基于 LZW 的海底声学探测数据压缩方法研究

晋　松[1], 裴彦良[2, 3, 4], 阚光明[2, 3, 4], 吴爱平[1], 付青青[1]

(1. 长江大学 电工电子国家级实验教学示范中心, 湖北 荆州 434023; 2. 青岛海洋科学与技术试点国家实验室海洋地质过程与环境功能实验室, 山东 青岛 266061; 3. 自然资源部第一海洋研究所, 山东 青岛 266061; 4. 海洋沉积与环境地质自然资源部重点实验室, 山东 青岛 266061)

摘要: 针对海底声学探测仪器采集数据量大而存储容量有限、数据传输带宽不足的实际问题, 基于Lempel-Ziv-Welch(LZW)无损压缩算法, 研究海底声学探测数据的实时压缩方法, 提高数据压缩效果、节省传输带宽。并在 LZW 无损压缩算法的基础上结合数据存储的特点对压缩结果进行内存重新分配, 极大提高压缩比(压缩数据大小/原始数据大小)。利用海底地震仪(OBS)采集的原始声学探测数据进行测试验证, 结果表明该方法对于OBS声学探测数据有很好的压缩比, 可用于对OBS采集的声学探测数据进行压缩处理, 对于海底探测仪器的研发有很好的指导意义。

关键词: Lempel-Ziv-Welch(LZW)无损压缩算法; 无损压缩; 海底地震仪; 海底声学探测数据
中图分类号: P716^{+}.83　　**文献标识码**: A　　**文章编号**: 1000-3096(2019)04-0036-10
DOI: 10.11759/hykx20180721001

声学探测方法在海洋地球物理勘探中有着广泛的应用[1-5], 随着探测精度要求的提高, 海底声学探测技术向着多维、多参数、多分量、多节点阵列和高采样率的方向发展, 必然造成采集数据量的急增, 海量数据的存储和传输出现瓶颈, 目前的海底声学探测设备很少有搭载数据压缩模块, 因此, 研究海底声学探测数据实时压缩方法, 快速、高效地进行数据采集和存储是海底声学探测设备研发急需解决的问题之一。

数据压缩技术是指把输入数据流中的冗余信息去除掉, 运用较小的输出数据流保存输入数据流中的全部信息或部分主要信息[6]。常用的声学压缩方法有: 多级树集合分割(SPIHT)算法[7]、Free Lossless Audio Codec(FLAC)算法[8]、Monkeys's Audio(APE)算法[8]、Huffman 算法[9]、Lempel-Ziv-Welch(LZW)压缩方法[9]等。SPIHT 算法在编码和解码过程中不仅需要消耗大量内存且存在多次重复运算, 不利于实时压缩。FLAC 和 APE 算法只能对 WAV 格式的数据进行压缩, 压缩之前要将数据格式进行转化, 这样不能满足实时压缩的需求。Huffman 编码是基于数据源的统计概率特性, 基于统计概率的信源编码是Shannon信息论的基本特征, 但是在某些场合要确定实际信源的统计特性相当困难。

本文采用LZW无损压缩算法对海底声学探测数据进行压缩, 该压缩算法是基于数据串特性, 采用串表压缩的方式, 使其具有较好的自适应性, 对于图像、文本、音频、视频等不同的数据都适用。常见的.txt, .dat, .bin, .jpg, .gif, .doc 等类型数据也都可以采用 LZW 算法来进行压缩[10]。目前对于 LZW 压缩算法的改进, 多于算法本身改进, 如在压缩时监视压缩比, 清除匹配概率较小的词条来提高压缩比[10]、引入 Hash 函数提高字典匹配速率[11]、利用编码前缀时存在的冗余数据, 设计将前缀映射为编码来提高压缩比[12]等等。对于如何减小压缩结果所占用的内存空间, 目前的研究并不是很多。

对声学探测数据, LZW 算法不仅有着很好的压缩效果, 而且易于硬件实现。同时本文在传统 LZW

收稿日期: 2018-07-21; 修回日期: 2018-09-18
基金项目: 青岛海洋科学与技术国家实验室"问海计划"(2017WHZZB0401); 青岛海洋科学与技术国家实验室海洋地质过程与环境功能实验室开放基金(MGQNLM-KF201705); 国家自然科学基金(41527809); 泰山学者工程专项经费(TSPD20161007); 长江大学电工电子国家级实验教学示范中心开放课题(SFZX2018015)
[Foundation: Wenhai Program, No.2017WHZZB0401; Laboratory for Marine Geology, Qingdao National Laboratory for Marine Science and Technology, No.MGQNLMKF201705; National Natural Science Foundation of China, No. 41527809; the Research Fund for the Taishan Scholar Projectof Shandong Province, No. TSPD20161007; National Demonstration Center for Experimental Electrical & Electronical Education, Yangtze University, Jingzhou, Hubei, China, No.SFZX2018015]
作者简介: 晋松(1995-), 男, 安徽滁州人, 硕士, 现在主要从事海洋信息技术研究, 电话: 18607216817, E-mail: 365698588@qq.com; 吴爱平, 通信作者, wuaping@yangtzeu.edu.cn

