International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 2 Issue 1, January- 2013

Review of Data Compression and Different Techniques of Data Compression

P. S. Asolkar, P. H. Zope, S. R. Suralkar
SSBT college of Engineering and technology, Jalgaon

Abstract

This paper presents the basic of data compressor and the different types of data compressor techniques. Various data compressor techniques which have developed in past are compared. A comprehensive list of references is reported and the data compression techniques are classified in to following main types: 1) lossless compression; 2) lossy compression. The objective of this paper is to identify various compression techniques that can be useful in the monitoring systems which are used in home care and hospital system comparative study of data compression techniques and the algorithms has been described in this paper.

Index Terms— Data compression, lossless, lossy, compression techniques.

1. Introduction

Compression is used just about everywhere. All the images you get on the web are compressed, typically in the JPEG or GIF formats, most modems use compression, HDTV will be compressed using MPEG-2, and several file systems automatically compress files when stored, and the rest of us do it by hand.

The neat thing about compression, [1] is that the algorithms used in the real world make heavy use of a wide set of algorithmic tools. Furthermore, algorithms with strong theoretical foundations play a critical role in real-world applications. In this, the generic term *message* for the objects we want to compress, which could be either files or messages. The task of compression consists of two components, an *encoding* algorithm that takes a message and generates a “compressed” representation (hopefully with fewer bits), [2] and a *decoding* algorithm that reconstructs the original message or some approximation of it from the compressed representation. These two components are typically intricately tied together since they both have to understand the shared compressed representation. The difference is made between *lossless algorithms*, [3] which can reconstruct the original message exactly from the compressed message, and *lossy algorithms*, which can only reconstruct an approximation of the original message. Lossless algorithms are typically used for text, and lossy for images and Sound where a little bit of loss in resolution is often undetectable, or at least acceptable. Lossy is used in an abstract sense, however, and does not mean random lost pixels, but instead means loss of a quantity such as a frequency component, or perhaps loss of noise.

For example, one might think that lossy text compression would be unacceptable because they are imagining missing or switched characters. Consider instead a system that reworded sentences into a more standard form, or replaced words with synonyms so that the file can be better compressed. Technically the compression would be lossy since the text has changed, but the “meaning” and clarity of the message might be fully maintained, or even improved.

2. Data Compression

Compression:
The process of coding that will effectively reduce the total number of bits needed to represent certain information. Compression ratio is defined as the ratio of number of bits before compression to the number of bits after compression.
Compression issues:
There are basically two types of compression techniques. The one compression technique is lossless and the technique of compression is lossy technique.

3. LOSSLESS COMPRESSION

Lossless data compression has been suggested for many space science exploration mission applications either to increase the science return or to reduce the requirement for on-board memory, station contact time, and data archival volume.

A Lossless compression technique [5] guarantees full reconstruction of the original data without incurring any distortion in the process. The Lossless Data Compression technique recommended preserves the source data accuracy by removing redundancy from the application source data. In the decompression processes the original source data is reconstructed from the compressed data by restoring the removed redundancy. The reconstructed data is an exact replica of the original source data. The quantity of redundancy removed from the source data is variable and is highly dependent on the source data statistics, which are often non-stationary.

The Lossless Data Compression algorithm can be applied at the application data source or performed as a function of the on-board data system as shown in the following figure. The performance of the data compression algorithm is independent of where it is applied. However, if the data compression algorithm is part of the on-board data system, the on-board data system will, in general, have to capture the data in a buffer. In both cases, it may be necessary to rearrange the data into appropriate sequence before applying the data compression algorithm. The purpose of rearranging data is to improve the compression ratio.

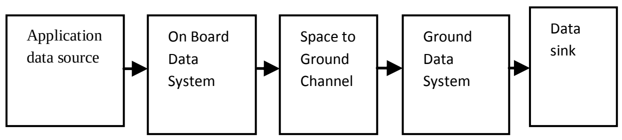

Fig. 1 Packet telemetry for lossless compressor [5]

4. LOSSY COMPRESSION

Lossy compression is compression in which some of the information from the original message sequence is lost. This means the original sequences cannot be regenerated from the compressed sequence. Just because information is lost doesn't mean the quality of the output is reduced.

