RAID控制器中矩阵重构方法研究

姜国松[1,2] 邹 辰[2] 谢长生[1]
(华中科技大学计算机学院外存储系统国家重点实验室 武汉 430074)[1]
(武汉邮电科学研究院 武汉 430074)[2]

摘 要 对于数据重构,纠删码提供了一个特定的编码方法,用于保护那些在磁盘阵列中的多重失效。在RAID的应用中,用纠删码为条块数据丢失建模,以便优化重构算法来重构整个条块。换句话说,它们只应用于高度相关的扇区故障,也就是在丢失磁盘上连续的扇区。定位了两个更一般的问题:①由分散或不相关的擦除导致丢失的数据的恢复;②由单个磁盘(存在许多故障时)导致的部分但连续的丢失数据的恢复。对两个问题所建议的方法是完全一般化的,并且能够应用于任何纠删码,但是此方法最适合基于异或的编码。对于分散的擦除,典型的是为每一个丢失扇区的数据规定了两种结果:要么这个丢失的数据被宣布为不可恢复,要么宣布为可恢复。并且,为只依赖于可读扇区的重构提供一个规则。简而言之,这个方法既完整又具有建设性。

关键词 冗余磁盘阵列,条带,条块,生成矩阵,纠删码

Research on Matrix Reconstruction in RAID Controller

JIANG Guo-song[1,2] ZOU Chen[2] XIE Chang-sheng[1]
(National Storage System Laboratory, Huazhong University of Science and Technology, Wuhan 430074, China)[1]
(Wuhan Research Institute of Post & Telecommunication, Wuhan 430074, China)[2]

Abstract Erasure code provides a specific coding method for data reconstruction which is for the protection of those multiple failure in the disk array. In RAID applications, with Erasure code strips for data loss modeling to optimize reconstruction algorithm to the entire strip reconstruction. In other words, they are only applied to a high degree of fault-related sectors which is the sequential sector on the lost disk. We addressed two more general issues: data recovery from scattered or not related erase sectors which lead to the loss; data recovery from single disk failure(existing many faults) which lead to the partial loss. The two issues of the proposed method are completely generalized and can be used in any Erasure code, but best suited to XOR-based Erasure code. For scattered erasure code, we typically provide for two results for every loss sector data: either the lost data is declared unrecoverable or it is declared recoverable and a formula is provided for the reconstruction that depends only on readable sectors. In short, the methodology is both complete and constructive.

Keywords RAID, Stripe, Strip, Generator matrix, Erasure code

1 引言

针对磁盘阵列的基于异或的纠删码将丢失数据从粗的层面上模型化为整个磁盘的丢失,但是从更精确层面上是模型化为编码的整个符号的丢失。

实际上,一个符号典型的是映射为一个条块。更确切地说,具有符号的一个比特位的多个连续的扇区对应于一个或多个扇区,具有各自不同的符号的多个连续的扇区驻留在不同的磁盘。相关条块的集合称为条带。为了处理磁盘故障,每一个纠删码是与特定的高度依赖于纠删码结构的重构算法一起的。当两个磁盘失效时,重构遵循这些对角线路径,起始于一些初始点。换句话说,重构既是几何定义的,又是递归定义的。BCP码[4]几乎没有几何设计,但是仍然有一个递归重构算法。

纠删码因而被看成有相互关系的扇区故障:当磁盘失效时,一个条块内的所有扇区就丢失了。但是,渐增的磁盘容量连同一个相当稳定的比特误差率意味着在一个给定的条带内存在多个不相关或分散的扇区错误,特别是与一个或多个磁盘失效结合。例如,两个磁盘失效加上一个扇区失效可能经常发生,使得一个2磁盘容错码甚至不可能提供充分的可靠性。假如所有的相关和不相关的擦除发生在至多 t 个磁盘内时(这里 t 是纠删码的容错数),那么一个方法是模拟所有受影响磁盘,并根据特定码的重构算法重建丢失。但是,这有两个缺陷。首先,非常明显的是这种方法的效率是非常低下的,

到稿日期:2008-08-08 返修日期:2008-11-11 本文受国家自然科学基金项目(60603074)资助。
姜国松(1976—),男,博士生,研究方向为计算机存储系统,E-mail:gsjiang@fiberhome.com.cn;邹 辰(1969—),男,高级工程师,研究方向为嵌入式系统开发、光通信;谢长生(1957—),男,硕士,教授,博士生导师,研究方向为NAS与SAN存储体系、iSCSI存储设备、图像存储与处理和计算机系统结构。