压缩算法的基础上提出对了对压缩结果进行内存重新分配的方法，极大提高了压缩比，对于海底声学探测仪器的研制具有重要的意义。

1 LZW 算法

1.1 LZW 算法原理

LZW 是 Terry A.Welch 提出的一种基于字典的压缩算法，所谓字典算法，即认为信源输出的信息序列是由一本字典中的各种词条构成的。LZW 压缩有 3 个重要的对象，分别是数据流、编码流和字典。在编码时，数据流是输入对象，编码流就是输出对象，在解码时，编码流则是输入对象，数据流是输出对象[13]。而字典是在编码和解码时都须要用借助的对象，由词条及其索引组成。LZW 压缩算法的基本思想是建立字典，将输入字符串映射成定长的码字(字典的索引)进行输出[14]。该字典是根据信源的数据特点动态建立，编码过程也就是建立字典的过程。压缩过程中，字典中不断产生新字符串(字典中没有的字符串)，存储新字符串时也保存与新串的前缀相应的码字[15]。解码的过程也是不断建立字典的过程，只有解码建立的字典与编码过程中所建立的字典完全一致才能保证压缩解压过程中没有出现错误。

LZW 压缩算法和其他一些压缩技术的不同之处在于它是动态地标记数据流中出现的重复数据串[11]。它把在压缩过程中遇到的字符串记录在字典中，在下一次又碰到这一字符串的时侯，就用一个代码来表示它，通过用短代码表示相对较长的字符串来压缩数据量。

1.2 LZW 压缩流程

在编码开始之前，初始化字典，字典中包含信源所有可能的根，并为字典索引设置上限，随着数据的读入，编码进行的同时也不断扩充字典，一旦字典扩充至索引上限，说明字典已经填满，就清空字典扩充的部分并重新根据编码数据扩充字典，如此循环，直至所有数据全部编码完成。

LZW 压缩具体流程如图 1 所示。

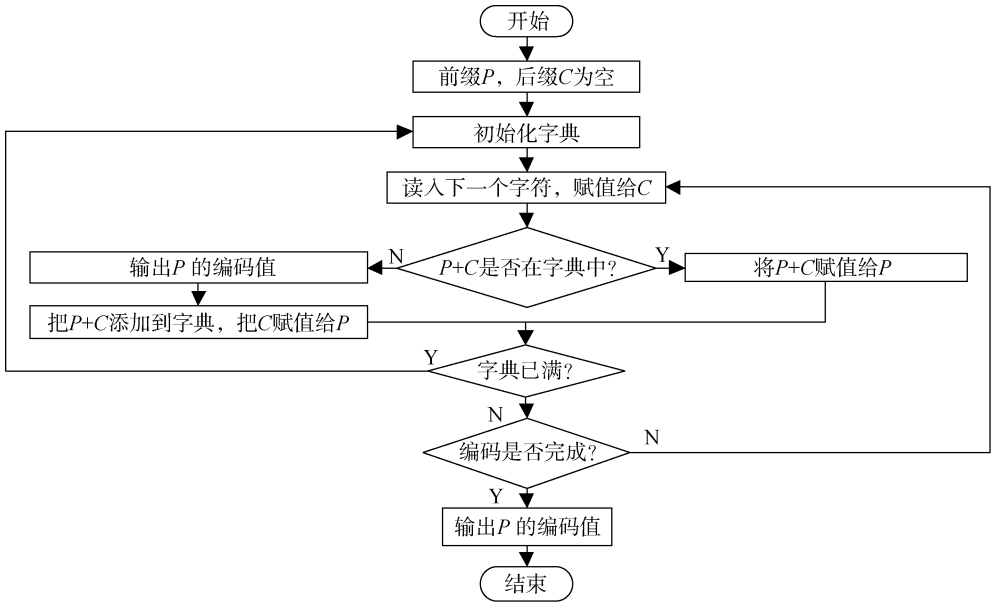

图 1　LZW 算法压缩流程图

Fig. 1　Flowchart of Lempel-Ziv-Welch (LZW) algorithm compression

以声学探测数据为例，编码开始之前，初始化字典时，字典的 0—255 号索引对应的位置用于存放十六进制声学探测数据所有可能的取值，即 0—FFH，同时设置字典的索引上限，索引大于 255 的位置用于存放字典扩充的数据。每次读入的数据赋值给后缀 C，判断前缀 P 与后缀 C 的组合 $P+C$ 在字典中是否存在，如果不存在，输出前缀 P 所对应的编码值，也就是字典中的位置(类似于现实字典中的页码)并将该组合加入字典，同时将后缀 C 的值赋给前缀 P。反之，如果字典中存在 $P+C$，将该组合的值赋给前缀 P 即可[16]。

在 LZW 进行编码的同时需要检测字典的索引是否已经达到设定值。如果字典的索引过大，查找数据的效率就会很低，因此有必要为字典的索引设置上

限，如果达到索引的设定值，表明字典填满，清空字典扩充的部分(256 号索引之后的部分为扩充部分)，保留前 256 个基础值，以此来提高编码的效率。这种为字典设置索引上限的方法虽然可以加快压缩速率，但是压缩比(压缩数据大小/原始数据大小)会有所降低。在 LZW 算法中，原始数据中子串重复的越多，压缩效果越好，原始数据量越大，压缩效果也越好，如果不为字典索引设置上限，字典索引就会过大，会导致压缩的速率降低，这就意味着需要在压缩速度与压缩比之间做好取舍的平衡。在压缩比可以接受的情况下，尽可能地提高压缩速率。

1.3 LZW 解压流程

在进行 LZW 解压时，同样需要建立字典并初始化，这一环节与压缩时的字典建立及初始化完全相同。但建立字典的过程相对于压缩时要推迟一步，这是因为建表所需要的未匹配字符要在解压缩下一个输入码字后才可得到。通过查找输入数据在字典中有无编码值进行解压。

