Lossless Compression of Hydroacoustic Image Data

Lixue Wu, Adam Zielinski, *Senior Member, IEEE*, and John S. Bird

***Abstract*— Despite rapid progress in improving mass-storage density and digital communication system performance, compression of hydroacoustic image data is still significant in many engineering and research areas since it can overcome data storage and transmission bandwidth limitations. In this paper, we present a novel and effective approach for lossless compression of hydroacoustic image data which consists of two stages. The first stage reduces the information redundancy. We propose several new techniques to remove redundancy between data samples, data blocks, and data frames. The second stage uses a newly developed cascade coding scheme. This simple scheme can achieve an efficiency of 97%. A decomposition algorithm is presented for finding the optimal cascade coding parameters. The algorithm decomposes a multivariable optimization problem into a series of one-variable optimizations. Our two-stage algorithm offers a compression ratio of 2–3 and provides an exact recovery of the original data. Because of its simplicity, the algorithm can be incorporated into a variety of echo sounder systems. The compression algorithms can also be implemented using low-level assembly language to meet the requirements of real-time applications.**

***Index Terms*— Data compression, echo sounding, fisheries acoustics.**

I. Introduction

DESPITE rapid progress in improving mass-storage density and digital communication system performance, compression of hydroacoustic image data is still significant in many engineering and research areas since it can overcome data storage and transmission bandwidth limitations. Hydroacoustic image data are initially shown on a computer-driven graphic display to ensure data quality during the data collection process. The data are also recorded so that analysis can be performed later on large amounts of data for numerous applications such as fish detection and identification, determination of abundance and distribution of fish stocks, studies of sea bottom structure, etc.

With improved instrumentation, such as multifrequency and multibeam echo sounder systems, the amount of data gathered has increased significantly. Without compression, we would receive 0.25–0.75 GB of data per day assuming that each sample is represented as a 2-byte unsigned integer value. Since the trend is toward increasing the sampling frequency of recorded data [1], the estimated amount of data will triple.

On today's echo sounder systems, entropy reduction [2] is widely used to compress data. In such a system, the output data samples are compared with a preassigned threshold. Those samples exceeding the threshold are tagged as alarms and must be stored and processed, while the remainder are considered as noise and discarded [3]. If the alarm event occurs infrequently, then a large compression of the signal is achieved. An entropy reduction operation would result in a compression ratio of 5:1. Here the degree of data reduction obtained as a result of the compression process is known as the compression ratio. This ratio measures the quantity of compressed data in comparison with the quantity of original data and is given by [4]

$$\text{Compression ratio} = \frac{\text{Length of original data string}}{\text{Length of compressed data string}}. \quad (1)$$

The entropy-reduced data can be further compressed using lossless data compression methods without any numerical noise being added during decompression, so that we can retrieve the data in its exact original form. Our study is focused on the data that result following entropy reduction. Therefore, we refer to the data after entropy reduction as the original data.

Data compression has been an ongoing area of research for several decades. Nevertheless, in the last several years, there has been an explosion of interest and activity in this area. Several review articles can be found in the special section of [5]. Topics addressed include lossless compression, speech compression, image compression, video compression, and emerging technology. These papers reflect the current state of data compression research and provide a reference to important techniques and areas. Lossless image compression methods [6], [7] are not suitable for application to hydroacoustic image data. In some cases, these methods even expand the original data. This is due to the additional pixels added to the image background, where targets are not presented, in forming a raster image. Lossless data compression methods such as Huffman and arithmetic coding [8] can be applied when the number of possible symbols to be encoded is small. However, the methods are not directly applicable to the compression of digital echo sounder outputs from typical analogue-to-digital (A–D) converters, where the numbers of symbols (sample values) are generally of order 10^3–10^6, corresponding to 12–16-bit outputs of A–D converters. We have applied some commercially available data compression software such as PKZIP and ZIP to hydroacoustic image data and only achieved a compression ratio of about 1.1–1.3.

In this paper, we present a novel and effective approach for lossless compression of hydroacoustic image data that consists of a two-stage algorithm. The first stage reduces the

Manuscript received October 31, 1995; revised October 4, 1996. This work was supported by the Natural Sciences and Engineering Research Council of Canada.

L. Wu was with the School of Engineering Science, Simon Fraser University, Burnaby, BC V5A 1S6 Canada. He is now with the National Research Council, Ottawa, ON K1A 0R6 Canada.

A. Zielinski is with the Department of Electrical and Computer Engineering, University of Victoria, Victoria, BC V8W 3P6 Canada.

J. S. Bird is with the School of Engineering Science, Simon Fraser University, Burnaby, BC V5A 1S6 Canada.

Publisher Item Identifier S 0364-9059(97)01417-9.

0364–9059/97$10.00 © 1997 IEEE

Authorized licensed use limited to: Univ of Science and Tech Beijing. Downloaded on September 23,2021 at 02:17:56 UTC from IEEE Xplore. Restrictions apply.

information redundancy. We present several new techniques to reduce or remove the redundancy between data samples, data blocks, and data frames. The original data are rearranged to form different types of data blocks. Different redundancy-reduction techniques are used for different data types in order to fully utilize the data statistics. After the redundancy-reduction operation, the second stage uses a new coding scheme called *cascade coding*. The technique benefits from its simplicity in which only subtractions and logical operations are performed. The cascade code can reach an efficiency of 97% (the term efficiency will be defined mathematically in Section II). A decomposition algorithm is presented for finding the optimal cascade coding parameters. This algorithm decomposes a multivariable optimization into a series of one-variable optimizations. With these new techniques, lossless data compression can be achieved. Our algorithm offers a compression ratio of 2–3 and provides an exact recovery of the original data. It can be incorporated into a variety of echo sounder systems. The compression algorithms can also be implemented using low-level assembly language to meet the requirements of real-time applications.

In most cases, one of the slowest computer operations in hydroacoustic image data analysis is accessing data on disk. When data compression is used to reduce storage requirements, overall program execution time may be reduced. This is because the reduction in storage will result in a reduction of disk-access attempts, while the encoding and decoding required by the compression technique will result in additional program instructions being executed. Since the execution time of a group of program instructions is normally significantly less than the time required to access and transfer data to a peripheral device, overall program execution time may be reduced. Our program based on the proposed coding scheme can read and decompress compressed data faster than it can read an equivalent amount of uncompressed data.

II. Background

A. Data Collection

A scientific echo sounder system provides several calibrated analogue outputs for data recording [9]. All echo signals located between the transducer and several meters below the sea bed that exceed a user-selected threshold are recorded, i.e., an entropy-reduction operation is performed. A typical single ping showing echo amplitude versus ocean depth is plotted in Fig. 1. The detector and A–D conversion system provide digitized voltage output and are controlled by a computer. Samples are collected into blocks along with header information on vessel position, time, and date obtained by serial link from the navigation systems. The data blocks are recorded on digital audio tape (HTI 340) or 150-MB DC600 cartridges (Simrad EK400). The data rate is approximately 2–6 MB per minute.