由于它要求要么已知或可读数据的重构，要么要求在处理的每一步进行检查以便了解是否一个重构是必需的。但更重要的是，这个方法并没有解决当多于 t 个磁盘由于扇区失效已经受影响时的更一般问题。在这种情况下，丢失扇区的部分或全部能够被重构是完全可能的，但是根据来自于编码的删除纠正能力，这个方法是不明显的演绎。

除此之外，当每个纠删码提供一个方法重构整个条块时，根据我们的了解，并没有文献包含任何明确定位重构丢失数据的部分条块问题的方法。这样的一个需求可能在一个没有完成的重建操作期间，主机读操作一个失效的磁盘时出现。当然，条块重构方法能够应用于这种情况，但是很可能这样的重构将恢复额外不必要的丢失扇区。也就是说，做了超过所要求的更多工作来为主机读操作服务，因此影响了性能。

本文定位于这些问题的两个方面。由于本文的方法能够应用于任意容错的纠删码，因此是通用的。此方法可应用于从完全的磁盘失效、分散的扇区失效以及两种情况的结合等场景中的任意失效情况。方法完全基于针对纠删码的生成矩阵。因此，一个通用的纠删码重构模块能够实现这个方法，并且使用生成矩阵作为输入参数之一。

对于第一个问题——分散(相关和/或不相关)的扇区丢失问题，方法提供了一个数学上的保证：对于每一个丢失的扇区，要么这个扇区的数据宣布为不可恢复，要么这个扇区的数据宣布为可恢复，并且生成一个重构公式。这个重构公式是一个包括已知或可读数据和校验扇区的线性方程。在这个方面，我们的方法是既完整、有建设性，又可以普遍应用的。它为了满足下面的要求提供了最好的保证。

应当注意的是对于 1 元容错码(例如 RAID1，RAID4 或 RAID5)，对于这两个问题的解决办法是相当简单且明显的。同样地，对于 Reed-Solomon 码[9,10]，Reed-Solomon 码里符号被映射为字节或字(不是扇区集)，标准的 Reed-Solomon 程序也直接定位这两个问题。更有趣的情况是 non-Reed-Solomon 多容错码。这样的编码就像那些大多数已经实际应用的编码一样是典型的基于异或的码。对一个特定码进行仔细、综合的分析可以提出对这个问题的解决办法。该法是通用的，同样非常明显的是，该方法能够扩展到更一般的编码。这个方法不仅能应用于 RAID 控制器，而且可以应用于任何像 dRAID(分布式 RAID)基于节点系统这样的纠删码类型。

对于第二个问题——部分条块重构问题，建议了一个混合解决方法：将针对完全重建的纠删码固有的递归方法和恢复分散扇区的方法联合起来。同时也建议了一个可选择的解决方法，那就是在许多情况下相当于特定码的方法，这个方法在某些情况下更好，而且可普遍地应用于任何纠删码。

这个方法是基于矩阵理论和伪逆的原则。许多编码用完全的反转来证明编码既具有标称的容错，又提供重构公式。但是应用这些编码仅仅只是恢复完全码的符号，在最大失效(这里唯一的反转存在)和不是更一般的符号内的比特位(也称为一个条块内的扇区)情况下就是在这个工作当中的定位。

2 相关工作

本文的两个主要结果是：

(1)将矩阵的伪逆操作应用于针对不相关丢失扇区的重构问题。

(2)联合特定码递归重构方法和这个矩阵方法的混合重构来有效地重构部分条块。

根据了解，对于这些问题的两个方面在已有文献里还没有明确的定位。和以前谈及的一样，矩阵求逆理论被用在一些编码满足条块容错的证明上。例如，Reed-Solomon 码通过求解一个线性方程组证明容错性。在这种情况下，矩阵求逆方法在最大失效时被用于全部条块的求解。相比之下，此方法对于任何擦除比特的分布定位于符号中的单个比特位。二进制的 BR 码[5]对于全部条块丢失有一个递归解决方法。

本文给这个递归式提供了一个闭合解。对于 EVENODD 码，给出了一个递归式，并很明确地指出这个递归式能够求解。对这个递归式的精确解等于在全部条块丢失的特殊情况下矩阵的解。BCP，ZZS，Xcode[2]和 RDP 等编码都有递归重构算法。后面的两种编码(以及 EVENODD 码)是几何的，并且很容易形象化；前面的编码更多是组合的，而且缺乏直观性。在任一种情况下，这些具有递归重构算法的编码是非常适合于本文的混合方法的。另外，许多混合方法应用于任何适合磁盘阵列的纠删码，这些纠删码可具有也可能不具有一个递归重构算法。

