RECENT ADVANCES IN LOSSLESS CODING TECHNIQUES

Gregory S. Yovanof
Eastman Kodak Company
Kodak Berkeley Research
Berkeley, CA 94704

ABSTRACT

Lossless data compression systems allow an exact replica of the original data to be reproduced at the receiver. Lossless compression has found a wide range of applications in such diverse fields as: compression of computer data, still images (e.g., medical or graphical images) and video (usually, in the form of entropy coding of the output of intra/inter-frame lossy schemes). It has been studied for over forty years and new compression algorithms are still continuously developed. This paper is a survey of current lossless techniques with results quoted for both sequential data files and still images.

1 INTRODUCTION

Data compression techniques can be grouped into two broad classes: a) lossless techniques, (also known as reversible, noiseless, bit preserving, or information preserving algorithms) that allow an exact replica of the original data to be reproduced at the receiver, and b) lossy (irreversible) techniques, that reconstruct an approximation to the original data, thereby achieving significantly higher compression. When data compression is motivated by the need to reduce storage requirements (e.g., in storing data on computer disks or tapes, or, in archiving X-ray material and other pictorial databases) it is usually important to employ lossless techniques. On the other hand, in data transmission applications (such as video transmission, teleconferencing, remote sensing via satellite) emphasis is usually placed on reducing the amount of transmitted data and the compression technique need not be information preserving as long as the decompressed data satisfies some fidelity criterion. Lossy compression is commonly applied to consumer-type still images and video, and it can be thought of as a filtering operation. However, there are several image compression applications where certain quality and/or legal considerations mandate the use of lossless compression (e.g., medical imaging, compression of graphics, deep space communication of imagery data).

Data compression algorithms can be further classified as one-dimensional schemes (e.g., for processing of sequential data such as computer programs, text files, numeric data, digital audio) and multidimensional ones (e.g., for processing of still images or video signals). Perhaps the best known one-dimensional lossless scheme is Huffman coding. Originally a two-pass technique (gathering source statistics on the first pass, and compressing the data according to its own statistics on the second pass), it has recently been made one-pass and adaptive. It was only in the last ten years that a new generation of "universal" coding schemes has emerged that are superior in most respects to Huffman coding. Universal schemes (a.k.a adaptive or dynamic schemes) have been developed for cases where the input statistics are not a priori known to the coder. They estimate the source statistics during the coding operation and adapt themselves thereto in order to maximize compression. Arithmetic codes and the Lempel-Ziv algorithm are the two most important members of this class of powerful schemes.

The rest of this paper is organized as follows. Section 2 surveys the two major groups of sequential algorithms: dictionary and statistical techniques. The subject of lossless compression of imagery data is briefly introduced in section 3. Several textbooks and review articles[1]-[5] have extensively surveyed multidimensional lossless techniques and compared their performance in the context of various applications. The interested reader should refer to these surveys for more information. We conclude this work with examples of practical implementations of lossless algorithms and some simulation results.

2 SEQUENTIAL DATA COMPRESSION SCHEMES

The schemes described in this section process data in a sequential manner, input symbol by input symbol, without special provisions for frame structure, end of line, etc. However, together with a two dimensional scanning strategy and may be with some preprocessing, these schemes can be used successfully for compression of 2D data, e.g., pictures. Sequential compression techniques can be classified as: dictionary and statistical techniques. Dictionary techniques are generally characterized by some sort of alphabet extension, viewing a given sequence as a concatenation of fixed- or variable-length blocks of symbols (subsequences) and encoding such subsequences with more compact codes. Statistical algorithms encode the original data without any extension one symbol at a time, exploiting the nonuniform distribution of the symbol probabilities; unlikely symbols are encoded using long codewords, whereas high probability symbols are encoded using fewer bits. It is difficult to establish the superiority of one class of algorithms over the other with regard to the compression performance. Indisputably, though, dictionary techniques provide the most practical trade off between flexibility, compression, and speed. [6] For an overview of both

statistical and dictionary compressors see Lelewer's and Hirschberg's survey [7] and the book by Bell et al.[8]

2.1 Dictionary Based Coding Techniques

Dictionary based compression systems (a.k.a textual substitutional codecs) remove redundancy by detecting and exploiting the presence of repeated patterns in long strings of symbols, e.g., common English words used in text files or certain keywords and identifiers found in inventory records. Typically, such compressors replace a long sequence of symbols in the input data stream with more compact codewords. Each time a unique string of symbols occurs, it is entered into a "dictionary" and is assigned a numeric value, a codeword. Subsequent occurrences of dictionary entries are replaced by their associated codewords. The Lempel-Ziv algorithm and its variations are good examples of this class. Other schemes that fall in the same class are some locally adaptive methods[9] [10] and the recency ranking code of Elias. [11] Fiala and Greene,[12] Storer,[13] and Williams[14] discuss textual substitution schemes in great detail, contain extensive bibliographies and provide extensive comparisons of the performance of most popular dictionary encoders on large sets of test files.

2.1.1 Lempel-Ziv Coding

Most practical compression programs, including the popular Unix utility compress, are based on a technique invented by Lempel and Ziv. The Lempel-Ziv (LZ) method - a string matching and parsing approach to data compression - converts variable length strings of input symbols into fixed length codes stored in a dictionary. A universal scheme, the LZ compressor is applicable to a wide variety of data with different types of redundancy. It does so by "learning" the redundancy of the data "on-the-fly"; no prescanning of the data is needed. Two realizations of the LZ method exist: the LZ77 and LZ78 schemes.