B. Data Format

To generate signals in the water column, a strong (several kilowatts), acoustic pulse is transmitted [10]. This transmission

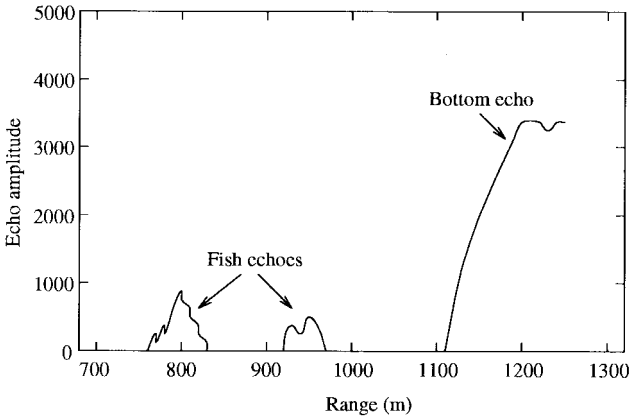

Fig. 1. Plot of a single ping showing echo amplitude represented by unsigned integer versus ocean depth in meters.

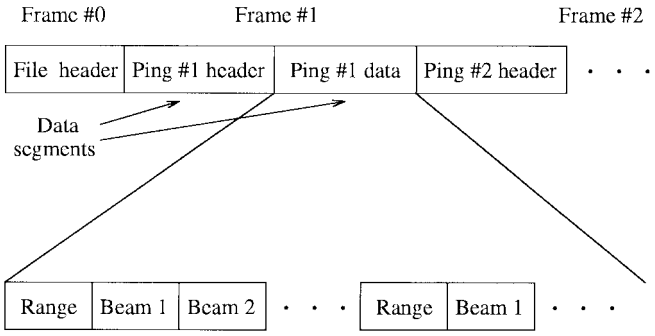

Fig. 2. Data structure of a typical hydroacoustic image data file acquired during a fisheries-related cruise.

and the subsequent echoes received prior to the next transmission are referred to as a "ping." The acquisition system records data based on the echoes of the ping, and the data are referred to as a "ping record." The ping record is preceded by a header which contains pertinent information such as transducer depth, time, position, ping number, etc. The exact format of the ping header is software-dependent. Thus, each ping record is associated with a particular instant in time. Data are only collected if the target detector determines that a target exists at that particular time. Each data entry consists of a 16-bit range word and several beam outputs from A–D converters. Information common to all pings, such as frequencies, number of beams, etc., is contained in a data block called the "file header" [3]. Shown in Fig. 2 is an illustration of the data structure of a typical hydroacoustic image data file acquired during a fisheries-related cruise.

C. Terminology

A basic data compression block diagram is illustrated in Fig. 3. Shown as black boxes, compression and decompression algorithms can be incorporated into a variety of echo sounder systems. As illustrated in Fig. 3, the original data stream is processed through an algorithm to produce a compressed data stream. This compression of the original data stream is

Authorized licensed use limited to: Univ of Science and Tech Beijing. Downloaded on September 23,2021 at 02:17:56 UTC from IEEE Xplore. Restrictions apply.

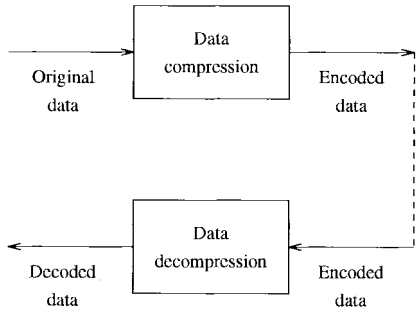

Fig. 3. Basic data compression block diagram.

sometimes referred to as an encoding process and the compressed data stream called an encoded data stream. To reverse the process, the compressed data stream is decompressed to recover the original data stream, which is referred to as the decoded data stream.

One term we mentioned in the introduction is the efficiency, which is defined here as [8]

$$\text{Efficiency} = \frac{\text{Entropy of source}}{\text{Average length of codeword}} \times 100\%. \quad (2)$$

In the above definition, the entropy of source is given by

$$\text{Entropy} = -\sum_k P_k \log_2 P_k \quad (3)$$

where P_k is the kth source symbol probability.

III. Data Compression

As illustrated in Fig. 2, the data file consists of data frames. Except for the first data frame (file header), each data frame consists of two data segments, the ping header and the ping record. The ping record can be rearranged to form two kinds of data blocks, range and echo amplitude, as shown in Fig. 4. Different encoding schemes are used for different data types in order to fully utilize the data statistics. Each method can be described as a two-stage process. In the first stage, a redundancy-reduction operation is performed. In the second stage, a cascade encoding scheme is employed. The process provides an exact recovery of the original data.

A. File Header

The file header contains information common to all pings. A typical data collection sequence involves several processors as discussed in [3]. The host processor is a computer which is responsible for the buffering and storage of data. It is also responsible for providing the user interface by accepting inputs from a keyboard. At the beginning of the data collection, the host processor first initializes all processors and configures them with appropriate parameters. Once this is accomplished, the host processor generates a data block containing these parameters and writes the file header.

The file header can be compressed using the Lempel–Ziv encoding technique [11], [12]. Their algorithms are based on a procedure that creates a new phrase as soon as a prefix of the still unparsed part of the string differs from all preceding phrases. Using the LZ-77 encoding algorithm, we can achieve an average compression ratio of more than $5:1$ for various file headers.

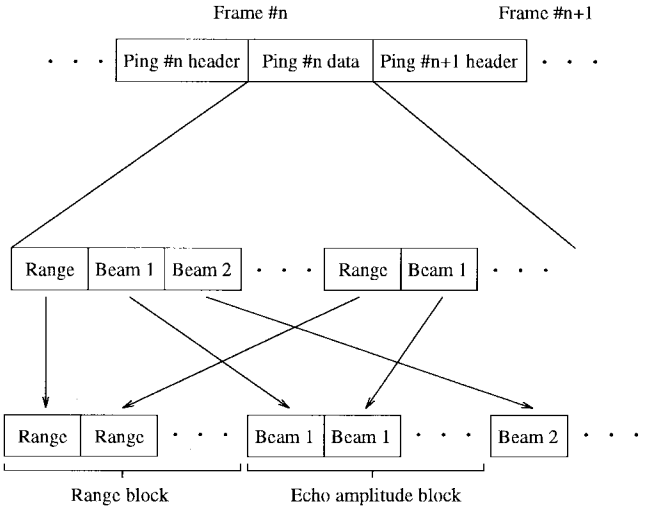

Fig. 4. Rearrangement of a ping record to form range and echo amplitude data blocks.