3 生成矩阵和校验矩阵

纠删码的生成矩阵 G[3]将输入的“字”(输入数据)转换成“码字”(数据和校验)。校验矩阵证实“码字”包含一致的数据和校验(校验擦除)。

生成矩阵是特定的一个列分块结构：每块对应一个条块，在一块中的每个列对应条块中的一个单元 。假定列只包含一个 1，那么这个单元包含用户数据。由于这个列是单位矩阵的一列，因此我们称这样的一个列为“单元列”。如果这个列包含多个 1 的话，那么它对应用户数据单元的一些集的异或算术单元。就是说，这个单元是一个校验单元。换句话说，生成矩阵 G 指定了数据和校验在条块上的布局、条带内条块的逻辑顺序和用于计算校验值的方程。例如，在 5 个磁盘上初始 $p=3$ 的 EVENODD(3,5)码的生成矩阵是

$$G=\left(\begin{array}{cc|cc|cc|cc|cc} 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 1 & 0 \end{array}\right)$$

虽然不作为一个要求，但是磁盘阵列的生成矩阵典型地有一个针对每个用户数据单元的单位列，以便数据总能逐步地复制到一些条块中的单元扇区，然后能以最少的 I/O 开销被读取。

让 D 作为输入的用户数据的行向量，那么行向量 S 可以通过下面的表达式给出：

$$S=D\cdot G \tag{1}$$

式(1)表示了存储在磁盘条带内的数据和校验单元。向量 D 是通过用户数据的逻辑地址来变址的。向量 S 表示数据和校验单元的物理地址，包括磁盘地址(实际上就是条块，由生成矩阵的块来确定的)和此盘上扇区的地址(单元或者条块内的偏移量，由块内的列来确定)。S 同样也是分块结构的，并且这些分块能与 G 中的块相匹配。

假如有 N 个数据单元输入到编码中，并且 Q 个校验单元要通过编码来计算，那么生成矩阵为 $N\times(N+Q)$ 维大小(注意：N 是条带内数据单元的总数，而不是条块数；相似地，Q 是条带内校验单元的总数，而不是校验条块数)。校验矩阵 H 是 $(N+Q)\times Q$ 维大小，并且直接来自于生成矩阵(反之也是一样)。信道使用检验矩阵来探测错误。在数据和校验从信道中读完之后，当校验对应列指示时，校验同数据进行异或操作，产生一个“伴随矩阵”。假如伴随矩阵不为 0，那么就发生了错误(要么在接收的校验符号，要么在相关的数据符号中的一个)。对于磁盘阵列中的纠删码，这是校验一致性检查。换句话说，像上面所说的 $S=D\cdot G$ 时，这个检验就是校验一致性检查。

$$S\cdot H==0 \tag{2}$$

校验矩阵是行分块的，正好对应于 G(或 S)的列分块，而且能够通过排列来包含一个嵌入的单位矩阵(对应校验单元)。假如 G 是系统矩阵，那么这样做是很容易的。对于上面例子中生成矩阵 G 的校验矩阵如下。

$$H=\left(\begin{array}{cccc}1&0&1&0\\0&1&0&1\\\hline 1&0&0&1\\0&1&1&1\\\hline 1&0&1&1\\0&1&1&0\\\hline 1&0&0&0\\0&1&0&0\\\hline 0&0&1&0\\0&0&0&1\end{array}\right)$$

简而言之，生成矩阵是为磁盘上的存储计算数据和校验的(以及它的布局)。当所有的数据和校验从磁盘上读完成时(例如在校验擦除期间)，校验矩阵能够用于寻找错误。

假如一个编码能够容忍 $t\geqslant 1$ 个磁盘或条块丢失，那么 G 必须拥有这样的特性：假如任意 t 个 G 的块被移除(或者被置为 0)，那么结果矩阵必须是行满秩(因为嵌入的单位矩阵)。

同样，式(2)暗示：

$$D\cdot G\cdot H=0$$

对于每个数据向量 D，应当保持 $D\cdot G\cdot H=0$。这就意味着 $G\cdot H=0$，以便在 H 中的每个向量是在 G 的零空间内。一个简单的维度参数显示出 H 实际上是 G 的零空间的基。

另外，应该比较清楚的是，假如 G 是系统性的，那么存在一个 $M\times N$ 矩阵 R_0，R_0 包含一个嵌入的 $N\times N$ 单位矩阵，因此 R_0 是 G 的一个伪逆。R_0 正好摘掉 G 的嵌入系统部分。假如 G 不是系统性的，伪逆 R_0 仍然能够构造出来，但不是那么简单了。