LZ77: The first algorithm introduced by Lempel and Ziv [16] compresses a data string by converting it into a sequence of items, each of which can be either a literal item or a copy item. Literal items consist of the text they represent. Copy items consist of an offset and a pointer that together point to a substring of the data stream already transmitted. This scheme is often identified as the " sliding dictionary method" for data compression, or data compression using " finite windows", for it requires that a finite number of input symbols be stored in a buffer during the encoding/decoding processes. This buffer consists of two parts: its initial part contains a finite number from the past history of the input, while the remaining much smaller part contains the future symbols to compress (the look-ahead-buffer). At each step during the encoding process, the look-ahead-buffer is compared to the rest of the window and the longest

string in the window that matches all or a prefix of the look-ahead-buffer is found. If this matched-string is longer than two input symbols [12] a copy item is transmitted informing the decoder that the string occurred earlier and providing it with sufficient information to locate the substring within the buffer. Otherwise, the first uncompressed input symbol is transmitted as a literal item. Control bits indicate whether items are literal or copy items. Compression is achieved by representing long input strings using copy items. For this scheme, the decompression is extremely fast, since it only involves some string copying, but the compression is slow, due to the complexity of the string matching procedure. The algorithm of Storer and Szymanski,[18] the A1 and FG schemes of Fiala and Greene, [12] and the hardware optimized scheme of Williams [19] are good examples of the LZ77 method.

LZ78: In an effort to speed up the encoding process, Lempel and Ziv proposed a second algorithm [17] in 1978 that converts strings of input symbols into fixed-length codes stored in a string-translation-table, a dictionary. The LZ78 scheme employs a greedy parsing algorithm - repeatedly matching the input stream to the words contained in a dictionary and returning pointers to the locations in the dictionary of the longest match. The dictionary is initialized with all single symbol strings over the input alphabet, and contains strings previously encountered in the message being processed. After every match, the matched string concatenated with the next symbol of the remaining input stream is added to the dictionary, and a unique identifier (its code value) is assigned to it. The codeword associated with the matched string is transmitted to the decompressor. The process continues until the input stream is exhausted. The decoder rebuilds the same dictionary as the data stream is reconstructed, so that no extra data is needed to transmit the dictionary. The major concern in implementation is storing the dictionary. Welch [15] proposed that each string be represented by the identifier of its prefix string and the extension symbol, so that each entry has a fixed length. Such an implementation is possible since the dictionary growth heuristic implied by the addition of the last parsed word concatenated with the first unmatched character causes the dictionary to contain every prefix of every word it holds. The data structure representing such a dictionary with the “prefix property” is an ordered labeled rooted tree (a trie). Each edge emanating from a vertex is labeled by a character of the alphabet. A vertex represents the string obtained by concatenation of all characters along the path from the root to the vertex (see Figure (1)). Several variations of the LZ algorithm exist that increase compression performance by sometimes significant amounts [20]. The major difference between the two LZ schemes is that LZ78 makes no attempt to optimally select strings for the dictionary. While the LZ77 scheme limits the scope of the search by confining the string-matching process within a finite-sliding-window over the message being compressed, the LZ78 scheme limits the search to only a subset of all possible strings in an infinite input buffer. This suboptimal searching routine leads to very fast

hardware implementations, but typically lowers the number of strings that can be matched, thereby reducing the achievable compression ratio. The popular Unix utility compress (based on the Welch's implementation of the LZ78 algorithm [15]), when applied to text files, yields compression ratios of up to 60%, depending on the nature of the text and the size of the dictionary, but has a large memory requirement (450 Kbytes). On the other hand, the FG algorithm (a member of the LZ77 class) requires less memory (186 Kbytes for encoding and 130 Kbytes for decoding) and achieves compression that is about 30 percent better than that provided by compress [12] but runs at half the speed of the compress routine.

2.2 Statistical Coding

A popular alternative to a dictionary based coding scheme is a statistical scheme. Statistical coding, often identified as " entropy" coding, generally refers to the class of compression schemes which compact the data by reducing statistical redundancy, having to do with similarities, correlation and predictability of the data. A statistical compressor may be viewed as consisting of two steps [22] [23]: a) source modeling - involving the construction of a mathematical model that sufficiently characterizes the statistical behavior of the input source and accurately predicts the probability of the individual input symbols, and then b) encoding these symbols with a number of bits which is close to -log2 of the predicted probabilities (the optimal code length according to the theory of source coding [21]). Efficient coding techniques are currently known. Huffman coding, for example, is quite efficient when the alphabet size is fairly large. Arithmetic coding investigations in recent years have resulted in a class of practical schemes which are relatively new with respect to the long-established Huffman coding.

