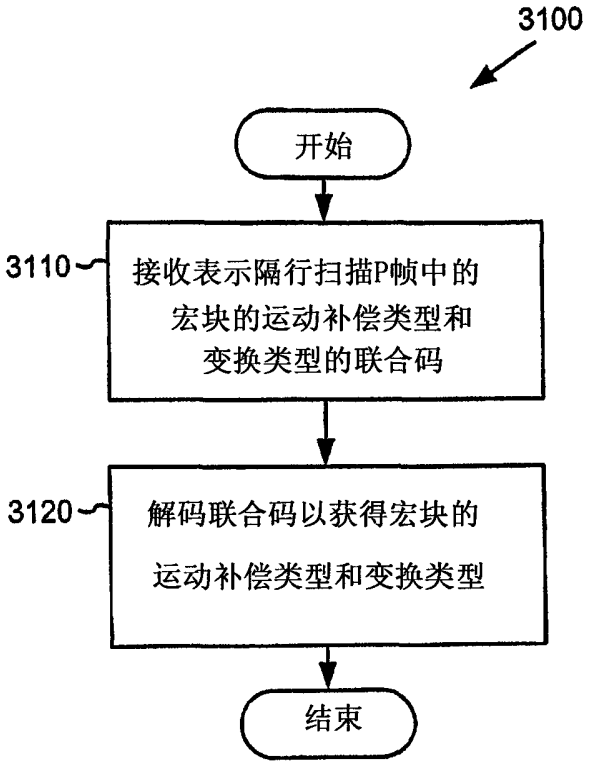

3100
开始
3110
接收表示隔行扫描P帧中的宏块的运动补偿类型和变换类型的联合码
3120
解码联合码以获得宏块的运动补偿类型和变换类型
结束

图 31

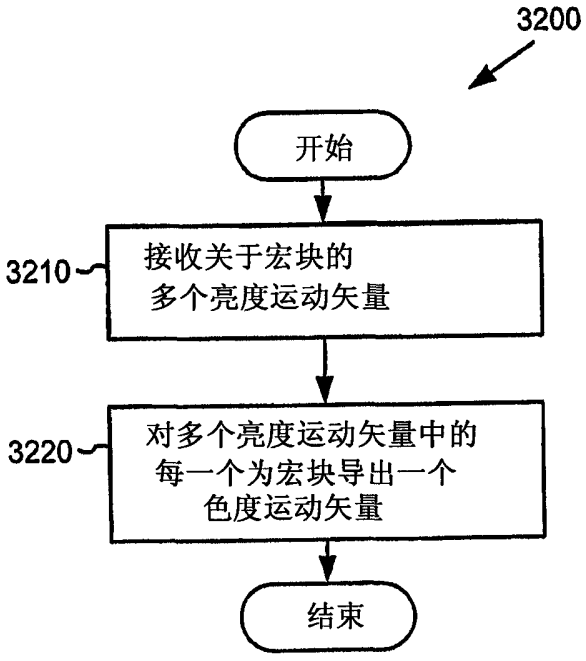

3200
开始
3210
接收关于宏块的多个亮度运动矢量
3220
对多个亮度运动矢量中的每一个为宏块导出一个色度运动矢量
结束

图 32

图 33

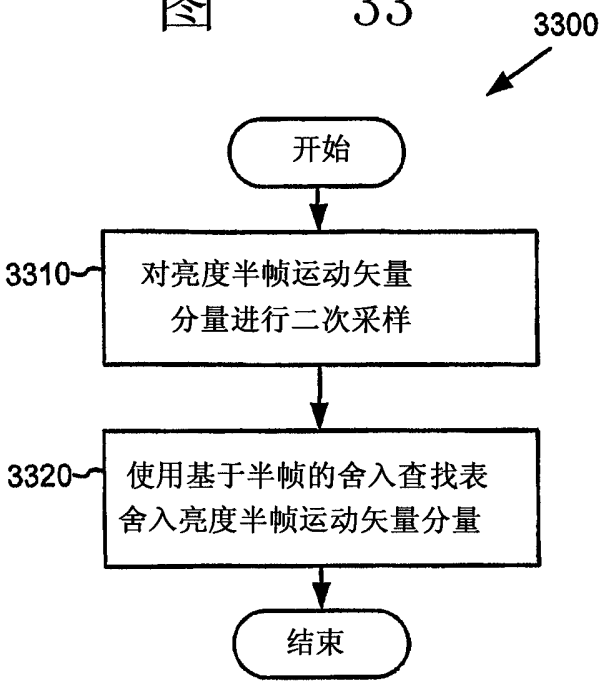
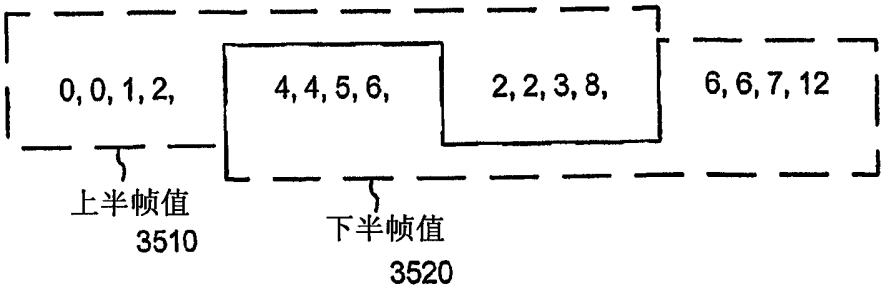


图 34

3400

```
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是半帧运动矢量)
    CMVY = (LMVY >> 4)*8 + s_RndTblField [LMVY & 0xF];
  } else {
  CMVY = (LMVY + s_RndTbl[LMVY & 3]) >> 1;
  }
```

图 35



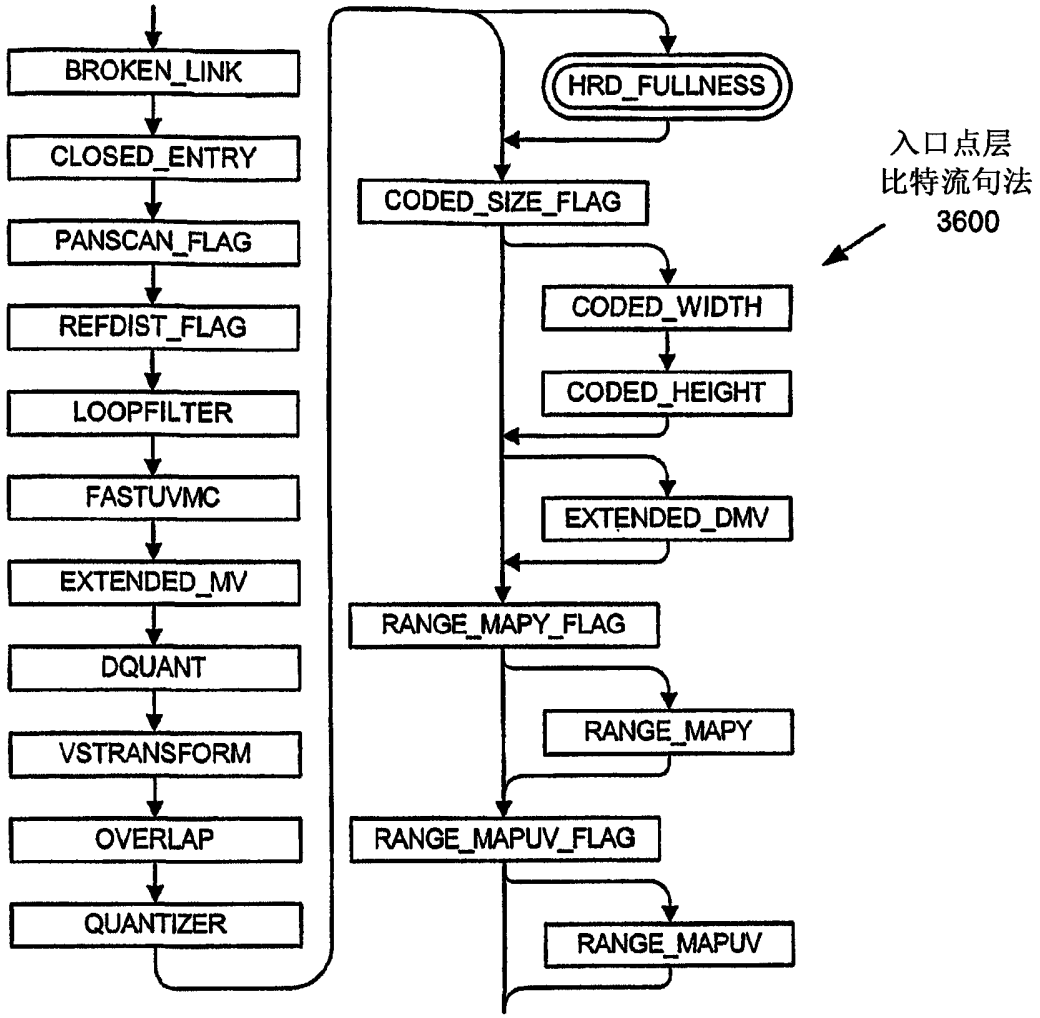

BROKEN_LINK
CLOSED_ENTRY
PANSCAN_FLAG
REFDIST_FLAG
LOOPFILTER
FASTUVMC
EXTENDED_MV
DQUANT
VSTRANSFORM
OVERLAP
QUANTIZER
HRD_FULLNESS
CODED_SIZE_FLAG
CODED_WIDTH
CODED_HEIGHT
EXTENDED_DMV
RANGE_MAPY_FLAG
RANGE_MAPY
RANGE_MAPUV_FLAG
RANGE_MAPUV
入口点层
比特流句法
3600