4 模拟扇区丢失及矩阵重构算法

本节为了解决两个问题中的第一个，并用理论展开。在上面指出了一个 t 容错码 G 必须有这样的一个性质：零化 G 的任意 t 块应当让 G 满秩，以便 G 的全伪逆必须存在。这建议能够通过零化或移除 G 中相关的单个列来模拟相关以及不相关的扇区丢失。应当确信的是，不相关扇区丢失结合起来必将导致一些或全部的数据丢失(一些或全部丢失的扇区有不可恢复的数据)，其它的结合可能不包括数据丢失。以直接的方法来讲，方法将正确地决定什么样的扇区成为数据丢失事件。什么样的扇区不会成为扇区丢失事件，方法也将提供一个这些扇区数据的重构公式。

假定在条带内检测到一组失效扇区 F(相关情况，或许是由于磁盘故障引起的；或者是不相关的，由媒体错误或媒体错误的组合引起的)。完全忽略 G 的块结构，让 $\hat{G}$ 为生成矩阵 G 的一个版本，成为零的列对应 F 中的扇区。假定能够找到一个 $M\times N$ 的矩阵 R，那么

- R 是 $\hat{G}$ 的一个局部伪逆；
- R 在所有对应 $\hat{G}$ 丢失列的行中都有零。

我们将 R 的列与 D 中的用户数据值联系起来。下面的定理包含我们主要的理论结果。

定理 让 G，$\hat{G}$ 和 R 如上面描述的一样。理论上任何可恢复的用户数据值对应 R 的一个非零列，并且非零比特的位置指出数据和校验单元，它们的异或运算结果等于数据值。一个特殊的情况是一个直接可读的数据值对应 R 中的一个单元列。一个数据丢失事件(不可恢复的数据值)对应 R 的一个全为 0 的列。

证明：让 $\hat{S}$ 为一个像在式(1)中的 S 一样的向量，但是在对应丢失单元的位置上是 0(G 中为 0 的列)。那么很明显

$$D\cdot G=S$$

因此，我们有

$$S\cdot R=D\cdot G\cdot R=D\cdot J_K=D$$

这里 $\hat{D}$ 是在对应 J_K 对角线上的 0 的所有位置为 0 的向量 D，J_K 也是对应 R 的全为 0 的列。

事实上 J_K 是由 $\hat{G}$ 唯一决定的，这意味着任何 J_K 的 0 对角元导致了在 $\hat{D}$ 中的一个 0，这对应于一个数据丢失事件。J_K 的任意非零对角元导致了在 $\hat{D}$ 中一个非零(不等同于 0)数据值。但是 J_K 的非零对角元对应 R 的非零列，并且零对角元对应 R 的全零列。这证明了第一部分和最后一部分的陈述。

现在再考虑一个 R 的非零列的问题。在那样一列中的每一个非零比特选择进入一个异或公式与来自 $\hat{S}$ 的数据或校验单元运算。由于 R 在对应 $\hat{S}$ 内为 0 位置的行位置上有 0，这样的一个公式是不能依赖任何丢失的数据或校验单元的。那么异或公式指出已知数据和校验单元的特定异或值等于与那列相关联的数据值。这就是说，这样的一列为重构提供了一个公式。这些证明了定理中第一个声明的其余部分。定理的第二个声明是很明显的。

强调一下，这个定理在关于失效扇区的位置上没有做假设，不管它们是相关还是不相关。因此，这个定理能够应用于全部磁盘/条块(高度相关)丢失的情况，甚至是在每个条块上有一个扇区丢失的情况(高度不相关)。它同样不依赖于纠删码的任何特殊结构。我们需要的所有信息都被嵌在生成矩阵内。

回忆一下，R 不必是唯一的，像式(1)一样，给定 $\hat{G}$ 的零空

间的一个基是很容易去构造其它的伪逆，这些其它的伪逆满足像在定理中 R 一样的性质。在下一节，我们讨论构造伪逆的方法和零空间的基。我们使用零空间基改善伪逆的稀疏性。