Given the availability of efficient coding techniques, compression depends on correctly estimating the local statistics of encoded symbols. Therefore, of overriding importance to statistical compression is the building of a good source model, which adapts to the local statistics of the encoded symbols on-the-fly, as the coding progresses. In order to improve the estimation accuracy, Rissanen and Langdon [23] conditioned the probability of the coded symbol on a " context" based on a number of symbols in the immediate vicinity of the current symbol. Source models that predict successive input symbols taking into account symbols already processed are generally referred to as "context models" Markov models are good examples of this type of modeling; the PPMC [8] algorithm, the state-of-the-art in context-modeling, is a variable order Markov model. Context modeling has proved to be the most promising of current general purpose data compression algorithms. However, while context-modeling algorithms provide excellent compression, they suffer from the disadvantages of being slow and requiring large amounts of memory space in which to

execute. Bell et al.[8] report encoding and decoding speeds of 2000 symbols per second (sps) for an order-3 context model as compared with 12000 sps for compress and 6000 sps for algorithm FG. Recently, a major research effort has been devoted to the designing of fast context modeling algorithms. [24]- [29]

2.2.1 Huffman Coding

Huffman coding[30] (HC) is one of the most popular schemes currently in use. It translates fixed-size pieces of input data into variable-length symbols whose average length approaches the normalized entropy of the input source. The general strategy is to assign codewords of relatively short length to those input symbols with the highest probability of occurrence. This method is also known as variable word-length coding (VLC). It can be shown that the HC performs optimally if all symbol probabilities are negative powers of two. But, its average code length is also constrained to be at least 1 bit per input symbol, regardless of how small the entropy is. Construction of a Huffman code involves the use of a tree (see Figure (2)). The symbols to be coded are arranged according to decreasing order of their probabilities. Construction of the tree proceeds from bottom-up as follows: the two symbols with the lowest probabilities are combined via a node and their probabilities are added to form the probability of a "combined symbol". The new set of symbols is again arranged according to decreasing order of probabilities and the procedure is repeated. This procedure continues until all symbols are combined into a single symbol whose probability is one. The resulting codebook is implemented as a code tree with branches that contain the codewords.

There are some major drawbacks to Huffman coding in this way: a) the codebook has to be transmitted as a header of the compressed data stream; b) its efficacy is limited by the size of the alphabet, since it encodes each symbol independently; and most important, c) the probabilities of the input symbols must be known before coding can begin. Gallager[31] introduced an adaptive version of Huffman coding (further developed by Knuth[32]) that alleviates some of the above problems. It is a fairly complex scheme, though, and it is difficult to implement in real time. In the meanwhile, a new class of universal statistical codes has emerged that generally outperform Huffman codes and lead to simple implementations. This class of powerful codes, collectively known as Arithmetic coding, can be thought of as a generalization of Huffman coding in which probabilities are not constrained to be powers of 2 and code lengths need not be integers.

2.2.2 Arithmetic Coding

Arithmetic coding[37] [6] is a compression method that maps a sequence of symbols to an interval of real numbers between 0 and 1. Shannon was the first to prove [21] that if the mapping is so constructed that the initial interval is divided among all possible sequences, allocating to each sequence a subinterval that is proportional to its probability, compression down to the source entropy is reached. Elias [33] presented a symbolwise recursive technique that accomplishes such a mapping. In its modern form, that leads to real-time dynamic implementations, arithmetic coding was introduced by Rissanen,[34] Pasco,[35] and further developed by Rissanen and Langdon.[36] General arithmetic coding compresses an m-ary symbol stream [6], but simplifies for binary symbols.[38] The technique of symbol decomposition [21] provides a means of converting any nonbinary source using an arbitrary but fixed alphabet size into an equivalent binary stream that can be coded by a binary code.

The encoding algorithm for arithmetic coding proceeds as follows (see Figure (3)): the "current interval" is initialized to [0,1), the interval of real numbers between 0 (inclusive) and 1 (exclusive). For each input symbol, the current interval is partitioned into subintervals, one for each possible alphabet symbol, with sizes proportional to the relative probability of occurrence of the input symbols, as estimated by the source model. The new "current coding" interval becomes the subinterval associated with the symbol that actually occurred in the input stream. This process continues until all input symbols are exhausted. Then, what is transmitted or stored instead of the original sequence (i.e., the coded message) is the binary expansion of a real number that lies in the final current interval. The number of bits in this expansion is so chosen that permits the decoder to distinguish the final current interval from all other possible final intervals. The decoding is also done recursively, using the same interval-partition process as the encoding. Since the code stream always points to a real number in the current interval, the decoding process is a matter of determining, for each reconstructed symbol, which subinterval is pointed to by the codestring.

In the encoding process, the length of the final subinterval is equal to the product of the probabilities of the individual symbols, which is the probability p of the particular sequence of input symbols. It can be shown that the number of bits in the binary expansion of the real number that distinguishes the final subinterval from all other possible subintervals is within one bit from -logp, the source entropy. Thus, arithmetic coding is optimal without the need for blocking the data. This method is also adaptive and does not need the probabilities of the symbols in the input in advance. These probabilities could be dynamically updated as the input is read and mapped into the interval. The system works as long as the encoder and the decoder have prior agreement about the precise way of estimating the probabilities. Furthermore, since

the encoding/decoding processes involve binary fraction arithmetic rather than concatenation of integer codewords, the more probable symbols can often be coded at a cost of much less than one bit per symbol. In all these respects, arithmetic coding is superior to the Huffman coding method. Arithmetic coding investigations in recent years have resulted in several practical coders. [6] [38] In particular, the state-of-the-art in binary adaptive arithmetic coding is presented by the Q-coder. [39] [40]

3 TWO-DIMENSIONAL LOSSLESS CODING SCHEMES

New developments in the desktop computer market (visualization, multimedia, color printers, high resolution monitors, digital video editing), and in the consumer electronic market (digital cameras, color facsimile, video games) are creating a tremendous need for high quality image compression systems. Lossless image compression poses a unique set of challenges. Images are highly correlated with respect to spatial, temporal, and chromatic variables. Sequential compression schemes, of the types described above, while fairly adept at exploiting the redundancy in repetitive symbol patterns, are less capable of removing correlations between pixel values in adjacent lines and frames of raster scanned images. High order context models are needed to capture the multidimensional structure of imagery data. Moreover, string matching algorithms do not usually perform well on picture data because, long repetitive patterns are only rarely found in pictures. A new class of codes is required for the efficient compression of imagery data. Several textbooks and review papers have been devoted to the subject of image compression. [1]- [5] In particular, run-length coding, contour coding, bit-plane coding, and block-coding are among those schemes that have been considered for use in image compression. Furthermore, recent developments in the area of sequential compression have led to a new generation of practical schemes which first compress the image using a highly efficient lossy algorithm, and then, encode the noisy residual with a lossless sequential coder.

