[19] 中华人民共和国国家知识产权局

[51] Int. Cl[7]
H04N 7/26
H04N 7/52 H04N 7/64

[12] 发明专利说明书

[21] ZL 专利号 98812591.9

[45] 授权公告日 2004 年 11 月 3 日

[11] 授权公告号 CN 1174631C

[22] 申请日 1998.10.23 [21] 申请号 98812591.9

[30] 优先权

[32] 1997.10.23 [33] US [31] 08/957,555

[32] 1998. 1. 2 [33] US [31] 09/002,553

[32] 1998. 7. 6 [33] US [31] 09/111,112

[32] 1997.10.23 [33] US [31] 08/956,632

[32] 1998. 1. 2 [33] US [31] 09/002,470

[32] 1997.10.23 [33] US [31] 08/956,870

[32] 1998. 1. 2 [33] US [31] 09/002,547

[32] 1998. 1.30 [33] US [31] 09/016,083

[86] 国际申请 PCT/US1998/022412 1998.10.23

[87] 国际公布 WO1999/021368 英 1999.4.29

[85] 进入国家阶段日期 2000.6.23

[71] 专利权人 索尼电子有限公司

地址 美国新泽西州

[72] 发明人 T·康多 Y·弗吉莫里
J·J·卡里格 S·戈萨尔

审查员 梁军丽

[74] 专利代理机构 中国专利代理(香港)有限公司

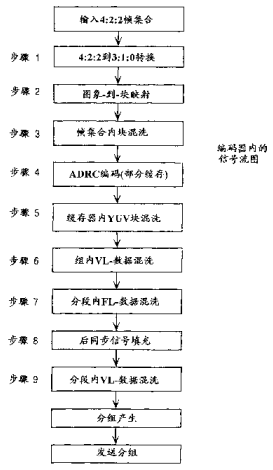
代理人 程天正 张志醒

权利要求书 3 页 说明书 42 页 附图 44 页

[54] 发明名称 部分缓存发送的数据以便提供增强的差错恢复

[57] 摘要

揭示了用于在信号传输之前缓存多个信号段的系统和方法。缓存器(800)被配置为防止发送信号解码过程中的差错传播。一定传输速率的一组编码级别从具有多个编码级别的门限表中选出。每组编码级别包括为选择编码比特而配置的多个范围。每个信号段使用该组编码级别来进行编码。在一个实施例中，用于在可能有损耗的通信信道上传输视频信号。



ISSN1008-4274

1. 一种限制在解码期间的差错传播的编码信号的方法，该方法包括：

将该信号分成多个数据段；

将每个数据段存储在至少一个缓存器中；

基于期望的发送速率和用于存储该数据段的至少一个缓存器的长度来确定可获得的用作编码比特的比特数；

选择用于编码每个数据段的门限组，该门限组具有多个范围，该多个范围中的每个范围分配一个不同的编码比特数，其中被分配用于编码每个数据段的编码比特数不大于可获得的比特数；以及

使用该门限组来编码该数据段。

2. 根据权利要求 1 的方法，其中选择该门限组以使得存储在缓存器内的编码数据最大化。

3. 根据权利要求 1 的方法，其中选择该门限组以使得被分配用于编码每个数据段的编码比特数在不超出可获得的比特数的情况下，尽可能接近可获得的比特数。

4. 根据权利要求 1 的方法，其中至少一个缓存器具有固定的长度。

5. 根据权利要求 4 的方法，其中没有被编码数据填充的固定长度的缓存器的比特被预定比特流图形填充，以便该预定比特流图形描述该编码数据的末尾。

6. 一种限制差错传播的解码数据的方法，该方法包括：

在至少一个缓存器中接收编码数据，该编码数据包括根据所选择的门限组来编码的数据段，该门限组具有多个范围，每个范围分配不同的编码比特数，其中被分配的该编码比特数不大于可获得的比特数，该可获得的比特数基于期望的发送速率和用于存储该数据段的缓存器的长度而用作编码比特；

确定是否缓存器内的编码数据的恢复已经被禁止；以及

如果该数据的恢复没有被禁止则解码该缓存器内的编码数据。

7. 根据权利要求 6 的方法，其中至少一个缓存器具有固定的长度。

8. 一种包括处理器的数字处理系统，该处理器被配置来用于编码信号，以使得在解码期间差错传播被限制，所述处理器被配置来用于将该信号分成数据段；将该每个数据段存储在至少一个缓存器中；基于期望的发送速率和用于存储该数据段的缓存器的长度来确定用作编码比特的可获得的比特数；选择用于编码每个数据段的门限组，该门限组具有多个范围，每个范围分配不同的编码比特数，其中被分配用于编码每个数据段的编码比特数不大于可获得的比特数；并且使用该门限组来编码该数据段。

9. 根据权利要求 8 的数字处理系统，其中选择该门限组以使得存储在缓存器内的编码数据最大化。

10. 根据权利要求 8 的数字处理系统，其中选择该门限组以使得被分配用于编码每个数据段的编码比特数在不超出可获得的比特数的情况下，尽可能接近可获得的比特数。

11. 根据权利要求 8 的数字处理系统，其中至少一个缓存器具有固定长度。

12. 根据权利要求 11 的数字处理系统，其中没有被编码数据填充的固定长度的缓存器的比特被预定比特流图形填充，以便该预定比特流图形描述该编码数据的末尾。

13. 一种包括处理器的数字处理系统，该处理器被配置来用于解码数据，以使得差错传播被限制，所述处理器被配置来用于在至少一个缓存器中接收编码数据，该编码数据包括根据所选择的门限组来编码的数据段，该门限组具有多个范围，每个范围分配不同的编码比特数，其中被分配的编码比特数不大于可获得的比特数，该可获得的比特数基于期望的发送速率和用于存储该数据段的缓存器的长度而用作为编码比特；确定是否缓存器内的编码数据的恢复已经被禁止；并且如果该数据的恢复没有被禁止则解码缓存器内的该编码数据。

14. 根据权利要求 13 的数字处理系统，其中至少一个缓存器具有固定长度。

15. 一种限制在解码期间的差错传播的编码信号的装置，该装置包括:

将该信号分成多个数据段的装置;

将每个数据段存储在至少一个缓存器中的装置;

基于期望的发送速率和用于存储该数据段的缓存器的长度来确定可获得的用作编码比特的比特数的装置;

选择用于编码每个数据段的门限组的装置，该门限组具有多个范围，每个范围分配一个不同的编码比特数，以使得被分配用于编码每个数据段的编码比特数不大于可获得的比特数; 以及

使用该门限组来编码该数据段的装置。

16. 根据权利要求 15 的装置，其中选择该门限组以使得存储在缓存器内的编码数据最大化。

17. 根据权利要求 15 的装置，其中选择该门限组以使得被分配用于编码每个数据段的编码比特数在不超出可获得的比特数的情况下，尽可能接近可获得的比特数。

18. 根据权利要求 15.的装置，其中至少一个缓存器具有固定的长度。

19. 根据权利要求 18 的方法，其中没有被编码数据填充的固定长度的缓存器的比特被预定比特流图形填充，以便该预定比特流图形描述该编码数据的末尾。

20. 一种限制差错传播的解码数据的装置，该装置包括:

用于在至少一个缓存器中接收编码数据的装置;

确定是否缓存器内的编码数据的恢复已经被禁止的装置;以及

如果数据的恢复没有被禁止则解码该缓存器内的编码数据的装置，

其中该编码数据包括根据所选择的门限组来编码的数据段，该门限组具有多个范围，每个范围分配不同的编码比特数，其中被分配的该编码比特数不大于可获得的比特数，该可获得的比特数基于期望的发送速率和用于存储该数据段的缓存器的长度而用作编码比特;

21. 根据权利要求 20 的装置，其中至少一个缓存器具有固定的长度。

部分缓存发送的数据以便提供增强的差错恢复

发明背景

1. 相关申请

本申请是 1998 年 1 月 30 日提交的题为“Source Coding to Provide for Robust Error Recovery During Transmission Losses（在传输损耗时提供增强的差错恢复的信源编码）”的美国专利申请系列号 No. 09/016,083 的继续申请，而它是 1998 年 1 月 2 日提交的题为“Image-to-Block Mapping to Provide for Robust Error Recovery During Transmission Losses（在传输损耗时提供增强的差错恢复的图象-到-块映射）”的申请系列号 No. 09/002,547、1998 年 1 月 2 日提交的题为“Source Coding to Provide for Robust Error Recovery During Transmission Losses（在传输损耗时提供增强的差错恢复的信源编码）”的申请系列号 No. 09/002,470、以及 1998 年 1 月 2 日提交的题为“Multiple Block Based Recovery Method to Provide for Robust Error Recovery During Transmission Losses（在传输损耗时提供增强的差错恢复的基于多块恢复方法）”的申请系列号 No. 09/002,553 的部分继续申请；它们是 1997 年 10 月 23 日提交的题为“Image-to-Block Mapping to Provide for Robust Error Recovery During Transmission Losses（在传输损耗时提供增强的差错恢复的图象-到-块映射）”的申请系列号 No. 08/965,632、1997 年 10 月 23 日提交的题为“Source Coding to Provide for Robust Error Recovery During Transmission Losses（在传输损耗时提供增强的差错恢复的信源编码）”的申请系列号 No. 08/957,555、以及 1997 年 10 月 23 日提交的题为“Multiple Block Based Recovery Method to Provide for Robust Error Recovery During Transmission Losses（在传输损耗时提供增强的差错恢复的基于多块恢复方法）”的申请系列号 No. 08/956,870 的部分继续申请。1998 年 1 月 30 日提交的申请系列号 No. 09/016,083、1998 年 1 月 2 日提交的申请系列号

No. 09/002,547、1998 年 1 月 2 日提交的申请系列号 No. 09/002,470、1998 年 1 月 2 日提交的申请系列号 No. 09/002,553、1997 年 10 月 23 日提交的申请系列号 No. 08/956,632、1997 年 10 月 23 日提交的申请系列号 No. 08/957,555、以及 1997 年 10 月 23 日提交的申请系列号 No. 08/956,870 都在这里结合参照。

2. 发明领域

本发明涉及对由于信号传输过程中出现数据损耗而带来的差错提供增强的恢复。

3. 技术背景

有很多技术用于重构由于信号传输过程中出现随机差错而造成的丢失数据。但是，这些技术不能处理连续数据分组的丢失。数据分组的连续丢失在本领域中描述为突发错误。突发错误导致对最终用户非常明显的重构信号质量下降。此外，用于促进高速通信的压缩方法使突发错误所引起的信号下降变得更严重，因此使重构信号变得更差。影响发送及/或存储信号的突发错误损耗的例子在其中压缩方法起了重要作用的高清晰度电视（“HDTV”）信号以及移动电信应用中可以看到。

HDTV 的出现将电视系统引入比目前 National Television Systems Committee（“NTSC”国家电视系统委员会）所建议的标准更高的分辨率。所建议的 HDTV 信号主要是数字的。因此，当彩色电视信号转换为数字使用时，亮度和色度信号使用八比特数字化是常见的。彩色电视的数字传输要求每秒 216 兆比特的标称比特率。HDTV 的传输速率要更高些，它标称地要求大约每秒 1200 兆比特。这种高传输速率大大超过当前无线标准所支持的带宽。因此，要求有效的压缩方法。

压缩方法也在移动电信应用中起着重要作用。通常，数据分组在移动电信应用的远程终端之间传递。移动通信中有限数目的传输信道要求分组传输之前具有有效的压缩方法。多种压缩方法被用于促进高传输速率。

自适应动态范围编码（“ADRC”）和离散余弦变换（“DCT”）编码提供了本领域已知的图象压缩技术。两种技术都利用了图象内的局部相关性实现高压缩比。但是，有效的压缩算法导致更严重的差错传

播，因为当随后解码时编码信号中的错误更显著。这种错误繁殖导致对用户非常明显的视频图象质量下降。

发明概要

本发明描述了对信号信源编码的方法。具体而言，对包括多个信号单元的信号进行处理。每个信号单元被编码构成比特流。给定比特流内的比特被分散到不同比特流中。因此，描述段单元组成的参数被分散到不同比特流中。分散步骤导致错误分散到多级中。因此，当分散步骤由解码器逆向执行时，突发传输错误变成分散的局部损耗组。

也描述了另一种用于多级混洗（shuffling）处理的方法。信号被定义为多级的，其中每个级别包括多个帧、多个像素、以及多个比特。在一个实施例中，混洗出现在每一级和级之间。多级混洗使突发错误损耗被分散到多级中，因此有利于出现损耗的图象区域的图象重构。

一种在解码期间限制差错传播的编码信号的方法，该方法包括：

将该信号分成多个数据段；

将每个数据段存储在至少一个缓存器中；

基于期望的发送速率和用作存储该数据段的至少一个缓存器的长度来确定可获得的用作编码比特的比特数；

选择用于编码每个数据段的门限组，该门限组具有多个范围，该多个范围中的每个范围分配一个不同的编码比特数，其中被分配用于编码每个数据段的编码比特数不大于可获得的比特数；以及

使用该门限组来编码该数据段。

一种限制差错传播的解码数据的方法，该方法包括：

在至少一个缓存器中接收编码数据，该编码数据包括根据所选择的门限组来编码的数据段，该门限组具有多个范围，每个范围分配不同的编码比特数，其中被分配的该编码比特数不大于可获得的比特数，基于期望的发送速率和用于存储数据段的缓存器的长度，该可获得的比特数是用作编码比特；

确定是否缓存器内的编码数据的恢复已经被禁止；以及

如果数据的恢复没有被禁止则解码该缓存器内的编码数据。

一种包括处理器的数字处理系统，配置该数字处理系统以编码信号，以便在解码期间限制差错传播，配置所述处理器以将该信号分成

多个数据段，将该每个数据段存储在至少一个缓存器中，基于期望的发送速率和用于存储该数据段的缓存器的长度来确定用作编码比特的可获得的比特数，选择用于编码每个数据段的门限组，该门限组具有多个范围，每个范围分配不同的编码比特数，其中被分配用于编码每个数据段的编码比特数不大于可获得的比特数，并且使用该门限组来编码该数据段。

一种包括处理器的数字处理系统，配置该数字处理系统以解码数据，以便限制差错传播，配置所述处理器以在至少一个缓存器中接收编码数据，该编码数据包括根据所选择的门限组来编码数据段，该门限组具有多个范围，每个范围分配不同的编码比特数，其中被分配的编码比特数不大于可获得的比特数，该可获得的比特数是用作基于期望的发送速率和用于存储该数据段的缓存器的长度的编码比特，确定是否已经禁止缓存器内的编码数据的恢复，并且如果该数据的恢复没有被禁止则解码缓存器内的该编码数据。

一种在解码期间限制差错传播的编码信号的装置，该装置包括：

将该信号分成多个数据段的装置；

将每个数据段存储在至少一个缓存器中的装置；

基于期望的发送速率和用作存储该数据段的至少一个缓存器的长度来确定用作编码比特的可获得的比特数的装置；

选择用于编码每个数据段的门限组的装置，该门限组具有多个范围，每个范围分配一个不同的编码比特数，以便被分配用于编码每个数据段的编码比特数不大于可获得的比特数；以及

使用该门限组来编码该数据段的装置。

一种限制差错传播的解码数据的装置，该装置包括：

用于在至少一个缓存器中接收编码数据的装置；

确定是否缓存器内的编码数据的恢复已经被禁止的装置；以及

如果数据的恢复没有被禁止则解码该缓存器内的编码数据的装置，其中该编码数据包括根据所选择的门限组来编码的数据段，该门限组具有多个范围，每个范围分配不同的编码比特数，其中被分配的该编码比特数不大于可获得的比特数，基于期望的发送速率和用于存储数据段的缓存器的长度，该可获得的比特数是用作编码比特。

附图的简要描述

本领域技术人员根据如下详细描述将明确本发明的目的、特性和优点，其中：

图 1 一般性说明了信号编码、传输及解码过程。

图 2 说明了分组结构的一个实施例。

图 3 是一个流程图，说明根据本发明概念的编码过程的一个实施例。

图 4 是一个流程图，说明根据本发明概念的解码过程的一个实施例。

图 5 说明根据本发明概念的图象-到-块映射的一个实施例。

图 5a 说明图象-到-块映射中使用的混洗模式的一个实施例。

图 6 说明示范的互补和互锁块结构。

图 7a、7b、7c 和 7d 说明帧组合内 Y 块的混洗模式的一个实施例。

图 8 说明缓存器 0 的累计 DR 分布的一个实施例。

图 8a 说明根据本发明概念的部分混洗处理的一个实施例。

图 9 说明根据本发明概念的缓存器内 YUV 块混洗处理的一个实施例。

图 10 说明根据本发明概念的组内 VL 数据混洗处理的一个实施例。

图 11 说明根据本发明概念的 3-块组合内的 Q 码级联的一个实施例。

图 11a 说明根据本发明概念的包括运动块在内的帧对的 Q 码级联的一个实施例。

图 12 说明 1/6 突发错误损耗引起的像素数据差错的一个实施例。

图 12a 说明根据本发明概念的对 Q 码进行混洗和分布 Q 码比特的一个实施例。

图 12b 说明重新分布 Q 码的 1/6 突发错误损耗所引起的像素数据差错的一个实施例。

图 12c 说明重新指定 Q 码的 1/6 突发错误损耗所引起的像素数据差错的一个实施例。

图 13 说明根据本发明概念的 MIN 混洗的一个实施例。

图 13a 说明在一个帧对中运动标志混洗和固定长度数据损耗的一

个实施例。

图 14 说明取模混洗的一个实施例。

图 14a 说明与取模混洗有关的取模混洗结果和固定长度数据损耗的一个实施例。

图 14b 说明与取模混洗有关的取模混洗结果和固定长度数据损耗的另一个实施例。

图 14c 说明与取模混洗有关的取模混洗结果和固定长度数据损耗的又一个实施例。

图 15 说明帧组合中可变长度数据缓存的一个实施例。

图 16 说明了根据本发明概念的段间 VL-数据混洗的一个实施例。

图 17 是一般性说明本发明的数据恢复处理的一个实施例的流图。

图 18 是本发明的 Q 比特和运动标志恢复过程一个实施例的流图。

图 19 是说明候选解码一个实施例的表。

图 20a、20b、20c、20d 说明图 18 的 Q 比特和运动标志恢复过程中所用的测量的实施例。

图 21 说明用于图 18 的 Q 比特和运动标志恢复过程的确定误差平方概率函数的表的一个实施例。

图 22 说明根据本发明概念的 Q 比特、运动标志以及辅助信息恢复过程的一个实施例。

图 23 说明双向 Q 比特和运动标志恢复过程的一个实施例中后同步信号的使用。

图 24a、24b 和 24c 说明评价候选解码的另一个实施例。

图 25 根据本发明一个实施例的概念说明平滑措施的使用。

图 26a、26b、26c、26d 和 26e 说明评价候选解码过程的另一个实施例。

图 27a 说明评价候选解码的另一个过程，图 27b 说明确定权重值的一个实施例。

详细描述

本发明提供了一种编码及安排信号流以便提供增强的差错恢复的方法。在下面的描述中，为了说明起见，提出了很多细节，以便提供对本发明的完整理解。但是，对于本领域技术人员很显然的是，这些

特定的细节是实施本发明所不必要的。另一方面，众所周知的电气结构和电路用框图的形式表示，这样就不会不必要地使本发明变得不清晰。

信号处理方法及结构是从信号是视频信号这样一个实施例的角度来描述的。但是，这里描述的方法及装置是希望应用于各种类型的信号的，包括声音信号或其它数字数据比特流，其中每个信号由多个信号单元组成。此外，这里描述的处理实施例使用自适应动态范围编码（“ADRC”）过程来压缩数据；但是也可以使用各种编码技术和算法。对于 ADRC 的更详细讨论，可以参见“Adpative Dynamic Range Coding Scheme for Future HDTV Digital VTR（未来 HDTV 数字录象机的自适应动态范围编码方案）”，Kondo, Fujimori 及 Nakaya, Fourth International Workshop on HDTV and Beyond, 1991 年 9 月 4 日-6 日，都灵，意大利。

在上述文章中，解释了三种不同类型的 ADRC。它们都是根据下式实现的：

非边缘匹配 ADRC:

$$DR = MAX - MIN + 1$$

$$q = \left\lfloor \frac{(x - MIN + 0.5) \cdot 2^{Q}}{DR} \right\rfloor$$

$$x' = \left\lfloor \frac{(q + 0.5) \cdot DR}{2^{Q}} + MIN \right\rfloor$$

边缘匹配 ADRC:

$$DR = MAX - MIN$$

$$q = \left\lfloor \frac{(x - MIN) \cdot (2^{Q} - 1)}{DR} + 0.5 \right\rfloor$$

$$x' = \left\lfloor \frac{q \cdot DR}{2^{Q} - 1} + MIN + 0.5 \right\rfloor$$

多级 ADRC:

$$DR = MAX - MIN + 1$$

$$q = \left[\frac{(x - MIN + 0.5)\cdot 2^{Q}}{DR}\right]$$

$$x' = \left[\frac{(q + 0.5)\cdot DR}{2^{Q}} + MIN\right]$$

这里 MAX'是在 $q=2^{Q}-1$ 情况下 x'的平均值；MIN'是在 q=0 情况下 x'的平均值；而且

$$DR' = MAX' - MIN'$$

$$q = \left[\frac{(x - MIN')\cdot(2^{Q} - 1)}{DR'} + 0.5\right]$$

$$x' = \left[\frac{q\cdot DR'}{2^{Q} - 1} + MIN' + 0.5\right]$$

这里 MAX 代表一个块的最高级别，MIN 代表一块的最低级别，x 代表每个抽样的信号电平，Q 代表量化比特数，q 代表量化码（编码数据），x'代表每个抽样的解码级别，方括号[ ]代表方括号内的值上进行的截短运算。

信号解码、传输以及随后的解码过程一般性地在图 1 中说明。信号 100 是输入编码器 110 的数据流。编码器 110 遵循自适应动态范围编码（“ADRC”）压缩算法并沿着传输介质 135 产生分组 1、...、N。解码器 120 从传输介质 135 接收分组 1、...、N，并产生信号 130。信号 130 是信号 100 的重构。

编码器 110 和解码器 120 可以用各种方式实现以便执行这里描述的功能。在一个实施例中，编码器 110 及/或解码器 120 作为存储在介质中并由通用或专门配置的计算机系统执行的软件来实现，计算机系统通常包括中央处理单元、存储器以及一个或多个输入/输出设备及协处理器。或者，编码器 110 及/或解码器 120 可以作为执行这里所述功能的逻辑来实现。此外，编码器 110 及/或解码器 120 可以作为硬件、软件或固件的组合来实现。

在本实施例中，信号 100 是由视频帧序列组成的彩色视频图象，每个帧包括代表隔行扫描视频系统中图象的信息。每个帧由两个场组成，一个场包含图象偶数行的数据，另一个场包含图象奇数行的数据。

数据包括描述图象中相应位置的彩色成分的像素值。例如，在本发明中，彩色成分由亮度信号 Y 以及色差信号 U 及 V 组成。显然，本发明的过程可以用于隔行扫描视频信号以外的其它信号。此外，显然本发明不限于 Y、U、V 彩色空间中的实现，而是可以用于其它彩色空间中表示的图象。

返回参考图 1，编码器 110 将 Y、U 及 V 信号分开，并分别根据 ADRC 算法独立地处理每组信号。为了使讨论简单，下面的叙述描述了 Y 信号的处理，但是编码步骤要对 U 和 V 信号重复。

在本实施例中，编码器 110 将信号 100 的跨越两个连续帧的 Y 信号组成三维块（“3D”），这里称为一个帧对。对于一个实施例，3D 块由来自给定帧对上的同一局部区域的两个 2D 块的组合而产生，其中二维 2D 块通过帧内或场内的局部像素组合而产生。希望这里描述的过程可以用于不同的块结构。信号组合将在下面图象-到-块映射部分中进一步描述。

继续本实施例，对于给定的 3D 块，编码器 110 计算构成 3D 块的 2D 块之间的像素值中是否存在变化。如果值中确实有变化，就设置运动标志。正如本领域已知的，当每个帧对内有局部的图象重复时，运动标志的使用使编码器 110 减少了量化码的数目。编码器 110 也检测 3D 块内的最大像素强度值（“MAX”）和最小像素强度值（“MIN”）。利用值 MAX 和 MIN，编码器 110 计算给定 3D 数据块的动态范围(“DR”)。对于一个实施例，在非边缘匹配 ADRC 情况下 DR=MAX-MIN+1。对于边缘匹配 ADRC，DR=MAX-MIN 。在另一个实施例中，编码器 110 逐帧对代表一串视频帧的帧流信号编码。在另一个实施例中，编码器 110 逐场地对代表一串视频域的域流信号编码。因此，不使用运动标志，而是将 2D 块用于计算 MIN、MAX 和 DR 值。

在本实施例中，编码器 110 针对一个门限表（未表示）参考被计算的 DR，以便确定用于编码 DR 所对应的块内像素的量化比特（“Q 比特”）数。对像素编码产生量化码（“Q 码”）。Q 码是用于存储或传输的相对压缩的图象数据。

在一个实施例中，Q 比特选择是从 3D 块的 DR 中得到的。因此，给定 3D 块内的所有像素使用相同 Q 比特来编码，以得到一个 3D 编码块。

3D 编码块的 Q 码、MIN、运动标志以及 DR 的集合称为 3D ADRC 块。或者，对 2D 块编码，并且给定 2D 块的 Q 码、MIN 以及 DR 的集合产生 2D ADRC 块。

可以实现多个门限表。在一个实施例中，门限表由一行 DR 门限值组成。Q 比特对应于用来对门限表一行内两个相邻 DR 之间的一套 DR 值进行编码的量化比特数。在另一个实施例中，门限表包括多行，行的选择根据所需的传输速率。门限表中的每一行用门限索引来标识。门限选择的一个实施例的详细描述在下面讨论部分缓存时描述。ADRC 编码及缓存的进一步描述在转让给本发明受让人的题为“High Efficiency Coding Apparatus（高效编码装置）”的美国专利 No. 4,722,003 以及同样题为“High Efficiency Coding Apparatus（高效编码装置）”的美国专利 No. 4,845,560 中揭示。

由此 Q 码被称为可变长度数据（“VL-数据”）。此外，DR、MIN 和运动标志称为块属性。块属性以及门限索引组成了固定长度数据（“FL-数据”）。此外，考虑到上面的讨论，术语块属性描述了与信号单元的成分有关的参数，其中信号单元包括多个成分。在另一个实施例中，FL-数据分组括 Q 比特码。其好处是 Q 比特信息不必在解码过程中从 DR 得到。因此，如果 DR 信息丢失或损坏，Q 比特信息还可以从 Q 比特码中确定。此外，如果 Q 比特码丢失或损坏，Q 比特信息可以从 DR 得到。因此，降低了恢复 DR 和 Q 比特的要求。

要求将 Q 比特码包括在内的缺点是每个 ADRC 块要发送附加的比特。但是，在一个实施例中，用于组合 ADRC 块的 Q 比特码被根据例如加法或级联这样的功能而合并。例如，如果 ADRC 块被组成三组，并且如果每个 ADRC 块的 Q 比特值分别为 3、3 和 4，则编码为 FL-数据的总和值是 11。因此，代表总和所需的比特数少于代表每个单独值所需的比特数，而且组中没有损坏的 Q 比特值可以用于确定 Q 比特值而不必执行下面描述的那种 Q 比特恢复处理。

也考虑了其它实施例。例如，运动标志数据也可以被编码。可以产生带有 Q 比特和运动标志数据的标志并将其用于去查找码表。编码的配置和功能可以根据应用而变。

帧、块属性以及 VL-数据描述了视频信号内的各种成分。这些成分

的边界、位置及数量依赖于视频信号的传输及压缩特性。在本实施例中，这些成分在视频信号比特流中可变而且被混洗，以便确保传输损耗时增强的差错恢复。

为了说明起见，以下描述规定 1/6 连续分组传输损耗容限，遵循视频信号的 ADRC 编码和混洗。因此，以下的成分定义和划分被用于一个实施例中。也考虑了其它实施例。一个数据集合包括一个视频数据的一部分或其它类型的数据信号。因此，在一个实施例中，一个帧集合是包括一个或多个连续帧的一类数据集合。一个段包括一个能够存储 1/6 部分的 Q 码以及被包括在帧集合中的块属性的存储器。此外，缓存器包括一个能够存储 1/60 部分的 Q 码以及被包括在帧集合中的块属性的存储器。数据混洗通过交换段内及/或缓存器内的成分来进行。此后，存储在段内的数据用于产生供传输的数据分组。因此，在如下描述中，如果段丢失，所有从段中产生的分组都在传输中丢失。类似地，如果一部分段丢失，那么从段中产生的相应数目的分组在传输中丢失。

尽管，以下描述提到了用 ADRC 编码的数据有 1/6 连续的分组损耗，但是考虑这里描述的方法和装置可用于被运用到各种编码/解码方案中的 1/n 连续的分组损耗容限的设计。

图 2 说明了用于在点对点连接以及网络中传输数据的分组结构 200 的一个实施例。分组结构 200 由编码器 110 产生并通过传输介质 135 发送。对于一个实施例，分组结构 200 包括 5 字节字头信息、8 个 DR 比特、8 个 MIN 比特、一个运动标志比特、5 个门限索引比特以及 354 比特 Q 码。这里描述的分组结构是说明性的，而且以便可以用于异步转移模式（“ATM”）网络中的传输。但是，本发明不限于所描述的分组结构，而且可以使用各种网络中所用的各种分组结构。

正如前面注意到的，传输介质（例如，媒介）135 没有被认为能够提供无差错传输，因此分组可能会丢失或损坏。正如前面注意到的，存在检测这种丢失或损坏的常规方法，但是实际上一般会出现图象恶化。因此本发明的系统及方法教导了从这种丢失或损坏中增强恢复的信源编码。在整个如下讨论中假设，突发损耗（即，几个连续分组的丢失）是最可能的错误形式，但是一些随机分组损耗也可能出现。

为了确保从一个或多个连续数据分组损耗中增强的恢复，本发明的系统及方法提供了多级混洗。具体而言，包括在所传输的分组中的FL-数据和VL-数据包含空间和时间分离的数据位置上的数据。对数据混洗确保任何突发错误被分散并有利于差错恢复。正如下面所描述的，混洗使块属性和Q比特值得以恢复。

数据编码/解码

图3是说明编码器110所执行的编码过程一个实施例的流图。图3还描述了用于防止图象恶化并有利于增强的差错恢复的混洗过程的概述。

在图3的步骤1中，一个也被称为显示成分的输入帧集合被抽取，以便降低传输要求。Y信号被水平抽取到原有宽度的3/4，U和V信号每个被抽取到原有高度的1/2以及原有宽度的1/2。这样得到3:1:0的视频格式，每个帧对中有3960个Y块、660个U块以及660个V块。正如前面注意到的，讨论将描述Y信号的处理；但是该处理可用于U及V信号。在步骤2，两个Y帧图象被映射到3D块。在步骤3，3D块被混洗。在步骤4，使用ADRC缓存和编码。在步骤5，编码的Y、U及V块在缓存器内混洗。

在步骤6，一组编码3D块的VL-数据以及它们相应的块属性被混洗。在步骤7，FL-数据在不同段上混洗。在步骤8，执行后同步信号填充，其中缓存器末尾的可变空间被填充了预定的比特流。在步骤9，VL-数据在不同段上混洗。

为了说明起见，以下混洗描述提供了在编码之前和之后处理像素数据的方法。对于另一个实施例，独立的数据值通过硬件被混洗/去混洗。具体而言，硬件将块值地址映射到不同的地址，以便实现混洗/去混洗处理。但是，地址映射对于与数据相关的值是不可能的，因为混洗必须遵守数据的处理。下面描述的组内VL-数据混洗包括与数据相关的值。此外，为了说明起见，以下的混洗描述出现在不连续的数据集合上。但是，对于另一个实施例，信号基于多种数据级别——从比特到像素到帧——而定义。混洗对于信号中定义的每个级别都是可能的并且跨越不同的信号数据级别。

图4是说明解码器120所执行的解码处理的一个实施例的流图。

优选地，转换和去混洗处理是图 3 所代表的逆过程。图 4 还以 Q 比特、运动标志、DR、MIN 以及像素数据的不同组合描述了所发明的差错恢复过程。在下面以不同实施例的不同 Q 比特、运动标志、DR、MIN 以及像素恢复的不同组合来描述差错恢复过程。

从图象到块的映射

在本实施例中，单帧一般包括 5280 个 2D 块，每个 2D 块包括 64 个像素。因此，一个帧对包括 5280 个 3D 块，因为来自第一帧的一个 2D 块和来自后一帧的 2D 块合起来构成一个 3D 块。

为了分别将帧或数据帧集合分成 2D 块或 3D 块，就要进行图象-到-块映射。此外，图象-到-块映射包括使用互补及/或互锁模式划分帧中的像素，以便促进传输损耗过程中增强的差错恢复。但是，为了提高给定 DR 值不太大的概率，每个 2D 块从局部区域中的像素构造。

图 5 说明一个图象的示范 16 像素段的图象-到-块映射处理的一个实施例。图象 500 包括构成单帧局部区域的 16 像素。图象 500 中的每个像素由一个强度值代表。例如，图象顶部左边的像素具有等于 100 的强度值，而图象低部右边的像素强度值为 10.