B. Ping Header

The ping header contains pertinent information such as transducer depth, time, position, ping number, etc. Most of the information contained in a ping header consists of environmental parameters. These parameters generally vary slowly. The succeeding ping headers may be nearly identical (i.e., there are no environmental changes between pings). The Lempel–Ziv techniques, which serve as the basis for the compression method used by modems, do not work well in this case. This is because the repetitive ping headers are separated by ping records, as illustrated in Fig. 2. Even though we rearrange the data and pack all ping headers together, the LZ-77 encoding algorithm can only reach a compression ratio of $3:1$. This is due to the limitation of the LZ-77 algorithm. The fixed-step increments, such as ping number and time, in succeeding ping headers are not utilized by the algorithm. Instead, the LZ-77 algorithm treats the increment as a string differing from all preceding phrases. Moreover, packing the ping header requires large-size buffers generally on the order of a MB. It also introduces a notable delay on the order of minutes. Less data buffering can alleviate these difficulties; however, it may degrade the compression benefit from the LZ-77 algorithm.

We consider a new compression method for the ping header. The method is described generally as a two-stage coding scheme. The first stage uses interframe compression, which compares each ping header to succeeding pings, removes redundant information, and encodes the ping header. The encoded ping header is further compressed at the second stage by a cascade encoder.

1) Difference Sequence: The ping header can be considered as a character stream. We may represent each character by its integer value and form an integer sequence as illustrated in

Authorized licensed use limited to: Univ of Science and Tech Beijing. Downloaded on September 23,2021 at 02:17:56 UTC from IEEE Xplore. Restrictions apply.

6	0	50	0	16	92	1	23	16	10	21	62
4	0	1	16	2	0	1	0	15	39	28	5
0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0								

(a)

7	0	70	0	16	92	1	23	16	10	22	61
4	0	1	16	2	0	1	0	15	39	27	5
0	0	0	0	0	0	0	0	0	0	0	0
2	0	1	0								

(b)

1	0	20	0	0	0	0	0	0	0	1	-1
0	0	0	0	0	0	0	0	0	0	-1	0
0	0	0	0	0	0	0	0	0	0	0	0
2	0	1	0								

(c)

Fig. 5. A typical difference sequence of two adjacent ping header data segments. (a) An integer sequence representing the values of characters in the ping no. i header data segment. (b) An integer sequence representing the values of characters in the ping no. $i+1$ header data segment. (c) A difference sequence representing the differences between the integer sequences in (a) and (b).

Fig. 5(a). This sequence is compared with the integer sequence representing a succeeding ping header as shown in Fig. 5(b). Their differences are taken into account and referred to as the difference sequence, as illustrated in Fig. 5(c). We see that there are many zeros in the difference sequence. The occurrence of zero is over 88%. The second most reoccurring value is 1, followed by the third most frequent value -1. These observations mirror the fact that the environmental parameters vary slowly. When an advanced data acquisition system is used by a simple echo sounder system, many environmental parameters may be set to zero due to the lack of relevant information such as the attitude of a towfish. In such a case, even more zeros can occur in difference sequences.

2) Pre-Encoding: Difference sequences can be encoded using optimum source codes, such as the Huffman code and the Shannon–Fano code [8]. These codes reach an efficiency of 100% only when the source message probabilities are negative powers of 2. As illustrated in Fig. 5(c), long strings of zeros occur in the difference sequences. The optimum source codes operate on one sample at a time and utilize no intersample dependence. Therefore, they are not suitable for the difference sequence. We find that an average number of successive zeros is about 6.7. The elimination of repetitive zeros can reduce the redundancy and is referred to as pre-encoding. An obvious technique for avoiding many consecutive zeros is to count the number of successive zeros between two adjacent nonzero samples and store only this number. Since successive nonzero

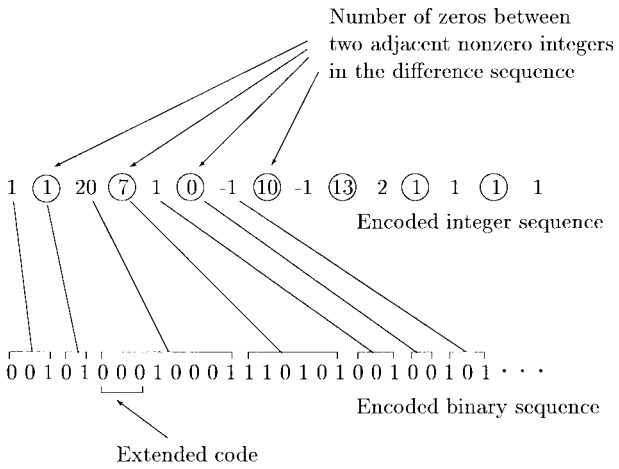

Fig. 6. A two-step coding scheme for ping header difference sequences.

samples are rare, we do not store the zero along with the intervening repetitive count as run-length coding does. Instead, we insert a zero between two successive nonzero samples to indicate that the number of successive zeros is zero. With this arrangement, nonzero samples and zero counts (numbers of successive zeros) appear alternately in the pre-encoded difference sequence. The pre-encoding process is illustrated in the top portion of Fig. 6 with the difference sequence taken from Fig. 5(c). Note that a zero is inserted between the nonzero samples 1 and -1. To ensure that the difference sequence always starts with a nonzero difference, one should arrange to write an always-changing parameter such as ping number at the beginning of the ping header.

3) Cascade Encoding: Pre-encoded difference sequences can be further encoded into binary sequences, as illustrated in the bottom portion of Fig. 6. Since the nonzero samples and zero counts in the pre-encoded difference sequence have different statistics, we deal with them separately.

The nonzero sample has a range from -255 to 255. Natural binary coding with 9 bit for each nonzero sample can be used. While simple and fast, it is far from optimum since the distribution of nonzero samples is not uniform. It is found that low-value samples occur frequently. We propose here an encoding scheme called cascade encoding to utilize these statistics. Several stages are involved in the encoding process. The cascade encoder bears some analogy to multistage and residual quantizers [15], [16]. Cascade encoders are realized as a sequence of small natural binary encoders, each one operating on the residual of the preceding natural binary encoder. The cascade structure is illustrated in Fig. 7 with two bits per residue assigned to each stage. At each stage, an extended code is utilized to indicate an oversized input sample that is the sample greater than $2^b - 1$, where b is the number of bits assigned to that stage. This sample is subtracted by an integer $2^b - 1$ to form a residue. The extended code and the residue are output. The residue is then passed to the next stage as an input. The process is repeated until a nonoversized input is found and a nonextended code is output. At the first stage, one extra bit (the first bit of the output code) is used to code the

Authorized licensed use limited to: Univ of Science and Tech Beijing. Downloaded on September 23,2021 at 02:17:56 UTC from IEEE Xplore. Restrictions apply.