4.1 伪逆构造

对于计算伪逆和零空间基有很多可能的算法。虽然数据结构和方法稍微不同，但是从根本上来说它们是相同的。

从现在开始，用标签 B 来表示一个矩阵，矩阵的列组成了一个针对一些零矩阵$\hat{G}$的零空间基。此外，由于只涉及不相关的扇区丢失，因此忽略 G 的块结构。作为结果，在不丧失一般性的情况下能够假定生成矩阵 G 在开始的 N 列中有系统化的单位子阵，在最右边的 Q 列中有校验列——称之为“左系统化”(假如不是的话，G 的列的置换和在$\hat{G}$、S 和$\hat{S}$中对应列的位置以及 B，H 和 R 中对应的行的位置将迫使变为这种情况)。

算法的输入是原始的生成矩阵 G(和/或校验矩阵 H)和一列数据或校验单元 F，F 被宣布为条带内的丢失(不可恢复)。

算法的输出将是两个矩阵，R 和 B。R 是$\hat{G}$的一个伪逆(通过将对应 F 中单元的 G 的列零化获得的)，B 是$\hat{G}$的一个零空间基。

算法用列操作和/或行操作来处理矩阵。列操作相当于通过简单的矩阵右乘法(对于行操作，操作符在左边)。考虑了 3 种简单的列(或者行)操作。

- 交换：将两个列(或者行)交换；
- 和运算并替换：将列 c 加到列 d(以 2 为模)，并用和替换掉列 d(对于行操作也是一样的)；
- 置零：将一个列(或者行)的所有元设置为 0。

前两个操作是可反向操作的(可逆的)，零化操作是不可逆的。

首选的算法称之为“列增”结构，能够被看成一个动态的或者在线算法。当新近丢失的扇区被检测到时逐渐地更新数据结构(通过在 F 中单元的增量处理来模拟)。

4.1.1 列增结构

本文的算法是一个增量算法。它以一个伪逆和矩阵 G 的零空间基开始，并递增地移除(模拟)一个丢失数据或校验单元，且在每一步维持伪逆和零空间基的性质。此算法是空间有效的，并且对于大多数设计良好的编码来说操作较少。此算法需要 R 中的空间只用于丢失的数据单元。只要伪逆是行满秩，这个处理是可逆的。也就是说，在任一步骤中，为丢失单元的数据重构建模是可能的，并且计算一个新的伪逆和零空间基等于一个矩阵，在这个矩阵里面，已恢复的单元是从未丢失过的。

在这个算法中，列操作是在一个工作空间矩阵上执行的。丢失的数据或校验单元检索 R 和 B 的一行。

算法 列增结构

(1)构造一个 $N+Q$ 维方形工作矩阵 W。首先是 N 列和行，放置一个单位矩阵。最后是 Q 列，放置校验矩阵 H。让 R 表示首先的 N 列，B 表示最后的 Q 列，那么 $W=(R|B)$，这里初始时 $B=H$，$R=(I_N,0)$。

(2)对于 F 中每一个丢失单元，让 r 表示对应于丢失单元的行。执行如下的操作：

a)找到在 B 中的任意列 b，列 b 在行 r 中有一个 1。假如列 b 在行 r 中不存在一个 1 的话，零化 R 中在行 r 中有一个 1 的任意列，然后继续下一个丢失单元(注意：零化这些列就是零化 W 中的整个行 r)。

b)将列 b 与 W(R 和 B 的共同部分)的每一列 c 和运算并替换，W 的行 r 中有一个 1。

c)零化 B 中的列 b，相当于将列 b 加到他自身。然后继续下一个丢失的单元，直到 F 列表已经处理完为止。

(3)(可选项)用 B 的列来提高 R 的非零列的权重(到目前为止对应丢失数据单元的处理)。

(4)输出 R(W 的开始 N 列)和 B 的非零列(来自于 W 的最后 Q 列)。

4.1.2 伪逆改进

在本节中，我们对实现在上面给定的列增构造算法的优化步骤(3)的一些方法加以概述。

下面的方法提供了找到一个最优 R 的系统方法。

算法 改进 R

(1)计算所有的零空间向量(通过采用所有可能的基向量子集的和)。

(2)对每一个 R 的非单位(和非零)列做如下操作，对每个零空间向量(从第一步)做如下操作：

a)将零空间向量加到 R 的列上生成一个新的公式。

b)假如生成的公式有较低的权重，那么用 R 替换掉它。

(3)结束。

当然，假定零空间有足够小的基集，这是唯一实际的。假定零空间基有非常少的向量，那么这个算法提供了一个穷举搜索办法来找到一个最佳的 R。一般而言，能用完全零空间的任意子集来找到更好的伪逆(在上面的第一步，只计算零空间的一些子集)，但或许不是最佳的。一个简单的选择是只用基向量自己，或者用基向量和所有成对的和。