解压的具体流程如图 2 所示。

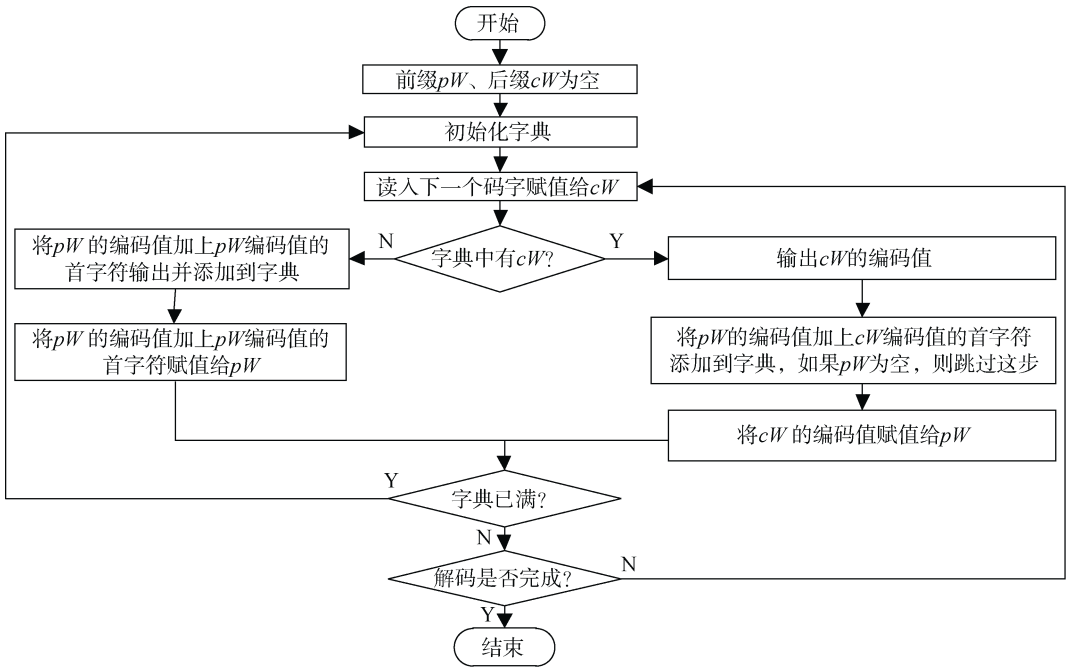

图 2　LZW 算法解压流程图

Fig. 2　Flowchart of LZW algorithm decompression

该环节利用压缩过程中生成的压缩文件进行解压，初始化字典与压缩环节一致，字典的 0~255 所对应的位置用于存放十六进制声学探测数据所有可能的取值，字典的索引上限也要一致。解压过程中读入的码字 *cW* 代表字典的索引位置，如果 *cW* 大于字典当前的已经填充的最大索引，则说明字典的索引还没有扩充到 *cW*，将前缀 *pW* 的编码值(字典中对应的字符串)加上 *pW* 编码值的首字符输出并添加到字典并赋值给 *pW*。如果 *cW* 小于字典当前的已经填充的最大索引，输出 *cW* 的编码值，并将 *pW* 的编码值加上 *cW* 编码值的首字符添加到字典同时将 *cW* 的编码值赋值给 *pW*。解压完成时，会生成一个和压缩环节完全一样的字典[16]。

2 LZW 压缩算法的软件实现

2.1 算法实现平台

本算法在 Windows 平台用 C 语言实现，采用的待压缩数据是 24 位 A/D 转换器采集的 OBS 原始声学探测数据，在用文本文件存储数据的情况下，压缩比较好，但是以二进制文件进行存储，压缩比较差，有可能出现压缩后的数据所占用的内存空间大于原始的待压缩数据，针对这一情况，本文提出了对压缩结果进行内存重新分配的方式来提高压缩比。

2.2 LZW 压缩方法的实现

以二进制格式存储的数据为例来说明压缩环节的具体实现步骤，采用的待压缩数据为 OBS 采集的

实际声学原始数据中截取的一小部分: 0x001974, 0x00455e, 0x0060c9, 0x006c41, 0x006daa, 0x006ae1, 0x006773, 0x006492, 0x006235, 0x006042, 0x005ef9, 设定的字典索引上限为 512。LZW 算法的思想是用单个索引尽可能地去表示更多的字符, 信源数据的重复性越好, 压缩的效果就越好, 为了提高数据的重复性, 将采集的数据按高位、中位和低位进行划分并分组, 原始的数据就变为: 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x19, 0x45, 0x60, 0x6c, 0x6d, 0x6a, 0x67, 0x64, 0x62, 0x60, 0x5e, 0x74, 0x5e, 0xc9, 0x41, 0xaa, 0xe1, 0x73, 0x92, 0x35, 0x42, 0xf9。压缩流程如表 1 所示(表中 *P*, *C* 分别表示压缩过程中, 字符串的前缀和后缀, 其值随着压缩的进行不断改变)。

表 1　压缩流程
Tab. 1　Compression process

步骤	*P*	*C*	*P*+*C*	字典中是否存在	输出	字典扩充内容	对应索引
1	-	0	0	是	-	-	-
2	0	0	0, 0	否	0	0, 0	256
3	0	0	0, 0	是	-	-	-
4	0, 0	0	0, 0, 0	否	256	0, 0, 0	257
5	0	0	0, 0	是	-	-	-
6	0, 0	0	0, 0, 0	是	-	-	-
7	0, 0, 0	0	0, 0, 0, 0	否	257	0, 0, 0, 0	258
8	0	0	0, 0	是	-	-	-
9	0, 0	0	0, 0, 0	是	-	-	-
10	0, 0, 0	0	0, 0, 0, 0	是	-	-	-
11	0, 0, 0, 0	0	0, 0, 0, 0, 0	否	258	0, 0, 0, 0, 0	259
12	0	19	0, 19	否	0	0, 19	260
13	19	45	19, 45	否	25	19, 45	261
…	…	…	…	…	…	…	…