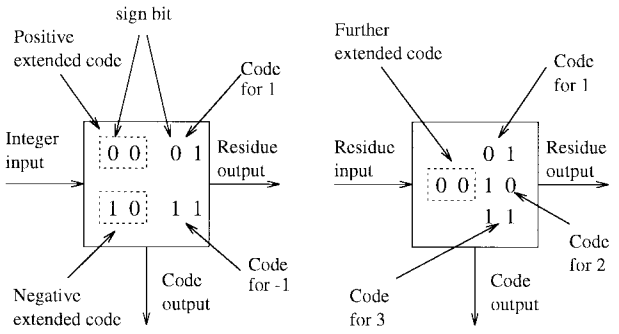

Fig. 7. Structure of cascade stages for coding nonzero samples in difference sequences. Two bits are assigned to each stage. In the first stage, the first bit of the output code is used to indicate the sign of the input integer.

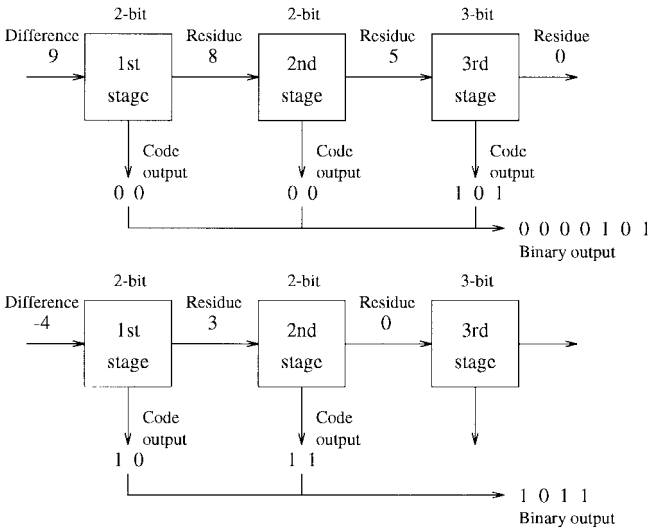

Fig. 8. A cascade procedure for encoding nonzero samples in difference sequences. Three stages are shown in the illustration. Two, two, and three bits are assigned to the 1st, 2nd, and 3rd stage, respectively.

sign of samples. Therefore, residues passed to the second stage are positive integers. The binary representation of the sample can be obtained by collecting the code outputs at all active stages (i.e., stages having nonzero input). Fig. 8 illustrates the cascade procedure for encoding two nonzero samples, 9 and -4, in difference sequences. Three stages are shown in the illustration. Two, two, and three bits per residue are assigned to the first, second, and third stage, respectively.

The codes 001, 00 010 001, 001, and 101 in the bottom portion of Fig. 6 are the cascade codes representing the first four nonzero samples (uncircled numbers in Fig. 6) 1, 20, 1, and -1, respectively, of the difference sequence illustrated in Fig. 5(c). Three and five bits per residue are assigned to the first and second stage, respectively. In the first stage, the first bit of the output code is used as the sign bit. The remaining two bits can code the unsigned integers 1, 2, and 3 using the natural binary codes 01, 10, and 11, respectively. The code 00 together with the sign bit is utilized to form the extended codes 000 and 100 for positive and negative oversized samples, respectively. Since the samples 1 and -1 are not oversized samples, they are coded at the first stage by the codes 001 and 101, respectively. The second stage is not active for these two samples. For the sample 20, an extended code 000 is output from the first stage since $20 > 2^2 - 1$. The residue of 17 obtained from $20 - (2^2 - 1)$ is also output and passed to the second stage. Since $17 < (2^5 - 1)$, the residue of 17 is not an oversized input for the second stage and can be coded by the natural binary code 10001. Collecting the code outputs from the first and second stages, we obtain the code 00 010 001 for the sample 20.

Cascade coding is a scheme in which the original data sample is chopped into several residues. These residues are encoded at separate stages using a modified natural binary coding scheme for the sake of simplicity and fast performance. The coding efficiency depends highly on the choices of coding parameters b_i and N, where the bit allocation b_i is the number of bits required for residues at the ith stage and N is the total number of stages. Knowing the probability of each sample, the average number of bits, n, needed to represent samples in cascade coding can be expressed as

$$\begin{aligned} n = b_1 &\sum_{i=1}^{2^{b_1-1}-1} \text{Prob}\,(\text{sample} = i) + (b_1 + b_2) \\ &\cdot \sum_{i=2^{b_1-1}}^{2^{b_1-1}+2^{b_2}-2} \text{Prob}\,(\text{sample} = i) + (b_1 + b_2 + b_3) \\ &\cdot \sum_{i=2^{b_1-1}+2^{b_2}-1}^{2^{b_1-1}+2^{b_2}+2^{b_3}-3} \text{Prob}\,(\text{sample} = i) + \cdots b_1 \\ &\cdot \sum_{i=-1}^{-(2^{b_1-1}-1)} \text{Prob}\,(\text{sample} = i) + (b_1 + b_2) \\ &\cdot \sum_{i=-2^{b_1-1}}^{-(2^{b_1-1}+2^{b_2}-2)} \text{Prob}\,(\text{sample} = i) + (b_1 + b_2 + b_3) \\ &\cdot \sum_{i=-(2^{b_1-1}+2^{b_2}-1)}^{-(2^{b_1-1}+2^{b_2}+2^{b_3}-3)} \text{Prob}\,(\text{sample} = i) + \cdots. \end{aligned} \tag{4}$$

The average bits per sample can be minimized by selecting optimal values of the bit allocation b_i and the number of stages N [17]. Although this optimization problem involves only one constraint:

$$\begin{aligned} &\sum_{i=1}^{N} b_i \le a \\ &\quad b_i \ge 0, \quad i = 1, \cdots, N; \qquad \text{all } b_i \text{ integers} \end{aligned} \tag{5}$$

where a is the number of bits needed to represent each sample using natural binary coding, the objective function given by (4) is not separable [18]. That is, the average number of bits n cannot be expressed in the form of $\sum_{i=1}^{N} f_i(b_i)$. Also, the total number of stages N is not fixed and needs to be optimized. Therefore, the problem cannot be solved by dynamic programming [18]. One method is that all possible values of b_i and N are tried; subsequently, the one set is chosen which produces the minimum n. Since this method is based

Authorized licensed use limited to: Univ of Science and Tech Beijing. Downloaded on September 23,2021 at 02:17:56 UTC from IEEE Xplore. Restrictions apply.

on trial, intensive computations may be required, which would lead to an unacceptable time delay in coding processing. In order to obtain the optimal coding parameters more efficiently, we propose a new method called *decomposition*, whereby a multivariable optimization problem is solved by consecutive one-variable optimizations.