3.1 Run-Length Coding

Run-length coding [41] [42] (RLC is one of the most extensively studied and simplest methods among the image coding schemes. With RLC sequences of identical input symbols (a run) are encoded as a count field (run-length) plus an identifier of the input symbol. It has extensively been applied to the compression of facsimile images. One- and two-dimensional versions of run-length encoding are part of both the Group 3 and the Group 4 CCITT standards [43] [44] for the compression of bilevel facsimile images as well as the STANAG 5000 standard for tactical digital facsimile equipment. In general, RLC is most practical for images with few gray scales (e.g., graphical images) and it does not perform well on halftone images.

3.2 Contour Encoding

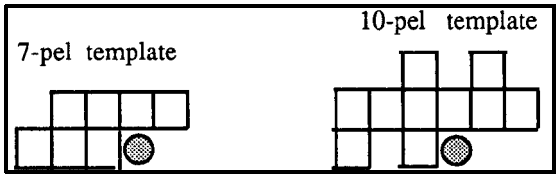
A contour-encoder functions by splitting an image into distinct objects; an object being a region within which all pixels have a constant grey-level. The image may then be represented by the boundaries or contours of the grey levels in each object. Each contour is uniquely encoded by specifying its grey level, the location of one point on its boundary, and a sequence of directionals which give the chain-code of the outer boundary of the object as it is traced in a clockwise direction. Each chain-code is composed of line segments connecting adjacent pixels in a horizontal, vertical, or diagonal direction.[45] [46] The basic premise of contour encoding as a means of data compression is that an image (especially, graphical images) will contain a relatively small number of contours compared to its total number of pixels. [47] [48] The efficiency of a contour-encoding scheme can be increased by further encoding the boundary information with a standard sequential scheme, like the LZ algorithm or an arithmetic coder.[49] Contour coding is a rather complex process, however, and suited only for relatively low picture transmission rates (e.g, video games, cartoon movies, etc.).

3.3 Bit-plane Coding

Constant wordlength PCM image coding can be conceptually organized into bit planes corresponding to the pixels' positions with the most significant bits occupying the lower plane. In most natural images the lower plane bits seldom change, while the upper plane bits fluctuate almost randomly. Standard bilevel image coding algorithms can be used to compress the individual bit planes. These algorithms can be designed separately to match the statistics of each bit plane. In order to increase the coherence of the bit planes, i.e., to create large uniform areas of 1's and 0's, the binary pixel values are usually converted into Gray coded [50] values prior to compression. [2]

Common bit-plane encoding algorithms involve the use of run-lengths of the 1's and 0's, or 2 D variable-block coding [2]. The most powerful approach, though, to compressing bilevel images has proved to be the arithmetic scheme of Langdon and Rissanen.[23] This scheme conditions the probability of a pixel's being black on a template of pixels surrounding it. Two different models have been used (shown in the following figure), wherein each binary pel is encoded based on the context of a set of 7 and 10 neighboring pels, respectively, selected from those above and to the left of the current one (the shaded circle marks the position of the pixel about to be coded). Each pixel's polarity is then coded arithmetically according to this probability. The application of the resulting binary adaptive arithmetic coder to raw bit plane data is straightforward.[49] Several investigations have explored the usefulness of conditional

modeling schemes for bilevel images [51] as well as bit plane data. [2] A similar approach has been used in the emerging JBIG standard as the coding scheme for compressing facsimile images. [52]



Another grey-scale image compression technique similar in principle to bit plane coding has been introduced by Rice. [53] Rice's scheme functions by removing the least significant k bits (k varying between 1, and 10 bits for a 16-bit input symbol) from the binary representation of the input symbol and coding the remaining bits with a variable length entropy coder. The removed k least significant bits are transmitted uncompressed along with the selected code word for the most significant part of the input symbol. This scheme has been employed by the Voyager 2 spacecraft for the transmission of imagery information, and it has been recommended by CCSDS for use as a basis for developing a standard data compression scheme for the transmission of telemetry data. [54] A VLSI implementation of Rice's scheme has recently appeared in the form of a chip set. [55]

3.4 Block Coding

Block coding was first developed for bilevel facsimile images. However, at the expense of some increase in complexity, it can be made adaptive and generalized to multilevel picture coding. [4] [56] [57] Block coding groups contiguous pixels in blocks of size nxm, where n and m are the number of pixels in the horizontal and vertical directions, respectively. These blocks are then encoded by an entropy coder according to their probabilities of occurrence.