有可供选择的构造能够应用于计算伪逆。它们之中是一个行增变量，这个变量类似上述的列增方法，但是使用行操作代替列操作。大多数步骤与列增构造的操作相同。在(2)b 步骤上，对于所选择 B 的 b 列中在 $s\neq r$ 位置上的每个 1，和运算并将行 r 替换 B 的行 s；在 R 中镜像这个操作。在(2)c 步骤中，零化 B 中的行 r，然后继续下一个丢失单元的处理。这个算法有与列变量同样的性质(包括可逆性)，但是，典型的是非常耗时，且要求更多的行操作。

作为可选择的办法，是既有关于列的版本，也有关于行的版本，它们与计算逆的经典算法平行。也就是以两个矩阵开始：原始的生成矩阵和一个 $(N+Q)$ 单位矩阵。零化生成矩阵的列对应每一个丢失数据和校验单元的单位矩阵。在已修改的生成矩阵上执行列(或行)操作，将矩阵转换为列(或行)的约化梯形。在单位矩阵上平行这些操作的每一个操作，结果矩阵既包含伪逆又包含零空间基。由于它们利用第一步中丢失单元的完整集，因此这些变化是静态、离线构造 。跟以前一样，列版本的计算或多或少会少一些。

4.1.3 列增构造的逆

如上所述，增量处理能够用于以一个完全的在线条带开始，一步一步地，当在条带内检测到媒体错误时，维护一组针对条带内每一个数据单元的重构公式(或者是非重构声明)。

当检测到新的媒体错误时，矩阵就被更新，并且产生新的公式。

假定阵列已经有了一些媒体错误，但是没有数据丢失事件，并且假定一个数据单元通过它在 R 中的公式被重构了。假如这个重构的数据在条带内被替换了，它对于更新反映这点的公式将是有帮助的。对于这个问题有两个原因。首先，知道能够通过一个单位列在 R 中替换这个公式（不再需要旧的公式）。第二它可能是这样的一种情况：其它丢失单元能够通过更好的公式重构，这个公式包含了最新的被重构的单元。我们应当更新 R 来反映这个事实。

一个方法将用任意算法来重新计算对于修订的丢失扇区集的公式。但是，增量算法建议可能有能力去逆操作这个处理。也就是说，直接去更新 R 和 B 来反映数据单元已经被重构的事实（例如它在 R 中的列用一个单位列来替换）。

要完全地反转前面章节的这种增量结构，必须是这种情况：在通过的每一步不能有信息丢失。算术上，只要我们执行一个不可逆的矩阵操作，这个就会发生，也就是由一个不可逆矩阵对应乘法。本质上这种情况只发生在构造中的一个位置：无论什么时候，我们在零空间基不能找到在所要求的行中有一个 1 的向量。这刚好对应于有数据丢失事件的这种情况。

因此，我们有下面的结果：只要我们从未遇到数据丢失分支，那么原则上步骤的顺序能够颠倒。但是，我们在下面给出的算法在数据丢失事件之后起作用，只要已恢复的单元在 R 中有一个重构公式，也就是说它本身不是一个数据丢失事件。显然，考虑恢复对应数据丢失事件的一个单元到矩阵几乎没有意义（定理规则表示这在理论上是不可能的）。

下面的算法在一个（可恢复）数据单元情况下执行这个增量恢复步骤。这个算法的输入是一个工作空间矩阵 $W=(R|B)$，（可能）由增量算法生成，并且有下面的性质：

$$G \cdot W=(I_N|0)$$

这里 $\hat{G}$ 是生成矩阵，带有针对在假定丢失单元 F 集里的每一个数据和校验单元的零化列。其它的输入是一个数据单元元素，即一个 W 的行数 $r \leqslant N$。算法的输出是一个修订的矩阵 W，所以上面的公式赞同 $\hat{G}$ 以适当的单位列代替列 r。这个新的矩阵 W 在位置 r 将有一个单位列（像以前一样，这个算法不直接记录 $\hat{G}$ 的改变）。注意这个处理不依赖于哪一个单元正在从增量期间被移除的单元集中恢复（它不需要是最后被移除的单元）。假定 B 包含有足够的全为 0 的列，所以它总共有 Q 列。