注: “-”表示无数据, 表 2 同。

表 1 给出了待压缩数据的部分压缩流程, 采用上述 LZW 算法的编码方式, 所有待压缩数据压缩完成得到的输出结果为: 0, 256, 257, 258, 0, 25, 69, 96, 108, 109, 106, 103, 100, 98, 96, 94, 116, 94, 201, 65, 170, 225, 115, 146, 53, 66, 249。

2.3　压缩结果的内存重新分配

在压缩环节中, 由于字典的索引上限为 512, 压缩输出结果用二进制表示, 不会超过 9 位, 因此用 9 位二进制数据就可以表示所有可能的输出结果, 但由于存储器都是以 8 位数据为基本存储单元, 输出的每个数据占了 2 个字节(16 位), 其中高 7 位都是 0, 但是这高 7 位是无用的。为了去除多余的高 7 位, 用 9 位来表示输出结果, 这里以 8 个输出为一组, 每次向存储器申请 9 个字节的空间, 其中 2 到 9 个字节存放 8 个输出数据的低 8 位, 第一个字节存放 8 个输出数据的高 1 位。以上述输出结果的前 8 个数据为例, 内存分配如图 3 所示。

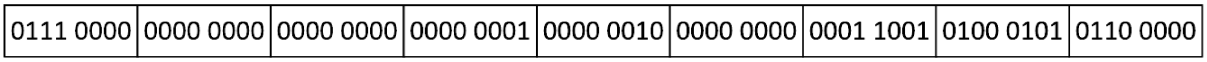

图 3　内存分配图
Fig. 3　Memory allocation chart

每个方框代表一个字节, 2 到 9 个字节是输出数据的低 8 位, 第一个字节的中的 8 位数据从左到右分别代表 2 到 9 字节中数据的高位。这 9 个字节中存储的数据用二进制表示分别为: 0 0000 0000, 1 0000 0000, 1 0000 0001, 1 0000 0010, 0 0000 0000, 0 0001 1001, 0 0100 0101, 0 0110 0000。换算成十进制为别为: 0, 256, 257, 258, 0, 25, 69, 96。若数据不足 8 个, 将 9 个字节中有效部分输出即可。

上述内存分配的方式实现了用 9 个字节表示 8 个输出数据, 在原始待压缩数据中, 每个数据用一个

字节表示，所占用的内存为 264 bit，压缩之后所占用内存为 243 bit。随着数据量的不断提高，压缩比会越来越好。

2.4 LZW 解压方法的实现

解压过程中，每次读取 9 个字节的数据量，按照压缩环节的内存分配原理，将首个字节中的 8 位数据，依次按顺序分配给剩余 8 个字节的数据，作为其高位。将经过内存分配后的数据还原，再利用还原后的数据进行解压。以压缩环节的输出数据为例，将经过内存分配的数据还原，得到结果：0, 256, 257, 258, 0, 25, 69, 96, 108, 109, 106, 103, 100, 98, 96, 94, 116, 94, 201, 65, 170, 225, 115, 146, 53, 66, 249。将这组数据按照解压流程进行解压，解压流程如表 2 所示(表中 *pW*, *cW* 分别表示解压过程中，字符串的前缀和后缀，其值随着解压的进行不断改变)。

表 2 解压流程
Tab. 2 Decompression process

步骤	*pW*	*cW*	字典中是否存在	输出	字典扩充内容	对应索引
1	-	0	是	0	-	-
2	0	256	否	0, 0	0, 0	256
3	0, 0	257	否	0, 0, 0	0, 0, 0	257
4	0, 0, 0	258	否	0, 0, 0, 0	0, 0, 0, 0	258
5	0, 0, 0, 0	0	是	0	0, 0, 0, 0, 0	259
6	0	25	是	19	0, 19	260
7	19	69	是	45	19, 45	261
8	45	96	是	60	45, 60	262
9	60	108	是	6c	60, 6c	263
10	6c	109	是	6d	6c, 6d	264
11	6d	106	是	6a	6d, 6a	265
12	6a	103	是	67	6a, 67	266
13	67	100	是	64	67, 64	267
…	…	…	…	…	…	

表 2 给出了待解压数据的部分解压流程，所有数据解压完成之后得到解压输出为：0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x19, 0x45, 0x60, 0x6c, 0x6d, 0x6a, 0x67, 0x64, 0x62, 0x60, 0x5e, 0x74, 0x5e, 0xc9, 0x41, 0xaa, 0xe1, 0x73, 0x92, 0x35, 0x42, 0xf9。由于在压缩过程中，将 24 位声学探测数据分为了高位、中位和低位分类压缩，因为在解压时需要将数据还原成 24 位的声学数据，首先将数据变为没有按位分组的排列顺序：0x00, 0x19, 0x74, 0x00, 0x45, 0x5e, 0x00, 0x60, 0xc9, 0x00, 0x6c, 0x41, 0x00, 0x6d, 0xaa, 0x00, 0x6a, 0xe1, 0x00, 0x67, 0x73, 0x00, 0x64, 0x92, 0x00, 0x62, 0x35, 0x00, 0x60, 0x42, 0x00, 0x5e, 0xf9。再将数据按高位、中位和低位进行合并，就得到了原始的声学探测数据：0x001974, 0x00455e, 0x0060c9, 0x006c41, 0x006daa, 0x006ae1, 0x006773, 0x006492, 0x006235, 0x006042, 0x005ef9。