图 36

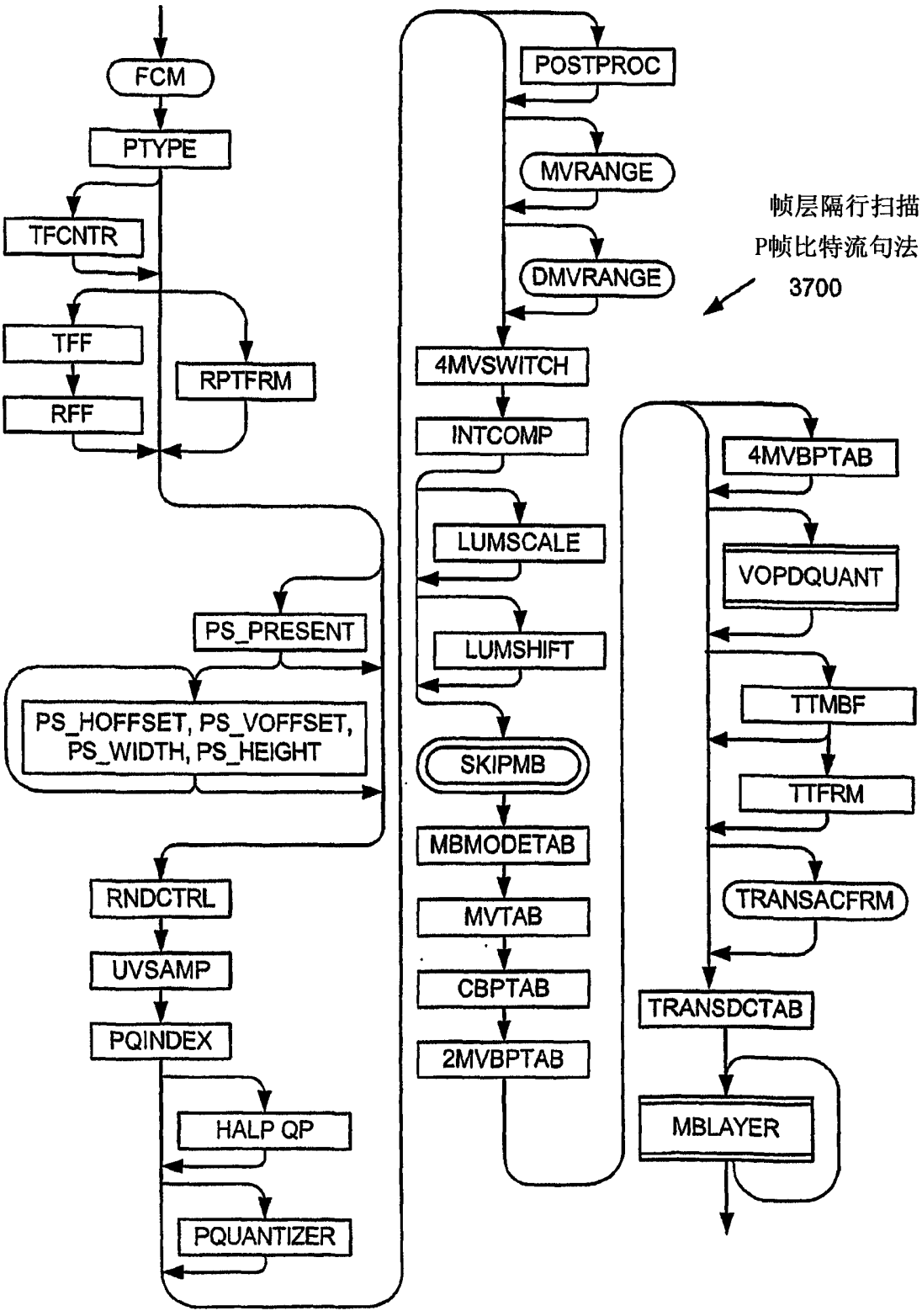

FCM
PTYPE
TFCNTR
TFF
RFF
RPTFRM
PS_PRESENT
PS_HOFFSET, PS_VOFFSET,
PS_WIDTH, PS_HEIGHT
RNDCTRL
UVSAMP
PQINDEX
HALP QP
PQUANTIZER
POSTPROC
MVRANGE
DMVRANGE
4MVSWITCH
INTCOMP
LUMSCALE
LUMSHIFT
SKIPMB
MBMODETAB
MVTAB
CBPTAB
2MVBPTAB
帧层隔行扫描
P帧比特流句法
3700
4MVBPTAB
VOPDQUANT
TTMBF
TTFRM
TRANSACFRM
TRANSDCTAB
MBLAYER

图 37

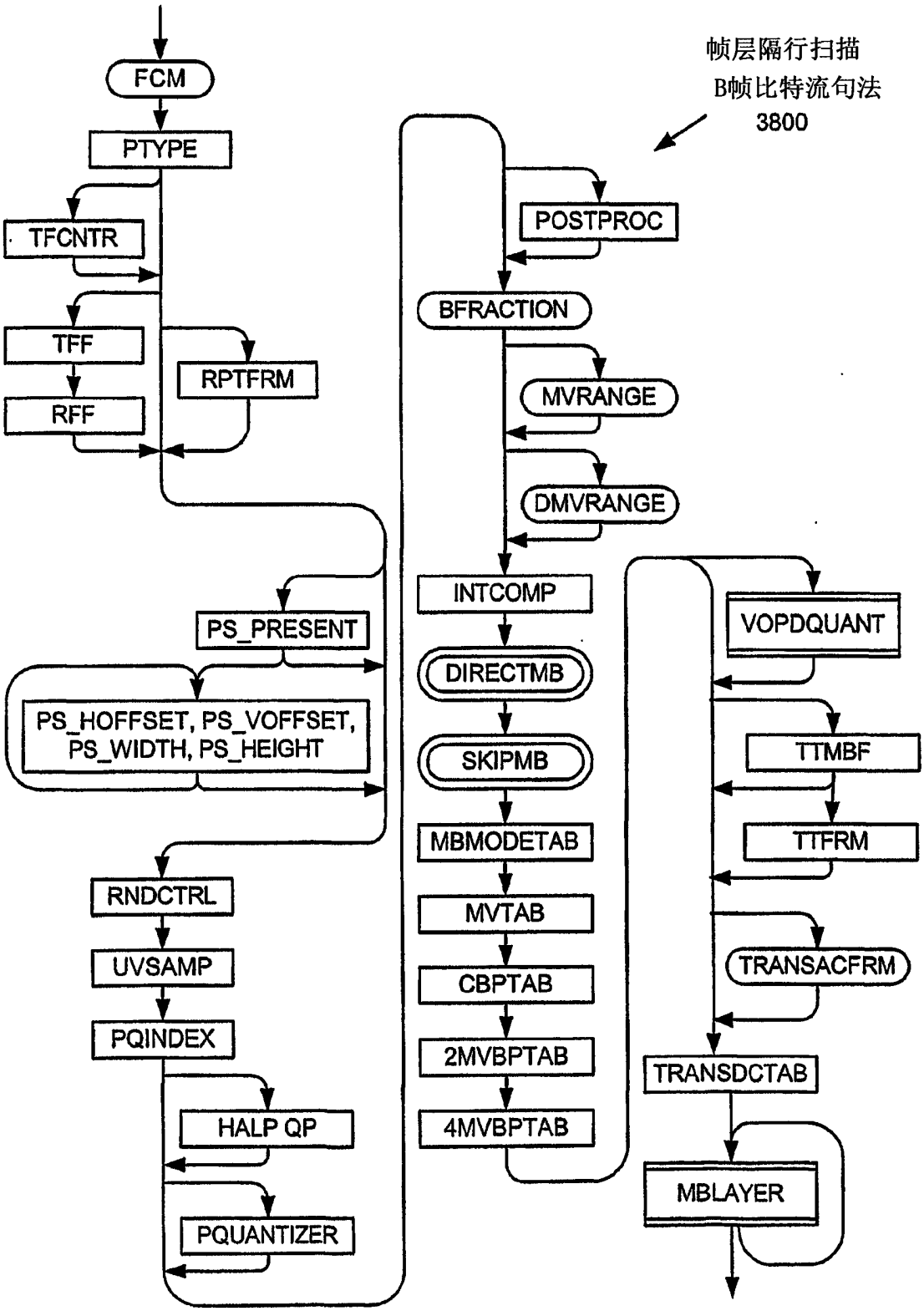

帧层隔行扫描
B帧比特流句法
3800
FCM
PTYPE
TFCNTR
TFF
RFF
RPTFRM
PS_PRESENT
PS_HOFFSET, PS_VOFFSET,
PS_WIDTH, PS_HEIGHT
RNDCTRL
UVSAMP
PQINDEX
HALP QP
PQUANTIZER
POSTPROC
BFRACTION
MVRANGE
DMVRANGE
INTCOMP
DIRECTMB
SKIPMB
MBMODETAB
MVTAB
CBPTAB
2MVBPTAB
4MVBPTAB
VOPDQUANT
TTMBF
TTFRM
TRANSACFRM
TRANSDCTAB
MBLAYER