For example, random noise has very high information content, but when present in an image or a sound file, we would typically be perfectly happy to drop it. Also certain losses in images or sound might be completely imperceptible to a human viewer (e.g. the loss of very high frequencies).

For this reason, lossy compression algorithms on images can often get a factor of 2 better compressions than lossless algorithms with an imperceptible loss in quality. However, when quality does start degrading in a noticeable way, it is important to make sure it degrades in a way that is least objectionable to the viewer (*e.g.*, dropping random pixels is probably more objectionable than dropping some colour information).

For these reasons, the ways most lossy compression techniques are used are highly dependent on the media that is being compressed. Lossy compression for sound, for example, is very different than lossy compression for images.

5. Lossless Compression Techniques

The main types of lossless compression techniques includes[4] the Huffman coding, Arithmetic coding, Run length coding, Variable length coding and the most recently used coding which is known as the Golomb-Rice coding.

1. Huffman Codes:

David Huffman developed the algorithm as a student in a class on information theory at MIT in 1950. The algorithm is now probably [6] the most prevalently Huffman codes are optimal prefix codes generated from a set of probabilities by a particular algorithm, the Huffman Coding Algorithm.

The algorithm is now probably [6] the most prevalently used component of compression algorithms used component of compression algorithms, used as the back end of GZIP, JPEG and many other utilities. The Huffman algorithm is very simple shown in Fig.3. And is most easily described in terms of how it generates the prefix-code tree.

1. Start with a forest of trees, one for each message. Each tree contains a single vertex with weight wi = pi.

2. Repeat until only a single tree remains.

3. Select two trees with the lowest weight roots (w1 and w2).

4. Combine them into a single tree by adding a new root with weight w1+w2, and making the two trees its children. It does not matter which is the left or right child, but the convention will be to put the lower weight root on the left if w1≠w2.

For a code of size n this algorithm will require n−1 steps since every complete binary tree with n leaves has n−1 internal nodes, and each step creates one internal node. If we use a priority queue with O(log n) time insertions and find-mins (*e.g.*, a heap) the algorithm will run in O(n log n) time.

The key property of Huffman codes is that they generate optimal prefix codes. *The Huffman algorithm generates an optimal prefix code.* Since Huffman coding is optimal we know that for any probability distribution S and associated Huffman code C $H(S) \leq la(C) \leq H(S) + 1$.

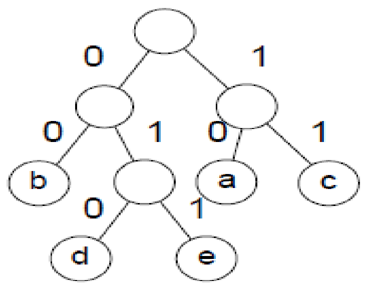

Fig. 2 Binary tree for Huffman Code [6]

2. Run Length coding:

This encoding method is frequently applied to images (or pixels in a scan line).[5] It is a small compression component used in JPEG compression. In this instance, sequences of image elements $X_1, X_2, \ldots, X_n$ are mapped to pairs $(c_1, l_1), (c_1, L_2), \ldots, (c_n, l_n)$ where c_i represent image intensity or colour and l_i the length of the ith run of pixels (Not dissimilar to zero length suppression above).

Original Sequence: 111122233333311111222 can be encoded as**:** (1,4),(2,3),(3,6),(1,4),(2,4) The savings are dependent on the data. In the worst case (Random Noise) encoding is heavier than original file: 2*integer rather 1* integer if data is represented as integers.