3 压缩算法的性能测试

3.1 压缩比和压缩时间测试

压缩算法的性能测试环节在 Inter (R) Core (TM) i7-3537U CPU，主频 2.0~2.5 GHz, 4G 内存的计算机平台上，基于 OBS 声学探测数据进行测试。压缩结果以二进制格式进行存储，为了更好的反映压缩效果，利用 OBS 声学探测数据中截取了一部分重复性较好数据与没有截取的数据进行测试，分别从文本文件角度和二进制文件角度测试记录压缩比以及压缩时间，测试结果分别如表 3—表 6 所示。

表 3—表 6 中所有的第一个数据源中数据个数都为 3 751，第二个数据源中数据个数都为 7 502。表 3、表 4 中采用的数据存储方式分别以文本形式和二进制形式进行存储，第一个数据源是从 OBS 采集的数据中截取的重复性较好的数据，第二个数据源是将前者重复了一遍，从不同的数据量来看压缩效果，可以看出，对 LZW 压缩结果进行内存的重新分配，压缩比有很大的提升，同时不会消耗太多运行时间，相比之下，此种压缩方式要明显优于传统的 LZW 压缩算法。表 3—表 6 的不同之处在于，前者使用的是 OBS 原始探测数据中重复率较高的数据，后者使用的数据是 OBS 原始探测数据。同样可以看出在 LZW

压缩算法的基础上对压缩结果进行内存重新分配，　压缩比会大幅度提高。

表 3　抽样数据压缩对比表(文本)
Tab. 3　Comparison of sampling data compression (text file)

数据源	内存是否重新分配	数据占用内存/Byte		压缩比/%	压缩时间/s
		原始数据	压缩后数据		
ObsDataSamp.txt	是	60 868	8 428	86.15	0.471
	否	60 868	14 982	75.39	0.412
ObsDataSamp2.txt	是	121 739	16 805	86.20	0.524
	否	121 739	29 874	75.46	0.426

注: ObsDataSamp.txt 是以文本形式存储的 OBS 声学探测数据中截取的一部分重复性较好的数据，数据量为 3751；数据 ObsDataSamp2.txt 是将前者数据重复一遍，得到 2 倍的数据量，即 7502，表 5 同。

表 4　抽样数据压缩对比表(二进制)
Tab. 4　Comparison of sampling data compression (binary)

数据源	内存是否重新分配	数据占用内存/Byte		压缩比/%	压缩时间/s
		原始数据	压缩后数据		
ObsDataSamp.dat	是	11 253	8 428	25.10	0.471
	否	11 253	14 982	–33.14	0.412
ObsDataSamp2.dat	是	22 506	16 805	25.33	0.524
	否	22 506	29 874	–32.74	0.426

注：数据 ObsDataSamp.dat、数据 ObsDataSamp2.dat 分别与表 3 中数据 ObsDataSamp.txt、数据 ObsDataSamp2.txt 中数据内容相同，存储方式为二进制方式，表 6 同。

表 5　无抽样数据压缩对比(文本)
Tab. 5　Comparison of unsampled data compression (text file)

数据源	内存是否重新分配	数据占用内存/Byte		压缩比/%	压缩时间/s
		原始数据	压缩后数据		
ObsDataHex.txt	是	61 077	9 076	85.14	0.471
	否	61 077	16 134	73.58	0.453
ObsDataHex2.txt	是	122 157	18 142	85.15	0.647
	否	122 157	32 252	73.60	0.517

表 6　无抽样数据压缩对比(二进制)
Tab. 6　Comparison of unsampled data compression (binary)

数据源	内存是否重新分配	数据占用内存/Byte		压缩比/%	压缩时间/s
		原始数据	压缩后数据		
ObsDataHex.dat	是	11 253	9 076	19.35	0.471
	否	11 253	16 134	–43.38	0.453
ObsDataHex2.dat	是	22 506	18 142	19.39	0.647
	否	22 506	32 252	–43.30	0.517

上述测试说明LZW算法的压缩效果与数据源特性以及数据的存储方式有很大关系。数据源的重复特性越好，压缩的效果越好，并且对文本文件的数据压缩比要远远优于二进制文件，原因在于文本数据是以字符为单位分配存储空间，而二进制文件是以数据的大小为单位分配存储空间。OBS 采集的原始声学数据以二进制形式进行存储的情况下，如果不对压缩结果进行内存重新分配，压缩过后会出现压缩结果大于源文件的情况，必须采用内存的重新分配。同时从 4 个表中可以看出在数据量远超字典最大索引值的情况下，数据量的大小对压缩比影响较小。

3.2 压缩算法正确性验证

LZW 算法为无损压缩算法，在压缩过程中没有精度的损失，被压缩的数据能够通过解压恢复到压缩以前的原状态。为了保证整个压缩解压环节的正确性，对原始数据以及解压后的数据进行比对，两者完全一致，保证整个压缩解压环节没有错误。

分别将原始声学数据和解压缩得到的声学数据转化成电压值并绘制波形图，比对两者的波形，波形一致，则算法正确，反之则错误。为了避免累赘重复，这里给出三组数据的压缩验证对比图片，包含不同压缩方式以及不同原始数据的压缩验证，同时以 OBS 原始声学探测数据为例，通过 txt 格式给出了部分原始数据和解压数据的数据对比截图。结果如图 4—图 7 所示:

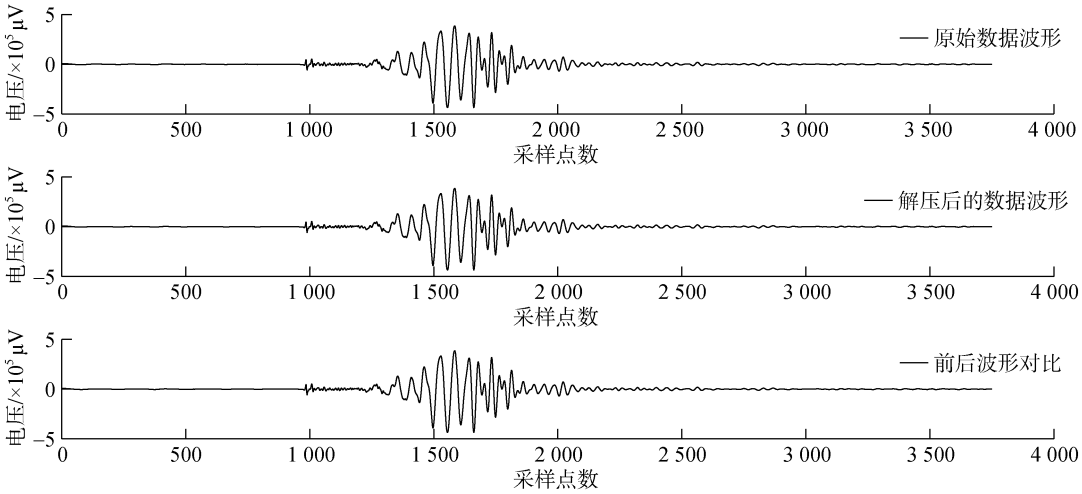

图 4　OBS 原始声学探测数据波形对比图

Fig. 4　Comparison of OBS original acoustic detection data waveforms

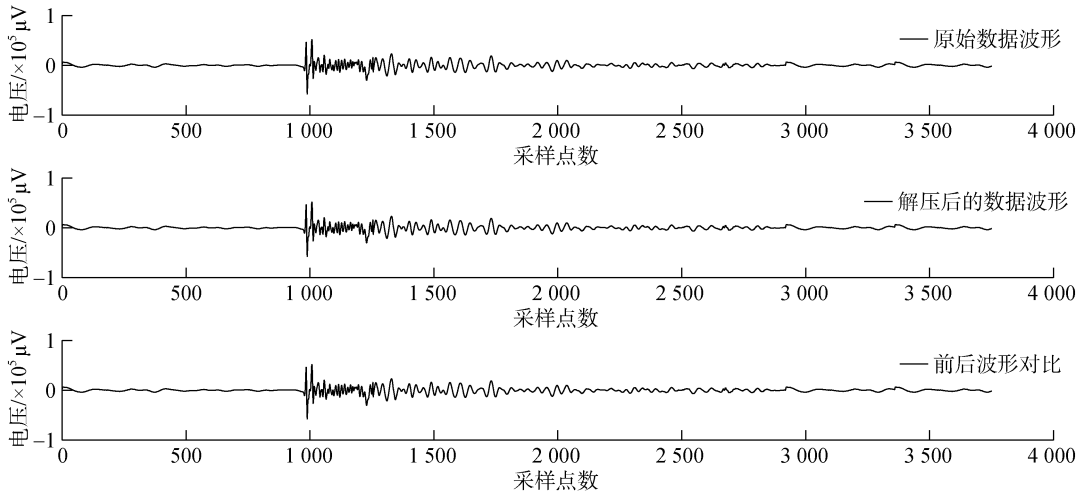

图 5　OBS 部分声学探测数据波形对比图

Fig. 5　Comparison of OBS partial acoustic detection data waveforms

图 4 是从 OBS 原始数据压缩并进行内存分配的压缩验证图，图中原始数据波形是根据压缩之前的数据绘制的波形图，解压后的数据波形是根据解压得到的数据绘制的波形，前后波形对比是将前两个波形绘制在同一坐标系中。在前后波形对比中由于两条波形完全重合，只能看到一条波形，说明了压缩数据和解压后的数据完全一致。图 5 中原始数据与图 6 相同，都是从 OBS 原始数据中截取的部分重复性较好的数据，但是前者采用了在 LZW 压缩算法的基础上进行了内存分配操作，后者是基于常规的 LZW 压缩算法，不含内存重新分配。从图 5、图 6 可以看出原始数据和解压后的数据都是完全重合，

说明了两种压缩方式都是正确的。图 7 中左侧 ObsDataHex.txt 数据为 OBS 原始数据中的部分数据，压缩之后对压缩结果进行内存重新分配，再对压缩结果进行解压，得到图中右侧 Obsjieya.txt 数据。可以看出两组数据完全一致，在压缩算法正确的基础上表明了压缩算法的无损特性。

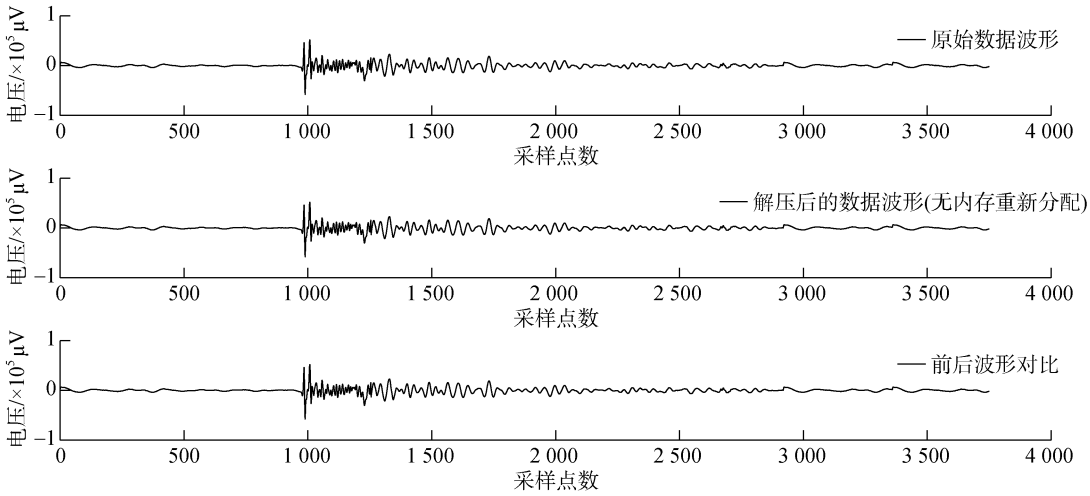

图 6　OBS 部分声学探测数据无内存重新分配操作波形对比图

Fig. 6　Comparison of the waveforms of OBS partial acoustic detection data without memory redistribution operation