图 38

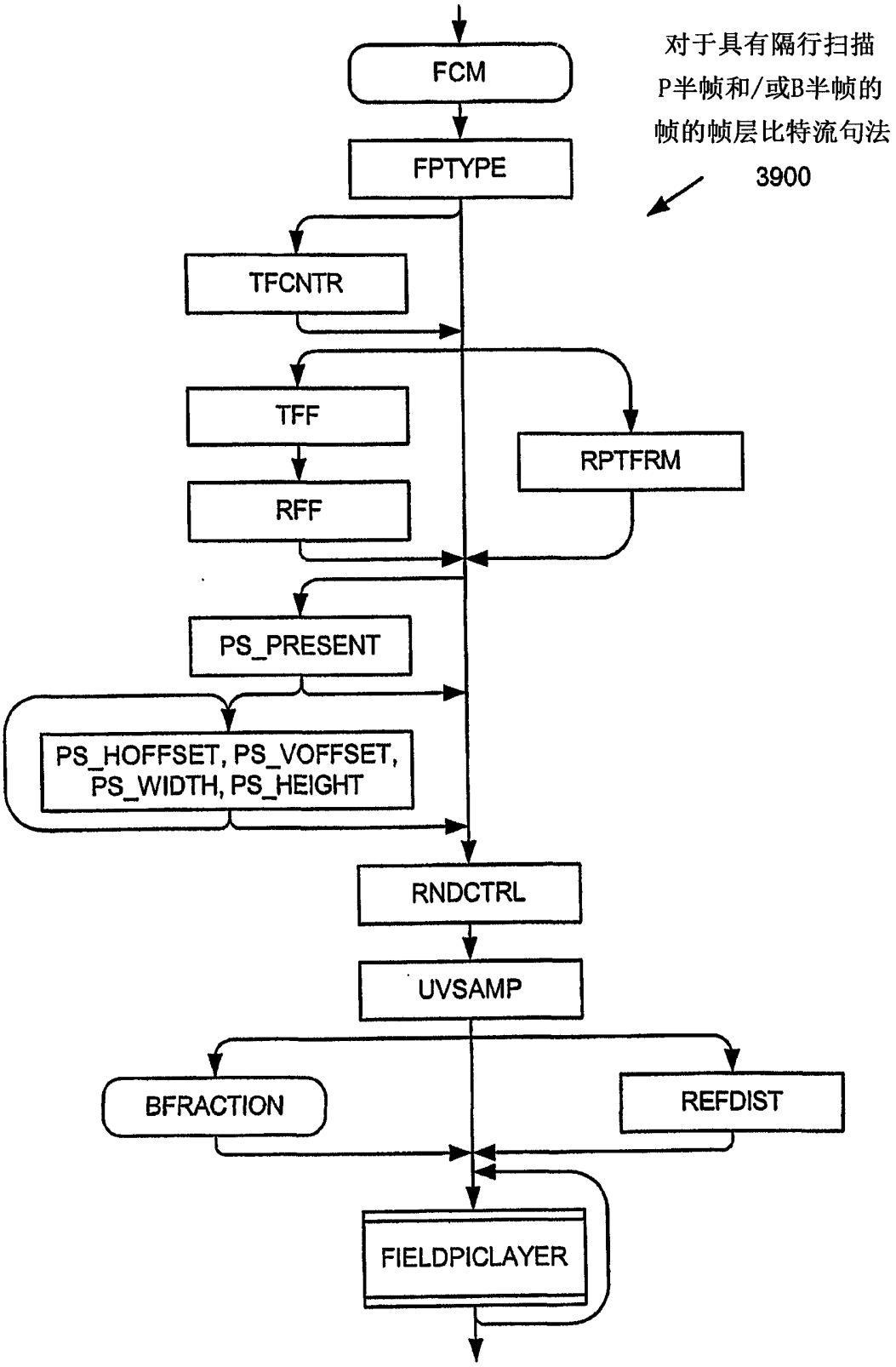

对于具有隔行扫描
P半帧和/或B半帧的
帧的帧层比特流句法
3900
FCM
FPTYPE
TFCNTR
TFF
RFF
RPTFRM
PS_PRESENT
PS_HOFFSET, PS_VOFFSET,
PS_WIDTH, PS_HEIGHT
RNDCTRL
UVSAMP
BFRACTION
REFDIST
FIELDPICLAYER

图 39

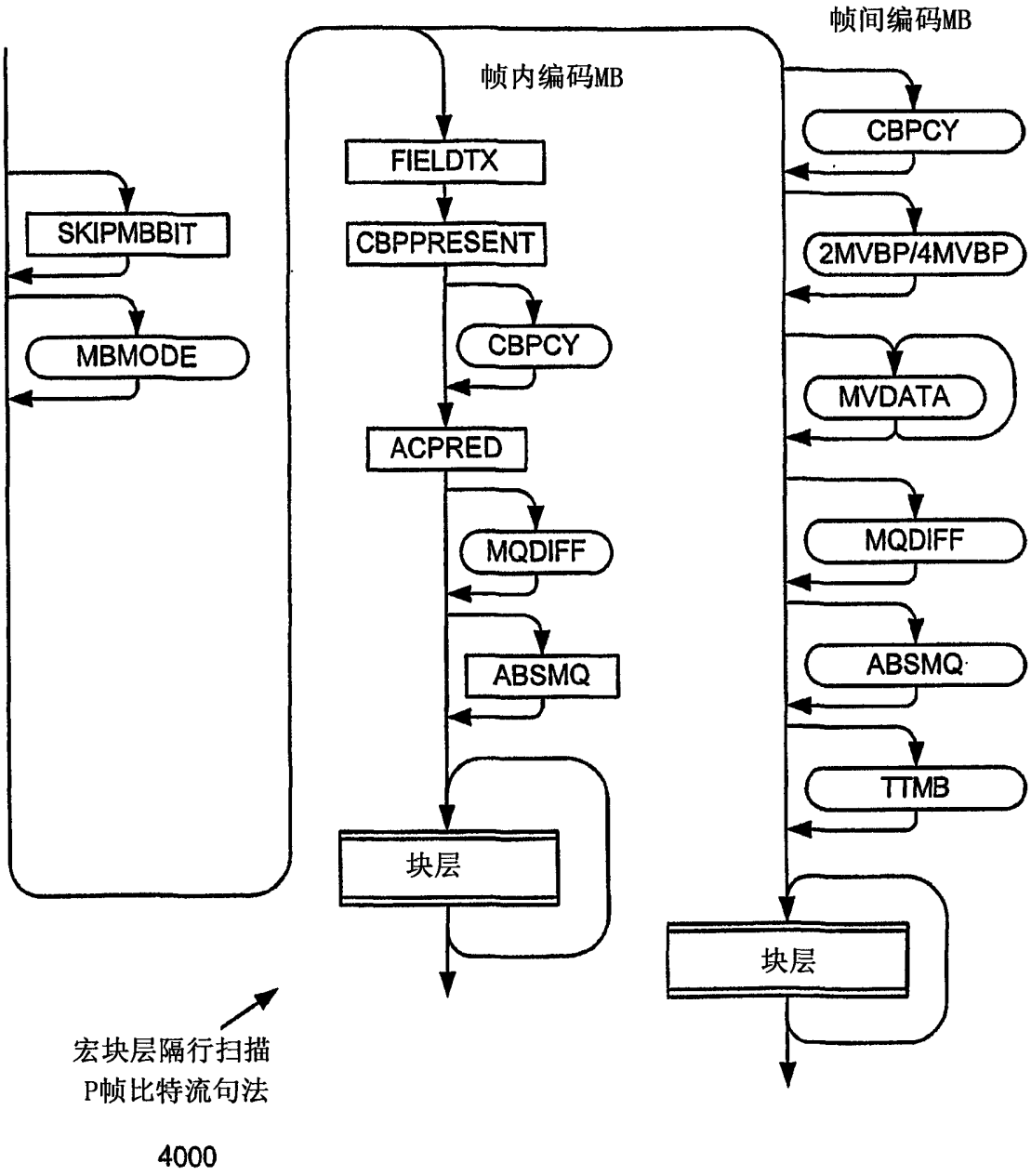

帧间编码MB
帧内编码MB
SKIPMBBIT
MBMODE
FIELDTX
CBPPRESENT
CBPCY
ACPRED
MQDIFF
ABSMQ
块层
CBPCY
2MVBP/4MVBP
MVDATA
MQDIFF
ABSMQ
TTMB
块层
宏块层隔行扫描
P帧比特流句法
4000