3. Arithmetic Coding:

Arithmetic coding is a technique for coding that allows the information from the messages in a message sequence to be combined to share the same bits. [7] The technique allows the total number of bits sent to asymptotically approach the sum of the self information of the individual messages (recall that the self information of a message is defined as log2 1 pi). To see the significance of this, consider sending a thousand messages each having probability .999. Using a Huffman code, each message has to take at least 1 bit, requiring 1000 bits to be sent. On the other hand the self information of each message is log2 1 pi = .00144 bits, so the sum of this self-information over 1000 messages is only 1.4 bits. It turns out that arithmetic coding will send all the messages using only 3 bits, a factor of hundreds less than a Huffman coder. Of course this is an extreme case, and when all the probabilities are small, the gain will be less significant. Arithmetic coders are therefore most useful when there are large probabilities in the probability distribution. The main idea of arithmetic coding is to represent each possible sequence of n messages by a separate interval on the number line between 0 and 1, *e.g.* the interval from .2 to .5. For a sequence of messages with probabilities p1, . . . , pn, the algorithm will assign the sequence to an interval of size $\prod_i^n = 1pi$, by starting with an interval of size 1 (from 0 to 1) and narrowing the interval by a Factor of pi on each message i. We can bind the number of bits required to uniquely identify an interval of size s, and use this to relate the length of the representation to the self information of the messages. [8] In arithmetic coding, a message is represented by an interval of real numbers between 0 and 1. As the message becomes longer, the interval needed ‘to represent it becomes smaller, and the number of bits needed to specify that interval grows. Successive symbols of the message reduce the size of the interval in accordance with the symbol probabilities generated by the model. The more likely symbols reduce the range by less than the unlikely symbols and hence add fewer bits to the message.

4. Golomb Rice Coding:

A Golomb-Rice code is a Golomb code where the divisor is a power of two, enabling an efficient implementation using shifts and masks rather than division and modulo. A Golomb code is variable-length code, a bit like Huffman; however, rather than being based on the data, like Huffman, it's based on a simple model of the probability of the values. To Golomb-code a number, find the quotient and remainder of division by the divisor. Write the quotient in unary notation, then the remainder in truncated binary notation. In practice, you need a stop bit after the quotient: if the quotient is written as a sequence of zeroes, the stop bit is a one. The length of the remainder can be determined from the divisor.

6. Lossy compression techniques:

The various types of lossy compression techniques includes the Scalar Quantization , Vector Quantization, in addition to this wavelet compression and fractal compression are also included under the category of lossy compression methods.

1. Scalar Quantization:

A simple way to implement lossy compression is to take the set of possible messages S and reduce it to a smaller set S′ by mapping each element of S to an element in S′.

For example take 8-bit integers and divide by 4 (*i.e.*, drop the lower two bits), or take a character set in which upper and lowercase characters are distinguished and replace all the uppercase ones with lowercase ones. This general technique is called *quantization*.

Since the mapping used in quantization is many-to one, it is irreversible and therefore lossy. In the case that the set S comes from a total order and the total order is broken up into regions that map onto the elements of S′, the mapping is called scalar quantization.

Applications of scalar quantization include reducing the number of colour bits or gray-scale levels in images (used to save memory on many computer monitors), and classifying the intensity of frequency components in images or sound into groups (used in JPEG compression).

2. Vector Quantization:

Scalar quantization allows one to separately map each color of a color image into a smaller set of output values. In practice, however, it can be much more effective to map regions of 3-d color space into output values. By more effective we mean that a better compression ratio can be achieved based on an equivalent loss of quality. The general idea of mapping a multidimensional space into a smaller set of messages S′ is called vector quantization. Vector quantization is typically implemented by selecting a set of representatives from the input space, and then mapping all other points in the space to the closest representative.

The representatives could be fixed for all time and part of the compression protocol, or they could be determined for each file (message sequence) and sent as part of the sequence. The most interesting aspect of vector quantization is how one selects the representatives.

7. ALGORITHM FOR LOW-POWER BIOMEDICAL SYSTEMS

The lossless data compression algorithm is largely based on a basic discrete pulse code modulation (DPCM) predictor followed by Golomb-Rice entropy coding, [4] whose block diagram and algorithm flow is shown in Fig. 3.