ObsDataHex.txt - 记事本

文件(F) 编辑(E) 格式(O) 查看(V) 帮助(H)

```
0x0,0x19,0x74,
0x0,0x45,0x5e,
0x0,0x60,0xc9,
0x0,0x6c,0x41,
0x0,0x6d,0xaa,
0x0,0x6a,0xe1,
0x0,0x67,0x73,
0x0,0x64,0x92,
0x0,0x62,0x35,
0x0,0x60,0x42,
0x0,0x5e,0xf9,
0x0,0x5e,0xb0,
0x0,0x5f,0x65,
0x0,0x60,0x91,
0x0,0x61,0x6b,
0x0,0x61,0x61,
0x0,0x60,0x67,
0x0,0x5e,0xd5,
0x0,0x5d,0x19,
0x0,0x5b,0x6a,
0x0,0x59,0xb5,
0x0,0x57,0xbb,
0x0,0x55,0x59,
0x0,0x52,0xa0,
0x0,0x4f,0xbc,
```

Obsjieya.txt - 记事本

文件(F) 编辑(E) 格式(O) 查看(V) 帮助(H)

```
0x0,0x19,0x74,
0x0,0x45,0x5e,
0x0,0x60,0xc9,
0x0,0x6c,0x41,
0x0,0x6d,0xaa,
0x0,0x6a,0xe1,
0x0,0x67,0x73,
0x0,0x64,0x92,
0x0,0x62,0x35,
0x0,0x60,0x42,
0x0,0x5e,0xf9,
0x0,0x5e,0xb0,
0x0,0x5f,0x65,
0x0,0x60,0x91,
0x0,0x61,0x6b,
0x0,0x61,0x61,
0x0,0x60,0x67,
0x0,0x5e,0xd5,
0x0,0x5d,0x19,
0x0,0x5b,0x6a,
0x0,0x59,0xb5,
0x0,0x57,0xbb,
0x0,0x55,0x59,
0x0,0x52,0xa0,
0x0,0x4f,0xbc,
```

图 7　部分 OBS 原始数据和解压数据对比图

Fig. 7　Comparison of partial OBS raw data and decompressed data

4　结论

本文提出了基于海底地震仪 OBS 采集的现场声学数据，利用 LZW 无损压缩算法对 OBS 采集的原始声学探测数据进行压缩并对结果进行内存重新分配，从文本文件角度和二进制文件角度分别对压缩效果进行测试比对，并对整个压缩流程的正确性进行了验证。结果表明，整个压缩环节稳定性高，且有较好的重复性。压缩的效果与数据的特性密切相关，对于 OBS 所采集的声学探测数据，LZW 压缩算法可以有很好的压缩比。同时，由于数据源是基于 A/D 转换器采集的原始数据，方便直接在硬件平台实现该压缩方法。进行内存重新分配，可以极大的改善压缩比，但是会相应的提高压缩时间，在压缩时间允许的情况下，采用带有内存分配的压缩方式更为适宜。后续工作会将此种压缩算法集成到硬件，开展进一步测试。

随着海洋探测技术的不断提高，探测的数据量激增，数据的存储与传输是海底声学探测设备研发急需解决的问题之一。本研究为海底探测仪器的设计提供了很好的数据压缩方法，在海底声学探测数据的处理方面有着广阔的应用前景。

参考文献:

[1] 李超，刘保华，支鹏遥，等. 海底地震仪的性能对比及在渤海试验中的应用[J]. 海洋科学，2015，39(3): 77-82.
Li Chao, Liu Baohua, Zhi Pengyao, et al. Performance comparison of ocean bottom seismometers and their application in the Bohai Sea[J]. Marine Sciences, 2015, 39(3): 77-82.

[2] 郭常升，李会银，成向阳，等.海底底质声学参数测

量系统设计[J]. 海洋科学, 2009, 33(12): 73-78.
Guo Changsheng, Li Huiyin, Cheng Xiangyang, et al. A design on an equipment for measuring acoustic properties of sea-floor sediment[J].Marine Sciences, 2009, 33(12): 73-78.

[3] 谷明峰, 郭常升, 李会银. 海底松散沉积物声学性质原位测量实验研究[J]. 海洋科学, 2008, 32(5): 1-5, 16.
Gu Mingfeng, Guo Changsheng, Li Huiyin. Experimental research of in situ acoustic properties measurement of seafloor unconsolidated sediment[J]. Marine Sciences, 2008, 32(5): 1-5, 16.

[4] 白燕, 毛洁, 樊炜, 等. 声学法深海热液速度场测量重建算法研究[J]. 海洋科学, 2011, 35(3): 42-49.
Bai Yan, Mao Jie, Fan Wei, et al. Reconstruction algorithm for acoustic measurement of velocity field of deep-sea hydrothermal vents[J]. Marine Sciences, 2011, 35(3): 42-49.