3.5 Lossy Plus Residual Techniques

Several data compression algorithms have been devised which reduce image redundancy by first decreasing the amount of correlation in the imagery data through the use of an efficient lossy scheme and then encoding the residual data (the differential between the original image and the reconstructed lossy replica) via a lossless coder to achieve distortionless reconstruction. Such compression algorithms are generally known as "lossy plus residual" techniques. The most commonly used decorrelation methods can be grouped in three classes: predictive, transform, and hierarchical (multiresolution) decorrelation. [1] [3] [65] [67] One of the advantages of those techniques is that they lead themselves nicely to progressive transmission, [65] [66] i.e.,

sending a sequence of low-resolution component images that produces, in stages. an increasingly accurate representation of the original image. The motivation for progressive coding is: I) low bit rate transmissions (e.g., videoconferencing over telephone lines, where one wants to send a rough sketch of the image first with details following), ii) browsing through remotely stored pictures, iii) creation of pictorial indices for large databases, iv) the ability of adapting the amount of data compression and image quality to the particular needs of different users or to the variable resolution components of a large communication network, and, v) sending images from a transmitter threatened with imminent destruction.

3.5.1 Predictive Decorrelation

Among the various compression methods, predictive techniques have the special advantage of relatively simple implementation. Predictive schemes exploit the fact that adjacent pixel values from a raster image are highly correlated. With a DPCM codec, the transmitter (and receiver) predict the value of the current pixel on the basis of the pixels which have already been transmitted (received). Then, only the prediction error needs to be transmitted. If a good predictor is used, the distribution of prediction error is concentrated near zero and the error image has a significantly lower entropy compared to the original image. The number of pixels used in the predictor (its order), and their locations with respect to the pixel to be coded have a direct bearing on the predictor performance. [58][59][60] Extensive studies with a variety of data have shown that a third order linear predictor is adequate for most applications. [1] The prediction-error sequence can be efficiently compressed by a lossless sequential scheme (such as: the LZ algorithm, [49] the modified Huffman coding, [1] and Arithmetic coders[61]) to achieve distortionless reconstruction. The Lossless function of the now emerging ISO/CCITT Joint Photographic Experts Group (JPEG) standard for continuous-tone still image compression comprises a two-dimensional, second-order predictor, and an entropy encoding scheme (either Huffman or Arithmetic coding) for compressing the error-signal. [62]

3.5.2 Transform Decorrelation

The Discrete Cosine Transform [1] (DCT), the Walsh-Hadamard transform (WHt) and the Subband Coding [68] [69] are three of the most widely used transform methods. These techniques can be used as “lossy plus residual” codes with an entropy coding scheme encoding the residual signal. [3] [67] The DCT coder is part of the JPEG standard for compression of still images, [62] the MPEG video codec [64] as well as the CCITT H.261 visual telephony standard. [63] In particular, the JPEG scheme can be made progressive, and lossless. Transform coding schemes are particularly fit for progressive transmission since they can be easily arranged to send low spatial-frequency

components of the image (corresponding to low resolution representation of the image) in early stages of the transmission with higher spatial-frequency components following.

3.5.3 Hierarchical Decorrelation

Hierarchical decorrelation functions by generating a sequence of different resolution representations of the image. This allows progressive image transmission and reconstruction: first the lowest resolution image is received, after which the higher levels of the image structure add finer detail. The S-transform, [70] [72] the Laplacian Pyramid[71] and the Hierarchical Interpolation scheme [3] (HINT) are the three representative examples of hierarchical methods. With the HINT method an image is transmitted progressively starting from the lowest resolution pixels and followed by higher and higher resolution pixels. Once the pixels at any particular resolution level have been transmitted, the pixels at the immediately higher level are decorrelated through interpolation of the already transmitted values and encoded via a Huffman code. Several investigations [3] [70] have shown that the HINT method is the most efficient scheme among the class of "lossy plus residual" techniques.

4 APPLICATIONS - HARDWARE SOLUTIONS

The Lempel-Ziv algorithm has been accepted for use in a variety of lossless data compression standards: the QIC-122 standard for data recorded on Quarter-Inch-Cartridges (also submitted to the ANSIX3B5 committee for consideration as an ANSI standard), the CCITI V.42/V.42bis standard for dial-up modems, the ACR-NEMA [72] standard for the compression of medical images. As we have mentioned before, with the LZ77 scheme decompression is quick and easy, but the compression is slow. Because of the difference between compression and decompression speeds, LZ77 is appropriate and desirable for applications which are compressed once and decompressed often, such as CD-ROM [75] or on-line dictionaries, etc. The LZ78 algorithm leads to much faster implementations by sacrificing some compression performance. Fast systolic array implementations of LZ algorithms have been investigated,[73] and several single-chip implementations of the LZ algorithm are already available on the market. Stack Inc., provides a chip operating at 750 Kbits/s and implementing a proprietary scheme based upon the LZ77 processor. Recently, InfoChips Systems Inc., introduced the IC-105 chip operating at up to 1.5 MBytes/s compression and 3.5 MBytes/s decompression rates, which implements another proprietary scheme similar to LZ77 and is designed to be used with all types of mass storage devices. Hewlett-Packard, Co., which uses an LZ78 algorithm in its tapedrives,[77] has recently announced a single-chip implementation of this algorithm operating at 2.5 Mbytes/s. This algorithm is expected to become the second standard

for the QIC industry. It will first appear in Digital Data Storage (DDS) and Digital Audio Tape (DAT) systems manufactured by HP and Ardat. A number of other suppliers[76] have also announced their intent to ship DAT drives with data compression. What is more important though, is that the data compression technology is no longer limited to tape, but is finding its place in random access disk as well. [74] Disk compression is available today in a software version or with add-in coprocessor cards. Engineers are still faced with the challenging task of developing a single-chip data compressor that could be effectively applied to Winchester drives and other random access devices.