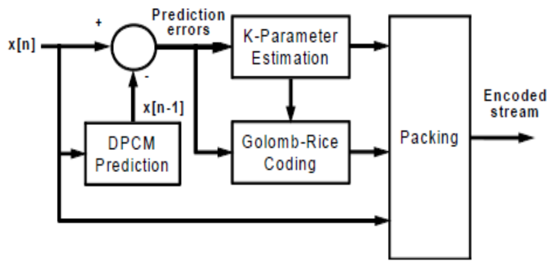

Fig.3. Basic DPCM Prediction Followed By Golomb- Rice Entropy Coding [10]

1. Set Window Size WS = 64,128 or 256

```
2. Set Sample Precision SP = 8 or 10
3. Output the First Sample x [0]
4. Set Sample Pointer S_PTR = 1
5. Set Sum_Abs_Err = 0
6. for n from S_PTR to S_PTR+WS-1:
SET SUM_ABS_ERR = SUM_ABS_ERR
+abs(x[n]-x [n-1])
7. SET K = ceil (log2 (SUM_ABS_ERR/WS))
8. Output Golomb- Rice Parameter K
9. for n from S_PTR to S_PTR+WS-1:
Output GR- ENCODE(x[n]-x [n-1], k)
10. Advance to next window: S_PTR = S_PTR + WS
11. Repeat steps 5 to 10 still all samples are processed
```

8. RESULT & ANALYSIS

The previous sample is taken as the prediction for the current sample, and the prediction error is obtained by subtracting the two.[4]

From a window size WS of prediction errors, the Golomb-Rice code scaling K parameter is estimated and the prediction errors are Golomb-Rice entropy coded using this parameter. Finally, the encoded stream is packed into data chunks of fixed width for output.

This algorithm is then implemented in brain-heart monitoring system and the Golomb-Rice coding gives the better compression ratio as compared to the other coding techniques in the particular brain-heart monitoring system.

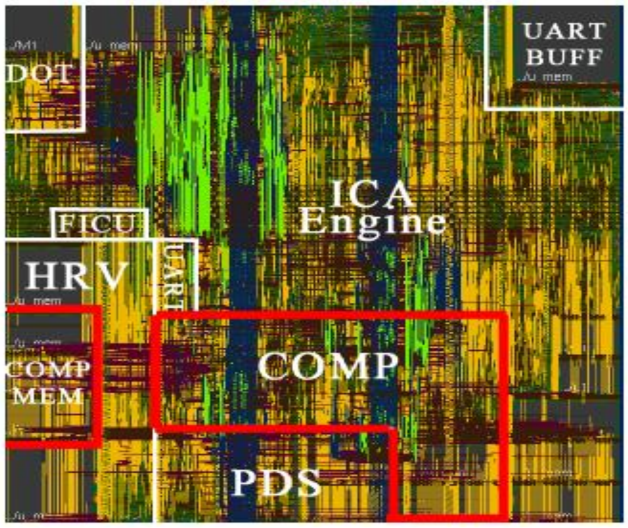

Fig. 4 Chip implementation in 65 nm CMOS technology [10]

The ECG/EEG/DOT data in chip is them implemented in 65 nm CMOS technology as shown in Fig. 4. For this technique, the compressor comprises 53,969 gates and occupying a total of 58k µm2. Simulated power consumption using a full operation test case reports 170µW under the condition of 24MHz clock frequency and 1.0V core supply voltage. Thus after implementing lossless data compressor algorithm, the compression ratio results are found out to be as shown in the table I

Table I. Compression Ratio Results [10]

Bio signal	**Compression Ratio**
ECG	2.38
EOG	1.37/1.55
DOT	2.10
AVERAGE	2.05

The results are calculated from MATLAB simulations whereas EEG and ECG raw data were obtained from EEGLab and the MIT-BIH Arrhythmia Database, respectively.

9. CONCLUSION

The data compression and to develop the various techniques of data compression is a still a challenging task for researchers and academicians. As reported by included references a large majority of research was oriented to data compression techniques. This paper presents the study and the comparison of various data compression techniques. After studying all these techniques it is found that the lossless data compressor technique is most effective over the lossy one. The papers published in the past years reflect experimental results for the compression techniques. Finally it is seen that the lossless Golomb-Rice entropy coding after implementing in the brain-heart monitoring system give the better compression results. In the future, it will be of interest to push further the limits of low power compression in terms of Ecomp and CR, possibly by investigating optimized implementations of more complex algorithms as well as more aggressive power optimizations at both algorithm and architectural levels.

10.REFERENCES

[1] I. Deslauriers and J. Bajcsy, “Serial turbo coding for data compression and the slepian-wolf problem,” in IEEE Information Theory Workshop,August 2003, pp. 296–299.