图 40

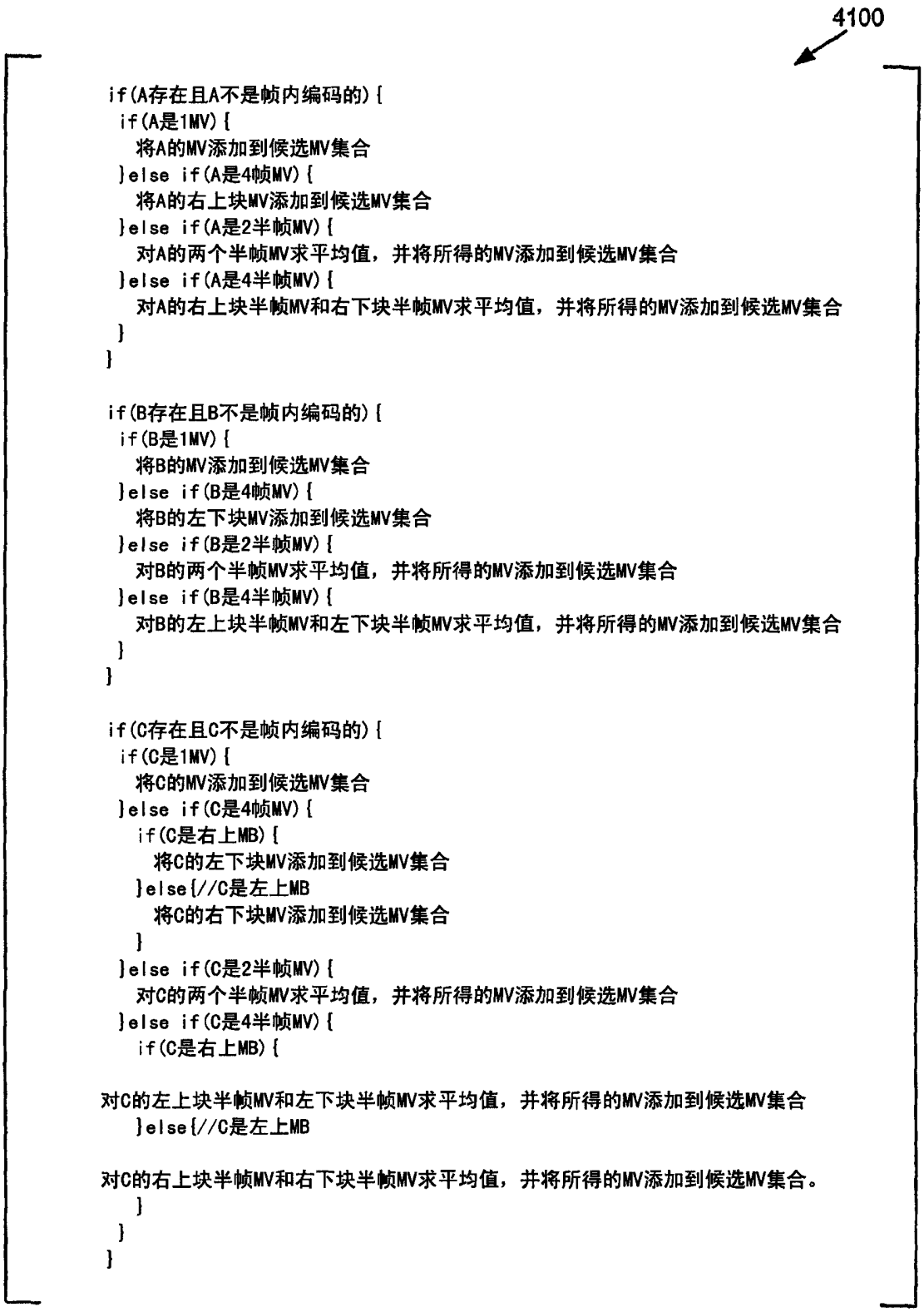

4100
if(A存在且A不是帧内编码的){
if(A是1MV){
将A的MV添加到候选MV集合
}else if(A是4帧MV){
将A的右上块MV添加到候选MV集合
}else if(A是2半帧MV){
对A的两个半帧MV求平均值，并将所得的MV添加到候选MV集合
}else if(A是4半帧MV){
对A的右上块半帧MV和右下块半帧MV求平均值，并将所得的MV添加到候选MV集合
}
}
if(B存在且B不是帧内编码的){
if(B是1MV){
将B的MV添加到候选MV集合
}else if(B是4帧MV){
将B的左下块MV添加到候选MV集合
}else if(B是2半帧MV){
对B的两个半帧MV求平均值，并将所得的MV添加到候选MV集合
}else if(B是4半帧MV){
对B的左上块半帧MV和左下块半帧MV求平均值，并将所得的MV添加到候选MV集合
}
}
if(C存在且C不是帧内编码的){
if(C是1MV){
将C的MV添加到候选MV集合
}else if(C是4帧MV){
if(C是右上MB){
将C的左下块MV添加到候选MV集合
}else{//C是左上MB
将C的右下块MV添加到候选MV集合
}
}else if(C是2半帧MV){
对C的两个半帧MV求平均值，并将所得的MV添加到候选MV集合
}else if(C是4半帧MV){
if(C是右上MB){
对C的左上块半帧MV和左下块半帧MV求平均值，并将所得的MV添加到候选MV集合
}else{//C是左上MB
对C的右上块半帧MV和右下块半帧MV求平均值，并将所得的MV添加到候选MV集合。
}
}
}

图 41

4200

```
//左上块MV
if(A存在且A不是帧内编码的){
 if(A是1MV){
   将A的MV添加到候选MV集合
 }else if(A是4帧MV){
   将A的右上块MV添加到候选MV集合
 }else if(A是2半帧MV){
   对A的两个半帧MV求平均值，并将所得的MV添加到候选MV集合
 }else if(A是4半帧MV){
   对A的右上块半帧MV和右下块半帧MV求平均值，并将所得的MV添加到候选MV集合
 }
}

if(B存在且B不是帧内编码的){
 if(B是1MV){
   将B的MV添加到候选MV集合
 }else if(B是4帧MV){
   将B的左下块MV添加到候选MV集合
 }else if(B是2半帧MV){
   对B的两个半帧MV求平均值，并将所得的MV添加到候选MV集合
 }else if(B是4半帧MV){
   对B的左上块半帧MV和左下块半帧MV求平均值，并将所得的MV添加到候选MV集合
 }
}

if(C存在且C不是帧内编码的){
 if(C是1MV){
   将C的MV添加到候选MV集合
 }else if(C是4帧MV){
   if(C是右上MB){
     将C的左下块MV添加到候选MV集合
   }else{//C是左上MB
     将C的右下块MV添加到候选MV集合
   }
 }else if(C是2半帧MV){
   对C的两个半帧MV求平均值，并将所得的MV添加到候选MV集合
 }else if(C是4半帧MV){
   if(C是右上MB){
     对C的左上块半帧MV和左下块半帧MV求平均值，并将所得的MV添加到候选MV集合
   }else{//C是左上MB
     对C的右上块半帧MV和右下块半帧MV求平均值，并将所得的MV添加到候选MV集合。
   }
 }
}
```