From the class of statistical compression schemes, Huffman coding has gained universal acceptance. It appears in all compression standards requiring some form of entropy coding. However, the class of Arithmetic codes is constantly gaining in popularity. An adaptive arithmetic scheme has been accepted by the JBIG committee for the now evolving standard for the compression of bilevel facsimile images. Arithmetic coding has also been part of the JPEG standard as an alternative option to Huffman coding. However, hardware implementations of Arithmetic coders have yet to become readily available. All available compressors conforming to the JPEG standard (e.g., the CL550 chip by C-Cube Inc., or the STI140 codec by SGS-Thompson) implement only the baseline function, employing the Huffman coding. So far, the only arithmetic coder that has been realized in hardware (as a single HCMOS chip[39]) is the IBM's bit-serial adaptive arithmetic coder (the Q-coder), designed for the compression of bilevel and grey-scale images. This chip achieves a throughput of approximately 1 Mbyte/s when applied to typical facsimile images. In general, the throughput rate of this algorithm varies with the compression ratio.

5 SIMULATION RESULTS - CONCLUSION

The performance of several of the algorithms described above is tested on a set of sample files. Table 1 is a comparison of compression performance of five of the sequential algorithms: 1) two-pass HC, 2) the LZ78 scheme, 3) the multisymbol arithmetic coding scheme of Witten, Neal and Cleary [6] (WNC) using a fixed distribution of probabilities for the input symbols based on typical english text, 4) the WNC scheme with an adaptive model, and 5) a statistical scheme employing a full first order Markov model, the tree decomposition technique for binarizing the input symbols and the binary Q-coder [39] for encoding the binary decisions. The files represent a variety of types and sizes; an on-line dictionary, unformatted english text, an engineering manual and a report formatted by a word processing package, a pin list from an engineering design, the c source code for the Unix utility compress, and the executable code for the Unix editor vi. All files consist of eight-bit input symbols. As a benchmark, with have also included in table 1 the zero-order (single symbol count)

entropy averages of the test files expressed in bits per symbol (bps). The HC approximated these entropy numbers within a few percent. (The entries in tables 1,2, denote the "effective number of bits per symbol" in the compressed data stream). Note that adaptive schemes performed better than the entropy measure, which is partially due to the fact that these schemes adapt to the local statistics of the source. As expected, the fixed WNC scheme failed to compress files with statistics drastically different from that of english text. But, even in the case of english text, adaptive models outperformed the fixed scheme. The Markov/Q-coder scheme achieved the best average performance, but the LZ78 scheme did better on two files and achieved a very good average performance. Note that the LZ78 scheme runs at about an order of magnitude faster than the Markov/Q-coder scheme. Table 1 compares the performance of lossless schemes on picture files. Both sequential (HC, LZ78, WNC/adaptive, Markov/NQ-coder) and two-dimensional schemes are tested. The two-dimensional schemes are: 1) a first-order, line-by-line predictive differential coding scheme followed by the sequential LZ78 encoder, and 2) the Independent function of the JPEG standard for lossless compression (two-dimensional, second-order lossless DPCM concatenated with a binary adaptive Arithmetic coder). The corpus of test files used consists of a set of six transverse head MR (Magnetic-Resonance) medical images with a 256x256x8-bit format. The JPEG scheme clearly outperformed all tested algorithms. Note that, even in this case of imagery data, the sequential schemes were able to perform better than the zero-order entropy measure.

In general the task of choosing the "best" compression algorithm for a specific application is quite complicated. In addition to the compression ratio there is a number of other factors that a system designer has to take into consideration when selecting the scheme that best fits into a given system. For example, the issues of implementation complexity, buffering requirements, encoder/decoder asymmetry, overall system compatibility are some important parameters in choosing a compression system.

6 REFERENCES

[1] A.K. Jain, "Fundamentals of Digital Image Processing" Prentice Hall, Englewood Cliffs, NJ 1989

[2] P. Melnychuk, M. Rabbani, "Survey of Lossless Image Coding Techniques," SPIE, Vol. 1075, 92-100, 1989

[3] P. Roos, M. Viergever, et al., "Reversible Intraframe Compression of Medical Images," IEEE Trans. Medical Imaging, vol 7(4), 328-336, Dec. 1988

[4] M. Kunt, O. Johnsen, "Block Coding of Graphics: A Tutorial Review," Proc. IEEE, Vol. 68(7), 1980