在一个实施例中，来自图象 500 不同区域的像素用于产生 2D 块 510、520、530 和 540。2D 块 510、520、530 和 540 被编码、混洗（如下所述）并发送。发送之后，2D 块 510、520、530 和 540 被重新合并并用于构成图象 550. 图象 550 是图象 500 的重构。

为了确保在可能的传输损耗下精确地代表图象 500，图 5 是一种用于重构图象 550 的互锁互补块结构，它的一个实施例在图 5 中表示。具体而言，用于产生 2D 块 510、520、530 和 540 的像素选择可以确保：在重构图象 550 时，互补及/或互锁模式用于重新合并块。因此，当特定的 2D 块属性在传输中丢失时，图象 550 的连续段在重构过程中不会失真。例如，如图 5 所示，2D 块 540 的 DR 在数据传输过程中丢失了。但是，在图象 550 的重构过程中，解码器利用邻块的多个相邻像素就可以恢复 2D 块 540 丢失的 DR。此外，正如后面将要描述的，互补模式与混洗的结合增加了相邻像素数目，优选地使得来源于其它块的相邻像素数最大化，显著地改善了 DR 和 MIN 恢复。

图 5a 说明了在图象-到-块映射处理的一个实施例中用于构成 2D

块的混洗模式的一个实施例。基于像素交替，图象被分成两个子图象，即子图象 560 和子图象 570。在子图象 560 中构成一个矩形，从而描述 2D 块的边界。为了讨论，2D 块被编号为 0、2、4、、7、、9、11、12、14、16、19、21 和 23。瓦块（tile）565 说明了子图象 560 内 2D 块的像素分布。

在子图象 570 中，2D 块指派被水平移了 8 个像素，垂直移了 4 个像素。这样当子图象 560 和 570 在重构过程中合并时，导致环绕 2D 块指派和重叠的包封。2D 块被编号为 1、3、5、6、8、10、13、15、17、18、20 和 22。瓦块 575 说明子图象 570 内 2D 块的像素分布。瓦块 575 是瓦块 565 的互补结构。因此，当特定的块属性在传输过程中丢失时，块属性可以从相邻像素中恢复，因为丢失的 2D 块仍存在。此外，存在具有类似块属性集合的重叠 2D 像素块。因此，在图象的重构过程中，解码器有来自相邻 2D 块的多个相邻像素，丢失的块属性可以从其中恢复。

图 6 说明其它的互补及互锁 2D 块结构.其它结构也是可以使用的。类似于图 5，图 6 中说明的这些 2D 块结构，无论给定 2D 块是否传输丢失，都确保提供周围的 2D 块。但是，在像素到随后的 2D 块映射过程中，模式 610a、610b 和 610d 使用水平及/或垂直移动。水平移动描述了：在开始新的 2D 块边界之前在水平方向上将瓦块结构移动预定数目的像素。垂直移动描述了：在开始新的 2D 块边界之前在垂直方向上将方块结构移动预定数目的像素。在应用中，可以只使用水平移动，也可以只使用垂直移动，或者可以使用水平和垂直移动的结合。

模式 610a 描述了用于图象-到-块映射的螺旋形模式。螺旋形模式遵循水平移动，在图象-到-块映射过程中产生随后的 2D 块。模式 610b 和 610d 说明了互补模式，其中像素选择被水平和垂直移动，从而在图象-到-块映射过程中产生随后的 2D 块。此外，模式 610b 和 610d 说明了 2D 块之间像素选择的交替偏移。模式 610c 说明使用不规则像素抽样来产生用于图象-到-块映射的 2D 块。因此，图象-到-块映射遵守任意映射结构，只要像素被映射到 2D 块仅一次。

图 5、图 5a 和图 6 描述了用于产生 2D 块的图象-到-块映射.显然，该处理可以用于 3D 块。如上所述，3D 块的产生遵守与 2D 块相同的边

界定义，但是边界划分延伸到随后的帧上，从而产生了 3D 块。具体而言，3D 块是通过收集用于定义第一帧中 2D 块的像素与来自随后帧中 2D 块的像素而产生的。在一个实施例中，来自第一帧的 2D 块与来自随后帧的 2D 块中的像素都是来自完全相同的位置。

帧集合内块混洗

给定图象的像素值在局部区域密切相关。但是，在同一图象的另一个区域，像素值可能有很不同的值。因此，在编码之后，一部分图象中空间接近的 2D 或 3D 块的 DR 和 MIN 值具有类似值，而另一部分图象中块的 DR 和 MIN 值可能非常不同。因此，当缓存器顺序填充了来自图象中空间接近的 2D 或 3D 块的编码数据时，会出现不成比例的缓存器空间利用。帧集合内块混洗在 ADRC 编码之前出现，并包括对在图象-到-块映射处理过程中产生的 2D 或 3D 块进行混洗。这个混洗过程确保在随后的 ADRC 编码过程中均衡的缓存器使用。

图 7a-7d 说明对 3D Y 块混洗的一个实施例。图 7a-7d 中的 3D Y 块通过将上述图象-到-块映射处理应用于只包含 Y 信号的帧对而产生。3D Y 块的混洗确保用于存储编码帧对的缓存器包含来自帧对不同部分的 3D Y 块。这样导致 ADRC 编码过程中相类似的 DR 分散。每个缓存器内类似的 DR 分散导致一致的缓存器利用。

图 7a-7d 也说明了使用物理上不连贯的 3D 块进行 3D 块混洗，以便确保连续分组的传输损耗产生的损坏的块属性被分散到图象上，而不是在图象的局部区域。

块混洗的设计在出现小、中或大的突发分组损耗时都能广泛地分散块属性。在本实施例中，小量突发损耗被认为是几个分组丢失；中等损耗是一个缓存器中可以存储的数据量的丢失；而大量损耗是一个段中可以存储的数据量的丢失。在 3D 块混洗过程中，每个三相邻块组从图象中相对较远的部分选出。因此，在随后的组内 VL-数据混洗过程（后面将详细描述）中，每个组由具有不同统计特性的 3D 块构成。将块属性损耗分散，可得到增强的差错恢复，因为未损坏的 3D 块周围环绕损坏的 3D 块，未损坏的 3D 块可以用于恢复丢失的数据。

图 7a 说明一个包含水平方向上的 66 个 3D Y 块、垂直方向上的 60 个 3D Y 块的帧对。3D Y 块被分配到段 0-5。如所说明的，3D Y 块的

指定按照 2 乘 3 列段，这样，来自每个段的一个 3D Y 块关联于一个段。因此，如果不进行进一步的混洗而且出现了第一个 880 分组的突发损耗，与段 0 关联的所有块属性都会丢失。尽管如此，正如后面所描述的，进行 FL-数据混洗会进一步地分散块属性损耗。

图 7b 说明用于进入段 0 的编号为“0”的 3D Y 块的扫描顺序。图 7a 的每个“0”3D Y 块编号为 0、1、2、3、...、659，表示它们在被输入段 0 的流中的位置。使用块编号来分配段指定，其余的 3D Y 块输入到段 1-5，因此导致一个帧对在多个段上混洗。

图 7c 说明包括一个段的 660 个 3D Y 块。编号为 0-65 的 3D Y 块输入到缓存器 0。类似地，与编号的 3D Y 块相邻的 3D Y 块输入到缓存器 1。该过程重复，直到填满缓存器 2-9。因此，在数据传输过程中一个缓存器的损坏导致来自图象不同部分的 3D Y 块的丢失。

图 7d 说明“0”3D Y 块在一个缓存器中的最后顺序。3D Y 块 0、1 和 2 占据缓存器的前三个位置。该过程对缓存器的其余部分重复。因此，三个 3D Y 块在数据传输过程中的丢失导致图象内相距很远的 3D Y 块的丢失。

图 7a-d 说明了帧集合的 3D Y 块的 3D 块分散的一个实施例。但是，在另一个实施例中，提供了 3D U 块和 3D V 块的 3D 块属性。3D U 块通过将上述的图象-到-块映射处理提供给只包含 U 信号的帧集合而产生。类似地，3D V 块通过将图象-到-块映射处理提供给只包含 V 信号的帧集合而产生。3D U 块和 3D V 块遵循上述的 3D Y 块分布。但是，如前所述，3D U 块和 3D V 块数量各相对 3D Y 块有 1:6 的比例。

图 7a-d 用于说明 Y 信号的帧集合内块混洗的一个实施例，通过这样的操作，高达 1/6 传输过程中分组损耗的突发错误可以容忍而且进一步保证了均衡的缓存器利用。本领域技术人员会理解的是段、缓存器以及 ADRC 块指定可以改变，以便确保对付 1/n 的突发错误损耗并修改缓存器利用。

部分缓存

如图 3 所示，ADRC 编码和缓存过程在步骤 4 出现。根据编码技术，图象-到-块映射处理过程中产生的 2D 或 3D 块被编码，从而产生 2D 或 3D ADRC 块。一个 3D ADRC 块包含 Q 码、MIN 值、运动标志以及 DR。

类似地，一个 2D ADRC 块包含 Q 码、MIN 以及 DR。但是 2D ADRC 块不包括运动标志，因为编码在单帧或单个域上执行。

多种缓存技术可以在现有技术中找到（例如，见 High Efficiency Coding Apparatus（高效编码装置），美国专利 4,845,560，Kondo 等，以及 High Efficiency Coding Apparatus（高效编码装置），美国专利 4,722,003，Kondo）。这两项高效编码装置专利都在这里结合参照。

下面提到的部分缓存处理描述了确定用于 ADRC 编码的编码比特的创造性方法。具体而言，部分缓存描述了从门限表中选择门限值的方法，该门限表设计为在远程终端之间提供恒定的传输率同时限制了差错传播。在另一个实施例中，门限表被进一步设计为提供最大的缓存器利用。在一个实施例中，缓存器是存储来自给定帧集合的 1/60 部分编码数据的存储器。门限值用于确定前面描述的图象-到-块映射处理所产生的 2D 或 3D 块中像素编码所用的 Q 比特数。

门限表包括多行门限值，也称为门限集合，门限表中的每一行用门限索引来编排。在一个实施例中，门限表按照如下方式组织：产生较大数量 Q 码比特的门限集合处于门限表的较高行。因此，对于具有可用的预定比特数的给定缓存器来说，编码器 110 在门限表中向下移动直到遇到产生小于预定比特数的门限集合。这个恰当的门限值用于对缓存器中的像素数据编码。

在一个实施例中，希望不超过 30Mbps 的传输速率。所希望的传输速率导致有 31,152 比特用于任何给定缓存器中的 VL-数据存储。因此，对于每个缓存器，计算累积的 DR 分布并从门限表中选出门限集合，以便将 3D 或 2D 块中的像素编码为 VL-数据。

图 8 说明缓存器 0 中所选的门限值和 DR 分布的一个实施例。图 8 的纵轴包括累积的 DR 分布。例如，值“b”等于其 DR 大于或等于 L_3 的 3D 或 2D 块数。横轴包括可能的 DR 值。在一个实施例中，DR 值的范围从 0 到 255。门限值 L_4、L_3、L_2 和 L_1 描述了用于确定缓存器编码的门限集合。

在一个实施例中，存储在缓存器 0 中的所有块使用门限值 L_4、L_3、L_2 和 L_1 来进行编码。因此，DR 值大于 L_4 的块，其像素值使用 4 比特编

码。类似地，属于那些 DR 值在 L_3 和 L_4 之间的块的所有像素用 3 比特来编码。属于那些 DR 值在 L_2 和 L_3 之间的块的所有像素用 2 比特来编码。属于那些 DR 值在 L_1 和 L_2 之间的块的所有像素用 1 比特来编码。最后，属于那些 DR 值小于 L_1 的块的所有像素用 0 比特来编码。L_4、L_3、L_2 和 L_1 的选择使得用于对缓存器 0 中的所有块编码的总比特数尽可能地接近 31,152 比特的极限，而不超过 31,152 极限。

图 8a 说明在一个实施例中使用部分缓存。帧 800 被编码并存储在缓存器 0-59 中。假设传输错误阻止了差错恢复，帧 800 的解码过程就会停止直到对丢失数据进行了差错恢复。但是，部分缓存限制了缓存器内的差错传播，因此允许对其余缓存器解码。在一个实施例中，传输错误阻止了缓存器 0 中块 80 的 Q 比特和运动标志恢复。部分缓存限制了差错传播到缓存器 0 中的其余块。因为缓存器 0 的结尾和缓存器 1 的开始由于固定缓存器长度而已知，差错传播限制到缓存器 0。因此，解码器 120 可以没有延迟地开始缓存器 1 内的块处理。此外，不同门限集合用于对不同缓存器编码，使编码器 110 最大化/控制一个给定缓存器内包含的 Q 码比特数，从而允许较高的压缩比。此外，部分缓存处理提供了恒定的传输率，因为缓存器 0-59 由固定长度组成。

在一个实施例中，缓存器的可变空间不完全用 Q 码比特填充，因为存在有限数目的门限集合。因此，固定长度缓存器中的其余比特填充了被称为后同步信号的预定的比特流图形。正如随后将要描述的那样，后同步信号允许双向数据恢复，因为后同步信号描述了缓存器末尾之前 VL-数据的结尾。

缓存器内 YUV 块混洗

Y、U 和 V 信号各具有独特的统计特性。为了改善 Q 比特和运动标志恢复处理（下面描述的），Y、U 和 V 信号在缓存器内复接。因此，传输损耗对特定信号基本上没有影响。

图 9 说明了缓存器内 YUV 块混洗处理的一个实施例，其中 YUV ADRC 块分别从 Y、U 和 V 信号中得到。缓存器 900 说明了帧集合内块混洗之后的 ADRC 块指定。缓存器 900 包括 66 个 Y-ADRC 块，其后接着 11 个 U-ADRC 块，其后再接着 11 个 V-ADRC 块。缓存器 910 表示了缓存器内 YUV 块混洗之后 YUV ADRC 块的组织。如所示，三个 Y-ADRC 块后面接

着一个 U-ADRC 块或三个 Y-ADRC 块，其后面又接着一个 V-ADRC 块。缓存器内 YUV 块混洗减少了缓存器内相邻块比特流之间的相似性。具有不同信号（即，YUV 比或其它色度空间）的缓存器内 YUV 块混洗的另一个实施例按照最初的图象格式也是可能的。

组内 VL-数据混洗

组内 VL-数据混洗包括三个处理步骤.这三个处理步骤包括 Q 码级联、Q 码重指定以及随机化级联的 Q 码。图 10 说明组内 VL-数据混洗的一个实施例，其中三个处理步骤连续应用于缓存器中存储的 Q 码。在另一个实施例中，处理步骤的子集应用于组内 VL-数据混洗。每个处理步骤独立地协助传输过程中数据丢失的差错恢复。因此，对每个处理步骤独立描述。差错恢复的详细描述在下面数据恢复的讨论中提供。

1.Q 码级联

Q 码级联保证 ADRC 块组一起解码。组解码有利于差错恢复，因为在下面详述的数据恢复处理过程中附加信息可从相邻块提供。对于一个实施例，Q 码级联独立地应用于存储在缓存器中的每个三 ADRC 块组。在另一个实施例中，组包括来自不同缓存器的 ADRC 块。在三个 ADRC 块上的 Q 码的级联被描述为产生一个级联的 ADRC 瓦块.图 11 和图 11a 表示了一个产生级联的 ADRC 瓦块的实施例。

图 11 说明从 2D ADRC 块中产生级联的 ADRC 瓦块的一个实施例。具体而言，对 2D ADRC 块 0、1 和 2 中所包括的每个 Q 码（q_0-q_{63}）执行级联，从而产生级联 ADRC 瓦块 A 的 64 个 Q 码。例如，2D ADRC 块 0 的第一 Q 码 q0,0（第 0 个量化值）级联到 2D ADRC 块 1 的第一个 Q 码 q0,1。两个级联的 Q 码再级联到 2D ADRC 块 2 的第一个 Q 码 q0,2，因此产生级联的 ADRC 瓦块 A 的 Q_0。处理一直重复到产生 Q_{63}。或者，在级联的 ADRC 瓦块 A 中产生的 Qi 由下式描述

$$Q_i = \lfloor q_{i,0}, q_{i,1}, q_{i,2} \rfloor \qquad i=0,1,2,...63.$$

此外，与级联的 ADRC 瓦块 A 中的每个 Qi 关联，有相应数目的 N 比特代表产生单个 Q_i 的级联比特总数。

图 11a 说明从包括运动块的帧对中产生级联的 ADRC 瓦块的一个实施例。运动块是具有设置的运动标志的 3D ADRC 块。当前面描述的图象-到-块映射处理所产生的两个 2D 块结构内的预定数目像素在第一帧

和后续帧之间改变其值时，就设置运动标志。在另一个实施例中，当第一帧和后续帧的 2D 块之间每个像素改变的最大值超过预定值时，就设置运动标志。相反，非运动（即静止）块包括一个未设运动标志的 3D ADRC 块。当第一帧和后续帧的两个 2D 块内的预定像素数的值没有改变时，运动标志保持不设置。在另一个实施例中，当第一帧和后续帧之间每个像素改变的最大值不超过预定值时，运动标志保持不设置。

运动块包括来自第一帧中编码的 2D 块以及后续帧中编码的 2D 块的 Q 码。对应于单个编码的 2D 块的 Q 码集合被称为 ADRC 瓦块。因此，一个运动块产生两个 ADRC 瓦块。但是由于没有运动，一个静止块只需包括运动块一半数目的 Q 码，因此只产生一个 ADRC 瓦块。在本实施例中，静止块的 Q 码通过在第一帧的 2D 块和后续帧中相应 2D 块之间对相应的像素值进行平均而产生。每个平均的像素值随后被编码，产生构成单个 ADRC 瓦块的 Q 码集合。因此，运动块 1110 和 1130 产生 ADRC 瓦块 0、1、3 和 4。静止块 1120 产生 ADRC 瓦块 2。

图 11a 中级联 ADRC 瓦块的产生可以把 ADRC 瓦块 0-4 的 Q 码级联到级联的 ADRC 瓦块 B。具体而言，对 ADRC 瓦块 0、1、2、3 和 4 中包括的每个 Q 码（q0-q63）进行级联，从而产生级联的 ADRC 瓦块 B 的 64 个 Q 码。或者，每个 Q 码（Qi）在级联的 ADRC 瓦块 B 中的产生由以下数学公式来进行描述：

$$Q_i = \lfloor q_{i,0}, q_{i,1}, q_{i,2}, q_{i,3}, q_{i,4} \rfloor \qquad i=0, 1, 2, \ldots 63$$

2. Q 码重指定

Q 码重指定可以确保传输损耗所引起的比特错误被局限在空间分离的像素内。具体而言，在 Q 码重指定过程中，Q 码被重新分布，而且重新分布的 Q 码比特被混洗。因此，Q 码重指定有利于差错恢复，因为未损坏的像素环绕每个损坏的像素。此外，因为像素损坏均匀地分布在 ADRC 块上，所以有助于 DR 和 MIN 恢复，DR 和 MIN 恢复在下面的数据恢复讨论中详述。

图 12 说明在 1/6 突发错误损耗的传输损耗过程中像素破坏的一个实施例。具体而言，2D ADRC 块 1210、1220 和 1230 中的每一个包括使用 3 比特编码的 64 个像素。因此，2D ADRC 块的每个像素（p_0 到 P_{63}）用 3 比特来代表。2D ADRC 块 1210 表示了当每 6 比特的第一比特丢失

时的比特损耗模式，用加黑的方块表示，类似地，当每 6 比特的第二比特或第四比特丢失时的比特损耗模式分别在 2D ADRC 块 1220 和 1230 中表示。图 12 说明：没有 Q 码重指定时，对于 1/6 突发错误损耗，2D ADRC 块 1210、1220 和 1230 所有像素的一半被破坏。

对于一个实施例，Q 码重指定被独立应用于缓存器中存储的每个级联的 ADRC 瓦块，因此确保了在去混洗时比特错误被局限在空间分离的像素内。在另一个实施例中，Q 码重指定被应用于缓存器中所存储的每个 ADRC 块。

图 12a 说明从级联的 ADRC 瓦块产生混洗的 Q 码比特流的 Q 码重指定的一个实施例。表 122 和表 132 说明 Q 码重分布。比特流 130 和 140 说明了 Q 码比特的混洗过程。

表 122 表示用于级联的 ADRC 瓦块 A 的级联的 Q 码。Q_0 是第一个级联的 Q 码，Q_{63} 是最后一个级联的 Q 码。表 132 说明 Q 码的重分布。对于一个实施例，Q_0，Q_6，Q_{12}，Q_{18}，Q_{24}，Q_{30}，Q_{36}，Q_{42}，Q_{48}，Q_{54} 和 Q_{60} 被包括在第一集合、分区 0 中。按照表 132，后面 11 个级联的 Q 码被包括在分区 1 中。这些步骤对分区 2-5 重复。一个分区的边界由表 132 中的垂直线描述。将级联 Q 码向 6 个分区的这种空间分离指定确保了 1/6 突发错误损耗将导致分散到一组连续像素上的比特损耗模式。

图 12b 说明重新分布 Q 码的 1/6 突发错误损耗所产生的比特模式损耗的一个实施例。具体而言，2D ADRC 块 1215、1225 和 1235 中每个包括用 3 比特编码的 64 个像素。因此，每个 2D ADRC 块的每个像素（p_0 到 P_{63}）用 3 比特代表。在 2D ADRC 块 1215、1225 和 1235 中用加黑的方块表示的比特损耗模式被局限在一组连续的像素上。因此，对于给定的段损耗，每个 2D ADRC 块 1215、1225 和 1235 内只有 11 个连续像素被破坏。在另一个实施例中，Q 码到分区的指定包括来自不同运动块的 Q 码，因此提供到 6 个段的空间和时间分离的 Q 码指定。这样就产生了在 1/6 突发错误损耗时附加的未损坏的空间-时间像素，并进一步促进了更增强的差错恢复。

参考图 12a，表 132 中重分布的 Q 码比特在所产生的比特流上被混洗，这样，比特流中的相邻比特来自相邻的分区。表 132 中所有分区的 Q 码比特被级联成比特流 130。对于给定的分区，比特流 130 中的相

邻比特被分散到所产生的比特流 140 中的每个第 6 个位置上。因此，比特流 140 中号码 0 到 5 的比特包括来自每个分区中第一 Q 码的第一比特。类似地，比特流 140 中号码 6 到 11 的比特包括来自每个分区中第一 Q 码的第二比特。该处理对所有 Q 码比特重复。因此，1/6 突发差错损耗将产生空间分离的像素损耗。

图 12c 说明了重指定（即，重分布并混洗）的 Q 码的 1/6 突发错误损耗所造成的比特模式损耗的一个实施例。具体来说，2D ADRC 块 1217、1227 以及 1237 每个包括 64 个用三比特编码的像素。因此，每个 2D ADRC 块的每个像素 P_0 到 P_{63}，可用三比特代表。在 2D ADRC 块 1217、1227 和 1237 中，用黑方块表示的比特损耗模式在空间分离的像素上分布，因此有利于像素差错恢复。

3. Q 码比特的随机化

利用屏蔽密钥来使 Q 码比特随机化，从而有助于解码器恢复丢失和损坏的数据。具体而言，在编码过程中，将密钥（用 KEY 标志）用于屏蔽 Q 码比特流。因此，解码器必须识别 KEY 的正确值，以便解除对 Q 码比特流的屏蔽。

在一个实施例中，KEY 用于屏蔽三个 ADRC 块的 Q 码重指定所产生的 Q 码比特流。正如前面所描述的，ADRC 块包括 FL-数据和 Q 码。屏蔽密钥的每个密钥元素（“d_i”）由 FL-数据值和相应 ADRC 块所关联的量化比特（“q_i”）数的组合来产生。在一个实施例中，运动标志和 Q 比特用于定义一个密钥。因此，在这个实施例中，密钥元素值从下面数学公式产生：

$d_i=5 \cdot m_i+q_i$，这里 $i=0, 1, 2$ 而且 $q_i=0, 1, 2, 3, 4$

变量 m_i 等于运动标志。因此，当相应的 ADRC 块是静止块时，m_i 等于 0，当相应的 ADRC 块是运动块时，m_i 等于 1。此外，变量 q_i 代表用于对相应 ADRC 块编码的量化比特。因此，对于四比特 ADRC 编码技术，q_i 具有的值为 0、1、2、3 或 4。在一个实施例中，一组三个 ADRC 块的 KEY 根据下式用三个密钥元素（“d_i”）定义：

$KEY=d_0+10 \cdot d_1+100 \cdot d_2$

因此，在运动标志或 Q 比特数据的恢复过程中，根据用于产生屏蔽密钥的值重新生成可能的密钥值。重生成的密钥值用于对所接收的

导致产生候选解码的 Q 码比特流解除屏蔽。重生成密钥值以及特定候选解码技术选择的详细描述在下面讨论数据恢复时提供。

在另一个实施例中，屏蔽密钥从各种元素中产生。因此，给解码器提供与一个元素有关的特定信息，而不必在传输介质上发送该元素。在一个实施例中，与 ADRC 块对应的 DR 或 MIN 值用于产生屏蔽密钥，以便屏蔽代表 ADRC 块的比特流。

图 10-12 说明在传输过程中容限达 1/6 分组数据损耗的组内 VL-数据混洗。本领域技术人员将会理解的是，总分区数以及比特划分可以改变，以保证对付 1/n 突发错误损耗。

段内 FL-数据混洗

段内 FL-数据混洗描述了在不同段上重新排列块属性。重新排列块属性提供了数据丢失的分散。具体而言，当来自一个段的 FL-数据在传输过程中丢失时，DR 值、MIN 值以及运动标志值的丢失不属于同一块。图 13 和 14 说明了段内 FL-数据混洗的一个实施例。

图 13 说明段 0 到 5 的内容。对于一个实施例，每个段包括 880 个 DR、880 个 MIN、880 个运动标志以及对应于 660 个 Y 块、110 个 U 块以及 110 个 V 块的 VL-数据。正如图 MIN 混洗 1300 中所示，段 0 的 MIN 值移动到段 2，段 2 的 MIN 值移动到段 4，段 4 的 MIN 值移动到段 0。此外，段 1 的 MIN 值移动到段 3，段 3 的 MIN 值移动到段 5，段 5 的运动标志值移动到段 1。

图 13a 说明运动标志混洗。如所示，在图运动标志混洗 1305 中，段 0 的运动标志值移到段 4，段 2 的运动标志值移到段 0，段 4 的运动标志值移到段 2。此外，段 1 的运动标志值移到段 5，段 3 的运动标志值移到段 1，段 5 的运动标志值移到段 3。损耗模式 1310 说明了段 0 在传输中丢失后的 FL-数据损耗。

对于特定的块属性，图 13 和图 13a 说明了在段之间对特定块属性的所有情况的混洗。例如，在图 13 中，来自段 0 的 880 个 MIN 值全体与段 2 的 880 个 MIN 值互换。类似地，在图 13a 中，段 0 的 880 个运动标志全体与段 4 中的 880 个运动标志互换。在连续分组的传输丢失过程中，这种全体性的块属性混洗导致一个块组的特定块属性的不均匀地丢失。在一个实施例中，一个块组包括三个 ADRC 块。

图 14 说明了用于 DR、MIN 以及运动标志的模三混洗过程的一个实施例。模三混洗描述在三个不同段中的三个块（即，一个块组）上共享的混洗模式。混洗模式对于三个不同段内的所有块组重复。但是，不同混洗模式被用于不同的块属性。因此，模三混洗过程将块属性分散到所有三个段中。具体而言，对于给定的块组，模三混洗确保在段传输丢失过程中特定块属性只有一次丢失。因此，在下面描述的数据恢复过程中，就产生了恢复块内数据丢失的、数目减少的候选解码。