[2] A. D. Liveris, Z. Xiong, and C. N. Georghiades, “Compression of binary source with side information at the decoder using LDPC codes,” IEEE Communications Letters, vol. 6, no. 10, pp. 440–442, October 2002.

[3] Fang, Wai-Chi; Chen, Chiu-Kuo; Chua, Ericson; Fu, Chih-Chung; Tseng, Shao-Yen; Kang, Shih; , "A low power biomedical signal processing system-on-chip design for portable brain-heart monitoring systems," Green Circuits and Systems (ICGCS), 2010 International Conference on , vol., no., pp.18-23, 21-23 June 2010.

[4]Held, G. Data Compression: Techniques and Applications. Wiley, New York, 1984. Explains a number of ad hoc techniques for compressing text.

[5] Lossless Data Compression. Recommendation for Space Data Systems Standards, CCSDS 121.0-B-1. Blue Book. Issue 1. Washington, D.C.: CCSDS, May 1997.

[6] Cormack, G.V., and Horspool, R.N. Algorithms for adaptive Huffman codes. Ir$ Process. Lett. 18, 3 (Mar. 19841, 159-166. Describes how adaptive Huffman coding can be implemented efficiently.

[7] Rissanen, J.J. Generalized Kraft inequality and arithmmatic coding. IBM 1. Res. Dev. 20 (May 1976), 198-203. Another early exposition of the idea of arithmetic coding

[8]Rissanen. J.J. Arithmetic codings as number representations. Acta Polytech. Stand. Math. 31 (Dec. 1979), 44-51. Further develops arithmetic coding as a practical technique for data representation.

[9]I. Witten, R. Neal, and J. Cleary. Arithmetic Coding for Data Compression. *Commun. ACM*, 30:520–540, June 1987

[10] E. Chua, C. Fu, and W. Fang, “VLSI implementation of a mixed bio-signal lossless data compressor for portable brain-heart monitoring systems,” in *Consumer Electronics (ICCE), 2011 IEEE International Conference on. IEEE, pp. 557–558.*

[11] J. Ziv and A. Lempel. A Universal Algorithm for Sequential Data Compression. *IEEE Transactions on Information Theory*, 23(3):337–343, 1977.

[12] Wongsawat, Y.; s, S.; Tanaka, T.; Rao, K.R.; , "Lossless multichannel EEG compression," Circuits and Systems, 2006. ISCAS 2006. Proceedings. 2006 IEEE International Symposium on , vol., no., pp.4 pp.-1614.

[13] Sriraam, N.; Eswaran, C.; , "An Adaptive Error Modeling Scheme for the Lossless Compression of EEG Signals," Information Technology in Biomedicine, IEEE Transactions on , vol.12, no.5, pp.587-594, Sept. 2008.

[14] Giurcăneanu C.D.; Tăbuş I.; Mereuţă S.;, Using contexts and R-R interval estimation in lossless ECG compression, Computer Methods and Programs in Biomedicine, Volume 67, Issue 3, March 2002, Pages 177- 186.

[15] Antti Koski, Lossless ECG encoding, Computer Methods and Programs in Biomedicine, Volume 52, Issue 1, January 1997, Pages 23-33.

[16] Fidopiastis, Cali; Hughes, Charles;, "Workshop 1: Use of psychophysiological measures in virtual rehabilitation," Virtual Rehabilitation, 2008, pp.xi-xi, 25-27 Aug. 2008.

[17]Electronics Society, 2007. IECON 2007. 33rd Annual Conference of the IEEE , vol., no., pp.46-51, 5-8 Nov. 2007.

[18] Arnavut, Z.; , "ECG Signal Compression Based on Burrows-Wheeler Transformation and Inversion Ranks of Linear Prediction," Biomedical Engineering, IEEE Transactions on , vol.54, no.3.

[19] Wu, X.; Memon, N.; , "Context-based, adaptive, lossless image coding," Communications, IEEE Transactions on , vol.45, no.4, pp.437-444, Apr 1997

[20] Yates, D.C.; Rodriguez-Villegas, E.; , "A Key Power Trade-off in Wireless EEG Headset Design," Neural Engineering, 2007. CNE '07. 3rd International IEEE/EMBS Conference on, pp.453-456.