We now describe our method for a more general case. Given the occurrence of symbols p_i, we rearrange p_i according to its value in a decreasing order to form a vector $\boldsymbol{p}$, where $\boldsymbol{p} = [p_1, p_2, \cdots, p_K]$ with $p_i \geq p_{i+1}$. Meanwhile, the symbol with occurrence p_i is mapped into a nonzero positive integer i. Thus, we can use the cascade coding technique to encode any symbols rather than only integers. For a given bit allocation b_i we form a bit vector $\boldsymbol{c} = [\boldsymbol{c}_1, \boldsymbol{c}_2, \cdots]$, where $\boldsymbol{c}_i$ is a subvector and has $2^{b_i} - 1$ similar elements given by $\sum_1^i b_i$. The length of $\boldsymbol{c}$ must be equal to or greater than the length of $\boldsymbol{p}$ for a feasible bit allocation. The average bits per sample $\overline{n}$ is then obtained as

$$\overline{n} = \boldsymbol{b}\boldsymbol{p}' \tag{6}$$

where $\boldsymbol{b}$ is a truncated vector of $\boldsymbol{c}$ by taking the first K elements.

The decomposition algorithm is stated as follows.

1) Begin with initial bit allocations for the first stage. Set $b_1 = 2, 3, \cdots, \text{ceil}(\log_2(K+1))$, and set $b_i = 1$ for $i > 1$, where $\text{ceil}(x)$ rounds x up to the nearest integer. Form bit vectors for each bit allocation.
2) Compute $\overline{n}$. Find two possible bit allocations which result in the least average bits per sample (minimum and minor minimum). The possible bit allocations (two) for this stage are then determined.
3) At the ith stage, set $b_i = 2, 3, \cdots, \max(b_{i-1}^*) - 1$, where b_{i-1}^* are two possible bit allocations obtained in the previous step, and set $b_k = 1$ for $k \geq 2^{b_i} - 1 + i$. Form bit vectors for each bit allocation and go to step 2.

The key to this decomposition algorithm is a setting that assigns only one bit to the stages after the stage currently being optimized. The algorithm begins with the first stage and repeats this setting until all the stages are optimized. This setting is due to the fact that the elements of $\boldsymbol{p}$ are in decreasing order. The decomposition algorithm shows some analogy to dynamic programming in dimensionality reduction. It uses information from earlier searches to decide how to locate bits on later searches. This flexibility pays off heavily and results in an efficient search algorithm.

To demonstrate the decomposition algorithm, we choose 1000 nonzero integer samples from a quantizer. The definition of the quantizer will be described later. Table I shows the bit allocations, bit vectors, and total bits required for 1000 samples with sample variance $\sigma^2 = 16$ and $K = 26$ in the process of computing the optimal solution. Table I contains 24 rows of data corresponding to 24 searches. These rows are grouped to 8 blocks showing how bit allocations are optimized consecutively at each stage. The bit allocation being optimized at a stage is underlined. The initial bit allocations for optimizing the first stage are shown in the first block (first 4 rows, numbered from 1 to 4) in Table I. Since searches 2 and 3 result in the minimum and minor minimum total bits, the possible optimal bit allocation for the first stage is $\{3\}$ or $\{4\}$. Thus, in optimizing the second stage, the bit allocation of the first stage is set to either 3 or 4 and the bit allocation of the second stage varies from 1 to 3. Since the average bits per sample $\overline{n}$ for bit allocation 1 of the second stage is already calculated in the first-stage optimization, we only need to calculate $\overline{n}$ for the bit allocations 2 and 3 of the second stage. The bit allocations for the third stage and beyond are set to 1. Bit allocations and total bits for 1000 samples in optimizing the second stage are shown in the second block (second 4 rows, numbered from 5 to 8) in Table I. Among the six bit allocations (No. 2, 3, 5–8), searches 3 and 7 lead to the minimum and minor minimum total bits. Therefore, the possible optimal bit allocation for the first and second stages is $\{3, 2\}$ or $\{4, 1\}$. This process is repeated until all stages are optimized. Note that there are fewer evaluations in optimizing the fifth, sixth, seventh, and eighth stages. Here, we set a suitable criterion to discard infeasible bit allocation, namely that a feasible bit allocation must satisfy

$$2^{b_i - 1} - 1 > K - i + 1. \tag{7}$$

For example, the bit allocation $\{3, 2, 3, 1, 3\}$ in optimizing the fifth stage is not feasible. Because there are only three possible symbols left for the fifth stage and beyond to encode, it is not favorable to assign three bits to the fifth stage. From Table I, we also see that there are many suboptimal bit allocations near the optimum, such as No. 23, 24, and 20. This indicates that cascade coding can provide stable performance even if the bit allocation is shifted from its optimum.

To test the performance of the cascade encoder, we apply cascade coding to 1000 positive integer test samples. Let X be a Gaussian random variable with zero-mean and variance σ^2. The test data samples are outputs of a quantizer $Q(X)$ given

TABLE I
BIT ALLOCATIONS, BIT VECTORS, AND TOTAL BITS REQUIRED FOR 1000 SAMPLES WITH SAMPLE VARIANCE $\sigma^2 = 16$ IN THE PROCESS OF COMPUTING OPTIMAL SOLUTION

No.	Bit allocation	Bit vector	Total bits
1	2, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1	[2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14]	4489
2	3, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1	[3, 3, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13]	4141
3	4, 1, 1, 1, 1, 1, 1	[4, 4, 4, 4, 4, 4, 4, 5, 6, 7, 8, 9, 10]	4121
4	5	[5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5]	5000
5	4, 2, 1, 1, 1	[4, 4, 4, 4, 4, 4, 4, 6, 6, 6, 7, 8, 9]	4144
6	4, 3	[4, 4, 4, 4, 4, 4, 4, 4, 7, 7, 7, 7, 7, 7]	4201
7	3, 2, 1, 1, 1, 1, 1, 1, 1	[3, 3, 3, 5, 5, 5, 5, 6, 7, 8, 9, 10, 11, 12]	4083
8	3, 3, 1, 1, 1	[3, 3, 3, 6, 6, 6, 6, 6, 6, 6, 7, 8, 9]	4282
9	3, 2, 2, 1, 1, 1, 1	[3, 3, 3, 5, 5, 5, 7, 7, 7, 8, 9, 10, 11]	4100
10	3, 2, 3	[3, 3, 3, 5, 5, 5, 8, 8, 8, 8, 8, 8, 8]	4190
11	4, 1, 2, 1, 1	[4, 4, 4, 4, 4, 4, 4, 5, 7, 7, 7, 8, 9]	4131
12	4, 1, 3	[4, 4, 4, 4, 4, 4, 4, 5, 8, 8, 8, 8, 8]	4157
13	3, 2, 1, 2, 1, 1, 1	[3, 3, 3, 5, 5, 5, 6, 8, 8, 8, 9, 10, 11]	4106
14	3, 2, 1, 3	[3, 3, 3, 5, 5, 5, 6, 9, 9, 9, 9, 9, 9]	4163
15	3, 2, 2, 2, 1	[3, 3, 3, 5, 5, 5, 7, 7, 7, 9, 9, 9, 10]	4105
16	3, 2, 2, 3	[3, 3, 3, 5, 5, 5, 7, 7, 7, 10, 10, 10, 10]	4118
17	3, 2, 1, 1, 2, 1, 1	[3, 3, 3, 5, 5, 5, 6, 7, 9, 9, 9, 10, 11]	4093
18	3, 2, 1, 1, 3	[3, 3, 3, 5, 5, 5, 6, 7, 10, 10, 10, 10, 10]	4119
19	3, 2, 2, 1, 2	[3, 3, 3, 5, 5, 5, 7, 7, 7, 8, 10, 10, 10]	4102
20	3, 2, 1, 1, 1, 2, 1	[3, 3, 3, 5, 5, 5, 6, 7, 8, 10, 10, 10, 11]	4088
21	3, 2, 1, 1, 1, 3	[3, 3, 3, 5, 5, 5, 6, 7, 8, 11, 11, 11, 11]	4101
22	3, 2, 1, 1, 2, 2	[3, 3, 3, 5, 5, 5, 6, 7, 9, 9, 9, 11, 11]	4095
23	3, 2, 1, 1, 1, 1, 2	[3, 3, 3, 5, 5, 5, 6, 7, 8, 9, 11, 11, 11]	4085
24	3, 2, 1, 1, 1, 1, 1, 2	[3, 3, 3, 5, 5, 5, 6, 7, 8, 9, 10, 12, 12]	4085