如 DR 取模混洗 1410 所示，一个段存储 880 个 DR 值。因此，DR 值编号为 0-879，根据给定的 DR 值所取自的块而定。在模三混洗中，三个段的 FL-数据内容被混洗。0-2 的计数用于识别为取模混洗而标识的三个段中的每个 DR 值。因此，属于编号 0、3、6、9...块的 DR 属于计数 0。类似地，属于编号 1、4、7、10...块的 DR 属于计数 1，属于编号 2、5、8、11..块的 DR 属于计数 2。因此，对于给定的计数，与该计数关联的 DR 值在段 0、2 和 4 上混洗。类似地，与同一计数关联的 DR 值在段 1、3 和 5 上混洗。

在 DR 取模混洗 1410 中，属于计数 0 的 DR 值不混洗。属于计数 1 的 DR 值混洗。具体而言，段 A 中的计数 1 DR 值移动到段 B，段 B 中的计数 1 DR 值移动到段 C，段 C 中的计数 1 DR 值移动到段 A。

属于计数 2 的 DR 值也被混洗。具体而言，段 A 中的计数 2 DR 值移动到段 C，段 B 中的计数 2 DR 值移动到段 A，段 C 中计数 2 DR 值移动到段 B。

MIN 取模混洗 1420 说明了 MIN 值的模三块属性混洗过程的一个实施例。一个段包括 880 个 MIN 值。在 MIN 取模混洗 1420 中，在 DR 取模混洗 1410 中用于计数 1 和计数 2 的混洗模式移到计数 0 和计数 1。具体而言，DR 取模混洗 1410 中用于计数 1 的混洗模式被应用于计数 0。DR 取模混洗 1410 中用于计数 2 的混洗模式被应用于计数 1，属于计数 2 的 MIN 值不混洗。

运动标志取模混洗 1430 说明了运动标志值的模三块属性混洗过程的一个实施例。一个段包括 880 个运动标志值。在运动标志取模混洗 1430 中，在 DR 取模混洗 1410 中用于计数 1 和计数 2 的混洗模式分别移到计数 2 和计数 0。具体而言，DR 取模混洗 1410 中用于计数 2 的混

洗模式被应用于计数 0。DR 取模混洗 1410 中用于计数 1 的混洗模式被应用于计数 2，属于计数 1 的运动标志值不混洗。

图 14a 说明取模混洗 1410、1420 和 1430 的取模混洗结果。取模混洗结果 1416 表示属于段 0 的块的每个属性的目标。在这个例子中，段 0 对应于图 14 的段 A。这个目标根据图 14 的取模混洗 1410、1420 和 1430 而规定。图 14a 也说明了在传输过程中段 0 丢失后块属性损耗的分散。具体而言，损耗模式 1415 表示在随后的去混洗被应用于最初用取模混洗 1410、1420 和 1430 进行混洗的接收数据上之后，DR、运动标志以及 MIN 值在 6 个段上的丢失。如图 14a 所示，块属性丢失周期性地分散在段 0、2 和 4 上，而段 1、3 和 5 没有块属性丢失。此外，空间损耗模式 1417 说明了在传输过程中段 0 丢失之后损坏的 FL-数据的去混洗空间分布。空间损耗模式 1417 表示随后的去混洗被应用于接收数据之后的 DR、运动标志以及 MIN 值的丢失。在空间损耗模式 1417 中，损坏的块被未损坏的块环绕，损坏的块属性可以用周围未损坏的块来恢复。

图 14 和图 14a 说明了模三混洗模式以及传输过程中段丢失之后块属性丢失的分散。在另一个实施例中，计数变量或段号可变以便改变丢失块属性的分布。图 14b 说明了取模混洗结果 1421 和损耗模式 1420。类似地，图 14c 说明了取模混洗结果 1426 以及损耗模式 1425。损耗模式 1420 和损耗模式 1425 说明了块属性丢失在 6 个段上的分布，与前面描述的三个段不同。

应该考虑到在另外的实施例中，将会分散各种块属性组合以便进行混洗处理。

段内 VL-数据混洗

在段内 VL-数据混洗过程中，在预定数目的段（例如，6 个段）之间的比特被设计为：在高达 1/6 分组传输损耗过程中，可确保空间分开和周期性的 VL-数据丢失。图 15 和 16 说明了段内 VL-数据混洗处理的一个实施例。

在本实施例中，需要接近于 30Mbps 的传输率。因此，所需的传输速率导致 60 个缓存器每个中的 VL-数据有 31,152 比特。其余空间被缓存器中包括的 88 个块的 FL-数据所用。图 15 包括传输率接近 30Mbps

的一个帧集合内 VL-数据缓存器的组织。如前面所描述的，部分缓存用于使每个缓存器内可用 VL-数据空间的使用最大化，而且未使用的 VL-数据空间用后同步信号填充。

图 16 说明保证空间分开和周期性的 VL-数据损耗的混洗过程的一个实施例。第一行说明来自图 15 的 60 个缓存器的 VL-数据重新组织成 1,869,120 比特的级联流。第二行说明每个第 6 比特组成新的比特流。因此，当解码器后来将该过程逆转时，高达 1/6 被传输数据的突发损耗被转换为周期性损耗，其中至少 5 个未损坏的比特将每组两个损坏比特分开。

第三行说明将流 2 的每个 10 比特组成新比特流，即流 3。组的边界也由段中的比特数定义。每节 10 个比特的流 2 分组保证 1/60 数据损耗在每组 2 个损坏比特之间产生 59 个未损坏的比特。这样当 88 个连续数据分组丢失时提供空间分开和周期性的 VL-数据丢失。

第四行说明将流 3 的每个 11 比特组成流 4。组的边界也由段中的比特数定义。每节 11 个比特的流 3 分组保证 1/660 数据损耗在损坏比特之间产生 659 个未损坏的比特，从而当 8 个连续分组传输丢失时产生空间分开和周期性的 VL-数据丢失。

流 4 内的每个 31,152 比特组连续重存储在缓存器 0-59 中，第一比特组存储在缓存器 0 中，最后比特组存储在缓存器 59 中。

本领域技术人员会理解的是，图 16 的分组要求是可变的，以便在高达 1/n 传输损耗时保证空间分开和周期性的 VL-数据丢失容限。

传输

前面描述的混洗过程产生带有混合了 FL-数据和 VL-数据的缓存器。对于一个实施例，按照分组结构 200，分组从每个缓存器中产生并在传输介质 135 上发送。

数据恢复

正如前面所注意的，用于数据比特流编码的本发明方法能够进行由于数据分组丢失而通常出现的增强的数据恢复。解码过程的概述已经在图 4 中表示。

参考图 4，分组中接收的数据通过多级去混洗过程（步骤 425、430、435 以及 440）来处理，其中通过分组接收的不同级别或不同部分的比

特流被去混洗以便恢复数据。根据本领域已知的概念（例如，Kondo,Fujimori,Nakaya,“Adaptive Dynamic Coding Scheme for Future HDTV Digital VTR（未来HDTV数字录象机的自适应动态编码方案）”，Fourth International Workshop on HDTV and Beyond，1991年9月4-6日，意大利，都灵），ADRC解码在步骤445应用于数据。

然后进行帧集合内的块去混洗，随后执行块-到-图象映射，即步骤450、455。步骤425、430、435、440、445、450和455是前面执行的对数据编码的处理步骤的逆处理，这里就不再详细讨论。但是，应该注意的是在一个实施例中，步骤425、430和440所代表的去混洗级别是与数据无关的。例如，所执行的去混洗处理通过地址映射或表查找来预定或规定。由于去混洗步骤425、430和440是与数据内容无关的，由于（例如）分组丢失造成的数据损耗不会防止去混洗步骤的执行。类似地，步骤450和455是与数据无关的。但是组内VL-数据去混洗处理是与数据内容相关的。更具体地说，组内VL-数据去混洗处理用于确定组中的块的量化码。因此在步骤435，如果分组丢失了，受影响的组就不能被处理。

在进行了去混洗、解码和映射之后（步骤425、430、435、440、445、450和455），执行恢复处理，以便恢复处于丢失分组中的Q比特和运动标志值。Q比特值通常由于DR丢失（由丢失分组造成）而丢失。当Q比特或运动标志值未知时，像素的Q码比特不能从数据比特流中确定。如果Q比特或运动标志值的确定不恰当，那么这个错误将作为随后块的起点而传播，因为缓存器中的数据将被错误地识别。

图17描述了恢复Q比特和运动标志值的一般过程。这个特定实施例描述了使用多数据块恢复Q比特和运动标志值的过程；但是，应该考虑到特定的块数不受这里的讨论所限制，而且可以是一块或多块。参考图17，基于步骤1705中比特流的出错检测，基于规定参数的候选解码针对所检测的三个块而产生。在步骤1715，每个候选解码在它是精确解码过程的几率上记分，在步骤1720，使用具有最佳记分的候选解码，该解码识别能够进行随后的受影响块的像素解码的Q比特和运动标志值。

返回参考图 4 的解码过程，一旦最佳解码被选出，由于丢失分组造成丢失的任何 DR 或 MIN 值在步骤 465 被恢复。本领域技术人员已知的各种恢复处理都可以用于恢复 DR 和 MIN，包括相邻块值的最小二乘或平均。作为一个例子，见 Kondo, Fujimori, Nakaya 的“Adaptive Dynamic Coding Scheme for Future HDTV Digital VTR（未来 HDTV 数字录象机的自适应动态编码方案）”，Fourth International Workshop on HDTV and Beyond, 1991 年 9 月 4-6 日，意大利，都灵。在本实施例中，本发明的图象-到-块映射过程以及由此产生的数据结构增加了相邻块的数目，因此提供附加数据并有利于更精确的 DR 或 MIN 恢复。具体而言，在一个实施例中，DR 和 MIN 如下恢复：

$$DR' = \frac{m \cdot \sum_i (y_i - MIN) \cdot q_i}{\sum_i q_i^2}$$

这里 DR'对应于恢复的 DR，q_i 是 ADRC 块的第 i 值，而且 $q_i \in \{0, 1, \ldots, 2^Q-1\}$；对于边缘匹配 ADRC，$m=2^Q-1$，对于非边缘匹配 ADRC，$m=2^Q$；$y_i$ 是相邻块像素的解码值，Q 是 Q 比特值；而且

$$MIN' = \frac{\sum_i \left(y_i - \frac{DR}{m} \cdot q_i \right)}{N}$$

这里 MIN'对应于恢复的 MIN，N 是用于求和的项数（例如，当 i=0-31 时 N=32）。在另一个实施例中，如果同一块的 DR 和 MIN 同时损坏，根据下式恢复 DR 和 MIN：

$$DR' = \frac{m \cdot \left[N \cdot \sum_i q_i \cdot y_i - \sum_i q_i \cdot \sum_i y_i \right]}{N \cdot \sum_i q_i^2 - \left[\sum_i q_i \right]^2}$$

$$MIN' = \frac{\sum_i \left(y_i - \frac{DR'}{m} \cdot q_i \right)}{N}$$

在步骤 470，ADRC 解码应用于在 Q 比特和运动标志恢复之前没有解码的那些块，并在步骤 475 执行像素恢复处理，以便恢复由于丢失分组或随机错误而可能出现的任何错误像素数据。此外在步骤 480 执

行 3:1:0->4:2:2 的返回转换，以便将图象置于所选的显示格式。

图 18 说明了本发明的解码过程中 Q 比特和运动标志恢复处理的一个特定实施例。在这个特定实施例中，处理的输入是相邻块信息、将要处理的三个块的块属性以及像素数据。也要输入表示丢失数据位置的错误标志。错误标志可以用本领域技术人员已知的各种方式来产生，而且这里将不会进一步讨论，只是说明标志表示哪些比特是损坏或丢失的分组所发送的。

在步骤 1805，产生候选解码。候选解码可以用各种方式产生。例如，尽管处理负担可能相当大，但是候选解码可以包括所有可能的解码。或者，候选解码可以基于预定的参数产生，以便减少待评估的候选解码数。

在本实施例中，候选解码基于对前面描述的组内 VL-数据混洗处理的比特流进行随机化所用的可能的密钥值来确定。此外，应该注意候选解码还受等待解码的比特长度以及对剩下多少块的了解所限制。例如，正如将要讨论的，如果处理最后一块，通常该块的解码长度是已知的。

继续本例，图 19 说明了本发明的可能情况，其中值 x 表示未知值（可能 由于分组丢失）。将进一步举例来解释。m_i 定义为第 i 块的运动标志，q_i 是第 i 块的量化比特数，n_i 是第 i 块的可能候选数，d_i 是前面在组内 VL-数据混洗中描述的第 i 块的密钥元素值。第 i 块在每个组内定义。在本例中，每个组内的块数是三。三块组的密钥按 $d_0+10\cdot d_1+100\cdot d_2$ 而产生。假设在第一块中运动标志未知，量化比特数是 2，则 m_0 等于 x，q_0 等于 2。根据上面描述的式子产生密钥元素，$d_i=5\cdot m_i+q_i$，d_0 的可能数字集合由{2 和 7}组成。因此，可能值（n_0）数是 2。假设第二块具有运动标志值 1 以及一个量化比特，d_1 的值是 $5\cdot 1+1=6$ 而且 $n_1=1$。第三块具有运动标志值 1 以及未知数目的量化比特。因此，数字 d_2 包括由{6，7，8，9}组成的集合而且 $n_2=4$。因此，这个组可能的候选数 M 是 $2\cdot 1\cdot 4=8$，而且用于产生候选解码的密钥在 662、667、762、767、862、867、962、967 中变化。这个处理优选地用于受数据丢失影响的每个组。

返回参考图 17，在步骤 1715，一旦根据密钥数据已经对数据解码，

就评估所产生的候选解码或在它是正确的数据解码的几率上记分。在步骤1720，具有最佳分的候选解码被选择使用。

各种技术可以用于对候选解码记分。例如，记分数可以从特定候选解码中块的像素与图象其它像素一致程度的分析中得到。优选地，记分数基于表示错误（例如误差平方和相关）的准则中得到。例如，对于相关，假设相邻像素在某种程度上紧密相关是相当有把握的。因此，一个很大或很小的相关表示候选解码是不是正确解码。

正如图18所示，分析了四种不同的准则，以选择最佳候选解码。但是，考虑可以分析一、二、三或更多种不同的准则以便选择最佳的候选解码。

参考图18，本实施例利用四个子记分准则，它们最后合并成最后记分数。具体而言，在步骤1815产生误差平方测量，步骤1820确定水平相关，步骤1825确定垂直相关，在步骤1830测量时间活动性（根据M个候选、N个块以及2帧/数据块，每个是M乘2N矩阵）。尽管使用了水平和垂直相关，但是应该认识到包括对角线相关在内的各种相关测量都可以检查。在步骤1835、1840、1845、1850，对每个准则产生置信度测量，以便对所产生的测量归一化，并在步骤1855、1860、1865和1870产生每个不同准则的概率函数。这些概率函数然后通过（例如）将概率值相乘产生一个记分数来进行合并，例如图18所示的步骤1875中的似然函数。候选解码的记分数随后与所有候选解码记分数比较，以便确定最可能的候选者。

应该认识到可以使用各种技术评估候选解码并对每个候选产生“记分数”。例如，置信度测量是用来归一化准则的一种方式。此外，除了下面描述的以外，各种置信度测量都可使用。类似地，将基于每个准则的概率值相乘产生总似然函数只是合并所检查的各种准则的一种方式。

编码过程有利于确定最佳候选解码，因为不是最可能候选的候选解码一般有比较差的记分数，而最可能候选的解码会有比较好的记分数。具体而言，前面在组内VL-数据混洗过程中描述的Q码随机化过程在这方面有帮助。

图20a、20b、20c和20d提供了对图18的步骤1815、1820、1825

以及 1830 处执行不同测量以便对特定候选解码产生记分和总分的说明。图 20a 说明评价候选解码的像素 x_i 与它的解码的相邻 $y_{i,j}$ 比较的误差平方，其中下标“i, j”对应于“i”的相邻地址。优选去掉一些最大项，以便消除由于峰值（这是由于图象的合理边缘而出现的项）带来的任何影响。优选地，三个 $(x_i-y_{i,j})^2$ 最大项被丢弃，以便去掉可能出现的峰值。图 20b 说明了时间活动性准则。这仅当是或假设是运动块才有用。时间活动性准则假设候选解码越好，块间差别越小。因此候选解码越差，块间差别越大。空间相关假设：更可能的候选解码导致很重的相关，因为真实图象易于以很慢的一致方式改变。图 20c 所示的水平相关处理以及图 20d 说明的垂直相关处理利用了这个假设。

图 18 中步骤 1835、1840、1845 以及 1850 的置信度测量提供了对前面步骤（步骤 1815、1820、1825 以及 1830）所确定的准则归一化的处理。例如，在一个实施例中，误差平方的置信度测量在间隔[0,1]取值，并且如果误差相等则置信度等于 0，如果一个误差是 0 则置信度等于 1。其它归一化测量或方法也可考虑。

类似地，空间相关的置信度测量为：

最大值(Y, 0)-最小值(X, 0)

这里 Y 是最佳相关值，X 是当前候选解码的相关。时间活动置信度测量根据下式确定：

置信度=(a-b)/(a+b)

这里 a=max(X, M_TH)，b=max(Y, M_TH)，其中 M_TH 是候选块的运动门限，Y 是最佳测量值，即最小的时间活动性，而且 X 等于时间活动性的当前候选测量。

在图 18 中步骤 1855、1860、1865 以及 1870，为每个不同的准则产生概率函数。可使用各种方法产生概率测度。例如，可以规定置信度测量的记分数。如果置信度测量大于预定值，例如 0.8，基分就降低 10；如果是在 0.5 和 0.8 之间，基分就降低 5。图 21 所示的实施例中，有一个表用于产生误差平方测度准则的概率函数。该表包括基于经验确定的任意存储的数据，包括置信度和误差平方测度以及已知的候选解码。更具体地说，该表可以通过使用未损坏数据并假设 DR 被破坏或丢失而产生。然后产生正确和非正确解码的密钥以及置信度测量。该

表反映了正确与非正确解码的概率比。使用这个表，针对特定的误差平方值（行）以及置信度值（列），可以确定概率。例如，因此可以看到，对于置信度测量为 0 的各种误差平方测度，大约有 40%到 50%候选正确的概率。如果置信度不为 0，而是很小，则概率下降很大。类似的概率表可以基于相应的经验确定的准则测度和置信度测度针对相关及时间测量而产生。

所产生的概率被认为是在本实施例中产生“记分数”的数据，正如前面注意到的，其它对候选解码记分的技术也可以使用。在步骤 1875，不同的概率合并成似然函数 $L_i=j\cdot P_{i,j}$，这里 j 是概率函数 $P_{i,j}$ 的倍数函数，$P_{i,j}$是块 j、候选 i 的概率函数。因此选择能使函数 L_i 最大的候选。

返回参考图 18，恢复某些在丢失分组中发送的块属性可能是必要的。因此，在步骤 1810，DR 和 MIN 值在必要情况下恢复。可以使用各种技术，从缺省值、平均、误差平方函数到更复杂的技术，包括如下文献中讨论的那些：Kondo、Fujimori 和 Nakaya 的“Adaptive Dynamic Range Coding Scheme for Future HDTV Digital VTR（未来 HDTV 数字录象机的自适应动态范围编码方案）”以及 Kondo、Fujimori、Nakaya 和 Uchida 的“A New Concealment Method for Digital VCR（数字录象机的新隐蔽方法）”，IEEE Visual Singal Processing and Communications，1993 年 9 月 20-22 日，澳大利亚，墨尔本。所恢复的值用于产生候选解码，如上所述。

或者，DR 和 MIN 值在 Q 比特确定过程中确定。在图 22 中说明了这种情况。具体而言，如上面注意到的，在本实施例中运动标志和量化比特数用于编码过程并在随后用于恢复过程以缩减可能的候选解码的数目。如前面注意到的，其它信息也可以用于对数据编码。因此，DR 值及/或 MIN 值也可以用于对数据编码。或者 DR 的一部分比特用于编码（例如，DR 的两个最低有效比特）。尽管 DR 数据被编码，可能的候选解码数随着变量的增加而大大增加了。参考图 22，因此产生了 K·M 个候选解码，这里 K 是未知数据的候选值数，例如如果 DR_1、DR_2和 DR_3之和的两比特被编码（DR_1、DR_2和 DR_3代表组中块的 DR 值），那么 K=4。DR 和 MIN 因此用提供的辅助信息恢复，例如被编码的 DR_1、DR_2和 DR_3

之和的两比特。这样以附加开销为代价改善了检查较大数目的候选解码的候选选择过程。

应该注意的是，通常被解码的块越邻近，Q 比特和运动标志的恢复过程越好。此外，在一些实施例中，该处理被提供给缓存器内的每个随后的块；如果所有的或一些 FL-数据可用，候选解码数就会减少，假定该块所有 FL-数据都可用，候选解码可能减到 1 个。但是，希望 Q 比特和运动标志恢复过程全部避免，因为该处理是比较耗时的。此外，希望使用尽可能多的信息进行 Q 比特和运动标志恢复。在一个实施例中，从缓存器的开始处理各块，直到到达一个具有丢失 Q 比特/运动标志信息的块。这被定义为前向 Q 比特和运动标志恢复。在另一个实施例中，参考缓存器的末尾来确定缓存器最后块的结尾位置，并从缓存器的末尾恢复数据直到到达具有丢失 Q 比特/运动标志数据的块。这被定义为后向 Q 比特和运动标志恢复。

正如前面注意到的，由于 VL-数据的长度，块在长度上可变；因此需要确定构成块的 VL-数据的比特数以便正确地定位随后块在缓存器中的位置。在编码过程中，预定的而且优选为容易识别模式的后同步信号被放入缓存器，以便填充未用的比特位置。在解码过程中，后同步信号将处于该块和缓存器末尾之间。由于该模式是容易识别的一种，通过检查比特模式使系统能够定位后同步信号的开始，并因此定位缓存器中最后一块的结尾。这个信息可以按两种方式使用。如果最后一块包含损坏的 Q 比特/运动标志数据而且最后一块的开始是已知的（例如，前面的块已经被成功解码），那么紧接着的前面一块的结尾与后同步信号的开始之间的差对应于该块的长度。这个信息可以用于计算该块的 Q 比特及/或运动标志。后同步信号的开始位置也可以用于执行从最后一块开始并进行到缓存器开始的 Q 比特和运动标志恢复。因此，Q 比特和运动标志恢复处理可以双向实现。

图 23 说明了后同步信号在双向 Q 比特和运动标志恢复过程中的使用。参考图 23，缓存器 2300 包括 N 组 VL-数据块的 FL-数据 2303。每组由多个块（例如，3 块）组成。在本例中，前两组 2305、2310 被解码，第三组 215 不能立即被解码，因为 DR/运动标志数据损坏了。在这里，需要 Q 比特/运动标志恢复过程以便恢复损坏的数据。不是继续在

前向处理这些组，该处理参考通过查找后同步信号模式 220 而确定的缓存器结尾。确定了后同步信号的开始，由此得知最后块组的结尾。由于 DR/运动标志数据表示了 VL-数据的长度，最后一块的 VL 数据开始（因此也是紧接着前一块的结尾）就可以被确定了。因此，可以对这些块解码，例如块 225、230、235，直到到达损坏数据的块 240。然后，使用上述 Q 比特/运动标志恢复处理恢复损坏的 215、240 以及受阻的块 250。

应该注意双向处理不限于一系列向前和向后处理；处理可以在任一方向或双向上进行。此外，在一些实施例中，可能希望与改善效率并行地进行这种处理。最后，注意未损坏的受阻块可以通过直接访问 Q 比特/运动标志信息、而不执行上述 Q 比特/运动标志恢复过程而恢复。

正如前面注意到的，各种记分技术可以用于确定最佳候选解码以便选用来进行解码。在另一个实施例中，评估使用每种候选解码的图象平滑度。在一个实施例中，进行拉普拉斯测量。拉普拉斯测量测出二阶图象表面性质，例如表面曲率。对于线性图象表面，即平滑表面，拉普拉斯测量产生接近于 0 的值。

该处理将参照图 24a、24b 和 24c 来解释。图 24a 说明拉普拉斯内核的一个实施例。注意，也可以使用其它实施例。 内核“L”代表一个 3×3 区域。为了测量图象区域的平滑度，图象的 3×3 子区域（图 24b）与内核卷积，而且卷积值被平均。区域和子区域的大小（以及有关的内核大小）可以根据应用而变。

该处理的一个实施例参考图 24c 描述。这个实施例使用一个内核和 3×3 的子区域尺寸以及 8×8 的区域尺寸，，其中各个元素用 i, j 来标识。在步骤 2460，候选解码值 x[i][j]被归一化。例如，可以根据下式对这些值归一化：

$$x'[i][j]=\frac{x[i][j]}{\sqrt{\sum_{i,j}(x[i][j]-X_{mean})^2}}, \qquad 0\le i,j<8$$

这里，$X_{mean}=\frac{\sum_{i,j}x[i][j]}{64}, \qquad 0\le i,j<8$

在步骤 2465，根据下式归一化值被用于计算表示平滑度的块的拉普拉斯值 Lx：

$$l[i][j]=\sum_{m=-1}^{1}\sum_{n=-1}^{1}L[m][n]\cdot x'[i+m][j+n], \qquad 0\le i,j<8$$

$$Lx=\frac{\sum_{i,j}l[i][j]}{64}$$

块拉普拉斯值越接近 0，图象部分越平滑。因此，记分数可以基于块拉普拉斯值来测量，具有最小拉普拉斯值的解码是正确的解码。

拉普拉斯评估也可以使用候选编码值 q[i][j]来实现。其基本过程与图 24c 的候选解码值情况相同。这个实施例使用一个内核和 3×3 的子区域尺寸以及 8×8 的区域尺寸，其各个元素用索引 i，j 标识。在步骤 2460，候选解码值 q[i][j]被归一化。例如，可以根据下式对这些值归一化：

$$q'[i][j]=\frac{q[i][j]}{\sqrt{\sum_{i,j}(q[i][j]-Q_{mean})^2}}, \qquad 0\le i,j<8$$

这里，$Q_{mean}=\frac{\sum_{i,j}q[i][j]}{64}$, $0\le i,j<8$

在步骤 2465，根据下式归一化值被用于计算表示平滑度的块的拉普拉斯值 Lq：

$$l[i][j]=\sum_{m=-1}^{1}\sum_{n=-1}^{1}L[m][n]\cdot q'[i+m][j+n], \qquad 0\le i,j<8$$

$$Lq=\frac{\sum_{i,j}l[i][j]}{36}$$

块拉普拉斯值越接近 0，图象部分越平滑。因此，分数可以基于块拉普拉斯值来测量，具有最小拉普拉斯值的解码是正确的解码。

同时也要注意其它变化。在另一些实施例中，可以使用较高阶的图象表面性质作为平滑度测量。在那些情况下，会使用较高阶的内核。

例如，四阶块拉普拉斯测量可以使用四阶内核来进行。级联使用二阶拉普拉斯计算可以实现四阶内核。

还要注意的是，评估过程取决于图象是否比预定水平具有更大的活动性或运动。如果被评估的图象部分具有比预定水平大的运动，那么优选地基于场而不是基于帧进行测量。参考图 25 对其进行解释。图 25 解释了使用平滑度测量的处理；但是注意这个处理可以使用各种类型的测量来实现。

图象区域的帧 2505 由场 0 和场 1 组成。如果没有在步骤 2510 检测到运动，平滑度测量就通过在步骤 2515 计算每帧内块的块拉普拉斯值来计算。如果检测到超过预定水平的较大运动，块拉普拉斯测量就在每个场上进行（步骤 2520、2525），在步骤 2530 合并（例如，平均）两个测量，从而产生平滑度测度。

运动可以用各种方式检测/测量。在一个实施例中，评估场间的改变程度，如果它超过预定门限就检测到运动。

运动检测以及产生恢复值的帧信息和场信息的使用（通常取代丢失或损坏的值），都可以提供给要求产生恢复值的任一部分处理。例如，运动检测和产生恢复值的帧信息及场信息的选择性使用可以提供给 DR/MIN 恢复、像素恢复以及 Q 比特及运动标志恢复处理。因此，基于所检测的运动水平，恢复处理将基于场或基于帧利用现有信息。此外，这个处理可以与加权值的使用结合，加权值基于特定方向（例如，水平或垂直）上的相关水平来选择。

在 Q 比特和运动标志恢复处理的另一个实施例中，基于块内和块间测量来评估候选解码。在如下讨论中，术语“块”指帧或场的一部分。块内测量可评估候选解码的图象部分，例如图象部分的平滑度。块间测量可测出候选解码与相邻图象部分一致的程度。图 26a 和 26b 说明组合的块间及块内评估。具体而言，图 26a 表示一个可接受的候选解码，因为块间和块内测量均良好，而图 26b 中块间测量较差，即使块内测量相当好。

块内测量的例子包括上述的平滑度测量。块间测量的例子包括前面所述的误差平方测量。另一种块间测量是候选 ADRC 块上一致的边界像素与边界像素总数的比。

针对图 26c、26d 和 26e 解释 ADRC 编码的 8×8 块的块间和块内评估的一个例子。图 26d 说明由产生候选解码值 x 的 q 值组成的编码值 2650、以及由 y 值组成的相邻的解码数据 2655 的数据的图象部分(块)。正如图 26c 的流程图所提出的，在步骤 2605 计算块内测量以便产生一个测度，例如块拉普拉斯 Lx。在步骤 2610，计算块间测量 Sx 以便产生相邻块之间一致性的一个测度。在步骤 2615，产生组合测度 Mx。组合测度提供用于选择候选解码的信息。

在本实施例中，作为处于候选解码的每个边界像素有效范围内的相邻数据数来计算 Sx（见图 26e）。作为一个实施例，图 26e 是说明有效范围的图，表示每个所观察的量化值 q_i 的有效范围。因此 $L_Q < DR < U_Q$，这里 L_Q、U_Q 分别代表对应于量化比特数=Q 的 DR 的下界和上界。优选 Sx 根据下式归一化：Sx=Sx/边界像素数。

在本实施例中，合并的测度 Mx 根据下式计算：Mx=Sx+(1-Lx)。或者，合并的测度可以被加权，从此使用下式：Mx=w·Sx+(1-w)·(1-Lx)，这里 w 是加权值，通常是由经验确定的加权值。