[5] 李劳钰, 王辉武, 吕连港. 基于声学方法的南黄海浮游动物垂直迁移季节变化研究[J]. 海洋科学, 2012, 36(9): 96-101.
Li Laoyu, Wang Huiwu, Lü Liangang. Seasonal variation of the zooplankton vertical migration in the southern Yellow Sea described by acoustic method[J]. Marine Sciences, 2012, 36(9): 96-101.

[6] 叶倩, 张俊兰, 冯雄伟. 浅析数据压缩技术[J]. 延安大学学报(自然科学版), 2008, 27(4): 29-33.
Ye Qian, Zhang Junlan, Feng Xiongwei. A brief analysis of data compression technology[J]. Journal of Yanan University (Natural Science Edition), 2008, 27(4): 29-33.

[7] 张伟, 师奕兵. 声波测井数据压缩的一种 SPIHT 改进算法[J]. 电子测量与仪器学报, 2008, 22(1): 15-19.
Zhang Wei, Shi Yibing. Improved SPIHT algorithm for acoustic wave logging data compression[J]. Journal of Electronic Measurement and Instrument, 2008, 22(1): 15-19.

[8] 贾安学, 乔文孝, 鞠晓东, 等. 声波测井井下数据压缩算法压缩效果测试[J]. 测井技术, 2011, 35(3): 288-291.
Jia Anxue, Qiao Wenxiao, Ju Xiaodong, et al. Effect test on compression algorithms of acoustic logging downhole data[J]. Well Logging Technology, 2011, 35(3): 288-291.

[9] 胡斌, 李忠强, 刘婷婷, 等. Huffman 与 LZW 算法在海洋观测浮标通信数据压缩中的应用研究[J]. 海洋科学, 2018, 42(1): 6-10.
Hu Bin, Li Zhongqiang, Liu Tingting, et al. Application of Huffman and LZW algorithms in data compression for ocean-observation-buoy communication[J]. Marine Sciences, 2018, 42(1): 6-10.

[10] 许霞, 马光思, 鱼涛. LZW 无损压缩算法的研究与改进[J]. 计算机技术与发展, 2009, 19(4): 125-127.
Xu Xia, Ma Guangsi, Yu Tao. Research and improvement on LZW lossless compression algorithm[J]. Computer Technology and Development, 2009, 19(4): 125-127.

[11] 邹学玉, 冯振, 张少华, 等. 基于LZW算法的声波测井数据压缩研究[J]. 测井技术, 2013, 37(3): 294-296.
Zou Xueyu, Feng Zhen, Zhang Shaohua, et al. On the sonic logging data compression based on LZW algorithm[J]. Well Logging Technology, 2013, 37(3): 294-296.

[12] 鄢海舟, 胥布工, 石东江, 等. 无损压缩算法 LZW 前缀编码优化及应用[J]. 计算机工程, 2017, 43(3): 299-303.
Yan Haizhou, Xu Bugong, Shi Dongjiang, et al. Prefix encoding optimization and application of lossless compression algorithm LZW[J]. Computer Engineering, 2017, 43(3): 299-303.

[13] 曹芳彤. 声波测井数据压缩算法的嵌入式开发与实现[D]. 西安: 西安科技大学, 2015.
Cao Fangtong. Embedded development and realization of acoustic logging data compression algorithm[D]. Xi'an: Xi'an University of Science and Technology, 2015.

[14] 郑翠芳. 几种常用无损数据压缩算法研究[J]. 计算机技术与发展, 2011, 21(09): 73-76.
Zheng Cuifang. Research of several common lossless data compression algorithms[J]. Computer Technology and Development, 2011, 21(9): 73-76.

[15] Tan S, Lau S, Tan C. Optimizing LZW text compression algorithm via multithreading programming[C]//IEEE. IEEE Malaysia International Conference on Communications. Kuala Lumpur, Malaysia: IEEE, 2010: 592-596.

[16] Arjun N, Negi A. An approach to self-embedding document watermarking based on LZW algorithm[C]//IEEE. Tencon IEEE Region 10 Conference. Taipei, Taiwan: IEEE, 2007: 1-4.

Data compression method for submarine acoustic detection based on LZW compression algorithm

JIN Song[1], PEI Yan-liang[2, 3, 4], KAN Guang-ming[2, 3, 4], WU Ai-ping[1], FU Qing-qing[1]

(1. National Experimental Teaching and Demonstration Center of Electrical and Electronic Engineering, Yangtze University, Jingzhou 434023, China; 2. Laboratory for Marine Geology, Pilot National Laboratory for Marine Science and Technology (Qingdao), Qingdao 266061, China; 3. First Institute of Oceanography, Ministry of Natural Resources, Qingdao 266061, China; 4. Key Laboratory of Marine Sedimentology and Environmental Geology, Ministry of Natural Resources, Qingdao 266061, China)

Received: Jul. 21, 2018
Key words: LZW compression algorithm; lossless compression; ocean bottom seismograph; submarine acoustic detection data

Abstract: Considering the practical problems of large data collection volume, limited storage capacity, and insufficient data transmission bandwidth of submarine acoustic detection instruments, this study investigates the real-time compression method of submarine acoustic detection data based on Lempel–Ziv–Welch (LZW) lossless compression algorithm. This method redistributes the storage space of the compression results on the basis of the LZW compression algorithm and data storage characteristics; hence, the compression ratio (compressed data bytes/data bytes before compression) can be considerably reduced. Based on a test using the original acoustic detection data of ocean bottom seismographs (OBS), the results show that this compression method features a good compression ratio for OBS acoustic detection data and can be used to compress acquired OBS acoustic detection data. This study can significantly guide the development of submarine detection instruments.

(本文编辑：刘珊珊)