Authorized licensed use limited to: Univ of Science and Tech Beijing. Downloaded on September 23,2021 at 02:17:56 UTC from IEEE Xplore. Restrictions apply.

TABLE II
CODING EFFICIENCIES, NUMBER OF SEARCHES FOR OPTIMAL BIT ALLOCATION, AND OPTIMAL BIT ALLOCATIONS FOR DIFFERENT SAMPLE VARIANCES

σ^2	Range	Entropy	Bits/sam.	Efficiency	Searches	Bit allocation
2	{-5, 5}	2.559	2.661	96.16	8	2, 1, 1, 1, 1
4	{ 8, 8}	3.083	3.140	98.17	20	2, 1, 1, 1, 1, 1, 1, 1
8	{-8, 10}	3.529	3.569	98.87	20	3, 1, 1, 1, 1, 1, 1, 1
16	{-14, 13}	3.942	4.083	96.55	24	3, 2, 1, 1, 1, 1, 1, 1, 1
32	{-16, 16}	4.513	4.620	97.67	14	4, 2, 2, 1, 1, 1
64	{-29, 23}	4.951	5.144	96.24	21	5, 2, 1, 3, 2
128	{-37, 40}	5.483	5.721	95.84	24	5, 3, 3, 3, 1, 2
256	{-54, 46}	5.965	6.246	95.50	19	6, 3, 3, 3, 1, 1
512	{-92, 74}	6.491	7.040	92.20	23	7, 3, 3, 3, 3, 1
1024	{-104, 104}	6.894	7.309	94.32	30	7, 3, 3, 3, 3, 3, 3

input X. Such a quantizer is defined as

$$Q(X) = \begin{cases} i, & 0 \le i-1 \le x < i \\ -i, & 0 \ge -(i-1) \ge x > -i \end{cases} \tag{8}$$

where i is a positive integer. One thousand samples are collected from the quantizer. The optimal bit allocation is obtained using the proposed decomposition method. The samples are then encoded by the cascade encoder. The coding efficiencies, number of searches for optimal bit allocation, and optimal bit allocations for different sample variances are tabulated in Table II. We see that cascade coding can achieve an efficiency of about 97% for a Gaussian random sequence with $\sigma^2 = 16$.

The cascade coding scheme benefits from its simplicity. It is much faster than other available coding methods, with the exception of natural binary coding. For an average of 1.5 stages involved in cascade coding, which is the normal case for hydroacoustic image data, the processing speed slow down is only about 50% compared with natural binary coding. Since only subtraction and logical operations are involved in the coding algorithm, data compression can be implemented using low-level assembly languages to meet the requirement of real-time applications.

Besides fast processing performance, cascade coding is more appropriate than other compression methods when the distribution of symbol probabilities is close to a stepwise function. Cascade codes can reach an efficiency of 100% if each step in the distribution function is a negative power of 2. Furthermore, only a few bits are needed for transmitting the side information (bit allocation). Therefore, cascade coding can still maintain high efficiency when the number of total symbols is large.

4) Zero Counts: Zero counts, the numbers of successive zeros in difference sequences, have a different distribution than that of nonzero samples. Since the zero count will never exceed the length of the ping header, the possible values (symbols) of zero counts are limited. Therefore, Huffman coding is more suitable for this case. Information about the source (zero counts) statistics needed for designing the Huffman code can be obtained by analyzing the first few ping headers. Since the zero counts reflect the slowly changing and unused environmental parameters in the ping header, the statistics of zero counts will not change during data acquisition. The codes 01, 110 101, and 00 in the bottom portion of Fig. 6 are the Huffman codes representing the first three zero counts (circled numbers in Fig. 6) 1, 7, and 0 of the difference sequence illustrated in Fig. 5(c), respectively.

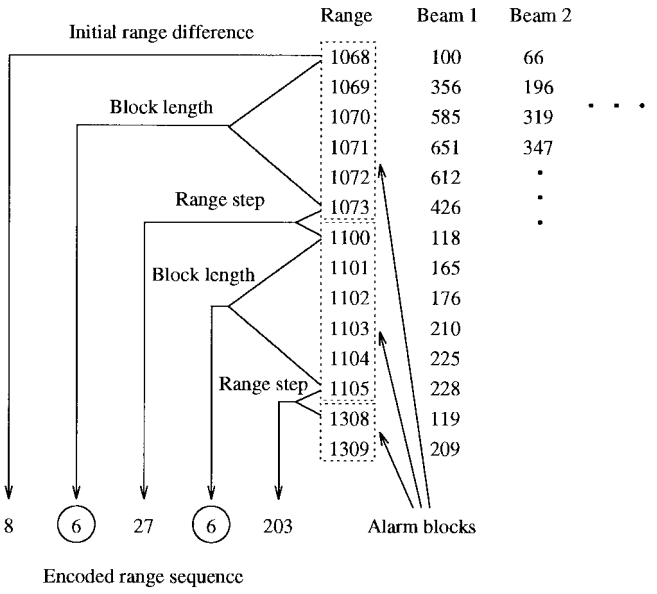

Fig. 9. Illustration of the encoding process for range data blocks.

C. Ping Record I: Range

The ping record can be further rearranged to form two kinds of data blocks, range and echo amplitude, as shown in Fig. 4. We now consider a new compression method for range blocks. The method is described generally as a two-stage coding scheme. The first stage uses intersample compression, which compares each range to succeeding ranges, removes redundant information, and encodes the range block. The encoded range sequence is further encoded in the second stage by a cascade encoder as described in the previous subsection.

The encoding algorithm for the range block is as follows.

1) The initial range is compared with the initial range in the previous ping record. Their difference is stored.
2) Consecutive ranges are grouped to form an alarm block as illustrated in Fig. 9. The alarm block length along with the range step between succeeding alarm blocks are stored.
3) The encoded range sequence is further compressed by the cascade encoder to obtain a binary sequence.