用于确定已经丢失/损坏的 DR 和 MIN 值的其它实施例也可考虑。例如，前面描述的公式可以被修改以便以更高精度恢复 DR 和 MIN 值。在另一个实施例中，使用中值技术。在中值技术的一个实施例中，MIN 值作为所有 MIN_i 值的中值恢复，如下计算：

$$MIN_i = y_i - q_i \cdot s$$

这里 q_i 代表编码的像素值，y_i 代表相邻 q_i 的解码像素。对于边缘匹配 ADRC，$s=DR/(2^Q-1)$。对于非边缘匹配 ADRC，$s=DR/2^Q$，这里 Q 代表每个像素的量化比特数（Q 比特值）。

所用的值可以是时间邻近或空间邻近的。y_i 值可以是相邻帧/场或相同场中的相邻像素的解码值。y_i 值可以是相邻帧/场或相同场中与 q_i 相同位置的像素解码值。

此外，任何 DR 及/或 MIN 恢复技术都可以与限幅处理结合以便改善恢复精度和防止数据在恢复过程中溢出。限幅处理将被恢复数据限制到预定值范围；因此范围外的那些值被限制到最接近范围边缘。在一个实施例中，限幅处理将值限制到范围 $[L_Q, U_Q]$ 中，这里 L_Q, U_Q 分别代表量化比特数=Q 个量化比特所代表的像素值范围的上界和下界，并进

一步将值限制到：MIN+DR＜ Num，这里 Num 代表最大像素值；在本实施例中 Num 为 225。在本实施例中，在可用的情况下， $U_Q+1= L_{Q+1}$。

可以将准则合并成单个式子，得到 DR 的无限制的恢复值（val'），最后限幅的恢复值（val）从下式得到：

$$val=max(min(val,\ min(U_Q, 255-MIN)),\ L_Q)$$

这里 min 和 max 分别代表最小和最大函数。

在另一个实施例中，用于产生恢复的 DR 及/或 MIN 的边界像素 y_i 可以被过滤以便只使用其中呈现最佳相关的那些，因此更好地恢复 DR 和 MIN。那些不满足准则的边界像素不使用。在一个实施例中，如果存在一个 DR 值使 $L_Q < DR < U_Q$ 而且原始像素 y_i 可能已经编码为 q_i，边界像素 y_i 被认为对 DR 计算有效。因此，如果满足下式则像素有效：

$$\frac{(y_i - MIN)m}{\max(q_i - 0.5, 0)} \geq L_Q$$

$$\frac{(y_i - MIN)m}{\min(q_i - 0.5, m)} < U_Q$$

这里 m 代表最大量化级别=2^Q-1。DR 恢复值（val'）可以根据下式计算：

$$val' = \frac{m \cdot \sum_i (y_i - MIN)q_i}{\sum_i q_i^2}$$

然后该值可以被限幅到有效范围。因此这个过程迫使 DR 恢复值进入门限表定义的有效区域内部，降低了其真正 DR 处于门限表边界附近的点的精度。

已经注意到由于量化噪声，静止 ADRC 块的 DR 逐帧轻微变化。如果这个变化跨过 ADRC 编码边界，而且如果 DR 在几个连续帧上恢复，那么具有有效像素选择的 DR 恢复值在每个交叉点易于超过目标，导致显示中可觉察的闪烁现象。在尝试减少这种现象出现的努力中，一个实施例修改了有效像素选择过程，以便放宽上界和下界，从而允许越到相邻有效区域中的边界像素。通过包括仅在边界外的点，恢复值将

更容易采取靠近上界或下界的值。放宽的边界 L'_Q 以及 U'_Q 通过放宽常数 r 计算。在一个实施例中，r 被设置为 0.5 值。其它值也可以使用：

$$L'_Q = rL_{Q-1} + (1-r)L_Q$$

$$U'_Q = (1-r)U_Q + rU_Q + 1$$

上面的讨论提出了当这些值已经损坏或丢失时恢复 DR 和 MIN 的多种方式。通过检查数据之间的时间和/或空间相关、并因此对相应的计算恢复值加权，可以实现进一步的提高。更具体地，如果在特定方向或时间上存在很大相关，例如水平相关，那么很可能图象特性在具有很强相关的方向上平滑地持续下去，因此使用高相关数据的恢复值通常产生较好的估计。为了利用这一点，边界数据被分解为相应的方向（例如，垂直、水平、场-到-场）并根据相关测量加权，以产生最终恢复值。

该过程的一个实施例参考图 27a 来进行描述。在步骤 2710，待恢复的 DR 或 MIN 值的恢复值在一个方向上产生，在步骤 2715，在另一个方向上产生恢复值。例如，如果处理是空间自适应的，那么沿着水平边界的边界像素用于产生第一恢复值，“hest（命令）”，沿着垂直边界的边界像素用于产生第二恢复值，“vest（上衣）”。或者，如果处理是时间自适应的，那么相邻场之间的边界像素用于产生第一恢复值，相邻帧之间的边界像素用于产生第二恢复值。

在步骤 2720，根据表示每个方向中相关程度的相关计算来对恢复值加权。在步骤 2725，加权的第一和第二恢复值被合并，以产生合并恢复值。应该注意该处理不限于只在两个方向上产生加权恢复值；很显然被加权并合并的恢复值数可以根据应用而变。各种已知技术可以用于产生表示特定方向上相关程度的相关值。此外，考虑到相关程度，有各种准则可以用于选择加权因子。通常，如果一个相关比另一个强，合并恢复值应该主要基于相应的恢复值。在一个实施例中，合并恢复值如下计算：

$$val' = \begin{cases} \alpha hest + (1-\alpha)vest: & hc \geq vc \\ (1-\alpha)hest + \alpha vest: & hc < vc \end{cases}$$

这里 hc 代表水平相关，vc 代表垂直相关，hest 代表仅基于左及右边界信息的 DR 恢复值，vest 代表仅基于上及下边界信息的 DR 恢复值，α代表加权值。加权值可以用各种方式确定。图 27b 说明了作为水平相关和垂直相关之差的函数来确定加权值的一个实施例。更具体来说，α被选为：

$$\alpha(|hc-vc|)=\begin{cases}0.5+0.25\cdot e^{-8(0.35-|hc-vc|)}: & |hc-vc|<0.35\\ 1-0.25\cdot e^{-8(|hc-vc|-0.35)}: & |hc-vc|\geq 0.35\end{cases}$$

正如上面注意到的，自适应相关处理用于 DR 和 MIN 恢复。但是优选地 MIN 恢复被限幅以便保证 MIN+DR＜255，因此可以使用函数 val=max(min(val', 255-MIN), 0)。此外，正如上面注意到的，可以评估时间相关并用于对恢复值加权。此外，可以进行时间和空间相关的合并。例如，可作为时间恢复值在场间产生一个恢复值。另一个恢复值作为空间恢复值在一个场内产生。最后的恢复值作为时间和空间相关的合并的合并值来计算。相关合并可以被运动量取代。其它变量也可以考虑。该方法也可以用于声音数据。

在另一个实施例中，使用对最小二乘技术的低复杂性修改。使用这个实施例，减少了由于恢复 DR 值而形成的闪烁。为了如下讨论，QV 代表一串来自其 DR 被恢复的、具有一组点 q_i 的图象段或 ADRC 块的编码值，Y 是一串从 QV 中各点的垂直或水平相邻点取得的解码值，这里 y_i 代表 q_i 的垂直或水平相邻点。由于每个点 q_i 可能有多达 4 个解码的相邻点，一个像素或点可能引起多达四个（q_i，y_i）配对。因此 DR 的非限制最小二乘估计（DR_{uls}）是：

$$(DR)_{uls}=\frac{2^{Q}\cdot\sum_{i}(y_i-MIN)\cdot(0.5+q_i)}{\sum_{i}(0.5+q_i)^2}$$

这里 Q 是量化比特数，MIN 是作为块属性传输的最小值。上式假设

非边缘匹配 ADRC；对于边缘匹配 ADRC，2^Q被 2^Q-1 代替，$(0.5+q_i)$被 q_i 代替。

优选地对非限制最小二乘估计进行限幅，以便保证与门限表和编码过程中实施的式子 MIN+DR≤255 的一致性（通常，对于非边缘匹配 ADRC，允许的 DR 值在 1-256 范围内）。因此最小二乘估计被限幅（DR_{lsc}）为：

$(DR)_{lsc}=max(min(UB, DR_{uls}), LB)$

这里 UB 代表上界，LB 代表下界，min 和 max 分别代表最小和最大函数。

在另一个实施例中，估计可以通过选择最适于 DR 估计的像素来提高，以便计算 DR 估计。例如，相比出现高活动性的那些区域，图象的平坦区提供较适于 DR 估计的像素。具体而言，边缘中的尖锐边缘可能降低估计的精度。下面实施例为选择用于计算 DR 估计的像素提供了计算量较轻的方法。

在一个实施例中，计算最小二乘估计（DR_{lso}），例如 Dr_{uls}或 DR_{lsc}。使用这个估计，编码值 QV 的列表被转换成候选解码值 X，其中 x_i 是从 q_i 得到的 X 成员。x_i 值是使用 DR 的第一估计构成的已恢复解码值。x_i 值根据下式定义：

边缘匹配 ADRC：$x_i = MIN + \left(0.5 + \frac{q_i \cdot DR_{lso}}{2^Q - 1}\right)$

非边缘匹配 ADRC：$x_i = MIN + \left(\frac{(q_i + 0.5) \cdot DR_{lso}}{2^Q}\right)$

假设 DR_{lso}是真正 DR 的合理估计，那么 x_i 离 y_i 比较近的任何地方可以被判决为低活动性区并因此是一个希望的匹配。新 X 和 Y 列表则可以通过只考虑 x_i 和 y_i 相接近地方的匹配而构成，并重新计算最小二乘估计以产生更新的估计。

用于确定“接近”的准则可以用各种方式来确定。在一个实施例中，使用误差函数的 ADRC 编码。这种方法的需求是因为它计算起来不昂贵。对于该处理，定义了由点 $e_i=|y_i-x_i|$组成的列表 E。定义 emin

和 emax 分别作为列表中的最小和最大值，则 eDR=emax-emin。编码误差值则可定义为：

$g_i=(e_i-emin)\,n1/eDR$

这里 n1 代表以类似于上述 ADRC 处理的方式重新量化 e_i 的量化级别平数。

因此，新列表 X 和 Y 通过只选择那些 g_i 小于一些门限的匹配而产生。如果新列表足够长，这些列表可用于产生精确的最小二乘估计 DR_{rls}。g_i 的门限和在精确确定最小二乘估计之前所需的匹配数优选地由经验确定。例如，在一个实施例中，对于包括 8x8x2 个水平二次抽样块的处理以及 n1 为 10，只使用与 $g_i=0$ 对应的匹配，而且仅当新列表包含至少 30 个匹配时才对估计进行精确地确定。

在另一个实施例中，DR 估计可以通过限幅潜在的 DR 值并重新计算 DR 估计来改善。具体而言，在一个实施例中列表 D 由成员 d_i 组成，该成员 d_i 包含使 x_i 等于 y_i 的 DR 值。更精确地：

$d_i=2^Q(y_i-MIN)/(0.5+q_i)$

改善可以通过对每个 d_i 限幅而看到。即，

$d_i'=max(min(UB,d_i),LB)$

其中将 DR_{cls} 计算为 d_i' 的平均值。限幅方法（DR_{cls}）可以与其它 DR 估计（例如加权平均中的 DR_{lse}）合并，以便产生最后 DR 值。例如，根据下式确定加权平均 DR_{est}：

$DR_{est}=w1\,(DR_{cls})+w2\,(DR_{lse})$

权重 w1 和 w2 优选地通过检查所得的估计以及从特定加权中产生的图象来经验性地确定。在一个实施例中，w1=0.22513，w2=0.80739。

本发明已经结合优选实施例进行了描述。对于本领域技术人员来说，根据前面的描述很多替换、修改、变化以及使用都是显而易见的。

图 1

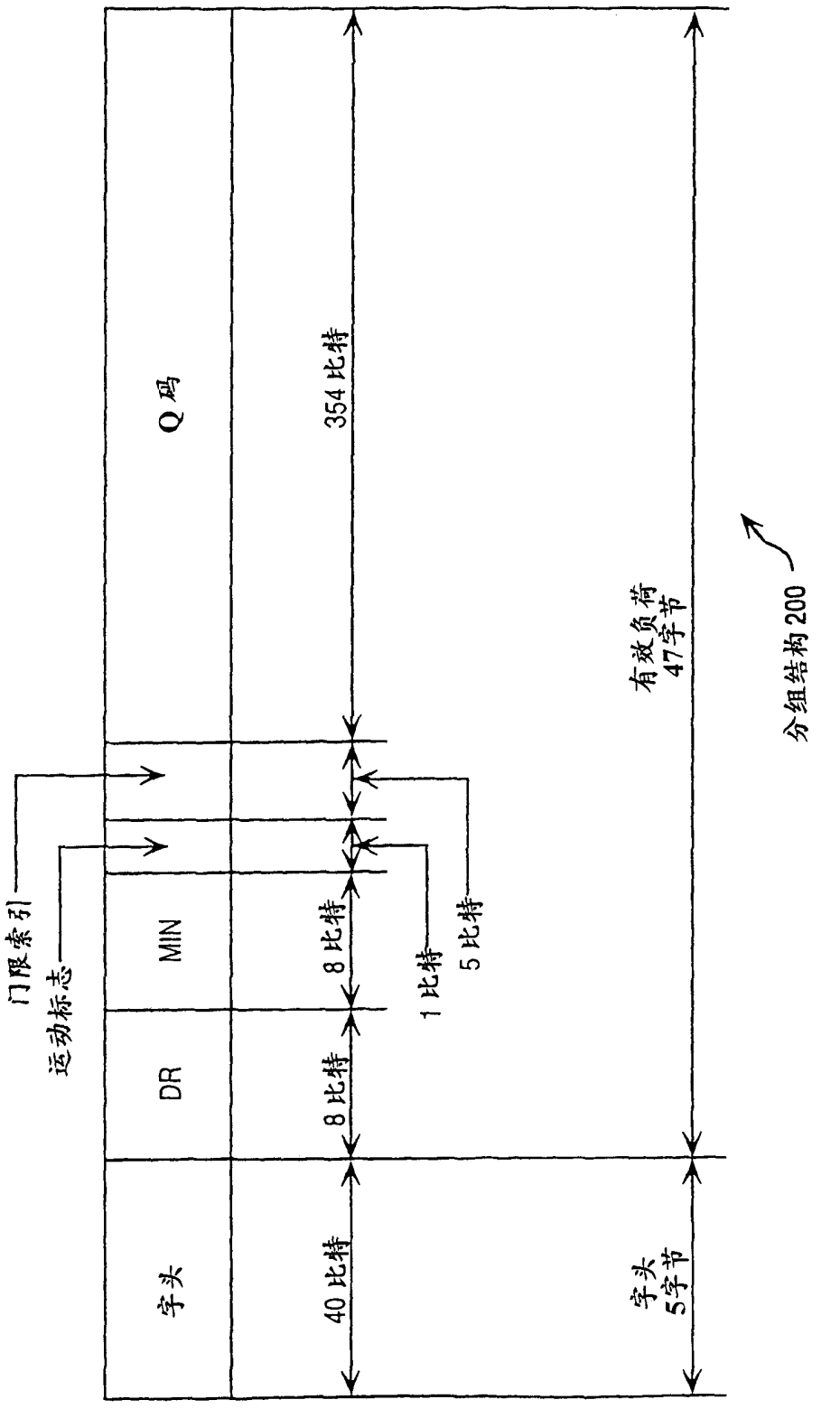
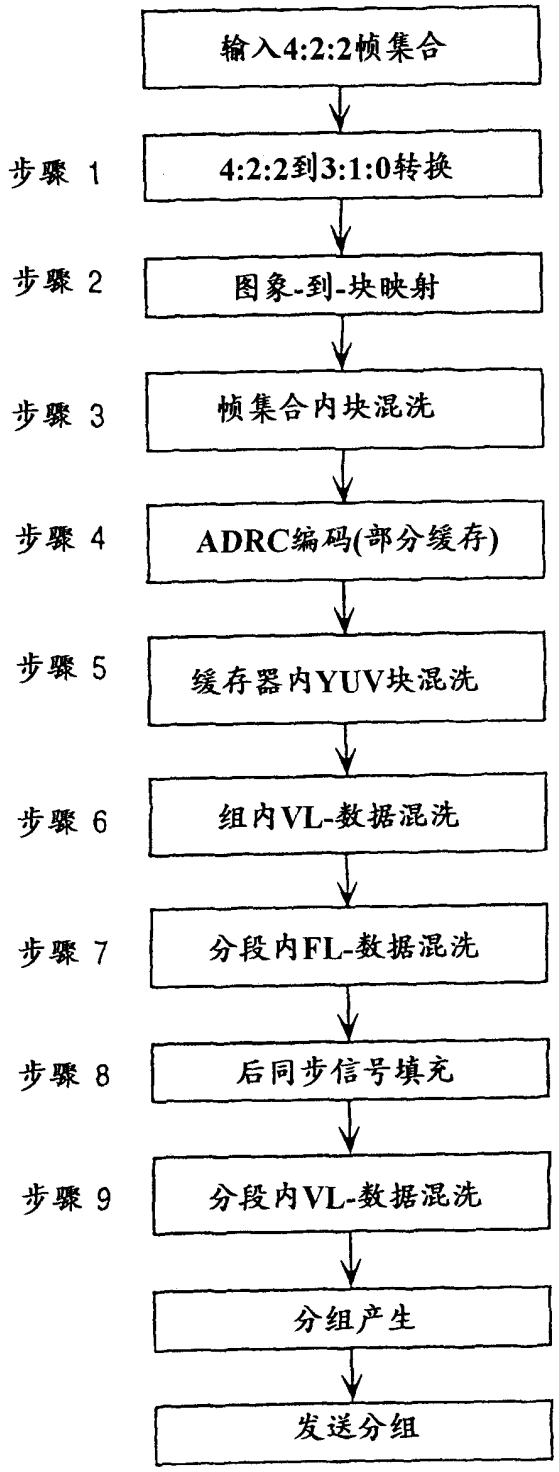

字头
DR
MIN
Q 码
门限索引
运动标志
40 比特
8 比特
8 比特
1 比特
5 比特
354 比特
字头
5字节
有效负荷
47字节
分组结构 200

图 2



编码器内的
信号流图

图 3

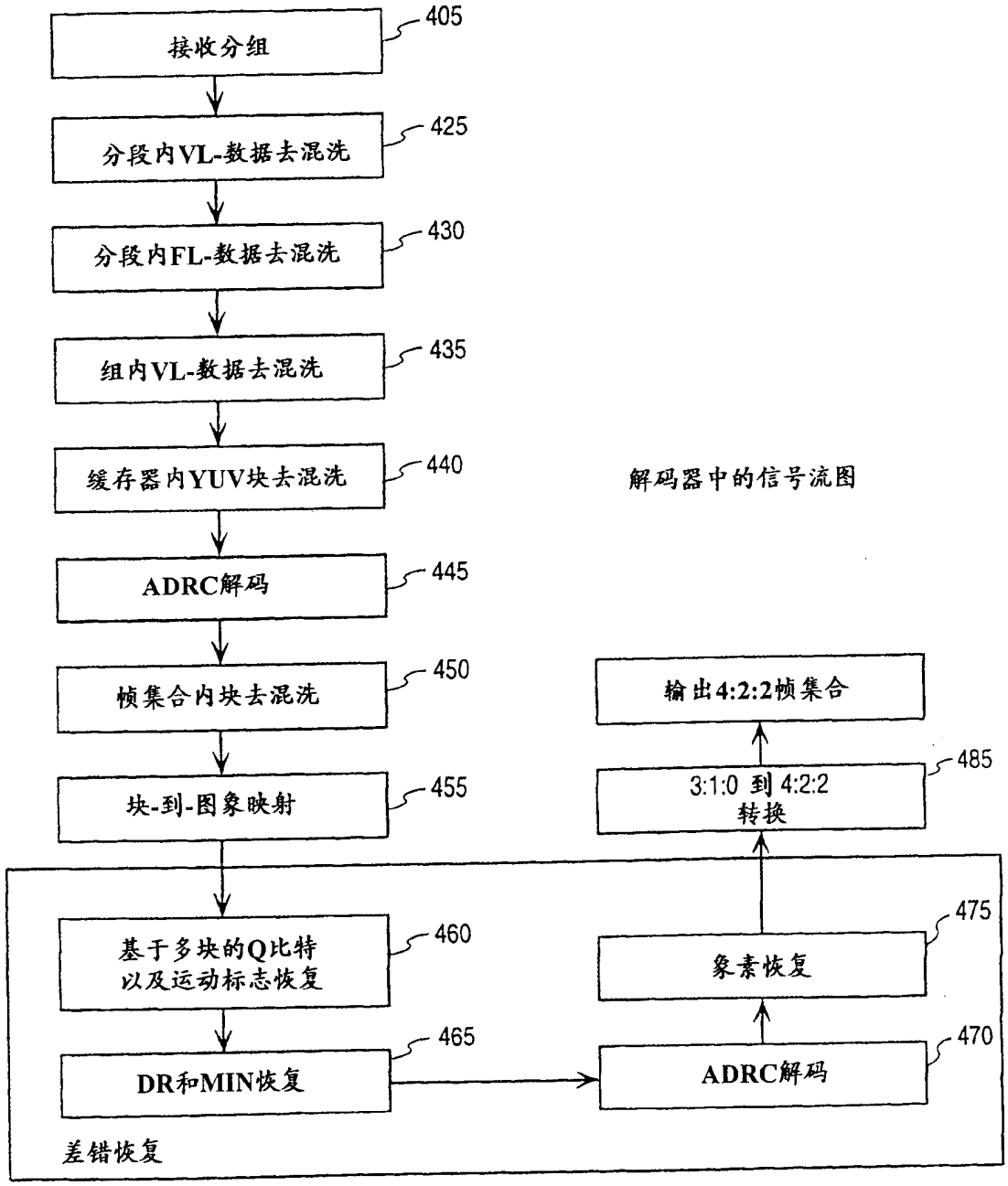

接收分组
405
分段内VL-数据去混洗
425
分段内FL-数据去混洗
430
组内VL-数据去混洗
435
缓存器内YUV块去混洗
440
ADRC解码
445
帧集合内块去混洗
450
块-到-图象映射
455
基于多块的Q比特
以及运动标志恢复
460
DR和MIN恢复
465
差错恢复
解码器中的信号流图
输出4:2:2帧集合
3:1:0 到 4:2:2
转换
485
象素恢复
475
ADRC解码
470

图 4

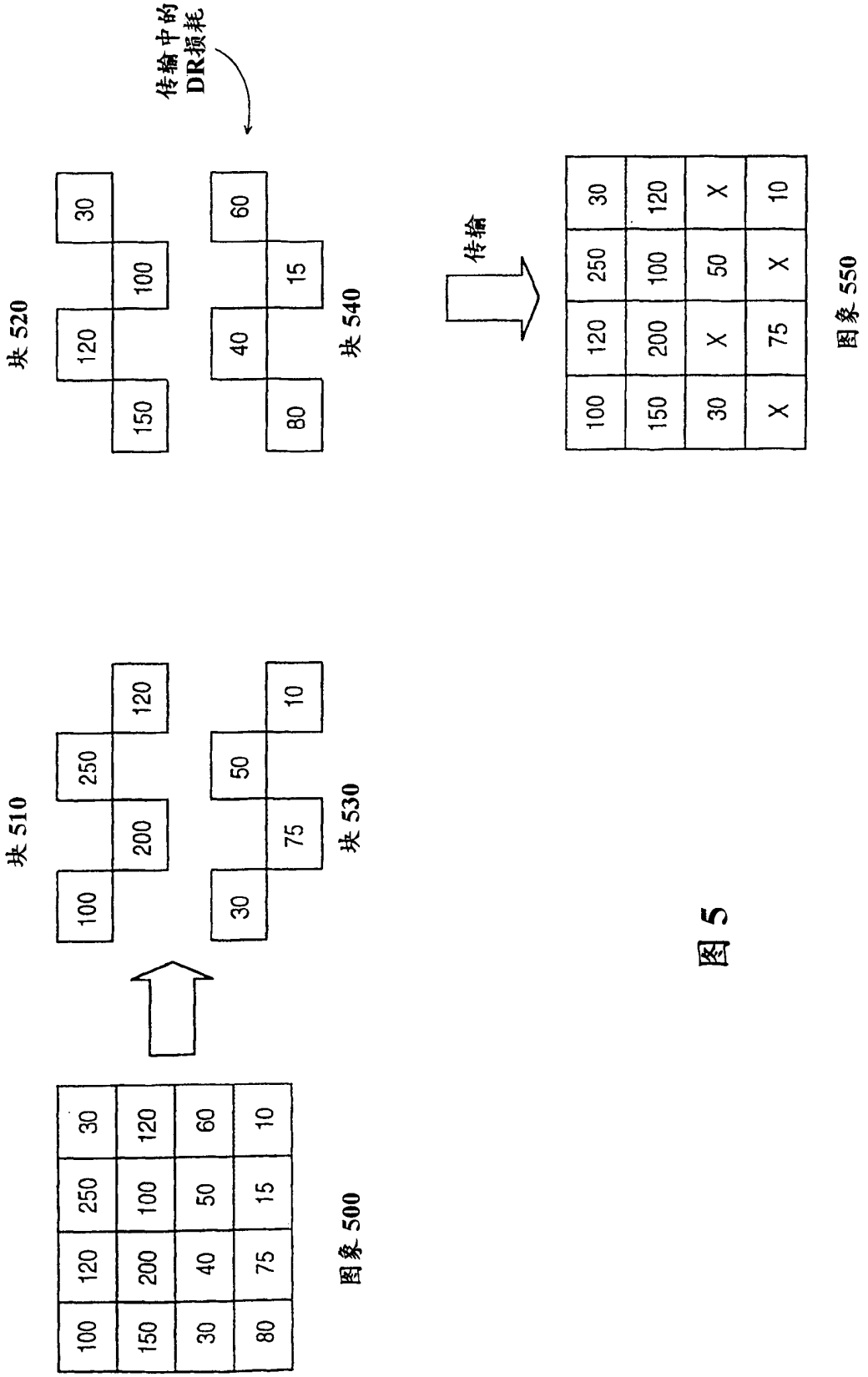
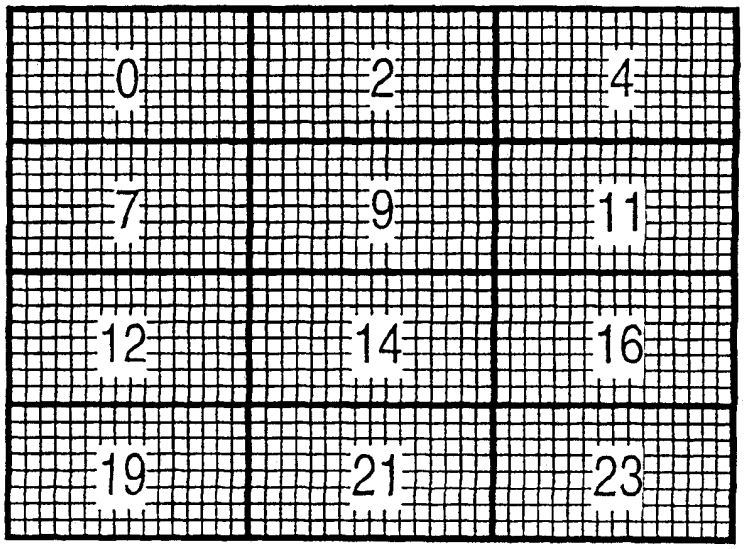
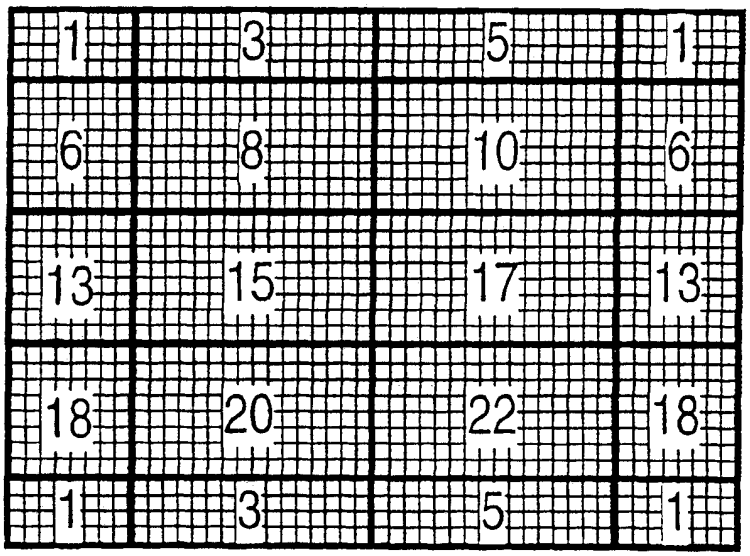

图象 500
100 120 250 30
150 200 100 120
30 40 50 60
80 75 15 10
块 510
100 200 250 120
块 520
150 120 100 30
块 530
30 75 50 10
块 540
80 40 15 60
传输中的
DR损耗
传输
100 120 250 30
150 200 100 120
30 X 50 X
X 75 X 10
图象 550