图 42

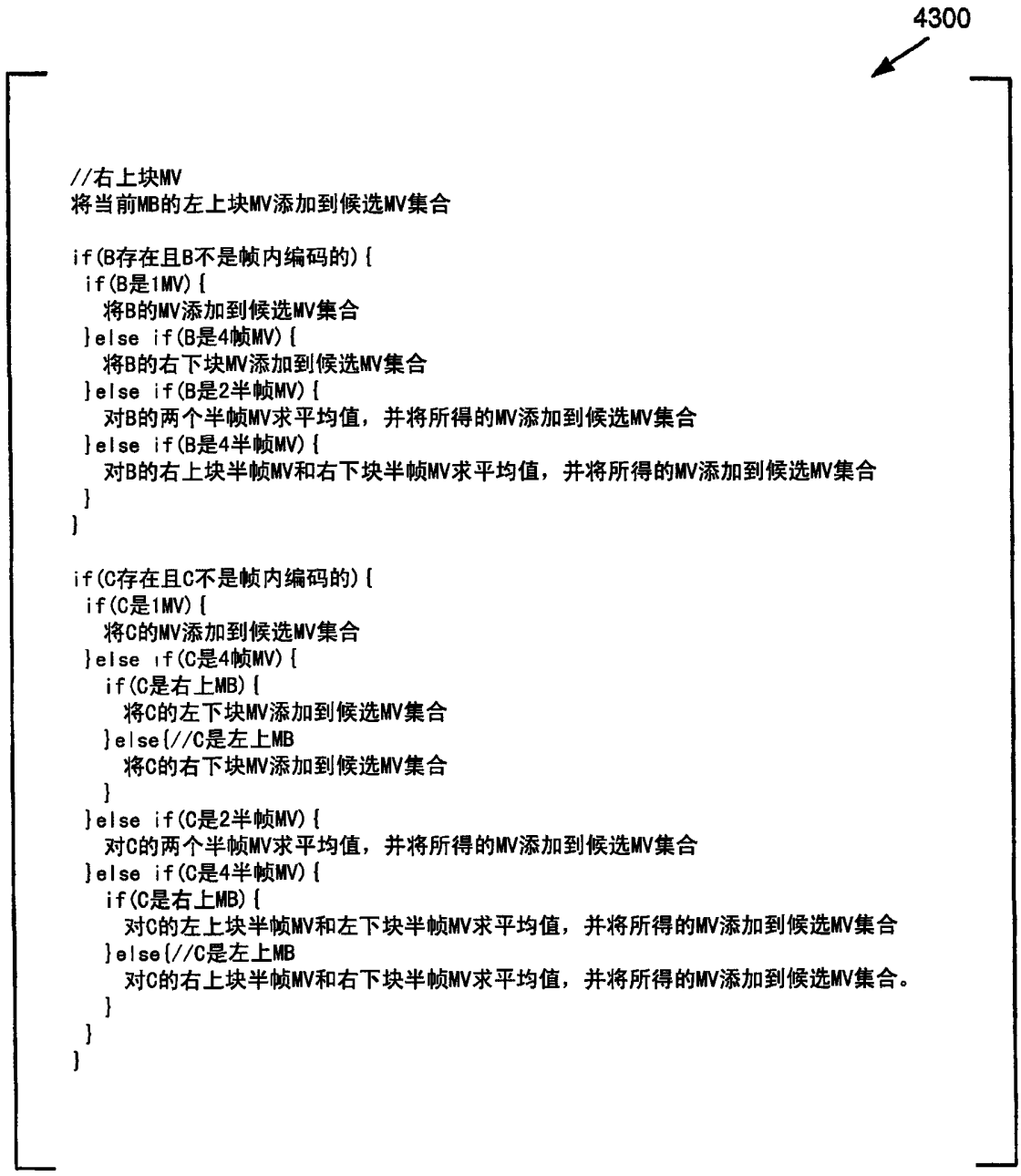

4300
//右上块MV
将当前MB的左上块MV添加到候选MV集合
if(B存在且B不是帧内编码的){
if(B是1MV){
将B的MV添加到候选MV集合
}else if(B是4帧MV){
将B的右下块MV添加到候选MV集合
}else if(B是2半帧MV){
对B的两个半帧MV求平均值，并将所得的MV添加到候选MV集合
}else if(B是4半帧MV){
对B的右上块半帧MV和右下块半帧MV求平均值，并将所得的MV添加到候选MV集合
}
}
if(C存在且C不是帧内编码的){
if(C是1MV){
将C的MV添加到候选MV集合
}else if(C是4帧MV){
if(C是右上MB){
将C的左下块MV添加到候选MV集合
}else{//C是左上MB
将C的右下块MV添加到候选MV集合
}
}else if(C是2半帧MV){
对C的两个半帧MV求平均值，并将所得的MV添加到候选MV集合
}else if(C是4半帧MV){
if(C是右上MB){
对C的左上块半帧MV和左下块半帧MV求平均值，并将所得的MV添加到候选MV集合
}else{//C是左上MB
对C的右上块半帧MV和右下块半帧MV求平均值，并将所得的MV添加到候选MV集合。
}
}
}

图 43

4400

```
//左下块MV
if(A存在且A不是帧内编码的){
 if(A是1MV){
   将A的MV添加到候选MV集合
 }else if(A是4帧MV){
   将A的右下块MV添加到候选MV集合
 }else if(A是2半帧MV){
   对A的两个半帧MV求平均值，并将所得的MV添加到候选MV集合
 }else if(A是4半帧MV){
   对A的右上块半帧MV和右下块半帧MV求平均值，并将所得的MV添加到候选MV集合
 }
}

将当前MB的左上块MV添加到候选MV集合
将当前MB的右上块MV添加到候选MV集合
```

图 44

4500

```
//右下块MV
将当前MB的左下块MV添加到候选MV集合
将当前MB的左上块MV添加到候选MV集合
将当前MB的右上块MV添加到候选MV集合
```

图 45

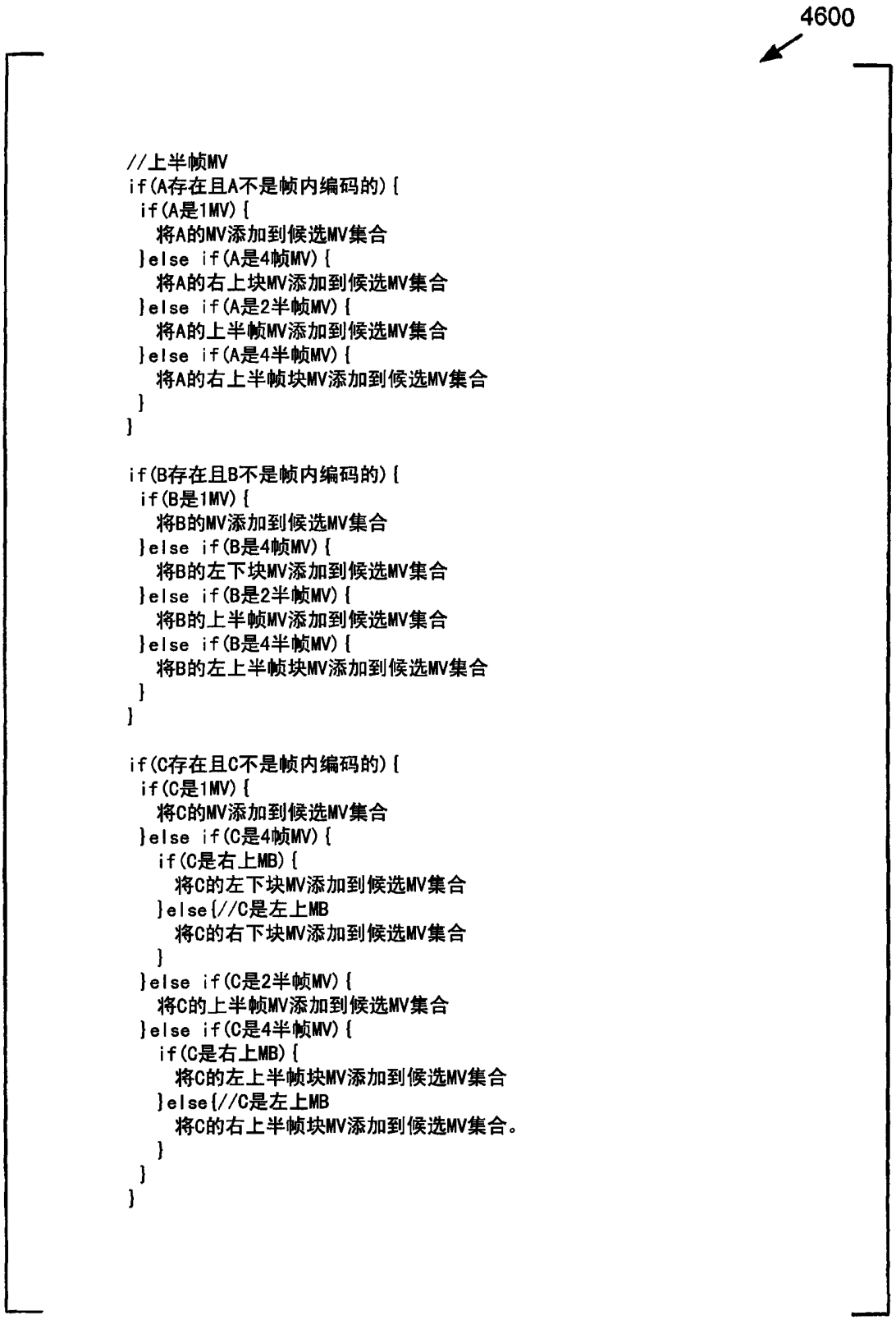

4600
//上半帧MV
if(A存在且A不是帧内编码的){
if(A是1MV){
将A的MV添加到候选MV集合
}else if(A是4帧MV){
将A的右上块MV添加到候选MV集合
}else if(A是2半帧MV){
将A的上半帧MV添加到候选MV集合
}else if(A是4半帧MV){
将A的右上半帧块MV添加到候选MV集合
}
}
if(B存在且B不是帧内编码的){
if(B是1MV){
将B的MV添加到候选MV集合
}else if(B是4帧MV){
将B的左下块MV添加到候选MV集合
}else if(B是2半帧MV){
将B的上半帧MV添加到候选MV集合
}else if(B是4半帧MV){
将B的左上半帧块MV添加到候选MV集合
}
}
if(C存在且C不是帧内编码的){
if(C是1MV){
将C的MV添加到候选MV集合
}else if(C是4帧MV){
if(C是右上MB){
将C的左下块MV添加到候选MV集合
}else{//C是左上MB
将C的右下块MV添加到候选MV集合
}
}else if(C是2半帧MV){
将C的上半帧MV添加到候选MV集合
}else if(C是4半帧MV){
if(C是右上MB){
将C的左上半帧块MV添加到候选MV集合
}else{//C是左上MB
将C的右上半帧块MV添加到候选MV集合。
}
}
}