The encoded range sequence has a simple structure. The sequence starts with an initial range difference followed by an alarm block length. The alarm block length is related to the size of a target. The alarm block length and range step appear alternately in the sequence. The sequence ends with the range step. It is not necessary to store the last alarm block length since it can be obtained for the ping record length stored in the ping header.

Optimal choice of cascade coding parameters depends on the sample statistics. The distribution of differences of initial ranges in two succeeding ping records shows the interframe relationship between succeeding pings. The distribution of alarm block lengths relates target size, while the distribution

Authorized licensed use limited to: Univ of Science and Tech Beijing. Downloaded on September 23,2021 at 02:17:56 UTC from IEEE Xplore. Restrictions apply.

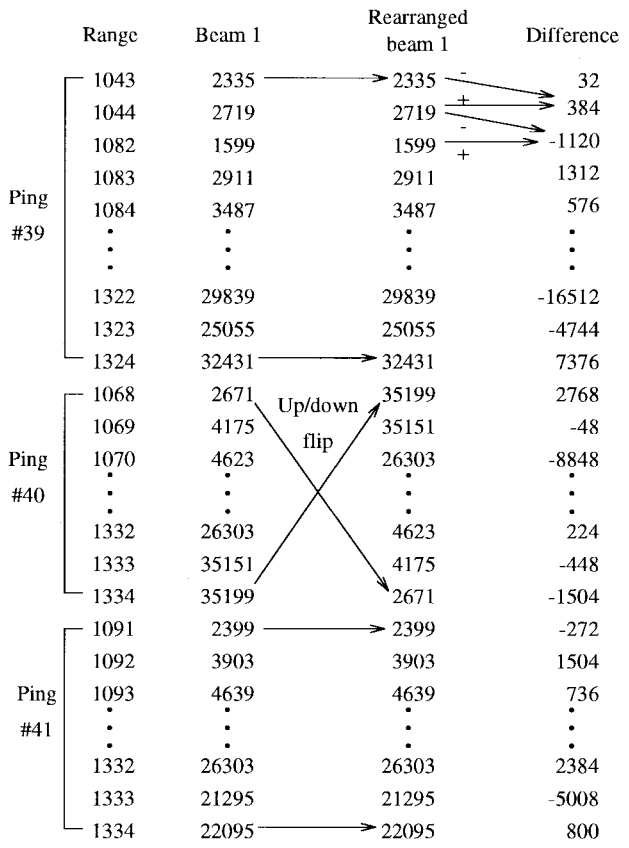

Fig. 10. Illustration of the algorithm for echo amplitude sample coding.

of range steps between two adjacent alarm blocks gives us the information about target density. All distributions reflect distinct physical parameters and therefore are different. To obtain high-coding efficiency, we choose different coding parameters according to different sample statistics. Three optimal cascade encoders are designed for encoding initial range differences, alarm block lengths, and alarm steps, respectively. The sample statistics needed for the design can be estimated from the analysis of the first several ping records. For dealing with a large volume of data, the estimates of the sample statistics should be updated after a period to maintain a high coding efficiency. However, there is always a tradeoff between coding efficiency and complexity.

D. Ping Record II: Echo Amplitude

The two-stage coding scheme, described in the previous subsection, can also be used to code echo amplitude samples. In the first stage, intersample compression is used to remove redundant information. The first-stage encoded echo sequence is further encoded at the second stage by a cascade encoder.

Data entry of a ping record starts from the echo of the first target encountered and ends with the echo from the sea bottom. The amplitudes of echoes from the sea bottom depend on bottom type and are highly correlated between consecutive pings. For this reason, we rearrange the echo amplitude block before forming an echo amplitude sequence. That is, the echo amplitude samples in an even ping number data block are flipped in the up–down direction. This is illustrated in Fig. 10 for pings 39–41. After this process, the echo data blocks are arranged consecutively to form the echo amplitude sequence. The echo amplitude sequence appears more smooth than that of the unflipped sequence.

TABLE III
ENTROPY OF ECHO AMPLITUDE SAMPLES AFTER UTILIZING DIFFERENT REDUNDANCY-REDUCTION METHODS

	Original data	Differential pulse	Linear prediction
Entropy	10.5801	10.1748	9.7056

Many methods can be used to code the echo amplitude sequence. The simplest method utilizes differential pulse code modulation, where the predicted difference is the same as the most recent difference between samples. The data is reconstructed from reverse difference prediction and nothing is lost. However, this scheme does not utilize first- and second-order data statistics and is therefore suboptimal. To achieve a better performance, linear prediction can be used to eliminate redundancy [13].

Our linear predictor differs from the usual linear predictor. Since we require exact recovery of the input integer data, the prediction and reverse prediction algorithms are designed to reverse each other exactly. Therefore, rounding operations are employed to ensure an integer input and an integer output in both encoding prediction and decoding reverse prediction [19].

The entropy of the encoded samples of a typical echo amplitude sequence from the first stage for different source coding methods is tabulated in Table III. The length of the sequence is 117 665. This sequence is divided into data frames to perform linear prediction. The length of each data frame is 1000, except for the last frame. A linear predictor with order 4 is optimally designed for each data frame. We see that linear prediction is more efficient at the expense of complexity.

The differences/residues from the output of the first stage are further encoded at the second stage using cascade coding. The high-dynamic range of the residues indicates that Huffman and arithmetic coding have difficulty achieving high compression efficiencies [14]. Cascade coding can alleviate the difficulty and still maintain a high efficiency.

E. Coding Example

With the methods described above, we have developed a new coding algorithm. We applied this algorithm to hydroacoustic image data provided by the Pacific Biological Station and compared the performance of our algorithm with those of Lempel and Ziv [11], [12] and the lossless version of the JPEG still-picture compression standard [20]. Table IV gives a comparison of the performance of LZW, JPEG, and our compression methods. The predictors used for the JPEG in the comparison are defined in [20, Table I]. Note that in coding echo amplitude samples we used differential pulse coding for simplicity. A higher compression ratio can be achieved if we use linear prediction coding.

Authorized licensed use limited to: Univ of Science and Tech Beijing. Downloaded on September 23,2021 at 02:17:56 UTC from IEEE Xplore. Restrictions apply.

TABLE IV
COMPRESSION RESULTS FOR HYDROACOUSTIC IMAGE DATA

File name	Original size (byte)	Compression method	Compressed size (byte)	Compression ratio
91her001.hyd	522,324	Proposed	225,900	2.31
91her001.hyd	522,324	LZW	422,900	1.24
91her001.hyd	522,324	JPEG, prediction 0	305,820	1.71
91her001.hyd	522,324	JPEG, prediction 1	369,539	1.41
91her001.hyd	522,324	JPEG, prediction 2	303,347	1.73
91her001.hyd	522,324	JPEG, prediction 3	368,977	1.42
91her001.hyd	522,324	JPEG, prediction 4	367,706	1.42

IV. CONCLUSION

This work has contributed three advances in lossless hydroacoustic image data coding: 1) two-stage lossless coding strategy; 2) new techniques for removing the redundancy between data samples, data blocks, and data frames; and 3) a new coding scheme, cascade coding, for efficiently coding samples with high dynamic range. These contributions result in coding efficiency gains and significant computational improvements.