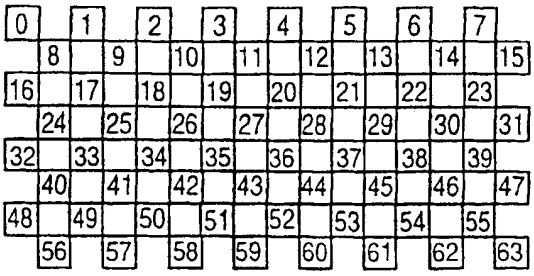
图 5



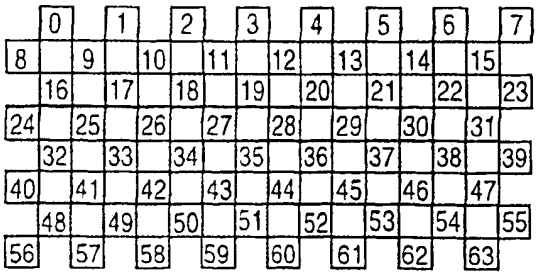
图象 560



图象 570



瓦块 565



瓦块 575

图 5a

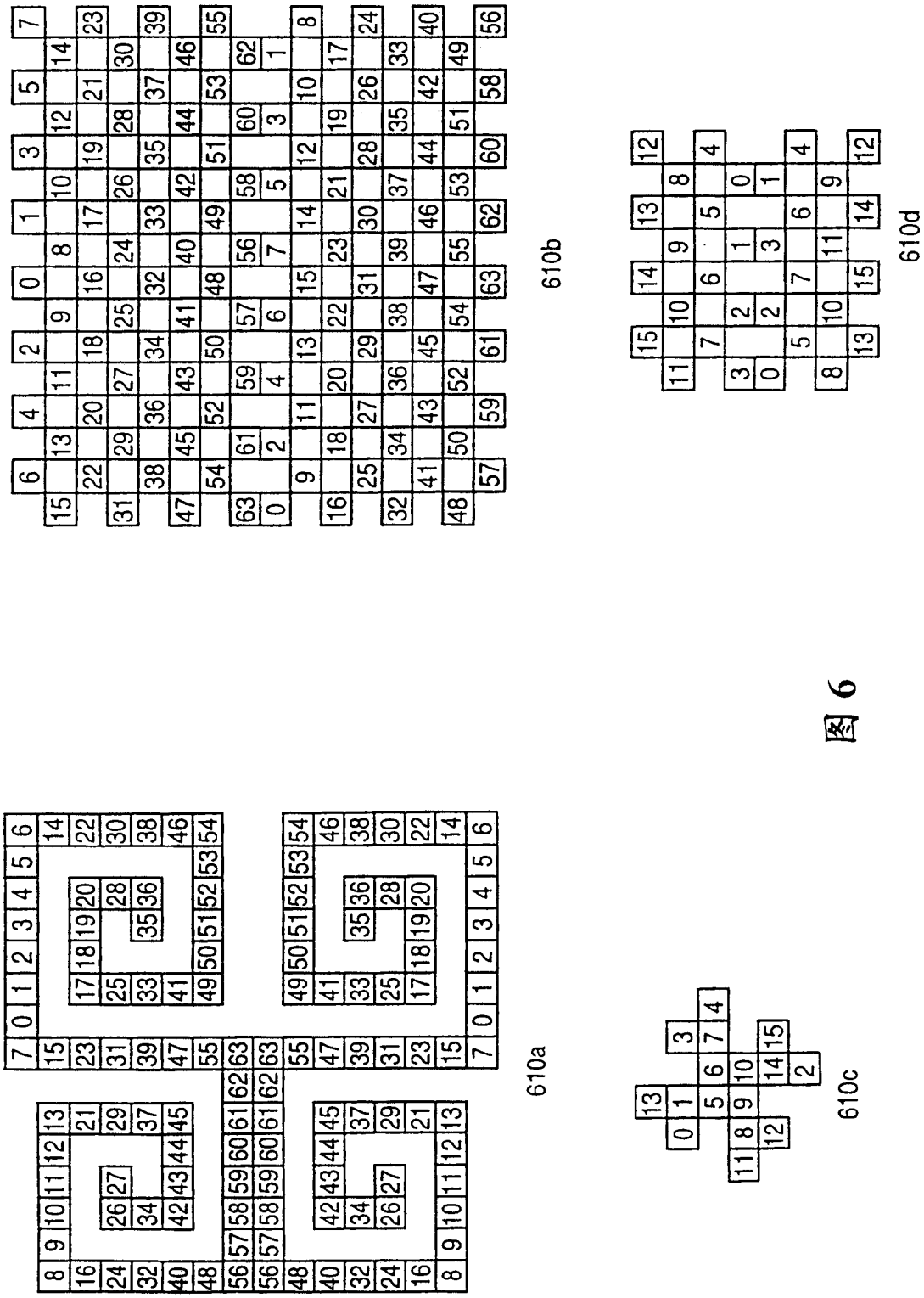


图 6

帧集合内块混洗
分段限定:Y块

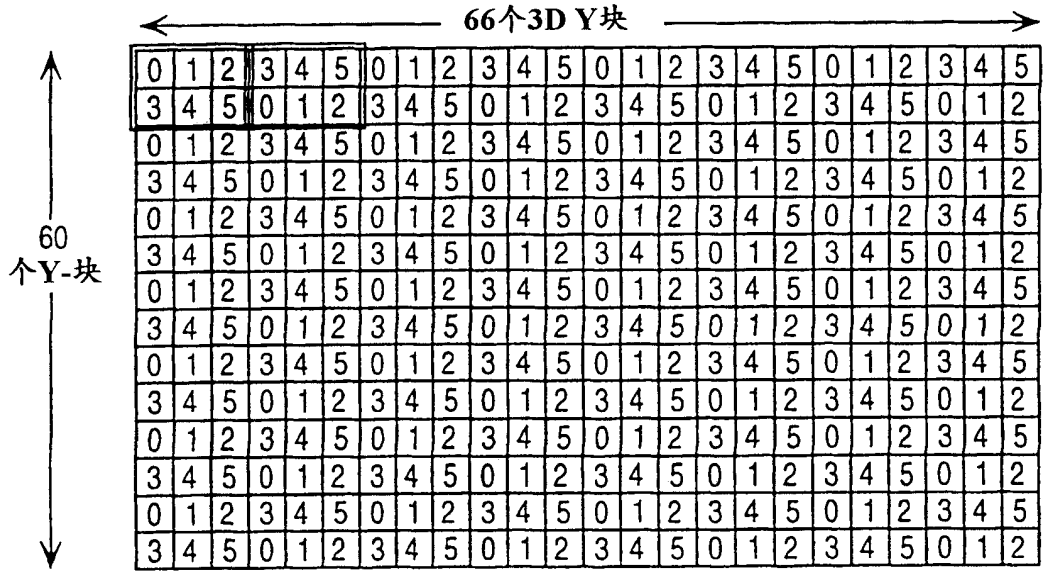


图 7a

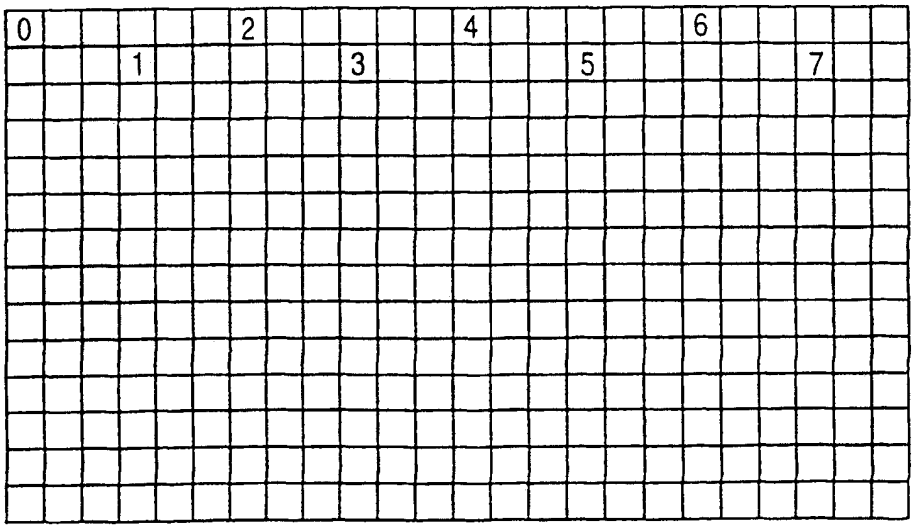


图 7b

帧集合内块混洗

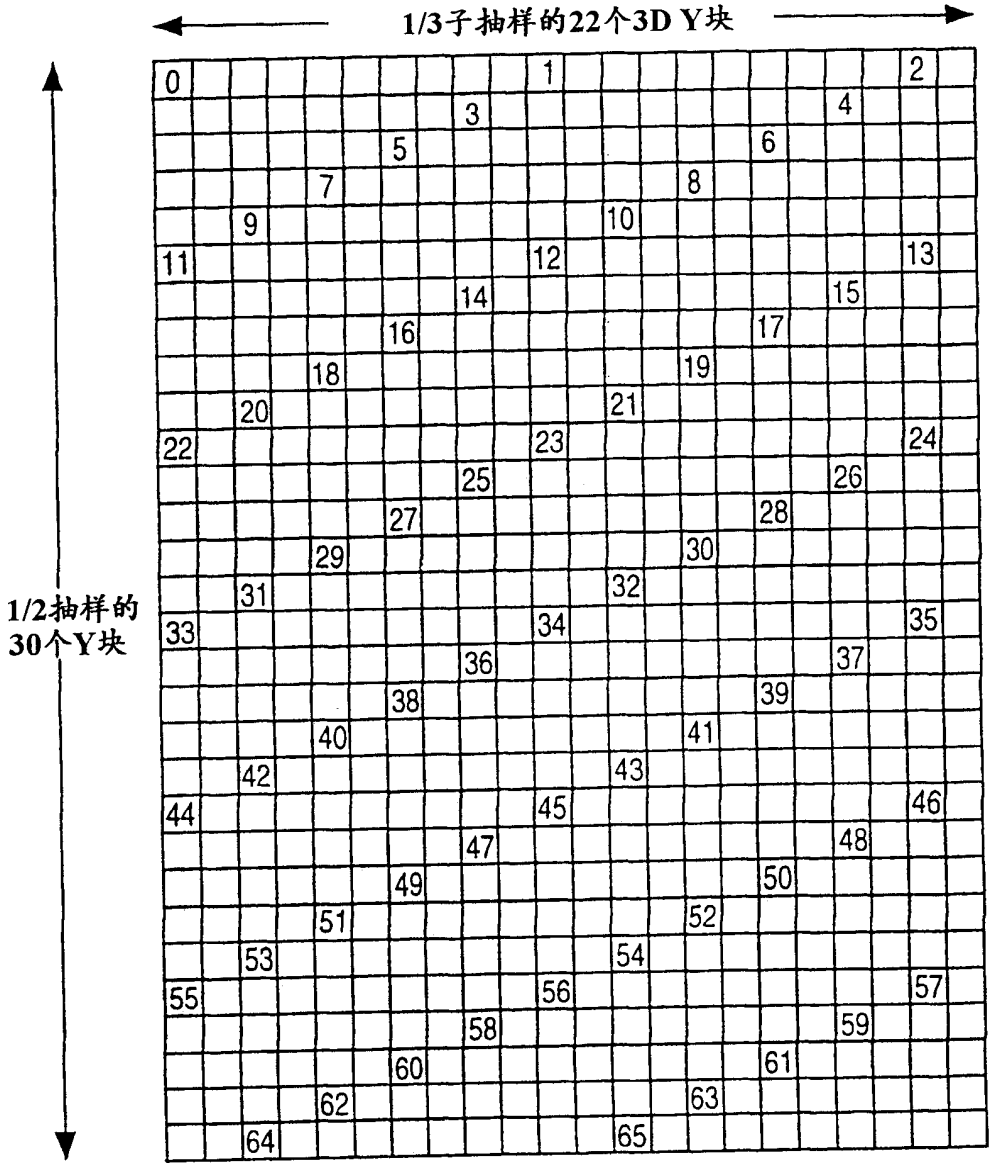


图 7c

帧集合内块混洗

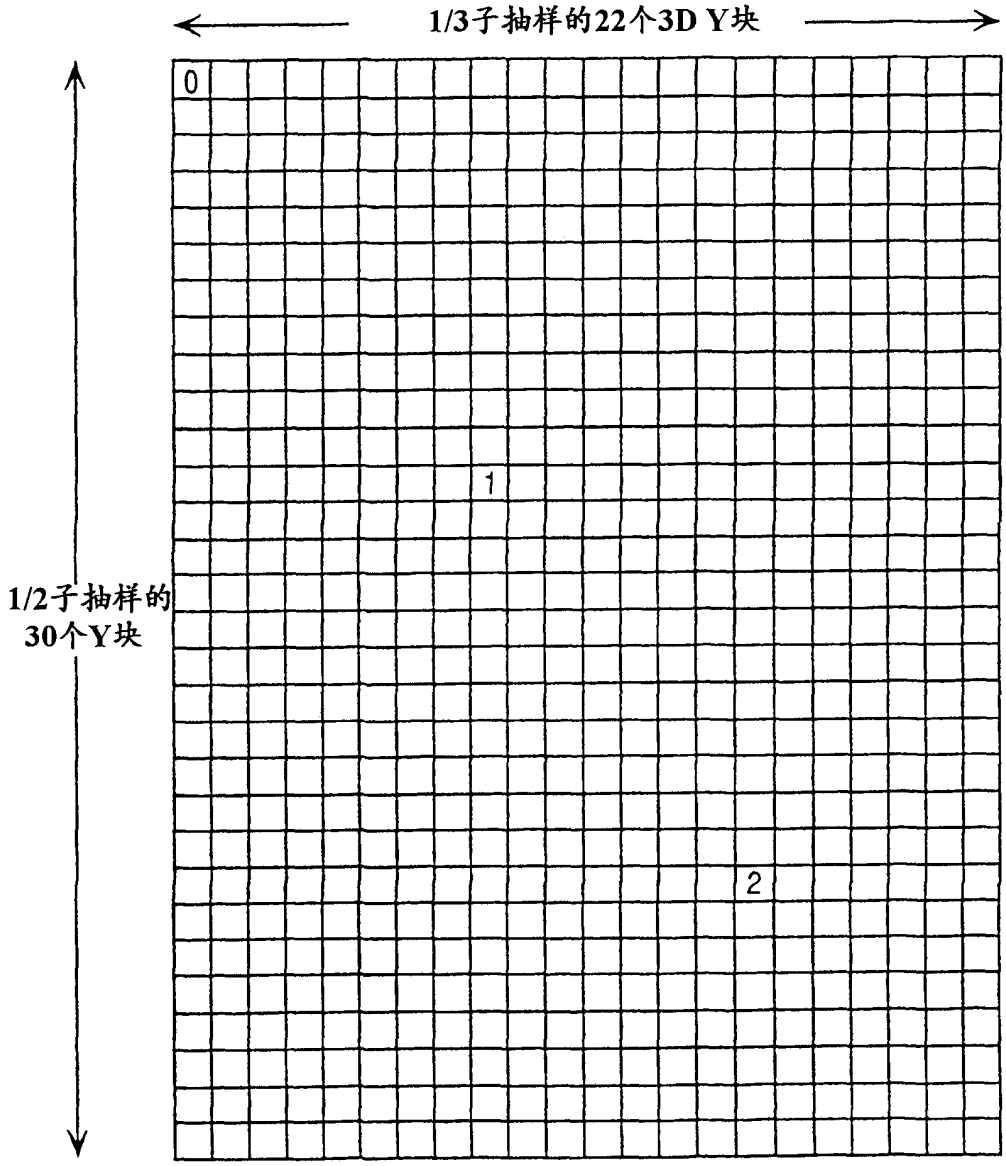


图 7d

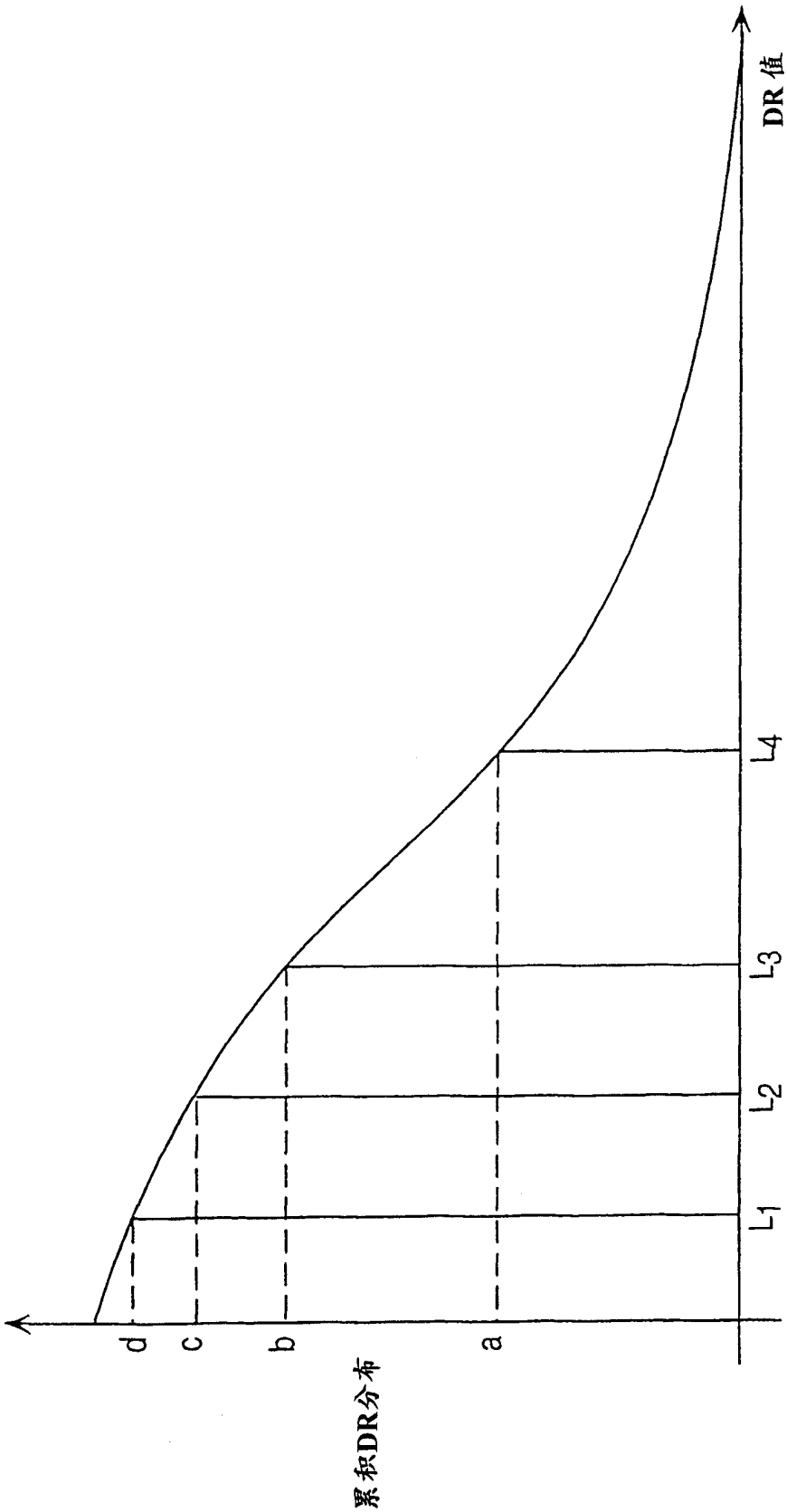


图 8

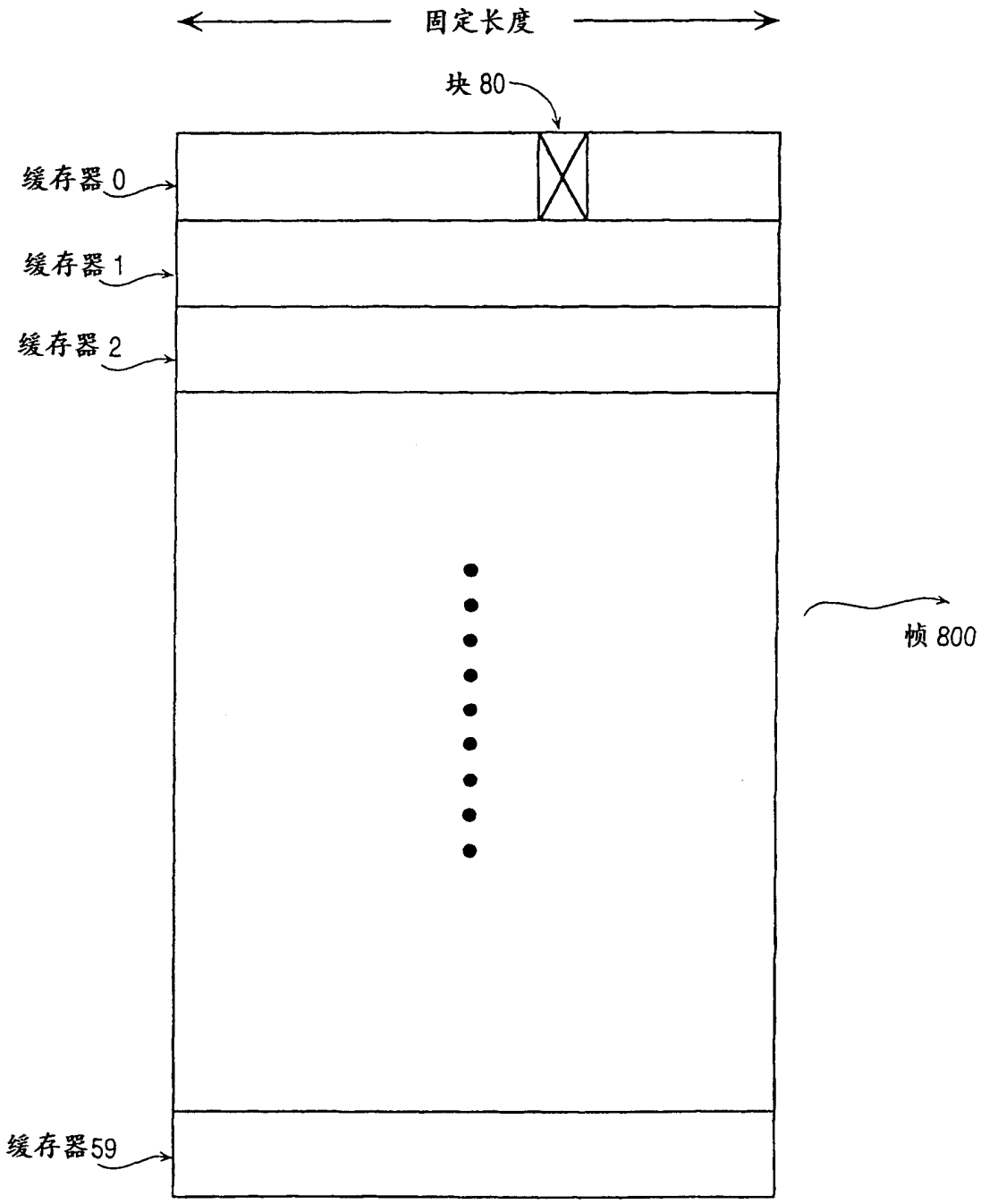


图 8a

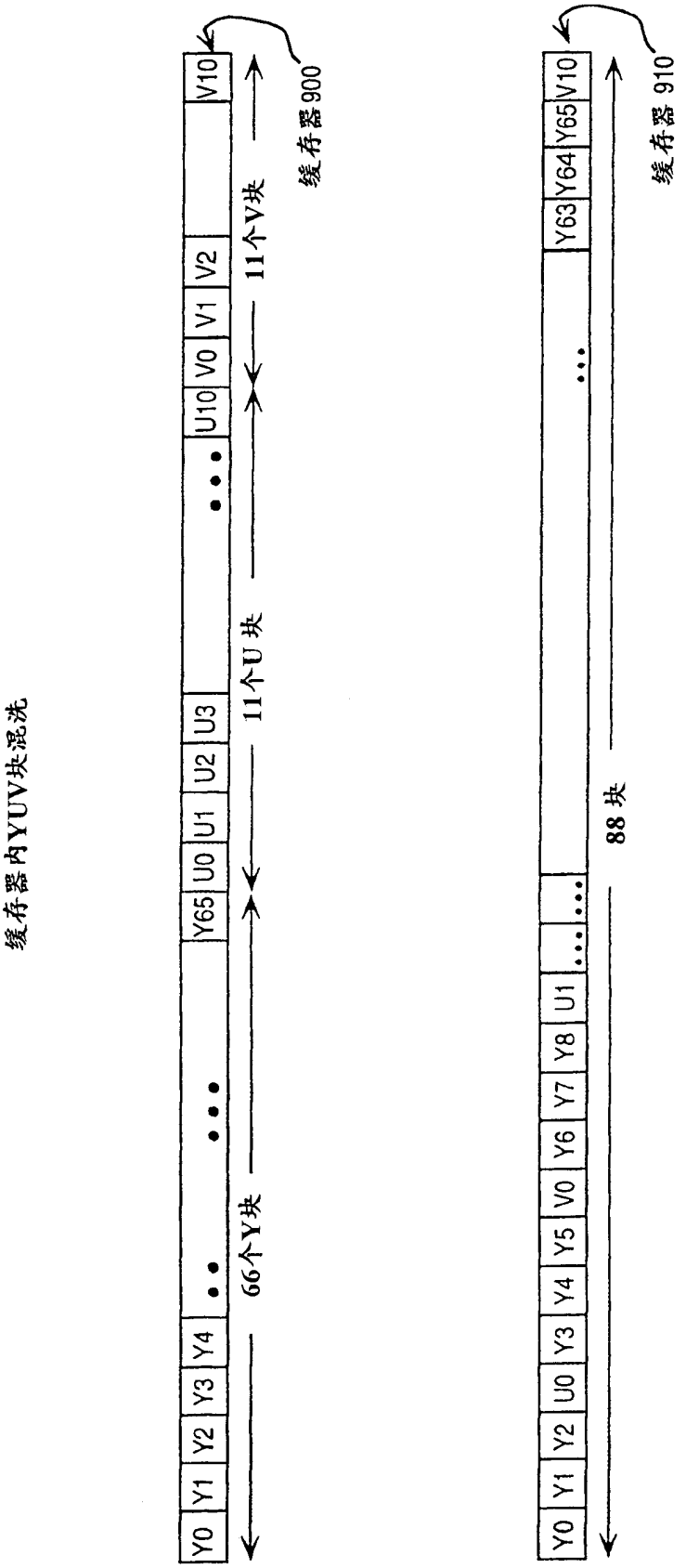

缓存器内YUV块混洗
Y0 Y1 Y2 Y3 Y4 ... Y65 U0 U1 U2 U3 ... U10 V0 V1 V2 ... V10
66个Y块
11个U块
11个V块
缓存器900
Y0 Y1 Y2 U0 Y3 Y4 Y5 V0 Y6 Y7 Y8 U1 ... ... Y63 Y64 Y65 V10
88块
缓存器 910

图 9

组内VL-数据混洗

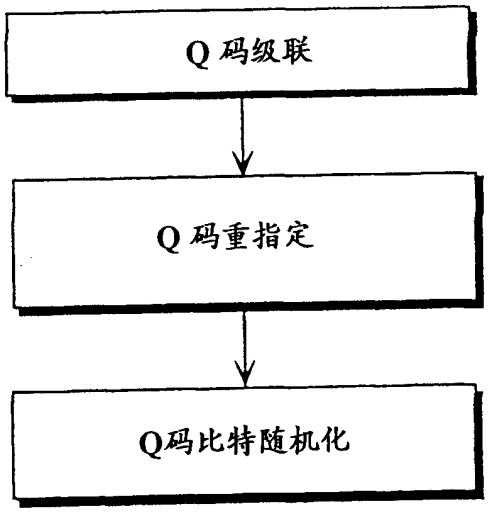


图 10

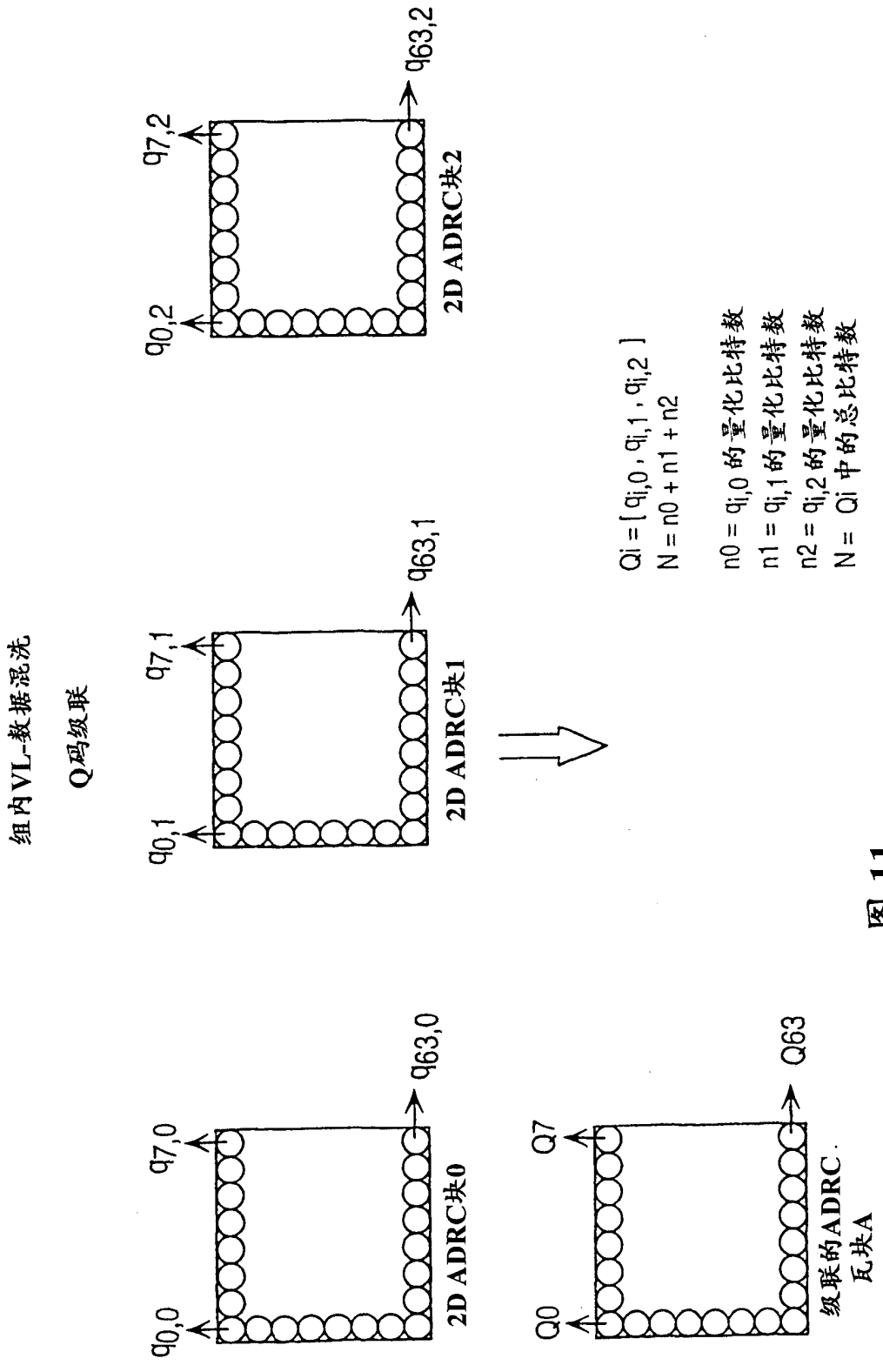

组内VL-数据混洗
Q码级联
q0,0
q7,0
q63,0
2D ADRC块0
q0,1
q7,1
q63,1
2D ADRC块1
q0,2
q7,2
q63,2
2D ADRC块2
Q0
Q7
Q63
级联的ADRC
瓦块A
Qi = [qi,0 , qi,1 , qi,2]
N = n0 + n1 + n2
n0 = qi,0 的量化比特数
n1 = qi,1 的量化比特数
n2 = qi,2 的量化比特数
N = Qi 中的总比特数