假如恢复的单元不是来自 F 集，那么这个算法就无事可做，因此我们假定丢失的单元是来自 F 的。

算法 增量结构的逆

（1）（可选）对于每一行有一个 1 的 W（开始的 N 列）的逆部分每一列 c，列 r 有（就是说，假如列 c 和列 r 的与操作等于列 r）如下操作：

a）将列 r 加到列 c 上，并用和替换列 c；即对于有一个 1 的列 r 的每个位置，在列 c 将相应的值设置为 0。

b）将列 c 中位置 r 的值设置为 1。

（2）在 W（在最后的 Q 列中）的零空间部分找到任意全零的列 b。

（3）将 W 中位置 (r,r) 和 (r,b) 上的值设置为 1。

（4）交换 W 的列 r 和列 b。

（5）（可选）用在 W 的 B 中的零空间基向量来减小在 W 的逆部分 R 中任意列的权重。

（6）返回已更新的 W。

这个算法是起作用的，因为它采用数据单元的重构公式并将其展开回零空间基，然后用一个单元列替换这个公式。

第一个可选步骤通过数据单元本身替换了在原始 W 中数据单元 r 的公式的任意事件。特别是它清楚地恢复到其它列依赖已恢复的数据单元。在这个处理中，它提高了这些公式的权重。

当单位矩阵被看到往前走时，这个算法没有必要完全逆反增量算法，因为它没有必要引起单位矩阵往回走。但是，这个差异似乎在零空间中。

4.1.4 恢复校验单元

为了将校验单元加回到矩阵，需要有来自生成矩阵 G 的原始校验列（对于数据列，知道一个演绎，就是此列是一个单元列，因此从外部看，不需要去明了这个）。假定这个校验是有 G 中的 c 列来索引的。

取得这个校验列，对于列中的每个 1，将其与 W 中对应 R 的列加起来（以 2 为模），并把结果放到 W 中 B 的一个全零列中（由于只有一个那样的列，这正好是对数据单元所作的）。用 1 替换这个新列的位置 c 上的 0。用这个校验列（恢复它）替换 G_0 的列 c（这刚好是对一个已恢复的数据列所作的，除了我们不得不设置 W 逆反部分的 (r,r) 位置为 1（在一个校验列情况下），在逆反部分没有那样的位置存在，因此这一步被忽略）。

结束语 在任意的阵列码中，为不相关和/或相关的扇区丢失建模，提供了一个直接方法和有建设性的算法来实现针对这些丢失扇区的可恢复性以及不可恢复性的一个通用和完整的解决方法。这个方法和算法满足了用户的约定，就是说理论上可恢复的将是可以恢复的。解决方法能够增量地或静态地被应用。相比纠删码的容错机制，通过当这些扇区分布到条带内的更多条块时怎样恢复丢失扇区的数据证明了直接方法的能力。

直接方法能够与编码制定的递归算法一起来定位部分条块数据有效重构的问题。作为选择，当一些数据被恢复时，增量方法能够被逆反来提供定位这个同样部分条块恢复问题的一个非常一般的方法。最后，证明了直接和递归的混合方法的重要性。

参 考 文 献

[1] Patterson D A, Gibson G A, Katz R H. A case for redundant arrays of inexpensive disks(RAID)[C]// Proceedings of the International Conference on Management of Data(SIGMOD). June 1988

[2] Zaitsev G V, Zinovev V A, Semakov N V. Minimum-check-density codes for correcting bytes of errors[J]. Problems in Information Transmission, 1983, 19: 29-37

[3] Deenadhayalan V, Hafner J L, Rao K K, et al. Matrix methods for lost data reconstruction in erasure codes[R]. RJ 10354. IBM Research, San Jose, CA, 2005

（下转第 294 页）

出“導播”二字。本文算法对两种颜色的文本都能较好处理，识别结果正确。图 1(c)中前景文本比较模糊，背景比较复杂。文献[3]算法分割结果比较残缺，导致中间几个字识别错误；本文算法分割结果比较完整，识别结果基本正确。从图 1 可以看出本文算法分割文本比较准确。

为了定量评价本文的算法，采用字符识别率作为评价指标。字符识别率(CRR)是 OCR 软件正确识别的字符数与文本行图像中所有字符数的百分比。设字符识别率为 CRR，OCR 软件正确识别的字符数为 N_1，文本行图像集合中所有字符数为 N，则

$$CRR=\frac{N_1}{N} \tag{7}$$

N_1 和 N 是从实验中由人工统计而来的。

表 1 比较了本文算法在使用不同特征时的字符识别率，由表 1 可知在对图像进行增强预处理后，同时使用颜色和笔画特征时可取得最高的识别率。

表 1　本文算法在使用不同特征时字符识别率的比较

特征向量	中文 CRR	英文 CRR	综合 CRR
颜色(未增强)	70.3%	72.4%	71.6%
颜色＋笔画(未增强)	78.2%	80.1%	79.3%
颜色(增强)	75.2%	78.3%	76.5%
颜色＋笔画(增强)	83.1%	85.2%	84.7%