[5] A. Todd-Prokopek, “Image data compression: A survey,” in Mathematics and Computer Science in Medical Imaging, M. Viergever and A. Todd-Prokopek, eds., NATO ASI series, vol. 39. pp. 167-195, Springer-Verlag, 1988
[6] I. Witten. R. Neal, J. Cleary, “Arithmetic Coding for Data Compression,” Commi. ACM, Computing Practices, 521-540, 1987
[7] D. Lelewer, D. Hirschberg, “Data Compression,” ACM Computing Surveys, 19(3):261-296, Sept. 1987
[8] T.C. Bell, J.G. Cleary, I.H. Witten, “Text Compression,” Englewood Cliffs, N.J.: Prentice Hall, 1990
[9] J.L. Bentley, et al., “A Locally Adaptive Data Compression Scheme,” Commun. ACM, vol. 29(4), 320-330, 1986
[10] R. Horspool, G. Cormack, “A Locally Adaptive Data Compression Scheme,” Commun. ACM 16(2):792-794, Sept. 1987
[11] P. Elias, “Interval and Recency Rank Source coding: Two On-Line Adaptive Variable-Length Schemes,” IEEE Trans. Inform. Theory, vol. IT-33(l), 3-10, Jan. 1987
[12] E.R. Fiala, D.H. Greene, “Data Compression with Finite Windows,” Commun. ACM, vol. 29(4),490-505, April 1989
[13] J.A. Storer, “Data Compression: Methods and Theory,” Rockville, Md.: Computer Science Press, 1988
[14] R.N. Williams, “Adaptive Data Compression,” Kluwer Academic Publishers, 1990
[15] T. Welch, “A Technique for High Performance Data Compression,” IEEE Computer Magazine, 17(6), 8-19, June 1984
[16] J. Ziv, A. Lempel, “A Universal Algorithm for Sequential Data Compression,” IEEE Trans. Inf. Theory, IT-23(3), 337-343, 1977
[17] J. Ziv, A. Lempel, “Compression of Individual Sequences via Variable-Rate Coding,” IEEE Trans. Inform. Theory, IT-24(5), 530-536, Sept. 1978
[18] J.A. Storer, T.G. Szymanski, “Data compression via textual substitution,” J. ACM, vol. 29(4), 928-951, 1982
[19] R.N. Williams, “An Extremely Fast Ziv-Lempel Data Compression Algorithm,” IEEE Data Compression Conf., Snowbird, Utah, 1991, pp. 362-371
[20] V.S. Miller, M.N. Wegman, “Variations on a theme by Ziv and Lempel,” IBM Res. Rep. RC 10630 (#47798), 1984. Combinatorial Algorithms on Words, NATO ASI Series F, 12 (1985), 131-140
[21] Shannon, C.E., “A Mathematical Theory of Communication,” Bell Syst. Tech. J., vol. 27, 379-423, 623-656, July 1948
[22] J.J. Rissanen, G.G. Langdon, Jr., “Universal Modeling and Coding,” IEEE Trans. Inform. Theory, IT-27(l), 12-23, Jan. 1981
[23] Langdon, Glen, G., Jr., Rissanen, J.J., “Compression of Black-White images with Arithmetic Coding,” IEEE Trans. Commun., COM-29, 858-867, June 1981

[24] Langdon, Glen, G., Jr., Rissanen, J.J., "A Double adaptive file compression algorithm," IEEE Trans. Commun., COM-31, 1253-1255, Nov. 1983
[25] J. Cleary, I. Witten, "Data Compression Using Adaptive Arithmetic Coding and Partial String Matching," IEEE Trans. Commun., COM-32(4), 396-402, April 1984
[26] J.J. Rissanen. "Stochastic Complexity and Modeling," Ann. Statist. 14, 1080, 1986
[27] D.A. Lelewer, D.S. Hirschberg, "Streamlining Context Models for Data Compression," IEEE Data Compression Conf., Snowbird, Utah, 1991, pp. 313-322
[28] T. Bell et al., "Modeling for Text Compression," ACM Computing Surveys 21(4):557-593, Dec. 1989
[29] D.M. Abrahamson, "An adaptive dependency source model for data compression," Commun. ACM 32(l):72-83, Jan. 1989
[30] D.A. Huffman, :A method for the construction of minimum-redundancy codes," Proc. Inst. Electr. Radio Eng., 40(9), 1098-1101, Sept. 1952
[31] R.G. Gallager, "Variations on a theme by Huffman," IEEE Trans. Inform. Theory, 24(6), 668-674, Nov. 1978
[32] D.E. Knuth, "Dynamic Huffman coding," J. Algorithms 6(2), 163-180, June 1985
[33] N. Abramson, "Information Theory and Coding," McGraw-Hill, Inc., New York. 1968
[34] J.J. Rissanen, "Generalized Kraft Inequality and Arithmetic Coding," IBM J. Res. Develop. 20, 198-203, 1976
[35] R.C. Pasco, "Source Coding algorithms for fast data compression." Ph.D. dissertation, El.Eng., Stanford Univ., CA, May 1976
[36] J. Rissanen, G.G. Langdon, Jr., "Arithmetic Coding," IBM J. Res. Dev. 23(2), 149-162, March 1979
[37] Langdon, Glen, Jr., "An Introduction to Arithmetic Coding," IBM J. Res. Dev. 28(2), 135-149, March 1984
[38] Langdon, Glen, G., Jr., J.J. Rissanen, "A Simple General Binary Source Code," IEEE Trans. Inform. Theory, IT-28(5), Sept, 1982
[39] W.B. Pennebaker, J.L. Mitchell, G.G. Langdon, Jr., R.B. Arps, "An overview of the basic principles of the Q-Coder adaptive binary arithmetic coder," IBM J. Res. Dev. 32(6), pp. 717-726, Nov. 1988
[40] J.L Mitchell, W.B. Pennebaker,"Optimal hardware and software arithmetic coding procedures for the Q-Coder," IBM J. Res. Dev. 32(6), pp. 727-736, Nov. 1988
[41] S.W. Golomb, "Run-Length Encoding," IEEE Trans. Inform.Theory, IT-12 399-401, July 1961