图 11

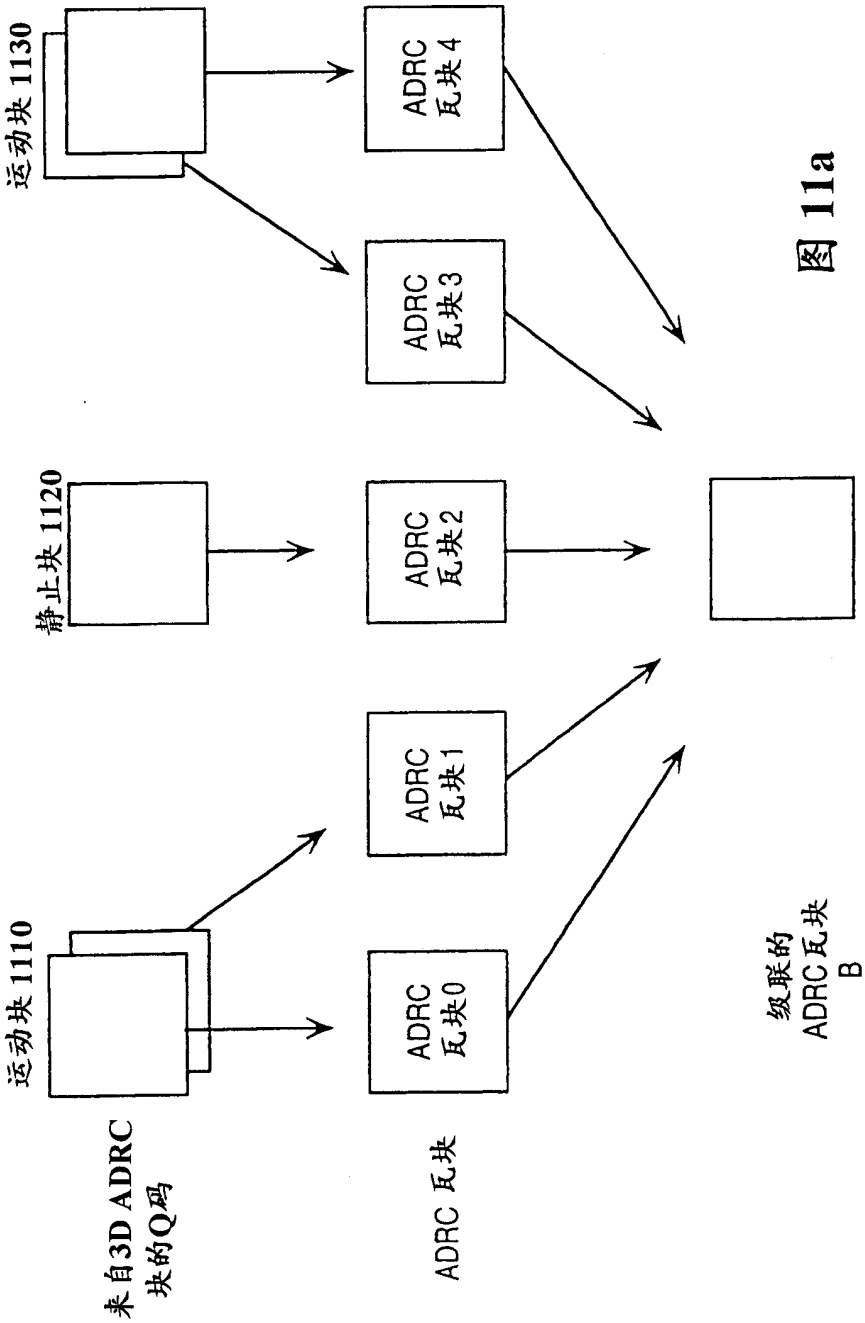

运动块 1130
静止块 1120
运动块 1110
来自3D ADRC
块的Q码
ADRC
瓦块0
ADRC
瓦块1
ADRC
瓦块2
ADRC
瓦块3
ADRC
瓦块4
ADRC 瓦块
级联的
ADRC瓦块
B

图 11a

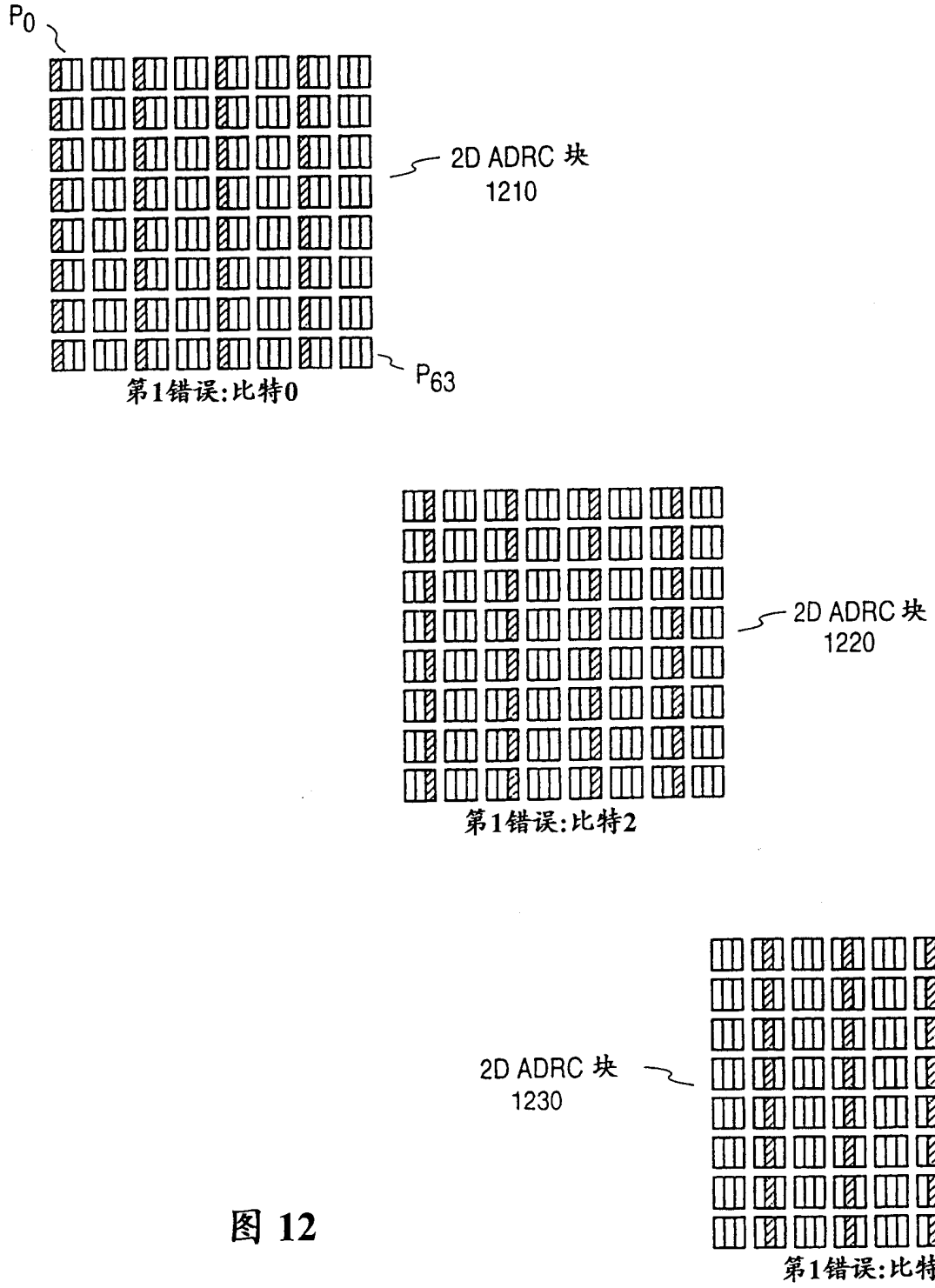

P0
2D ADRC 块
1210
P63
第1错误:比特0
2D ADRC 块
1220
第1错误:比特2
2D ADRC 块
1230
第1错误:比特4

图 12

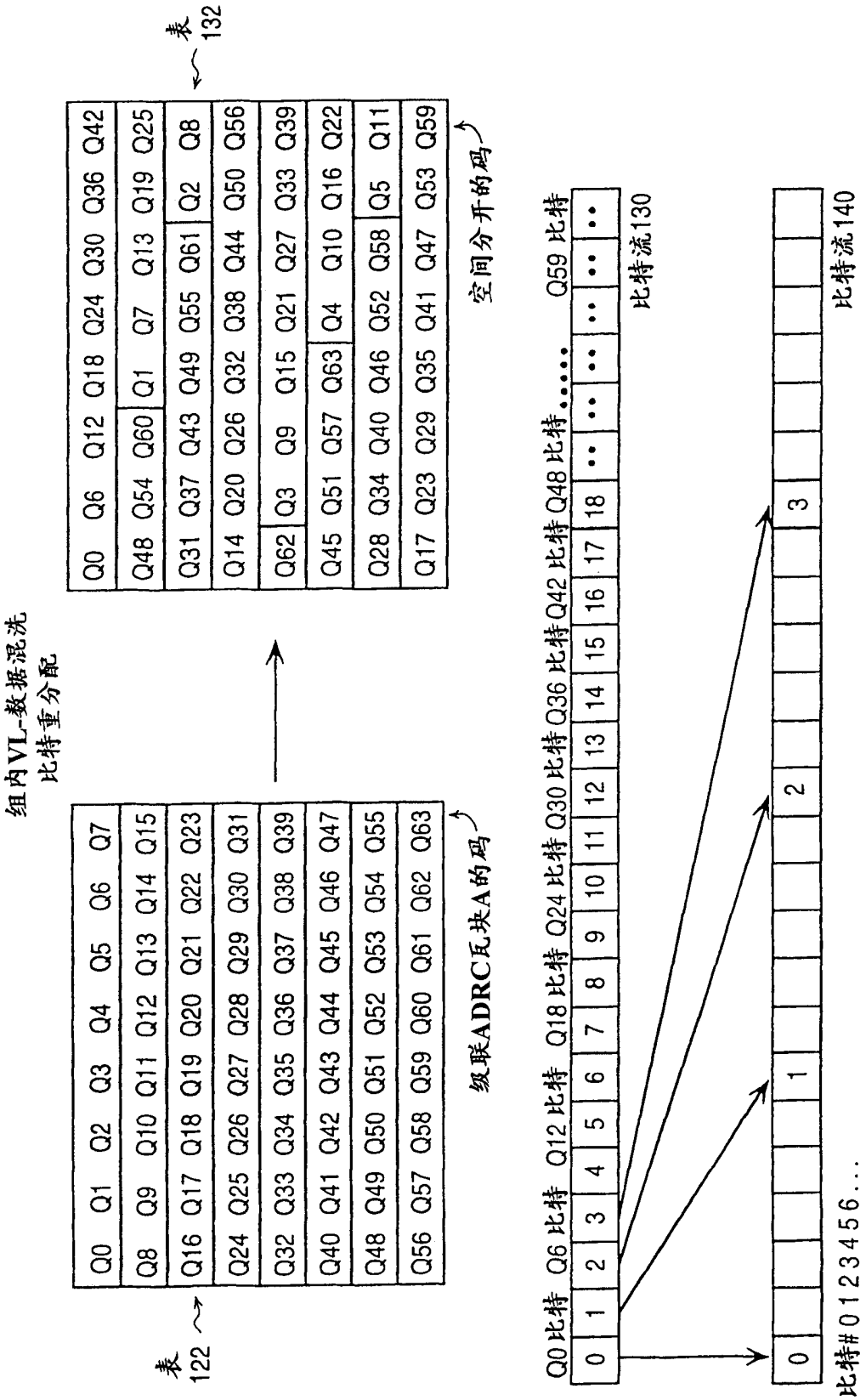


表 122
Q0 Q1 Q2 Q3 Q4 Q5 Q6 Q7
Q8 Q9 Q10 Q11 Q12 Q13 Q14 Q15
Q16 Q17 Q18 Q19 Q20 Q21 Q22 Q23
Q24 Q25 Q26 Q27 Q28 Q29 Q30 Q31
Q32 Q33 Q34 Q35 Q36 Q37 Q38 Q39
Q40 Q41 Q42 Q43 Q44 Q45 Q46 Q47
Q48 Q49 Q50 Q51 Q52 Q53 Q54 Q55
Q56 Q57 Q58 Q59 Q60 Q61 Q62 Q63
组内VL-数据混洗
比特重分配
级联ADRC瓦块A的码
表 132
Q0 Q6 Q12 Q18 Q24 Q30 Q36 Q42
Q48 Q54 Q60 Q1 Q7 Q13 Q19 Q25
Q31 Q37 Q43 Q49 Q55 Q61 Q2 Q8
Q14 Q20 Q26 Q32 Q38 Q44 Q50 Q56
Q62 Q3 Q9 Q15 Q21 Q27 Q33 Q39
Q45 Q51 Q57 Q63 Q4 Q10 Q16 Q22
Q28 Q34 Q40 Q46 Q52 Q58 Q5 Q11
Q17 Q23 Q29 Q35 Q41 Q47 Q53 Q59
空间分开的码
Q0比特 Q6比特 Q12比特 Q18比特 Q24比特 Q30比特 Q36比特 Q42比特 Q48比特..... Q59比特
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18
比特流130
0 1 2 3
比特流140
比特#0 1 2 3 4 5 6...

图 12a

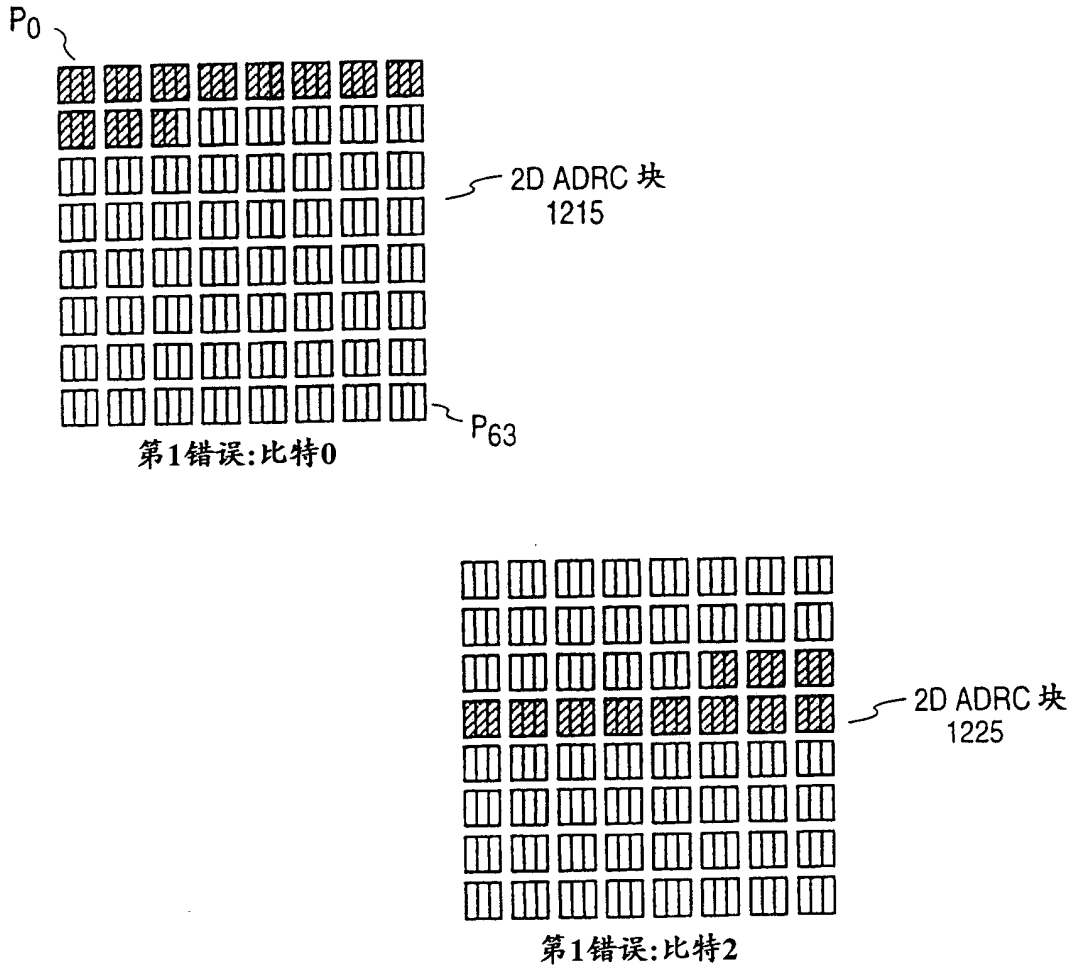
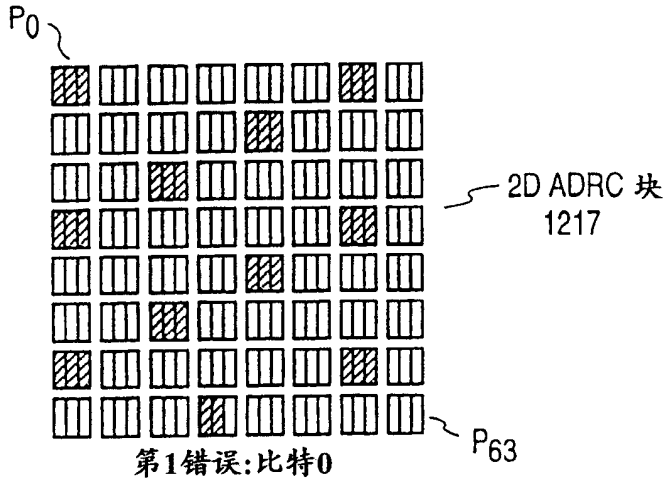
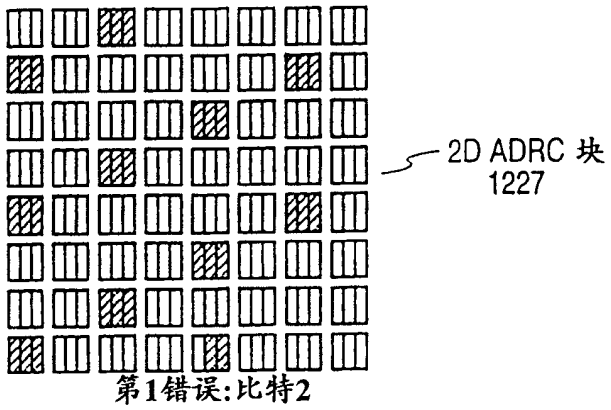

P0
2D ADRC 块
1215
P63
第1错误:比特0
2D ADRC 块
1225
第1错误:比特2
2D ADRC 块
1235
第1错误:比特4

图 12b


P0
2D ADRC 块
1217
P63
第1错误:比特0


2D ADRC 块
1227
第1错误:比特2

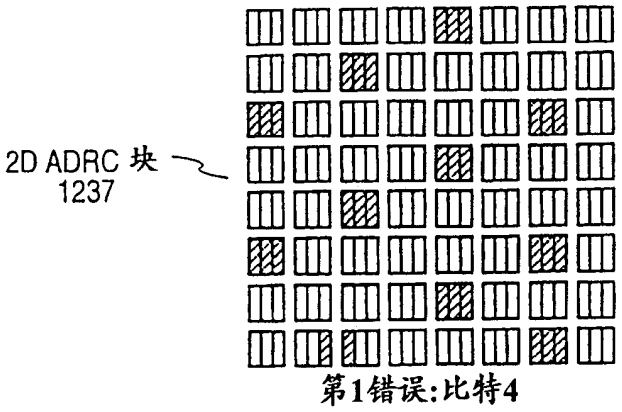
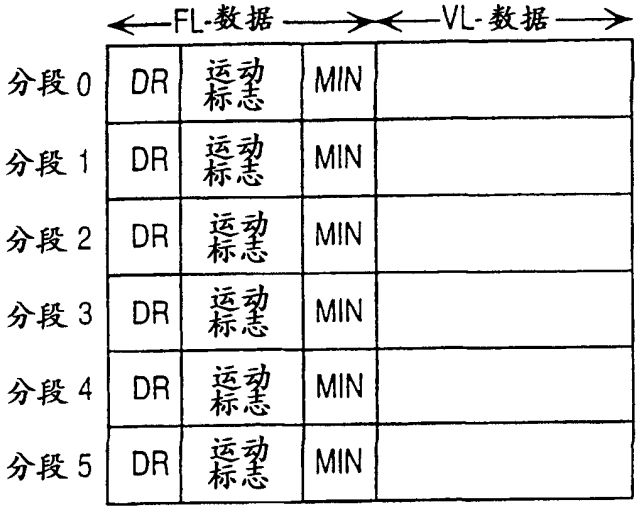

2D ADRC 块
1237
第1错误:比特4

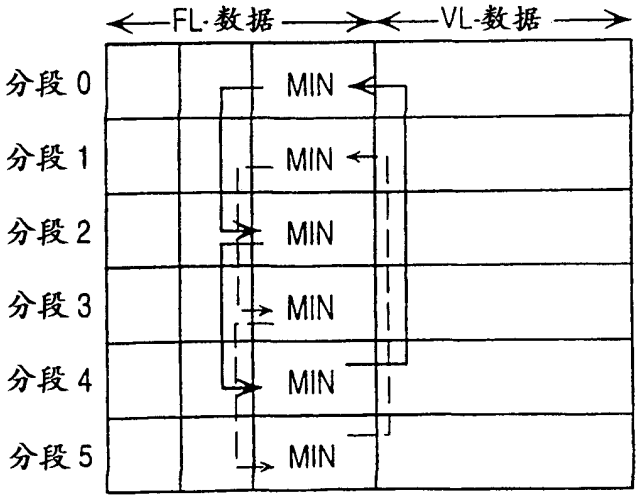
图 12c

分段内FL-数据混洗

原始



MIN 混洗1300

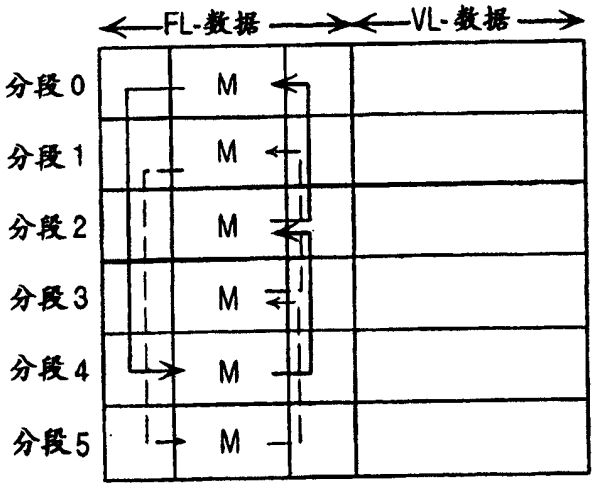


原始 混洗后

分段 0 --> 分段 2
分段 2 --> 分段 4
分段 4 --> 分段 0
分段 1 --> 分段 3
分段 3 --> 分段 5
分段 5 --> 分段 1

图 13

运动标志混洗 1305



原始的		混洗的
分段 0	-->	分段 4
分段 2	-->	分段 0
分段 4	-->	分段 2
分段 1	-->	分段 5
分段 3	-->	分段 1
分段 5	-->	分段 3

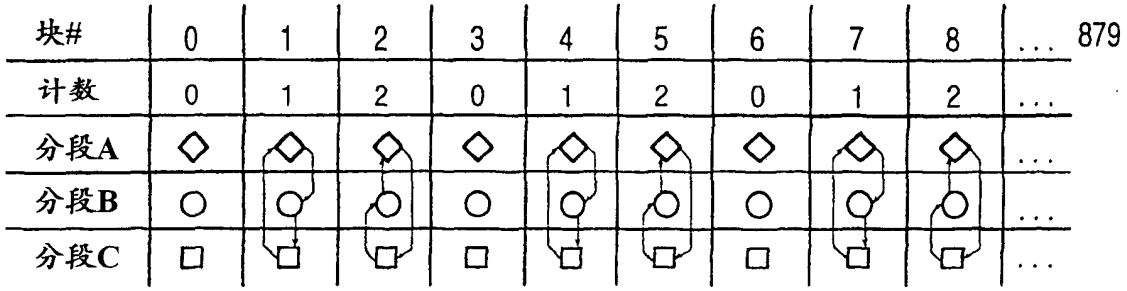
FL-数据混洗后
分段0的FL数据损耗模式

分段 \ 块#	0	1	2	3	4	5	• • • 879
0	DR	DR	DR	DR	DR	DR	• • •
1							• • •
2	M	M	M	M	M	M	• • •
3							• • •
4	MIN	MIN	MIN	MIN	MIN	MIN	• • •
5							

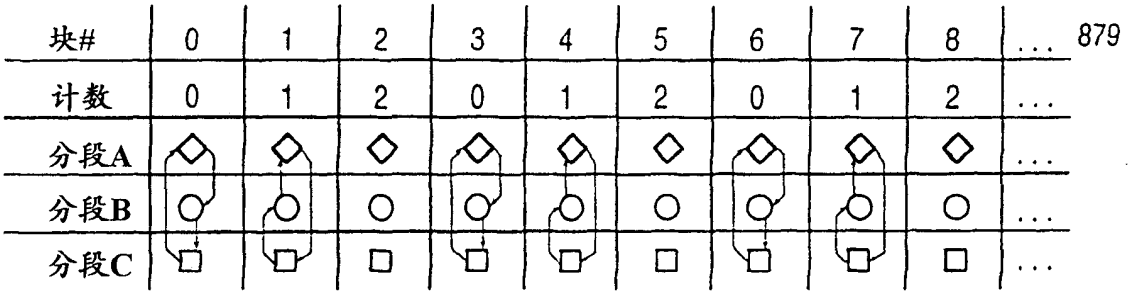
损耗模式1310

M:运动标志

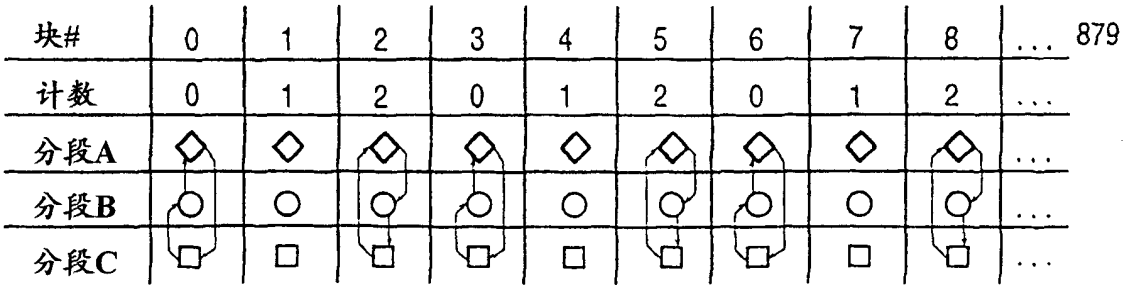
图 13a



DR 取模混洗1410

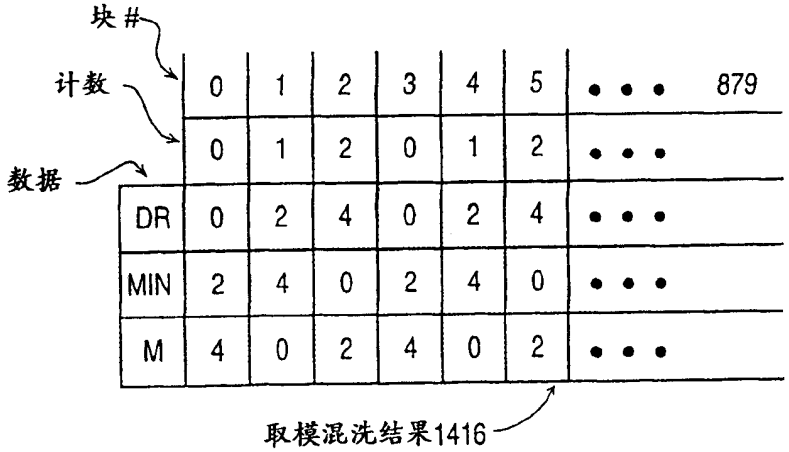


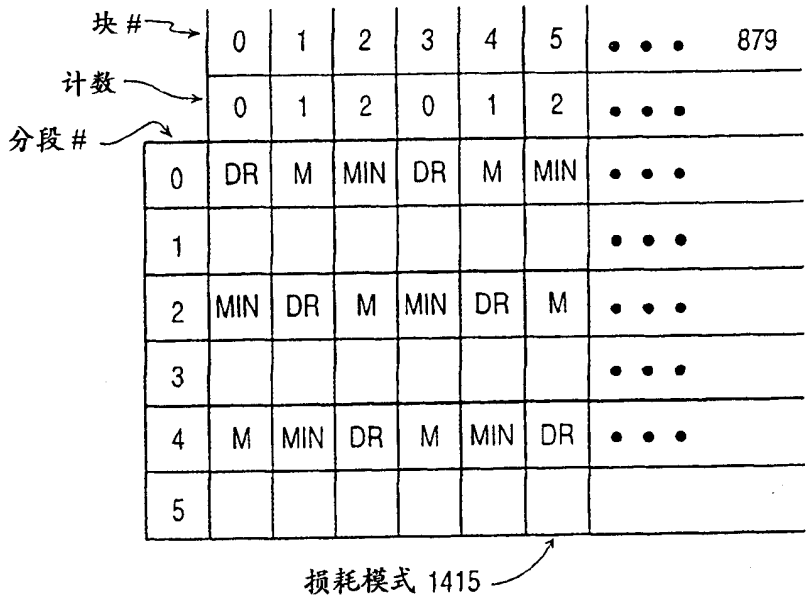
MIN 取模混洗 1420



运动标志取模混洗 1430

图 14

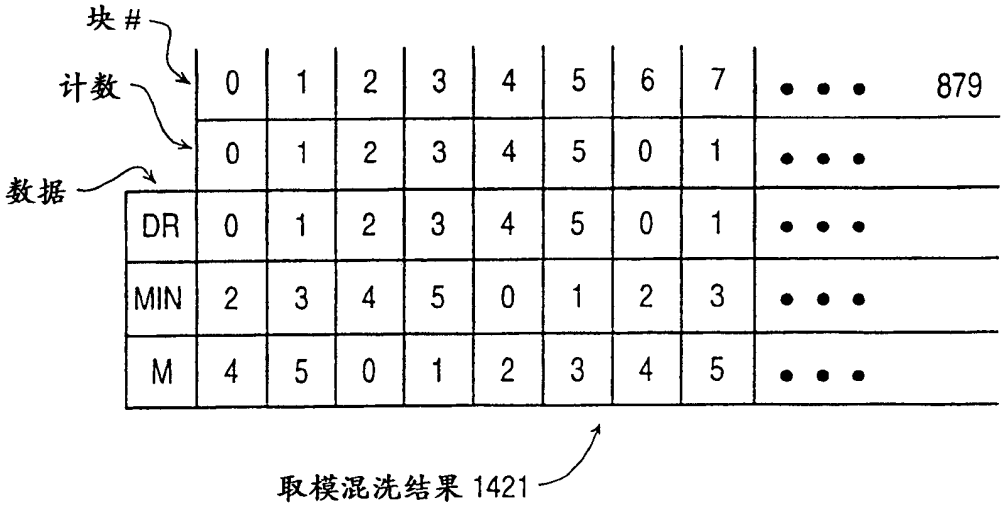




DR		MIN		M		DR		MIN		M	
	M		DR		MIN		M		DR		MIN
DR		MIN		M		DR		MIN		M	
	M		DR		MIN		M		DR		MIN
DR		MIN		M		DR		MIN		M	
	M		DR		MIN		M		DR		MIN
DR		MIN		M		DR		MIN		M	
	M		DR		MIN		M		DR		MIN
DR		MIN		M		DR		MIN		M	
	M		DR		MIN		M		DR		MIN