图 46

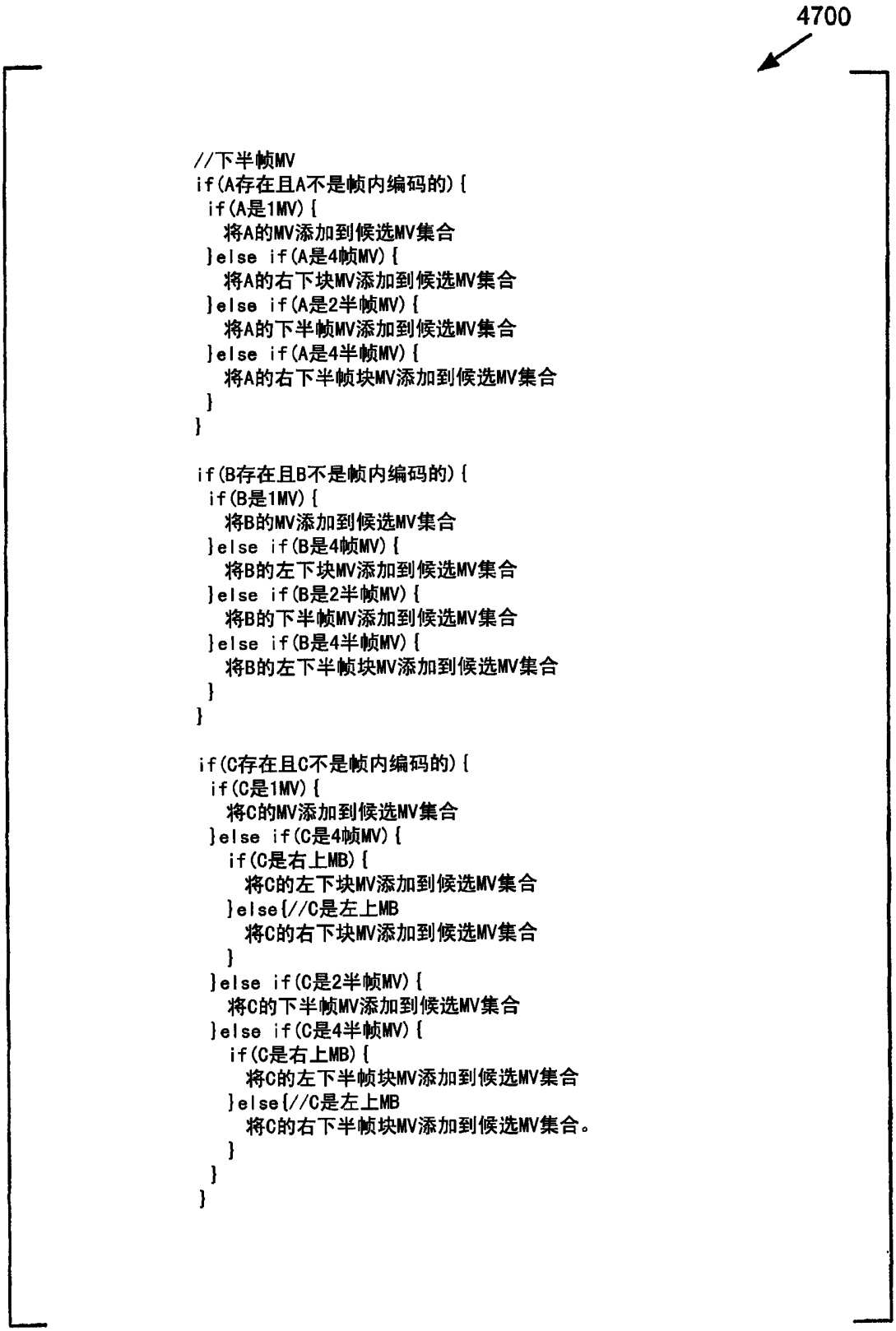

4700
//下半帧MV
if(A存在且A不是帧内编码的){
if(A是1MV){
将A的MV添加到候选MV集合
}else if(A是4帧MV){
将A的右下块MV添加到候选MV集合
}else if(A是2半帧MV){
将A的下半帧MV添加到候选MV集合
}else if(A是4半帧MV){
将A的右下半帧块MV添加到候选MV集合
}
}
if(B存在且B不是帧内编码的){
if(B是1MV){
将B的MV添加到候选MV集合
}else if(B是4帧MV){
将B的左下块MV添加到候选MV集合
}else if(B是2半帧MV){
将B的下半帧MV添加到候选MV集合
}else if(B是4半帧MV){
将B的左下半帧块MV添加到候选MV集合
}
}
if(C存在且C不是帧内编码的){
if(C是1MV){
将C的MV添加到候选MV集合
}else if(C是4帧MV){
if(C是右上MB){
将C的左下块MV添加到候选MV集合
}else{//C是左上MB
将C的右下块MV添加到候选MV集合
}
}else if(C是2半帧MV){
将C的下半帧MV添加到候选MV集合
}else if(C是4半帧MV){
if(C是右上MB){
将C的左下半帧块MV添加到候选MV集合
}else{//C是左上MB
将C的右下半帧块MV添加到候选MV集合。
}
}
}

图 47

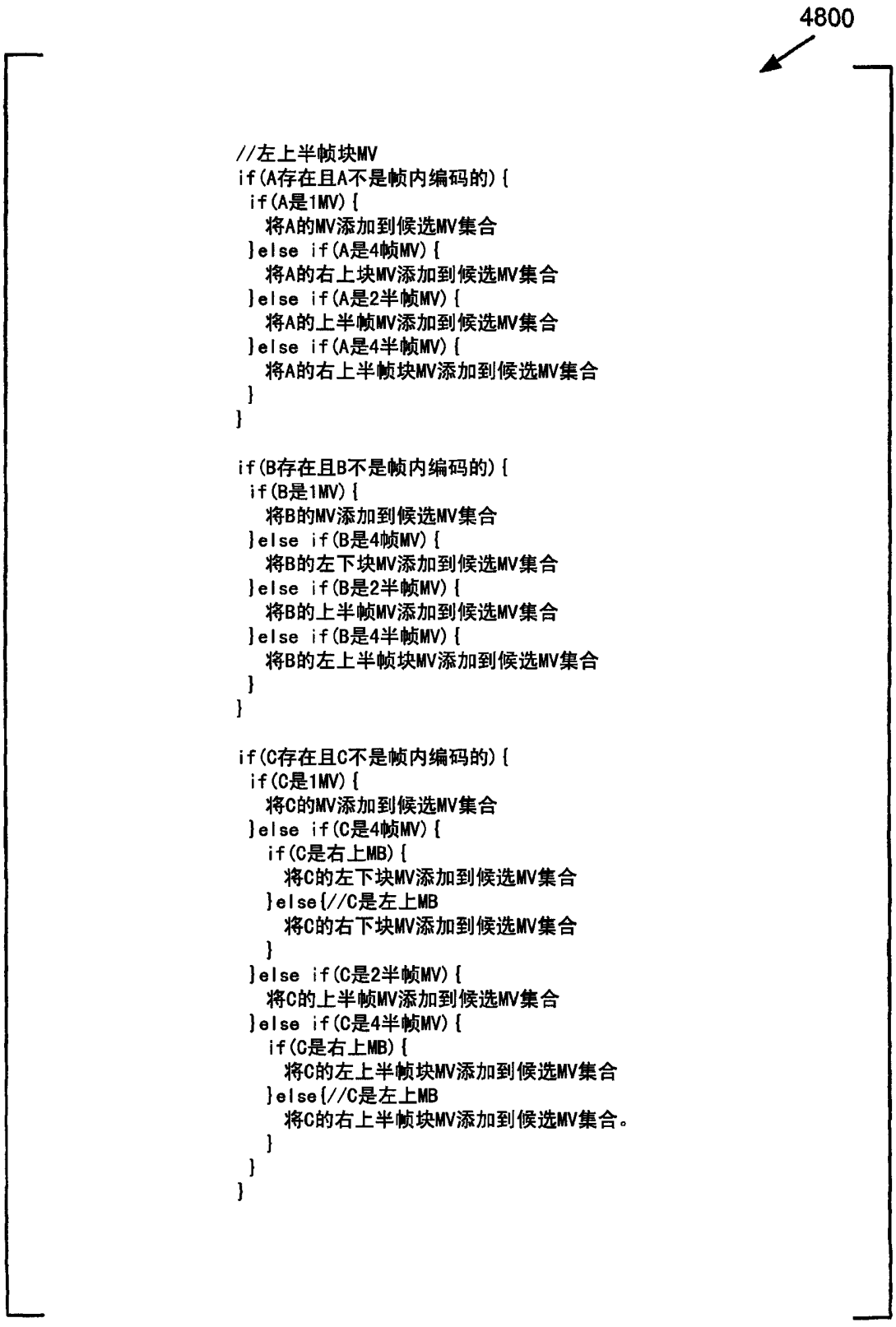

4800
//左上半帧块MV
if(A存在且A不是帧内编码的){
if(A是1MV){
将A的MV添加到候选MV集合
}else if(A是4帧MV){
将A的右上块MV添加到候选MV集合
}else if(A是2半帧MV){
将A的上半帧MV添加到候选MV集合
}else if(A是4半帧MV){
将A的右上半帧块MV添加到候选MV集合
}
}
if(B存在且B不是帧内编码的){
if(B是1MV){
将B的MV添加到候选MV集合
}else if(B是4帧MV){
将B的左下块MV添加到候选MV集合
}else if(B是2半帧MV){
将B的上半帧MV添加到候选MV集合
}else if(B是4半帧MV){
将B的左上半帧块MV添加到候选MV集合
}
}
if(C存在且C不是帧内编码的){
if(C是1MV){
将C的MV添加到候选MV集合
}else if(C是4帧MV){
if(C是右上MB){
将C的左下块MV添加到候选MV集合
}else{//C是左上MB
将C的右下块MV添加到候选MV集合
}
}else if(C是2半帧MV){
将C的上半帧MV添加到候选MV集合
}else if(C是4半帧MV){
if(C是右上MB){
将C的左上半帧块MV添加到候选MV集合
}else{//C是左上MB
将C的右上半帧块MV添加到候选MV集合。
}
}
}