REFERENCES

[1] R. B. Mitson and D. V. Holliday, "Future developments in fisheries acoustics," in *Developments in Fisheries Acoustics: A Symposium Held in Seattle*, W. A. Karp, Ed., 1990, pp. 82–91.

[2] T. J. Lynch, *Data Compression: Techniques and Applications.* Belmont, CA: Lifetime Learning, 1985.

[3] A. Stepnowski and R. S. Mitchell, "ECOLOG II: A real-time acoustic signal processing system for fish stock assessment," *Ultrason.*, vol. 28, pp. 256–265, July 1990.

[4] S. Ruth and P. Kreutzer, "Data compression for large business files," *Datamation*, vol. 18, no. 9, pp. 62–66, 1972.

[5] J. A. Storer, Ed., *Proc. IEEE*, special section on data compression, vol. 82, June 1994.

[6] R. J. Stewart, Y. F. Lure, and C. J. Liou, "An adaptive technique to maximize lossless image data compression of satellite images," *Robot. Comput.-Integr. Manuf.*, vol. 11, no. 2, pp. 111–116, 1994.

[7] N. D. Memon, K. Sayood, and S. S. Magliveras, "Lossless compression of multispectral image data," *IEEE Trans. Geosci. Remote Sensing*, vol. 32, pp. 282–289, 1994.

[8] T. C. Bell, I. H. Witten, and J. G. Cleary, *Text Compression.* Englewood Cliffs, NJ: Prentice-Hall, 1990.

[9] R. B. Mitson, *Fisheries Sonar*. Farnham, Surrey, U.K.: Fishing News Books, Ltd., 1983.

[10] C. S. Clay and H. Medwin, *Acoustical Oceanography: Principles and Applications.* London, U.K.: Wiley, 1977.

[11] A. Lempel and J. Ziv, "A universal algorithm for sequential data compression," *IEEE Trans. Inform. Theory*, vol. IT-23, pp. 337–343, 1977.

[12] ——, "Compression of individual sequences via variable-rate coding," *IEEE Trans. Inform. Theory*, vol. IT-24, pp. 530–536, 1978.

[13] A. Gersho and R. M. Gray, *Vector Quantization and Signal Compression.* Boston, MA: Kluwer Academic, 1992, pp. 83–125.

[14] L. Tan, "Theory and techniques for lossless waveform data compression," Ph.D. dissertation, University of New Mexico, Albuquerque, May 1992.

[15] A. Gersho and R. M. Gray, *Vector Quantization and Signal Compression.* Boston, MA: Kluwer Academic, 1992, pp. 451–459.

[16] R. L. Frost, C. F. Barnes, and F. Xu, "Design and performance of residual quantizers," in *Proc. Data Compression Conf.*, Snowbird, UT, Apr. 1991, pp. 129–138.

[17] D. J. Wilde, *Optimum Seeking Methods.* Englewood Cliffs, NJ: Prentice-Hall, 1964.

[18] G. Hadley, *Nonlinear and Dynamic Programming.* London, U.K.: Addison-Wesley, 1964.

[19] S. D. Stearns, L. Tan, and N. Magotra, "Lossless compression of waveform data for efficient storage and transmission," *IEEE Trans. Geosci. Remote Sensing*, vol. 31, pp. 645–654, May 1993.

[20] G. K. Wallace, "The JPEG still picture compression standard," *Commun. ACM*, vol. 34, no. 4, pp. 31–44, Apr. 1991.

Lixue Wu received the B.Sc. degree in radio physics from Each China Normal University, Shanghai, China, in 1982, and the M.A.Sc. and Ph.D. degrees in electrical engineering from the University of Victoria in 1989 and 1992, respectively.

From 1982 to 1985, he was a Research Assistant at the Institute of Modern Educational Technology, East China Normal University, where he was engaged in computer-aided design of satellite TV receivers. In 1986, he was a member of the research group of Canada-China International Co-operative Project and worked in the design and development of computer-aided learning systems at the University of Victoria. In 1993 and 1994, he was awarded a NSERC Post-Doctoral Fellowship, tenable at Simon Fraser University, Burnaby, BC, Canada. He is currently with the National Research Council of Canada. His main research interests are in the areas of acoustic signal processing, underwater acoustic telemetry, acoustic transducers and arrays, and data compression.

Dr. Wu is a co-recipient of the Shanghai Advanced Technology Premium in 1984. He is a member of the Acoustical Society of America.

Adam Zielinski (M'77–SM'79) received the B.Eng., M.Eng., and Ph.D. degrees from Wrocław Technical University, Poland, all in electrical engineering, in 1965, 1967, and 1971, respectively.

In 1967, he joined the Faculty of Electronics at Wrocław Technical University, where he was involved in teaching and research on remote control and telemetry systems. In 1972, he was awarded a Japanese Government Fellowship, tenable at the Tokyo Institute of Technology, where he conducted research on data transmission and digital filtering. In 1974, he joined the Department of Electrical Engineering at the University of New Brunswick, Canada, as a Post-Doctorate Fellow, where he worked on the resolution problem of overlapping radar signals. From 1975 to 1985, he was with Memorial University of Newfoundland working on ocean acoustic systems. During 1982–1983, he spent his sabbatical leave at the University of Hawaii and Tokyo Institute of Technology, working on technical aspects of tsunami warning systems. In 1985, he joined the University of Victoria, Department of Electrical and Computer Engineering, where he is presently a Professor. His areas of expertise and current interests include underwater acoustic sensing, communication, instrumentation, and signal processing. He has developed several new concepts and instruments which are documented in a number of publications.

Dr. Zielinski is a member of the Association of Professional Engineers of British Columbia.

John S. Bird received the B.S. degree in electrical engineering in 1973 from the University of British Columbia, Vancouver, BC, Canada, and the Ph.D. degree in electrical engineering from Carleton University, Ottawa, ON, Canada, in 1980.

He worked for the Defence Research Establishment Pacific, Victoria, BC, from 1973 to 1981, where he was engaged in signal processing and detection studies related to passive sonar. From 1981 to 1987, he continued these studies but in the context of land- and space-based surveillance radars while on loan to the Communications Research Centre from the Defence Research Establishment Ottawa. During this time, he also worked in the area of satellite spread spectrum communications. He is currently with the School of Engineering Science at Simon Fraser University in Burnaby, BC, where his research interests include signal processing, underwater acoustics, sonar, autonomous underwater vehicles, and ocean exploration.

Authorized licensed use limited to: Univ of Science and Tech Beijing. Downloaded on September 23,2021 at 02:17:56 UTC from IEEE Xplore. Restrictions apply.