表 2 对本文算法和文献[3]算法的结果进行比较，可见本文算法的中文文本识别率明显提高，这得益于本文算法利用了文本的笔画特征。

表 2　本文算法与文献[3]性能的比较

算法	中文 CRR	英文 CRR	综合 CRR
本文算法	83.1%	85.2%	84.7%
文献[3]算法	79.6%	84.5%	82.6%

结束语　本文提出了一种利用颜色和笔画特征的无监督聚类方法来进行复杂背景下的文本分割。首先在对文本行图像增强的基础上，应用颜色约简和直方图确定文本颜色。然后提取颜色的亮度、色度分量和笔画特征，并应用 K-均值聚类算法分割出文本和背景像素。最后应用后处理优化分割结果。利用 OCR 软件对分割结果的字符识别率作为评价指标，试验表明本文算法比已有的一些方法具有更好的性能。下一步工作要将本算法用于基于语义的图像视频检索。

参 考 文 献

[1] Lyu M R, Song J Q, Cai M. A comprehensive method for multilingual video text detection, localization, and extraction[J]. IEEE Transactions on Circuit and System on Video Technology, 2005, 15(2)

[2] Tsai C M, Lee H J. Binarization of color document images via luminance and saturation color features[J]. IEEE Trans. on Image Processing, 2002, 11(4)

[3] Gllavata J, Ewerth R, Freisleben B. Text detection in images based on unsupervised classification of high-frequency wavelet coefficients[C]// International Conference on Pattern Recognition. 2004

[4] Wernicke L R. A Localizing and Segmenting Text in Images and Videos[J]. IEEE Transact. on Circuits and Systems for Video Technology, 2002, 12(4): 256-258

[5] Odobez J M, Chen D. Robust Video Text Segmentation and Recognition with Multiple Hypotheses[C]// Proc. of IEEE International Conference on Image Processing 2002. Vol. II, Rochester, New York, 2002: 433-436

[6] Gllavata J, Ewerth R, Freisleben B. Finding Text in Images via Local Thresholding[C]// Proc. of the 3rd IEEE Int'l Symposium on Signal Processing and Information Technology. Darmstadt, Germany, 2003

[7] 冈萨雷斯. 数字图像处理[M]. 北京：电子工业出版社，2006：233-240

[8] Liu Qifeng, Jung Cheolkon, Kim Sangkyun, et al. Stroke Filter for Text Localization in Video Images[C]// IEEE International Conference on Image Processing. Oct. 2006: 1473-1476

[9] 付慧，刘峡壁，贾云得. 图像中多语种文本提取的高斯混合建模方法[J]. 计算机研究与发展，2007，44(11)：1920-1926

[10] 陈世亮，李战怀，袁柳. 一种基于概念层次的图像检索方法[J]. 计算机科学，2008，35(4)：139-141

(上接第 266 页)

[4] Baylor S, Corbett P, Park C. Efficient method for providing fault tolerance against double device failures in multiple device systems[P]. U. S. Patent 5,862,158. 1999

[5] Blaum M, Brady J, Bruck J, et al. EVENODD: an efficient scheme for tolerating double disk failures in RAID architectures [J]. IEEE Transactions on Computers, 1995, 44: 192-202

[6] Blaum M, Roth R M. On lowest density MDS codes[J]. IEEE Transactions on Information Theory, 1999, 45: 46-59

[7] Hafner J L, Deenadhayalan V, Kanungo T, et al. Performance metrics for erasure codes in storage systems[R]. RJ 10321. IBM Research, San Jose, CA, 2004

[8] MacWilliams F J, Sloane N J A. The Theory of Error-correcting Codes[M]. The Netherlands: Northolland, Amsterdam, 1977

[9] Plank J. A tutorial on Reed-Solomon coding for fault-tolerance in RAID-like systems[J]. Software: Practice and Experience, 1997, 27: 995-1012

[10] Reed I S, Solomon G. Polynomial codes over certain finite fields[J]. Journal of the Society for Industrial and Applied Mathematics, 1960, 8: 300-304

[11] Xu L, Bruck J. X-code: MDS array codes with optimal encoding [J]. IEEE Transactions on Information Theory, 1999, IT-45: 272-276

[12] Corbett P, English B, Goel A, et al. Row-diagonal parity for double disk failure[C]// Proceedings of the Third USENIX Conference on File and Storage Technologies. 2004: 1-14