空间损耗模式
1417

图 14a


块#
计数
数据
0 1 2 3 4 5 6 7 • • • 879
0 1 2 3 4 5 0 1 • • •
DR 0 1 2 3 4 5 0 1 • • •
MIN 2 3 4 5 0 1 2 3 • • •
M 4 5 0 1 2 3 4 5 • • •
取模混洗结果 1421

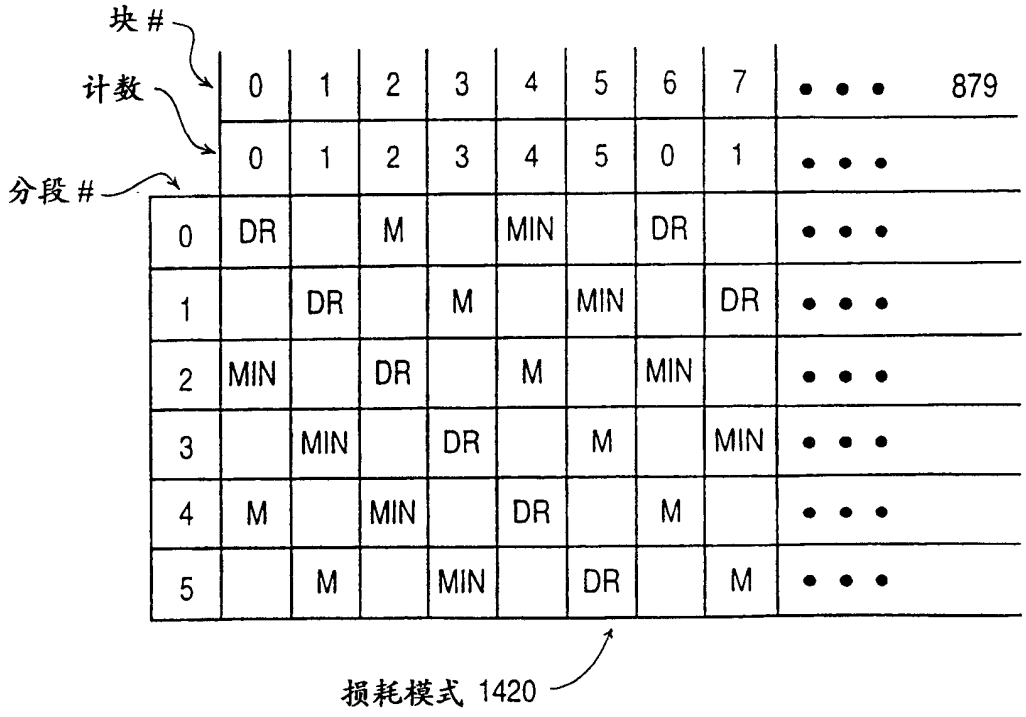
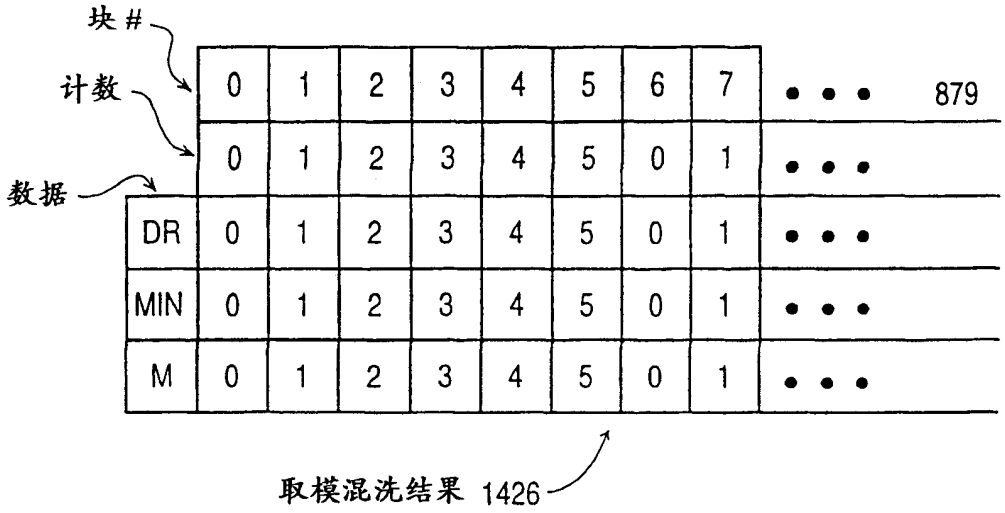

块#
计数
分段#
0 1 2 3 4 5 6 7 • • • 879
0 1 2 3 4 5 0 1 • • •
0 DR M MIN DR • • •
1 DR M MIN DR • • •
2 MIN DR M MIN • • •
3 MIN DR M MIN • • •
4 M MIN DR M • • •
5 M MIN DR M • • •
损耗模式 1420

图14b


块#
计数
数据
0 1 2 3 4 5 6 7 • • • 879
0 1 2 3 4 5 0 1 • • •
DR 0 1 2 3 4 5 0 1 • • •
MIN 0 1 2 3 4 5 0 1 • • •
M 0 1 2 3 4 5 0 1 • • •
取模混洗结果 1426

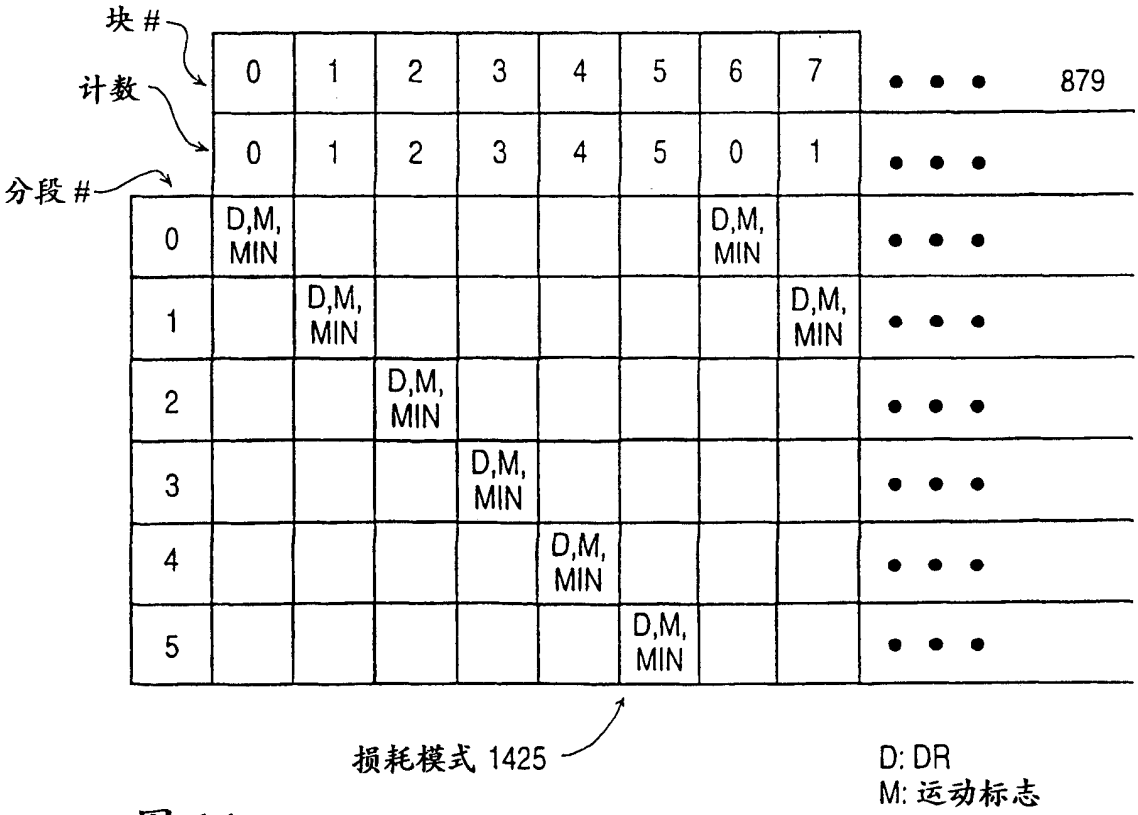

块#
计数
分段#
0 1 2 3 4 5 6 7 • • • 879
0 1 2 3 4 5 0 1 • • •
0 D,M, MIN D,M, MIN • • •
1 D,M, MIN D,M, MIN • • •
2 D,M, MIN • • •
3 D,M, MIN • • •
4 D,M, MIN • • •
5 D,M, MIN • • •
损耗模式 1425
D: DR
M: 运动标志

图 14c

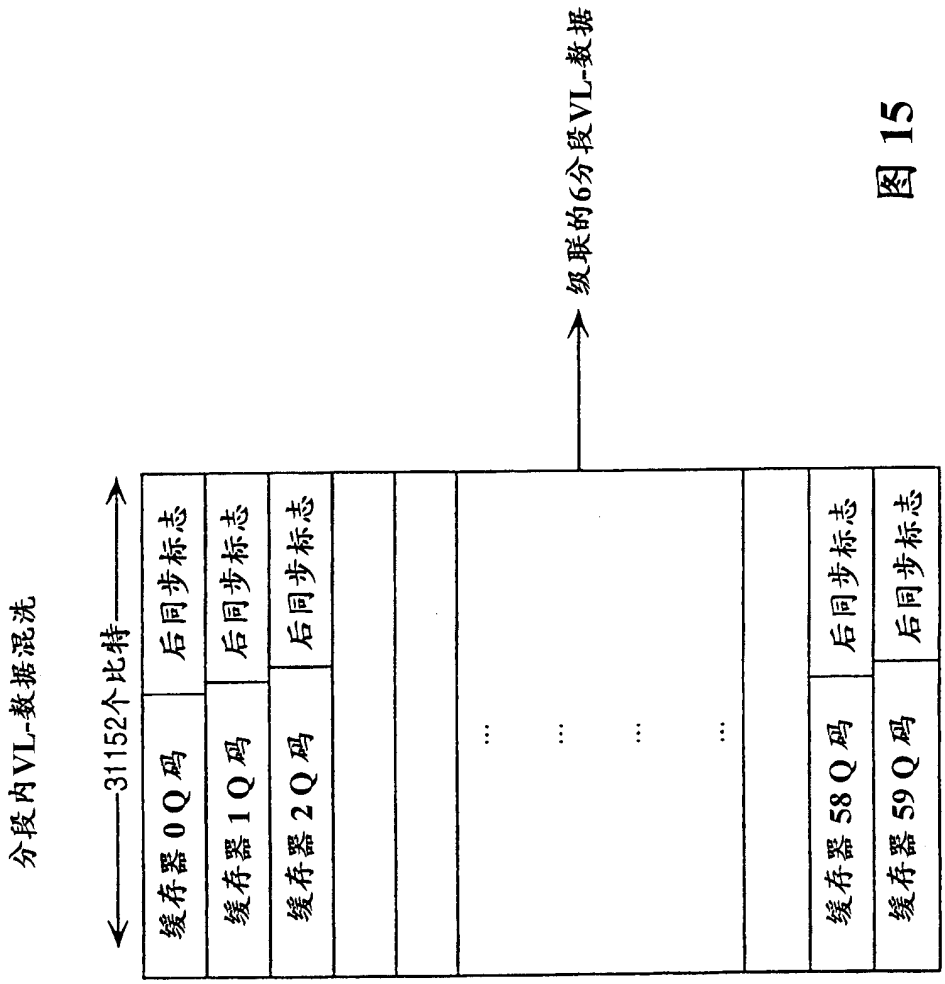

分段内VL-数据混洗
31152个比特
缓存器0Q码
后同步标志
缓存器1Q码
后同步标志
缓存器2Q码
后同步标志
缓存器58Q码
后同步标志
缓存器59Q码
后同步标志
级联的6分段VL-数据

图 15

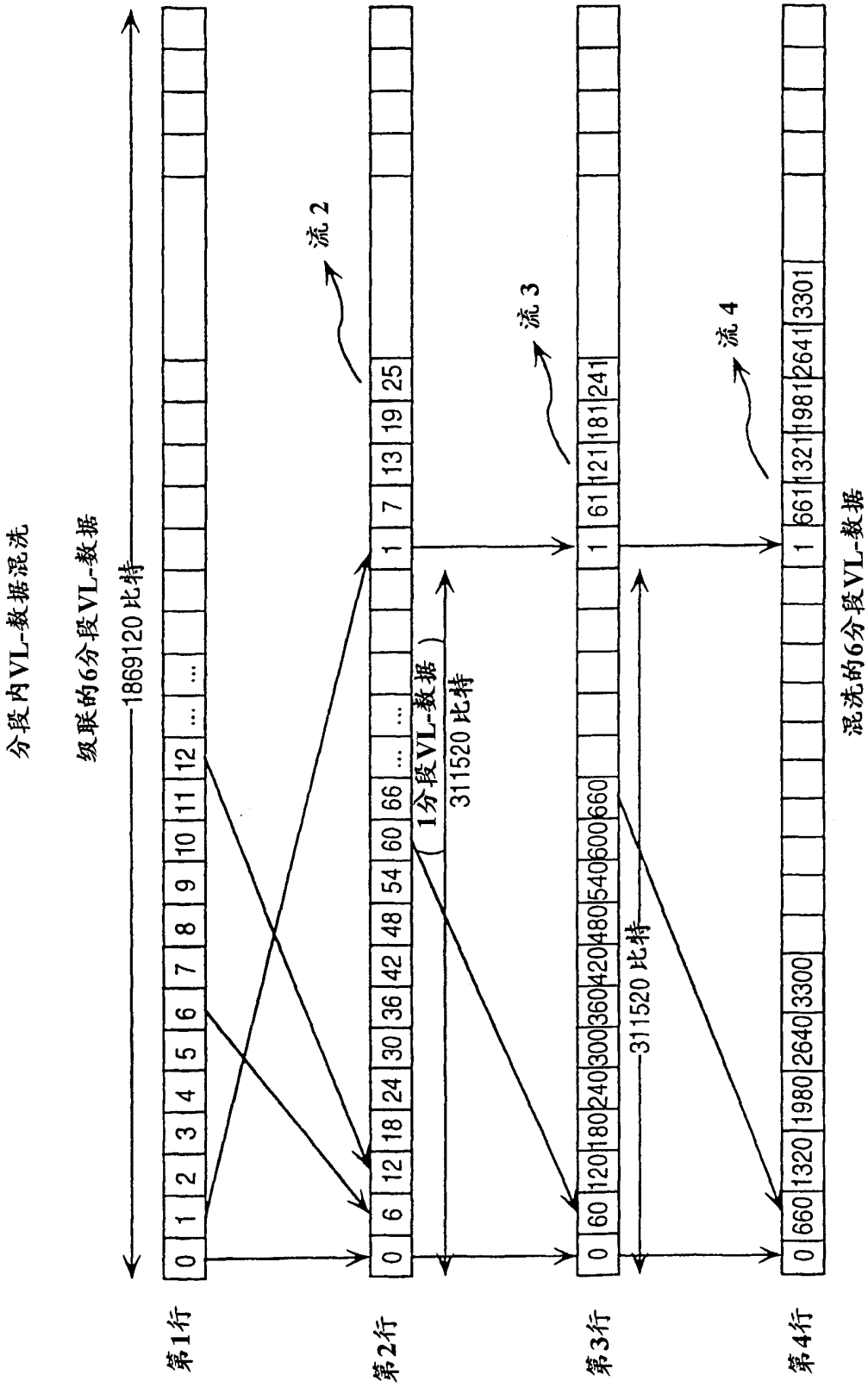

分段内VL-数据混洗
级联的6分段VL-数据
1869120 比特
第1行
第2行
第3行
第4行
流 2
流 3
流 4
1分段VL-数据
311520 比特
311520 比特
混洗的6分段VL-数据

图 16

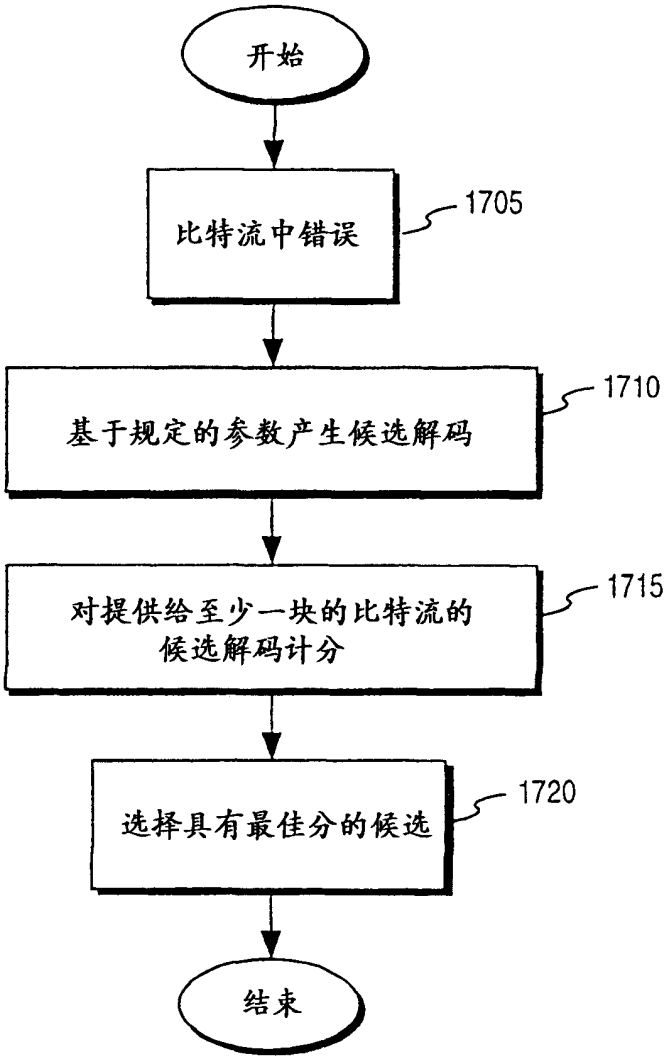

开始
比特流中错误
1705
基于规定的参数产生候选解码
1710
对提供给至少一块的比特流的候选解码计分
1715
选择具有最佳分的候选
1720
结束

图 17

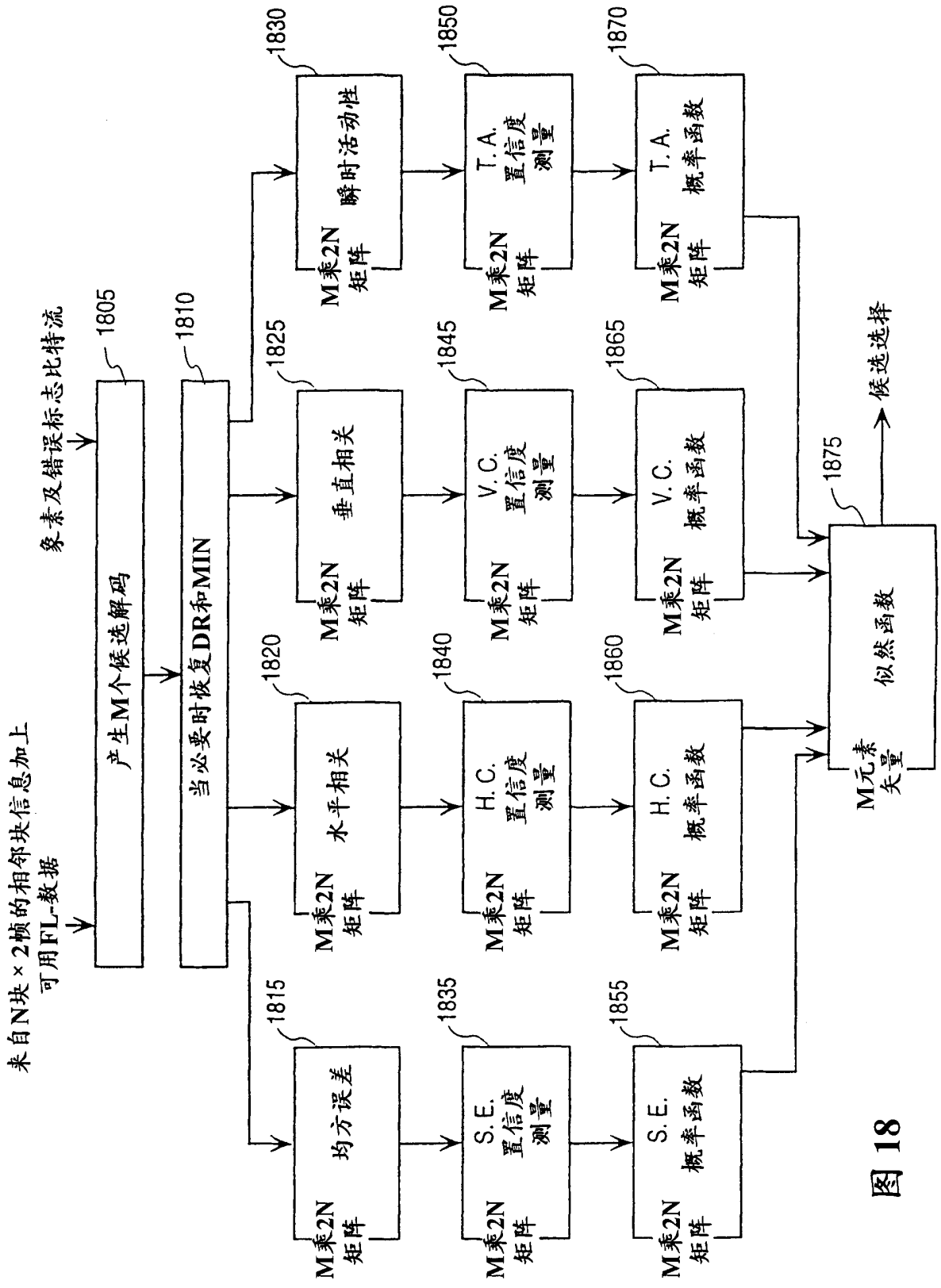

来自N块×2帧的相邻块信息加上
可用FL-数据
象素及错误标志比特流
产生M个候选解码
1805
当必要时恢复DR和MIN
1810
1815
均方误差
M乘2N
矩阵
1820
水平相关
M乘2N
矩阵
1825
垂直相关
M乘2N
矩阵
1830
瞬时活动性
M乘2N
矩阵
1835
S.E.
置信度
测量
M乘2N
矩阵
1840
H.C.
置信度
测量
M乘2N
矩阵
1845
V.C.
置信度
测量
M乘2N
矩阵
1850
T.A.
置信度
测量
M乘2N
矩阵
1855
S.E.
概率函数
M乘2N
矩阵
1860
H.C.
概率函数
M乘2N
矩阵
1865
V.C.
概率函数
M乘2N
矩阵
1870
T.A.
概率函数
M乘2N
矩阵
1875
似然函数
M元素
矢量
候选选择

图 18

- **(mi, qi)的可能情况 (x表示未知)**

(mi, qi)	d_i 的可能值	可能的数目(n_i)
(mi = m, qi = q)	{5m + q}	1
(mi = x, qi = 0)	{0}	1
(mi = x, qi > 0)	{qi, 5 + qi}	2
(mi = 0, qi = x)	{0, 1, 2, 3, 4}	5
(mi = 1, qi = x)	{6, 7, 8, 9}	4
(mi = x, qi = x)	{0, 1, 2, 3, 4, 6, 7, 8, 9,}	9

- **产生所有 $M = n_2 \times n_1 \times n_0$ 可能的密钥**

图 19

测量:均方误差

$$S.E. = \sum_i \sum_j (x_i - y_{i,j})^2 \quad *$$

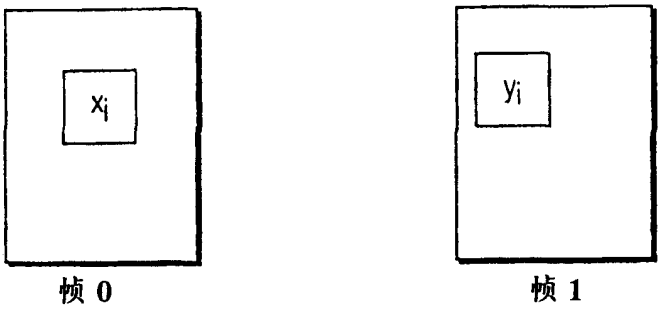
在此 x_i 是象素 i的(候选)解码值

y_{ij} 是 x_i 的水平或垂直相邻者

*注意:当前实现丢弃了3个最大项, 即 $(x_i - y_{i,j})^2$

图 20a

测量:瞬时活动性



x_i 和 y_i 是帧0的块j与帧的块j中相应的象素

T.A. = 块j内i的最大值 $|y_i - x_i|$

图 20b

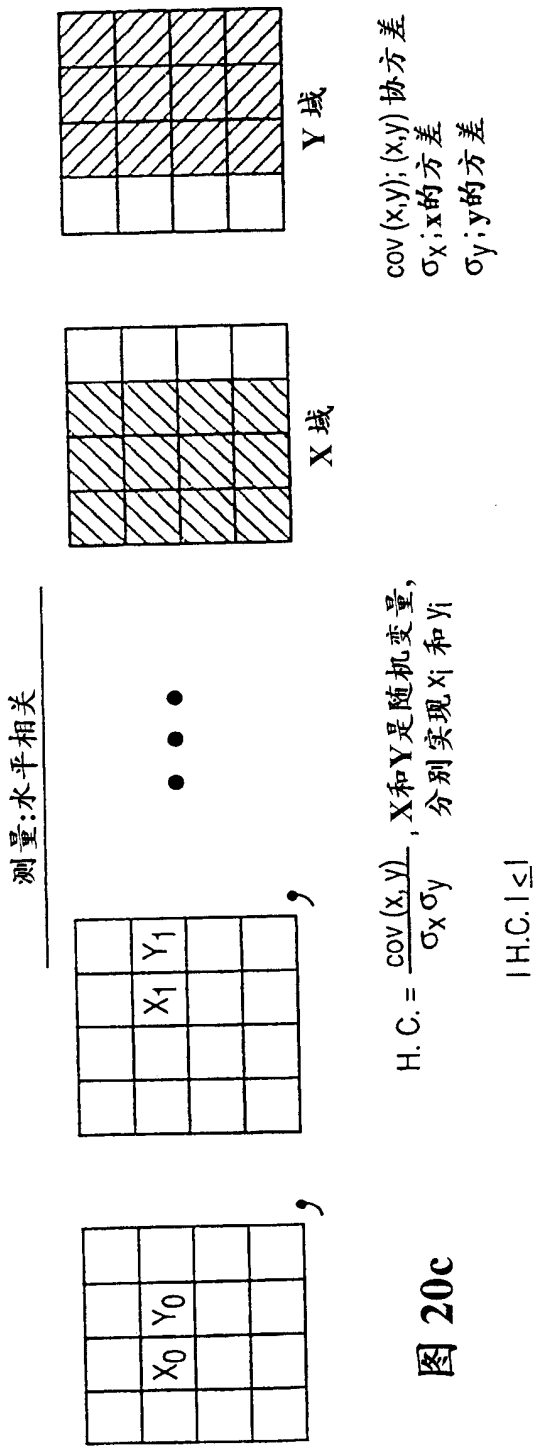

测量:水平相关
X域
Y域
X_0 Y_0
X_1 Y_1
H.C. = $\frac{cov(x,y)}{\sigma_x \sigma_y}$, X和Y是随机变量,
分别实现 x_i 和 y_i
$|H.C.| \leq 1$
cov(x,y): (x,y) 协方差
σ_x: x的方差
σ_y: y的方差

图 20c

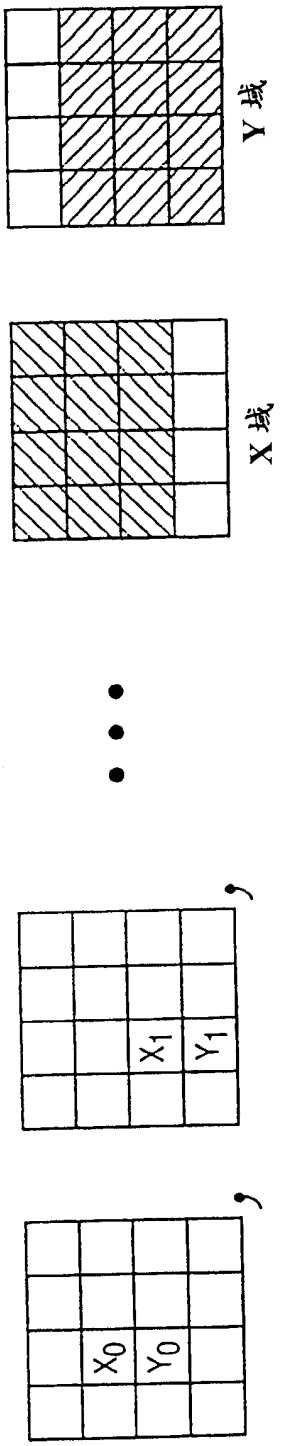

测量:垂直相关
X域
Y域
X_0 Y_0
X_1 Y_1
V.C. = $\frac{cov(x,y)}{\sigma_x \sigma_y}$, $|V.C.| \leq 1$