图 48

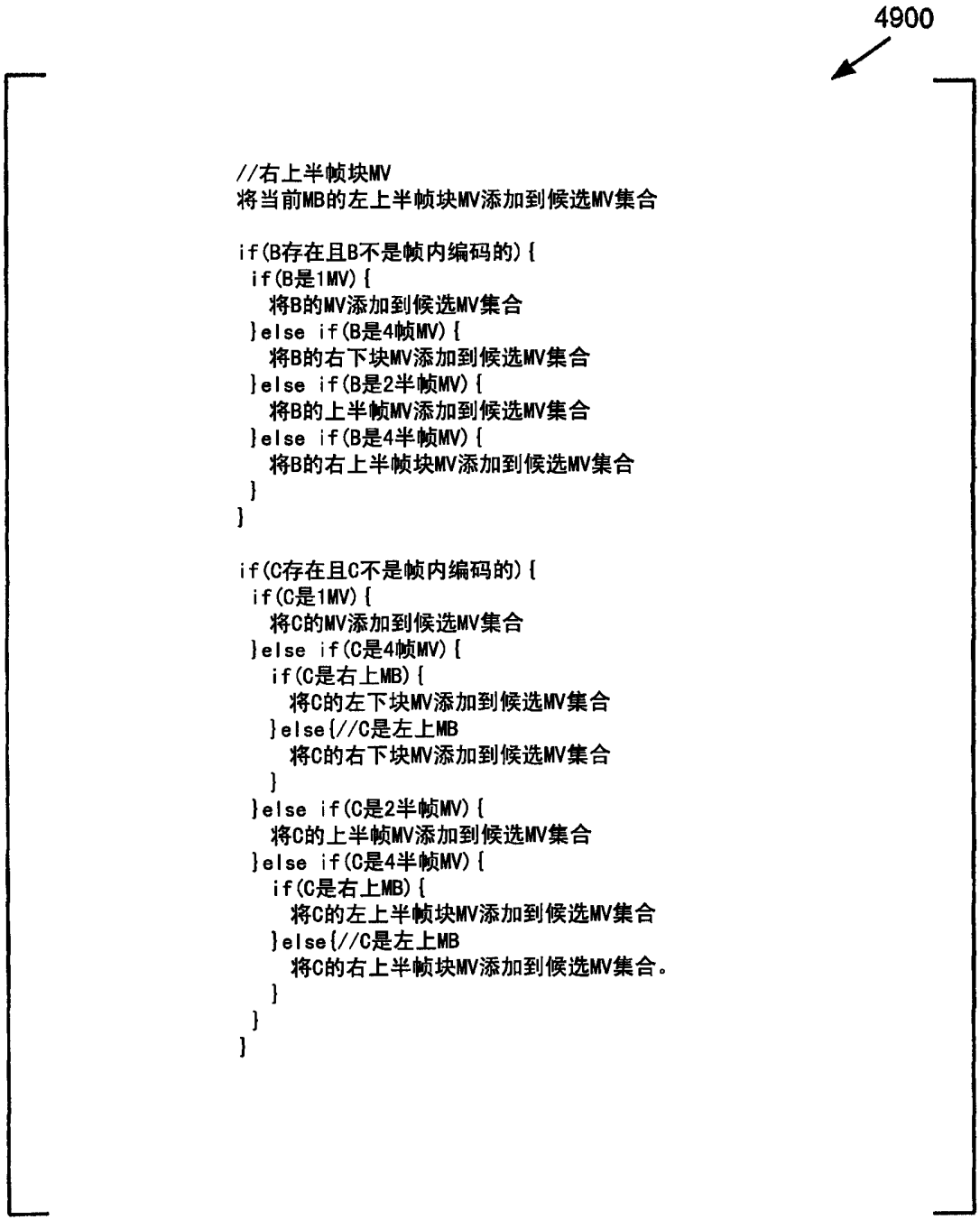

4900
//右上半帧块MV
将当前MB的左上半帧块MV添加到候选MV集合
if(B存在且B不是帧内编码的){
if(B是1MV){
将B的MV添加到候选MV集合
}else if(B是4帧MV){
将B的右下块MV添加到候选MV集合
}else if(B是2半帧MV){
将B的上半帧MV添加到候选MV集合
}else if(B是4半帧MV){
将B的右上半帧块MV添加到候选MV集合
}
}
if(C存在且C不是帧内编码的){
if(C是1MV){
将C的MV添加到候选MV集合
}else if(C是4帧MV){
if(C是右上MB){
将C的左下块MV添加到候选MV集合
}else{//C是左上MB
将C的右下块MV添加到候选MV集合
}
}else if(C是2半帧MV){
将C的上半帧MV添加到候选MV集合
}else if(C是4半帧MV){
if(C是右上MB){
将C的左上半帧块MV添加到候选MV集合
}else{//C是左上MB
将C的右上半帧块MV添加到候选MV集合。
}
}
}