图 20d

概率函数

S.E. 概率 (1997年9月测量)

S. E. 概率			S.E. <=2^(9)	2^(9) < S.E. <= 2^(11)	2^(11) < S.E. <= 2^(13)	2^(13) < S.E. <= 2^(15)
conf = 0			0.400607	0.486165	0.497679	0.499834
0	< conf <=	0.1	0.108873	0.097436	0.089769	0.102244
0.1	< conf <=	0.2	0.044734	0.014264	0.014457	0.004728
0.2	< conf <=	0.3	0.016635	0.001637	0.001385	0.001193
0.3	< conf <=	0.4	0.007419	0.000513	0.000001	0.000001
0.4	< conf <=	0.5	0.001317	0.000018	0.000022	0.000001
0.5	< conf <=	0.6	0.000264	0.000026	0.000001	0.000008
0.6	< conf <=	0.7	0.000141	0.000031	0.000001	0.000001
0.7	< conf <=	0.8	0.000078	0.000001	0.000001	0.000001
0.8	< conf <=	0.9	0.000001	0.000001	0.000001	0.000001
0.9	< conf <=	1	0.000001	0.000001	0.000001	0.000001

S. E. 概率			2^(15) < S.E. <= 2^(17)	2^(17) < S.E. <= 2^(19)	2^(19) < S.E.
conf = 0			0.499842	0.499689	0.498521
0	< conf <=	0.1	0.093066	0.091473	0.085688
0.1	< conf <=	0.2	0.003602	0.003743	0.003322
0.2	< conf <=	0.3	0.000001	0.000189	0.000192
0.3	< conf <=	0.4	0.000001	0.000075	0.000001
0.4	< conf <=	0.5	0.000001	0.000034	0.000001
0.5	< conf <=	0.6	0.000001	0.000001	0.000001
0.6	< conf <=	0.7	0.000001	0.000001	0.000001
0.7	< conf <=	0.8	0.000001	0.000001	0.000001
0.8	< conf <=	0.9	0.000001	0.000001	0.000001
0.9	< conf <=	1	0.000001	0.000001	0.000001

图 21

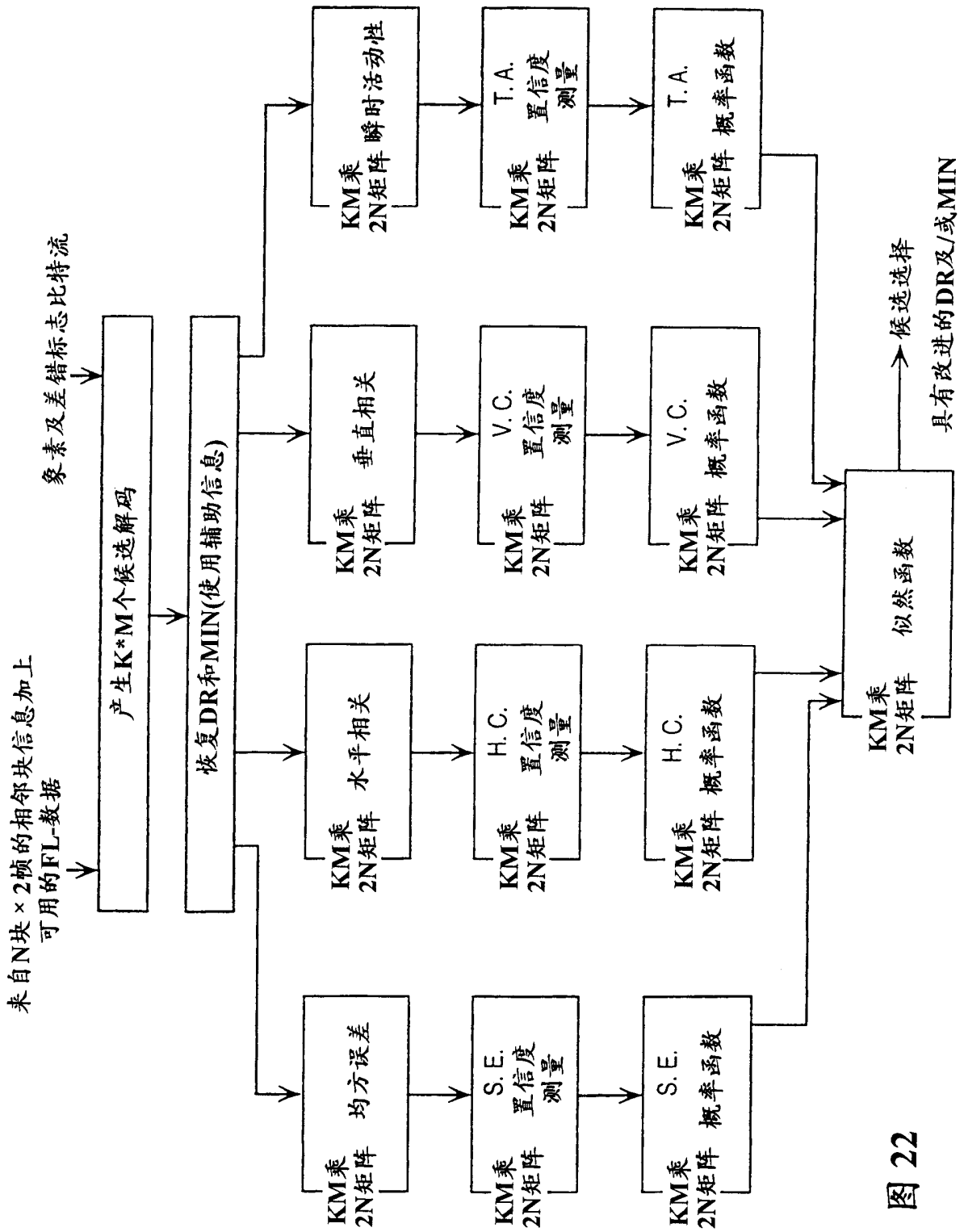

来自N块×2帧的相邻块信息加上
可用的FL-数据
象素及差错标志比特流
产生K*M个候选解码
恢复DR和MIN(使用辅助信息)
KM乘 2N矩阵 均方误差
KM乘 2N矩阵 水平相关
KM乘 2N矩阵 垂直相关
KM乘 2N矩阵 瞬时活动性
S. E. 概率函数
H. C. 概率函数
V. C. 概率函数
T. A. 概率函数
S. E. 置信度测量
H. C. 置信度测量
V. C. 置信度测量
T. A. 置信度测量
KM乘 2N矩阵 似然函数
候选选择
具有改进的DR及/或MIN

图 22

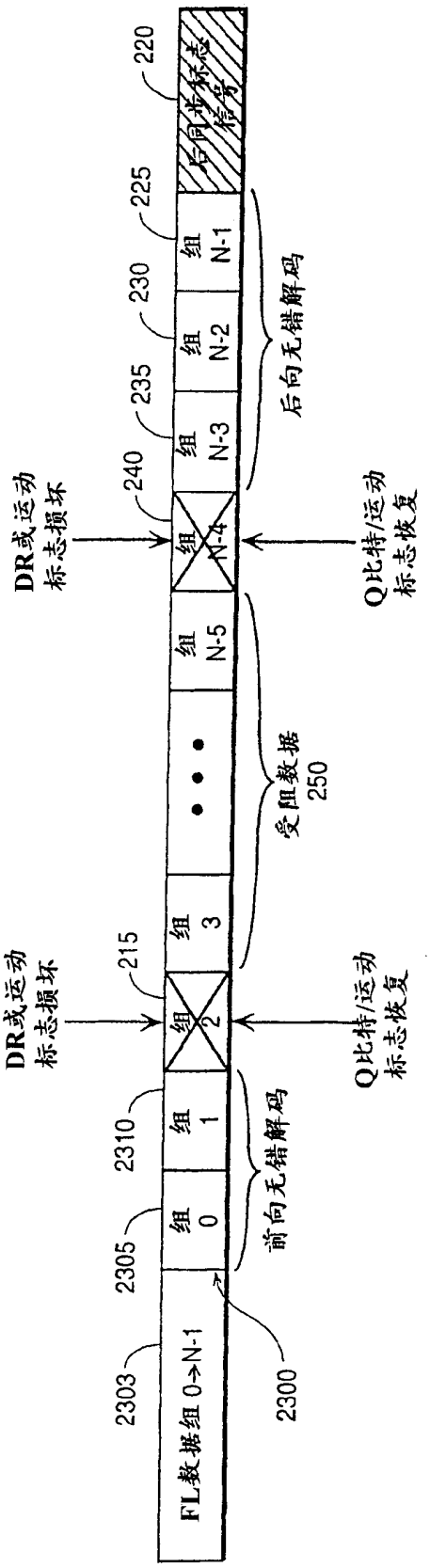

FL数据组 0->N-1
组 0
组 1
组 2
组 3
组 N-5
组 N-4
组 N-3
组 N-2
组 N-1
后向同步标志
2303
2305
2310
215
240
235
230
225
220
2300
DR或运动
标志损坏
Q比特/运动
标志恢复
前向无错解码
受阻数据
250
后向无错解码

图 23

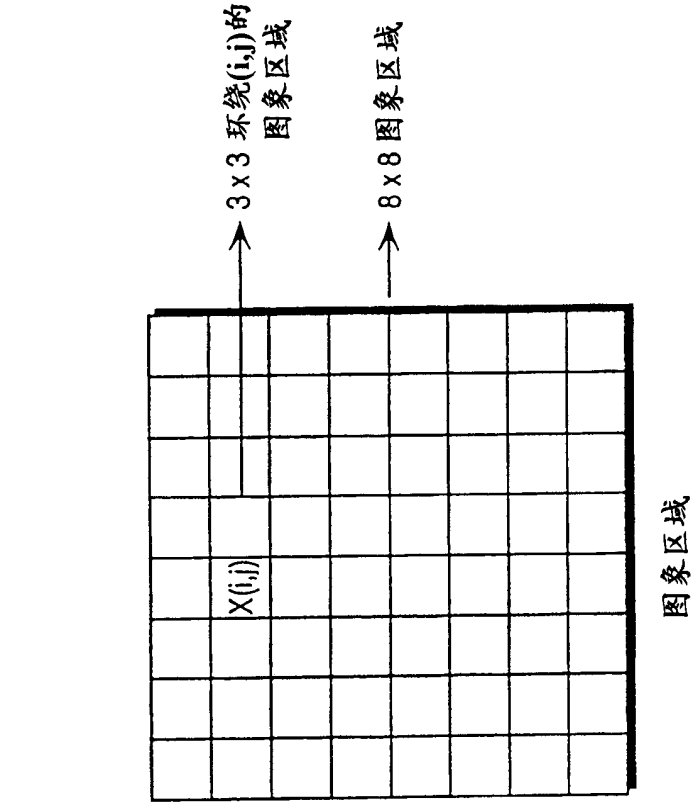

拉普拉斯测量
3 x 3 环绕(i,j)的图象区域
8 x 8 图象区域
X(i,j)
图象区域

图 24b

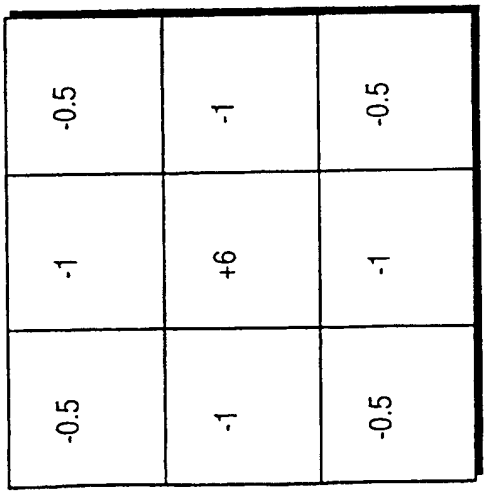

-0.5
-1
-0.5
-1
+6
-1
-0.5
-1
-0.5
拉普拉斯内核

图 24a

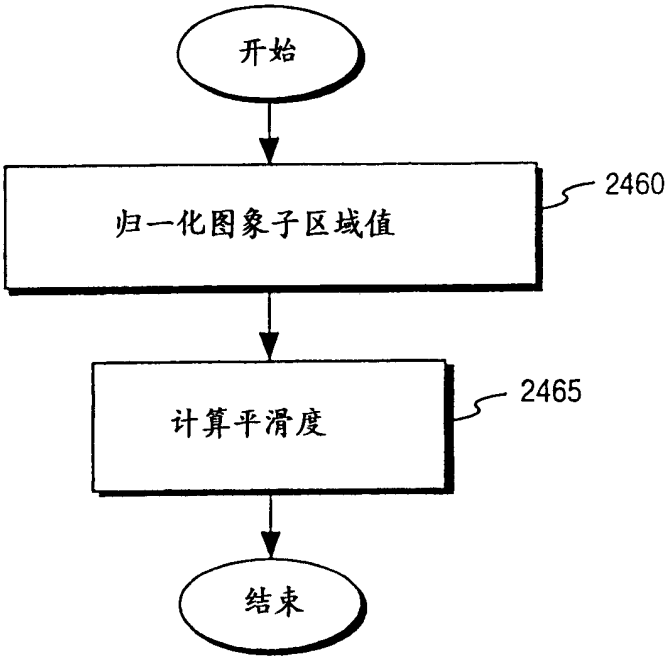


图 24c

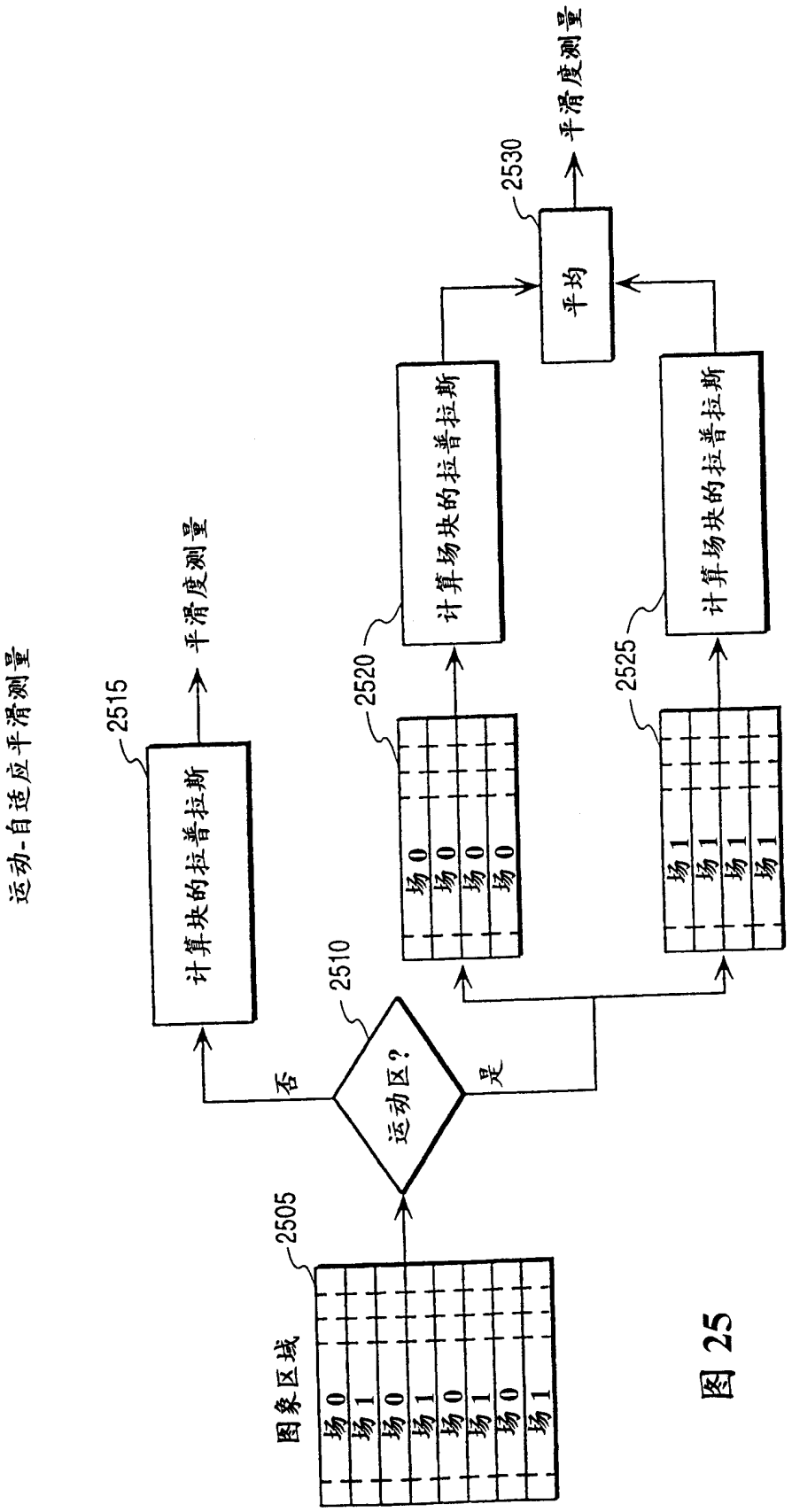

运动-自适应平滑测量
图象区域
2505
场 0
场 1
场 0
场 1
场 0
场 1
场 0
场 1
运动区?
2510
否
是
计算块的拉普拉斯
2515
平滑度测量
2520
场 0
场 0
场 0
场 0
计算场块的拉普拉斯
2525
场 1
场 1
场 1
场 1
平均
2530
平滑度测量

图 25

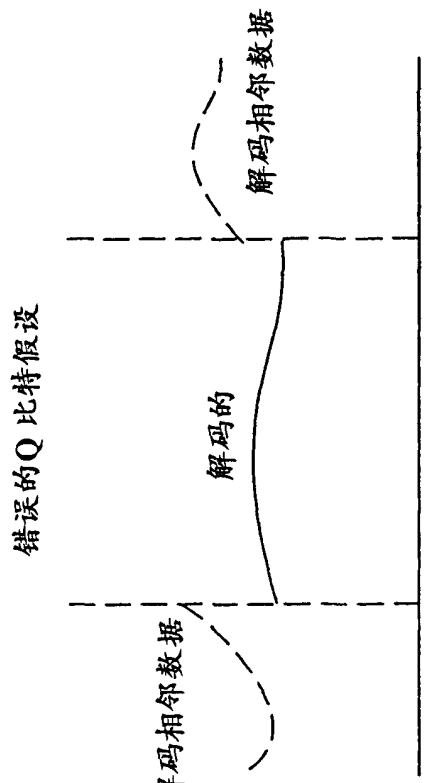


图 26b

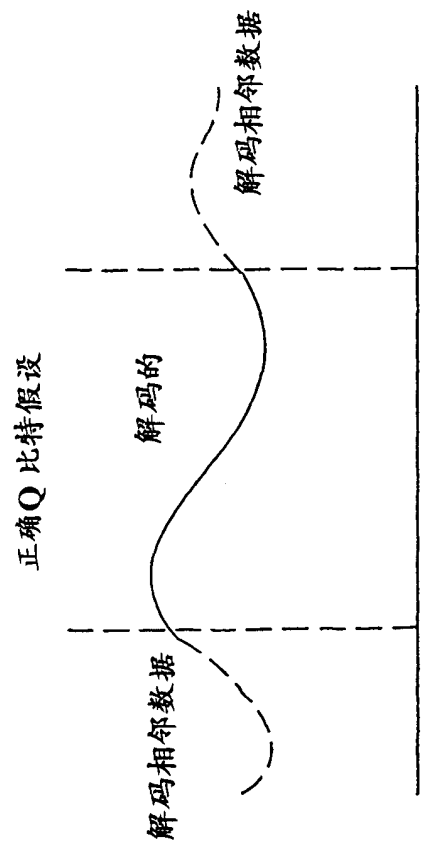


图 26a

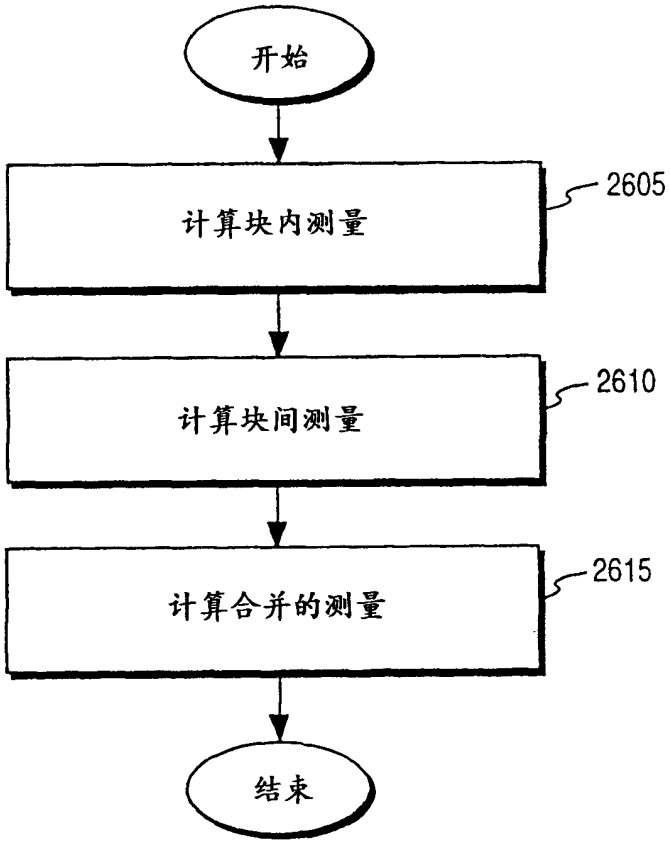


图 26c

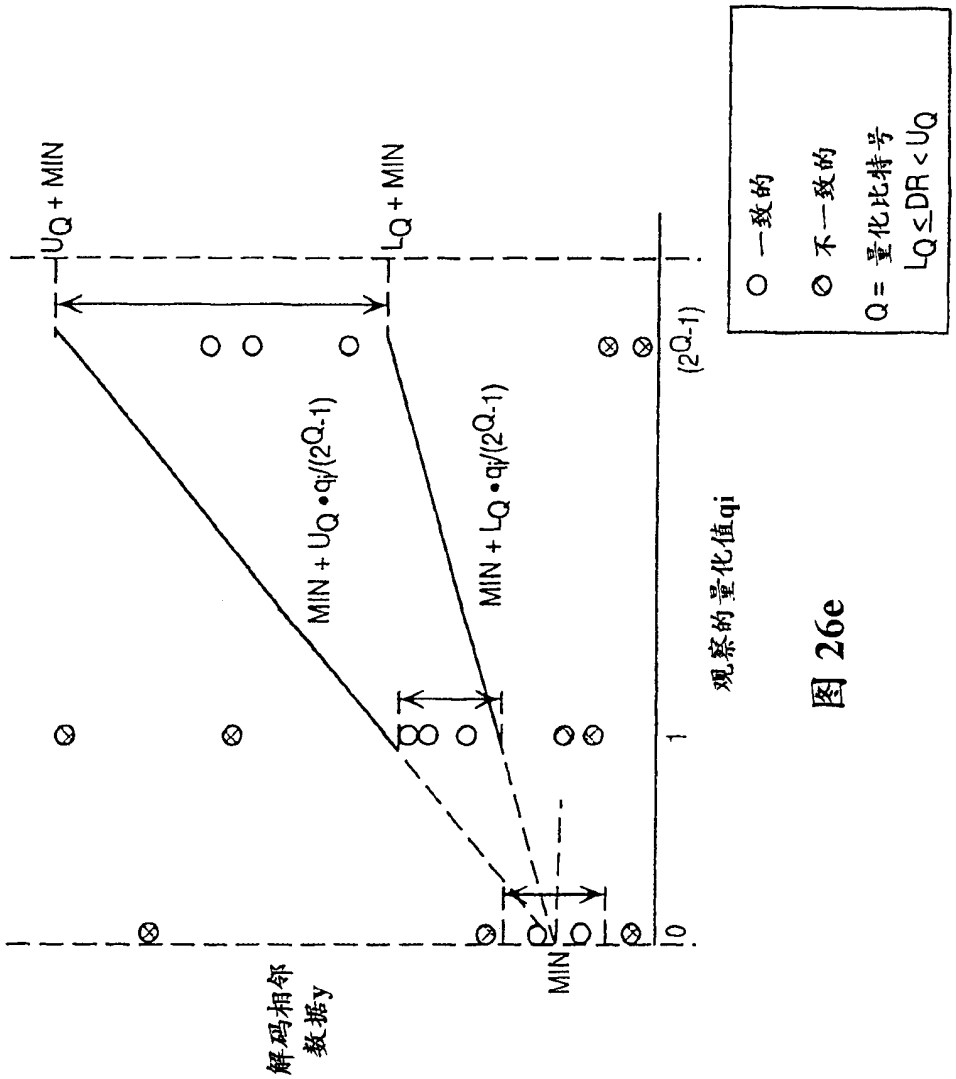

UQ + MIN
LQ + MIN
MIN + UQ • qi/(2Q-1)
MIN + LQ • qi/(2Q-1)
(2Q-1)
1
0
MIN
解码相邻数据y
观察的量化值qi
○ 一致的
⊘ 不一致的
Q = 量化比特号
LQ≤DR < UQ

图 26e

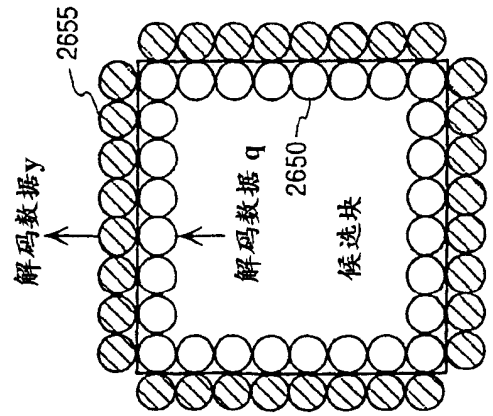

2655
2650
解码数据y
解码数据 q
候选块

图 26d

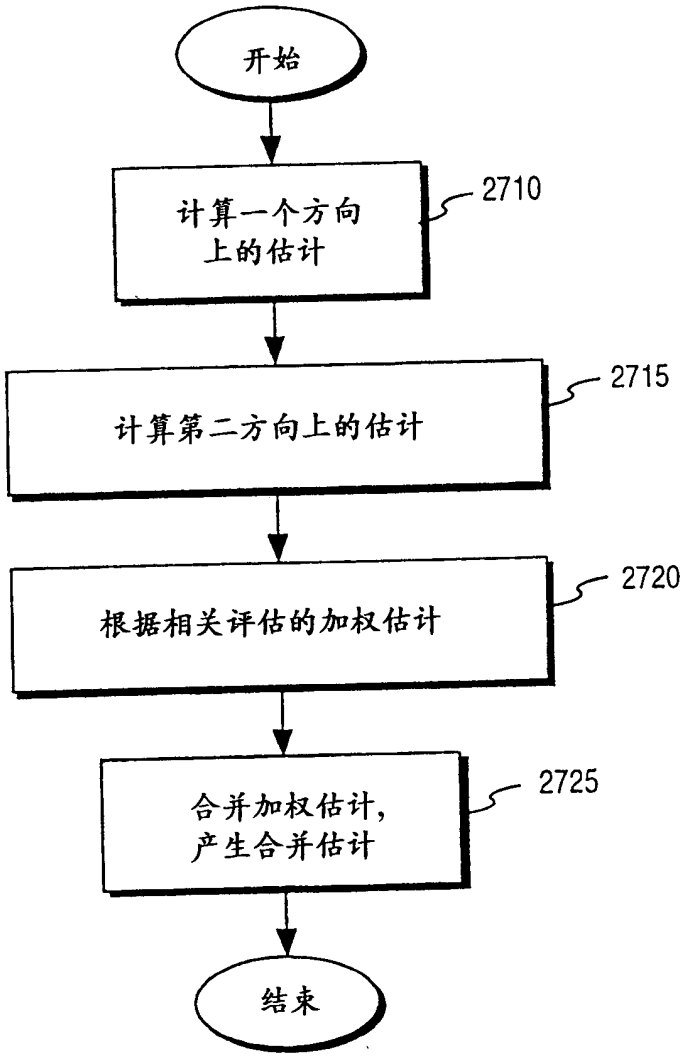

开始
计算一个方向上的估计
2710
计算第二方向上的估计
2715
根据相关评估的加权估计
2720
合并加权估计，产生合并估计
2725
结束

图 27a

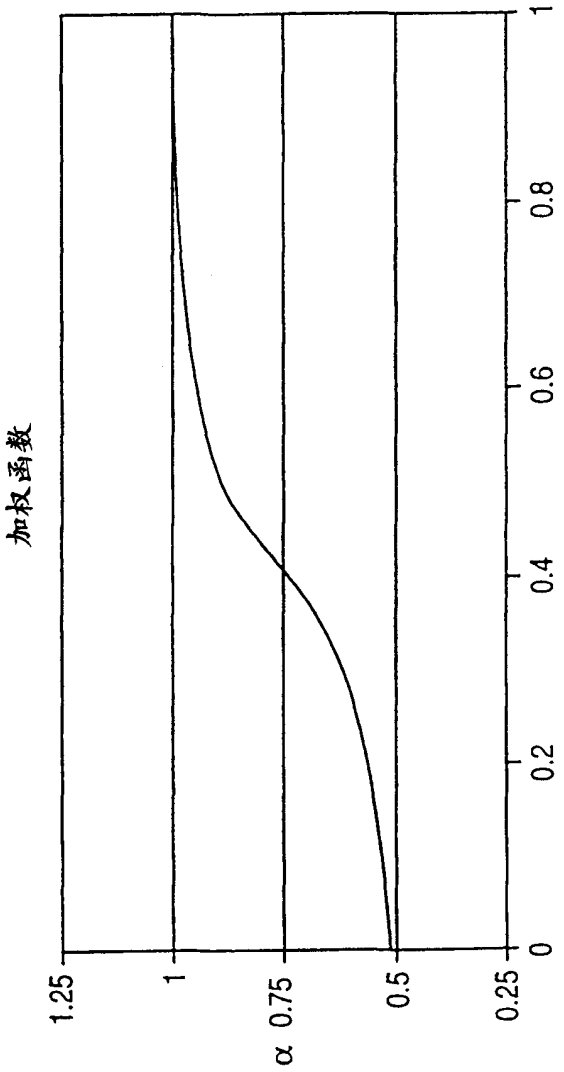


图 27b