图 49

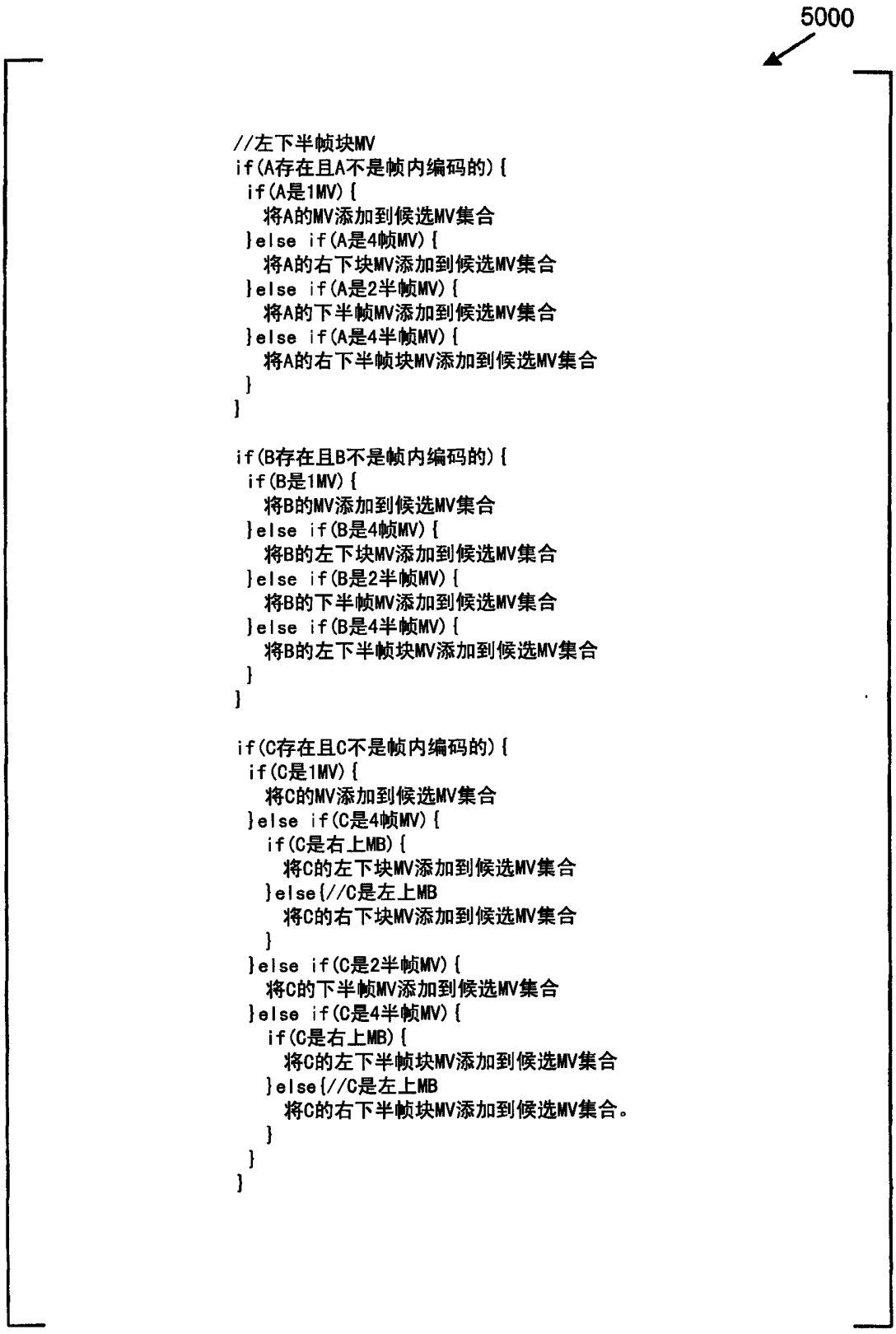

5000
//左下半帧块MV
if(A存在且A不是帧内编码的){
if(A是1MV){
将A的MV添加到候选MV集合
}else if(A是4帧MV){
将A的右下块MV添加到候选MV集合
}else if(A是2半帧MV){
将A的下半帧MV添加到候选MV集合
}else if(A是4半帧MV){
将A的右下半帧块MV添加到候选MV集合
}
}
if(B存在且B不是帧内编码的){
if(B是1MV){
将B的MV添加到候选MV集合
}else if(B是4帧MV){
将B的左下块MV添加到候选MV集合
}else if(B是2半帧MV){
将B的下半帧MV添加到候选MV集合
}else if(B是4半帧MV){
将B的左下半帧块MV添加到候选MV集合
}
}
if(C存在且C不是帧内编码的){
if(C是1MV){
将C的MV添加到候选MV集合
}else if(C是4帧MV){
if(C是右上MB){
将C的左下块MV添加到候选MV集合
}else{//C是左上MB
将C的右下块MV添加到候选MV集合
}
}else if(C是2半帧MV){
将C的下半帧MV添加到候选MV集合
}else if(C是4半帧MV){
if(C是右上MB){
将C的左下半帧块MV添加到候选MV集合
}else{//C是左上MB
将C的右下半帧块MV添加到候选MV集合。
}
}
}

图 50

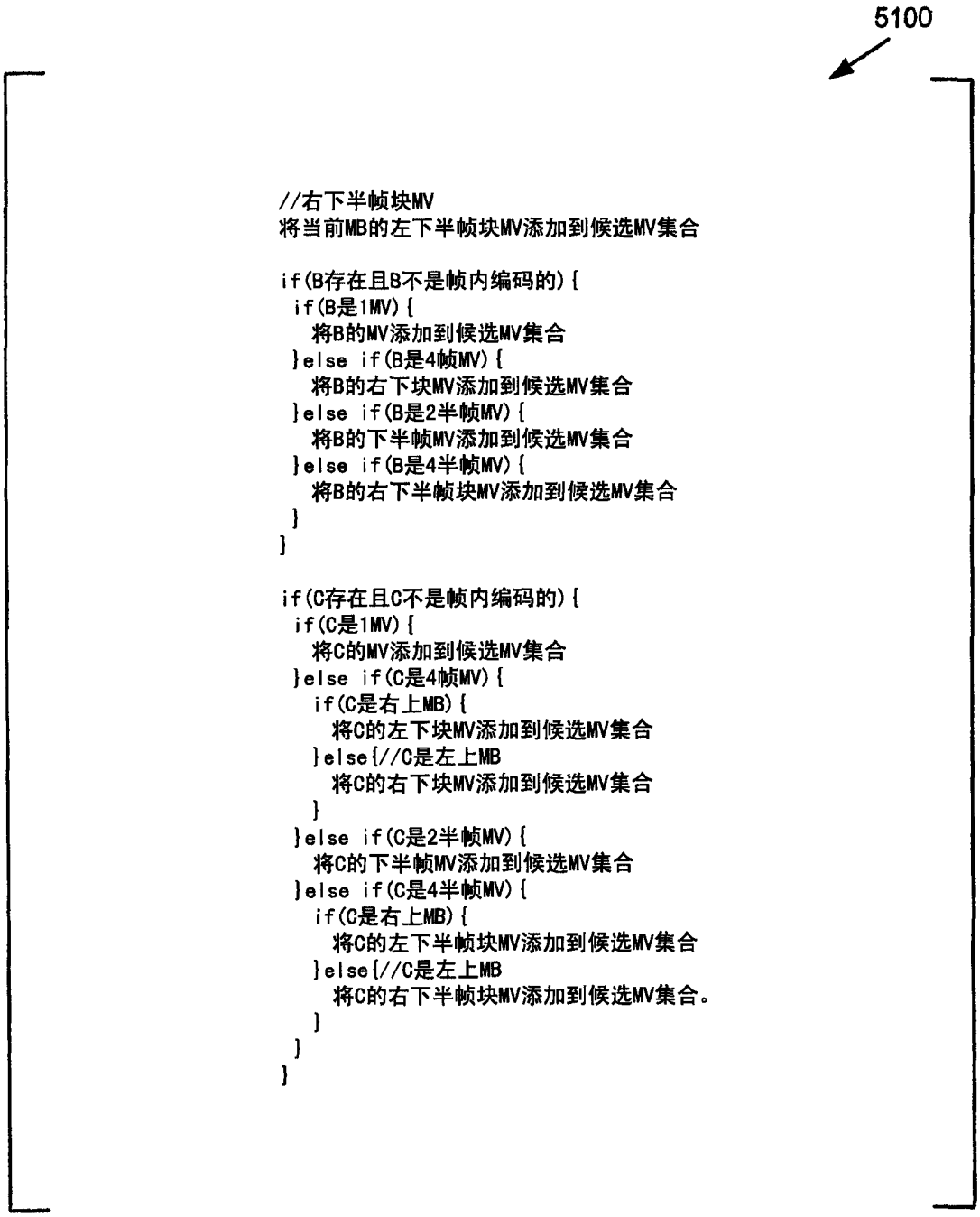

5100
//右下半帧块MV
将当前MB的左下半帧块MV添加到候选MV集合
if(B存在且B不是帧内编码的){
if(B是1MV){
将B的MV添加到候选MV集合
}else if(B是4帧MV){
将B的右下块MV添加到候选MV集合
}else if(B是2半帧MV){
将B的下半帧MV添加到候选MV集合
}else if(B是4半帧MV){
将B的右下半帧块MV添加到候选MV集合
}
}
if(C存在且C不是帧内编码的){
if(C是1MV){
将C的MV添加到候选MV集合
}else if(C是4帧MV){
if(C是右上MB){
将C的左下块MV添加到候选MV集合
}else{//C是左上MB
将C的右下块MV添加到候选MV集合
}
}else if(C是2半帧MV){
将C的下半帧MV添加到候选MV集合
}else if(C是4半帧MV){
if(C是右上MB){
将C的左下半帧块MV添加到候选MV集合
}else{//C是左上MB
将C的右下半帧块MV添加到候选MV集合。
}
}
}

图 51

图 52

5200

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

图 53

5300

```
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;
}
```

5400

```
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()    //使用由图像层中的MVTAB指示的表
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 54A

5410

```
offset_table[9] = {0, 1, 2, 4, 8, 16, 32, 64, 128}
index = vlc_decode()  //使用由图像层中的MVTAB指示的表
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 54B

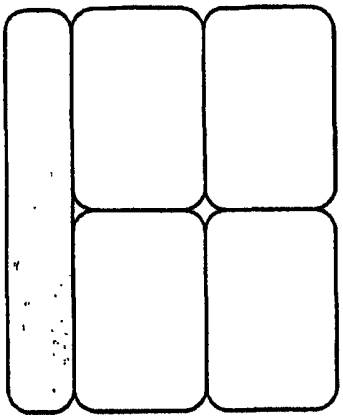


图 55A

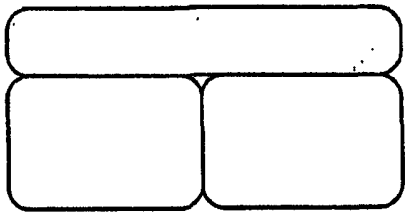


图 55B

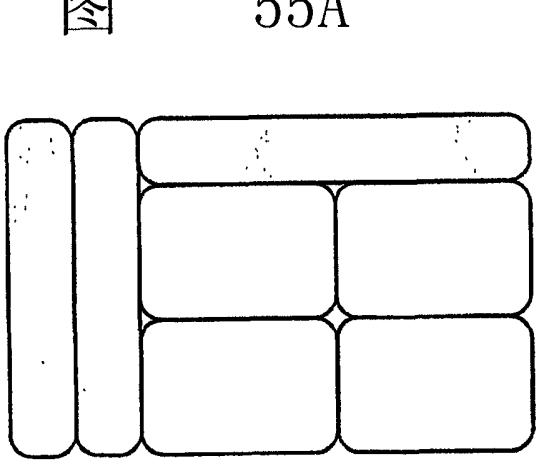


图 55C