[42] H. Tanaka, A. Leon-Garcia, "Efficient run-length encodings," IEEE Trans. Inform. Theory, IT-28(6), 880-890, 1982
[43] K. Kobayashi, "Advances in Facsimile Art," IEEE Comm. Magazine, 23(2), 27-35, 1985
[44] CCITT Recommendation T.6, "Facsimile coding schemes for Group 4 facsimile apparatus," Red Book (1984)
[45] H. Freeman, "Boundary Encoding and Processing," in "Picture Processing and Psychopictorics," B.Lipkin and A. Rosenfeld, eds., 241-266, Academic Press, 1970
[46] D. Graham, "Image Transmission by two-dimensional contour coding," Proc. IEEE, 55:336-346, Mar. 1967
[47] I. Sond, S. Kassam, "A New Noiseless Coding Technique for Binary Images." IEEE ASSP, vol. 34(4), 944-951, Aug. 1986
[48] M. Kunt, A. Ikonomopoulos, M. Kocher, "Second-Generation Image-coding Techniques," Proc. IEEE, Vol. 73(4), 549-574, 1985
[49] G.S. Yovanof. J.R. Sullivan, "Lossless Coding Techniques for Color Graphical Images," (Abstract only), IEEE Data Compression Conf., Snowbird, Utah, 1991, pp. 439
[50] M. Irshid, "Gray code weighting system," IEEE Trans. Inform. Theory, IT-33(6), 930-931, 1987
[51] A. Moffat, "Two-level context based compression of Binary Image," IEEE Data Compr. Conf., Snowbird, Utah, 382-391, 1991
[52] JBIG (Joint Bilevel Image Group) ISO-IEC/JTC1/SC2/WG9 & CCITT SG VIII, Draft Techn. Speification., Sept. 1990
[53] R.F. Rice, J.J. Lee, "Some Practical Universal Noiseless Coding Techniques," JPL Pub. 79-22, 1979
[54] Consultive Conunittee for Space Data Systems (CCSDS), "Universal Source Coding for Data Compression," Recommend for Space Data Systems Standards, White Book, March 1990
[55] J. Venbrux, N. Liu, "A Very High Speed Lossless Compression/Decompression Chip Set," (Abstract only) IEEE Data Compression Conf., Snowbird, Utah, 1991, pp. 461
[56] O. Mitchell, E. Delp, "Multilevel graphics representation using block truncation coding," Proc. IEEE, 68(7), 868-873, 1980
[57] J. Roy, N. Nasrabadi, "Hierarchical block truncation coding," Opt. Engin., 30(5), 551-556, 1991
[58] A. Rosenfeld, A.C. Kak, "Digital Picture Processing," vol. I & II Academic Press, 1982
[59] M. Kanefsky, Chung-Bin Fong, "Predictive Source Coding Techniques Using Maximum Likelihood Prediction for Compression of Digitized Images," IEEE Trans. Inform. Theory, IT-30(5), 722-727, Sept. 1984

[60] M. Rabbani, L. Ray, J. Sullivan, "Adaptive Predictive Coding with applications to radiographs," Medical Instrumentation, 20(4), 182-191, Jul-Aug. 1986
[61] P.G. Howard, J.S. Vitter, "New Methods for Lossless Irnage Compression Using Arithmetic Coding," IEEE Data Compression Conf., Snowbird, Utah, 1991, pp. 257-266
[62] Draft ISO 10198, "JPEG" Digital Compression and Coding of Continuous-tone Still Images," 1991
[63] CCITT Recommendation H.261, "Video Codec for Audio Visual Services at px64 kbits/s," 1990
[64] Committee Draft of Standard ISO 11172 "Coding of Moving Pictures and Associated Audio," ISO/MPEG 90/176, Dec. 1990
[65] K-H. Tzou, "Progressive image transmission: A review and comparison of techniques," Opt. Eng. 26, 581-589, 1987
[66] K. Knowlton, "Progressive Transmission of Gray-Scale and Binary Pictures by Simple, Efficient, and Lossless Encoding Schemes," IEEE Proc., 68(7):885-896, July 1980
[67] S. Elnahas, et al., "Progressive Coding and Transmission of Dicital Diagnostic Picture," IEEE Trans. Med. Imaging, MI-5(2),1986
[68] J. Woods, S. O'Neil, "Subband coding of images," IEEE Trans. Acoust., Speech, Signal Proc., ASSP-34, 1278-1288, 1986
[69] M. Smith, T. Barnwell, "Exact reconstruction techniques for tree-structured subband coders," IEEE Trans. Acoust., Speech, Signal Proc., ASSP-34, 434-441, 1986
[70] H. Blume, A. Fand, "Reversible and Irreversible Image Data Compression using the S-transform and Lempel-Ziv Coding," SPIE Medical Imaging III, vol. 1091, 1989
[71] R Burt, E. Adelson, "The Laplacian pyramid as a compact image code," IEEE Trans. Comm., COM-31, 532-540, 1983
[72] ACR-NEMA, "Digital Imaging and Communications Standard," Publication #300-1988 on page 7
[73] C. Thomborson, B. Wei "Systolic Implementations of a Move-to-Front Text Compressor," Proc. of the 1989 Symposium on Parallel Algorithms and Architectures, June 1989
[74] R.A. Monsour, D.L. Whiting, "Data Compression Breaks Through to Disk Memory Technology," Special Issue Computer Techn. Review, Vol. XI(6), 39-45, May 1991
[75] G.H. Steele, "Optical Disk Data Compression Foments Storage Revolution," Computer Techn. Review, XI(6), 53-60, May 1991
[76] K. Kelly, "Data Compression On Digital Audio Tape Closes Capacity Gap," Computer Techn. Review, XI(6), 77-84, May 1991

[77] M. Bianchi, et al., "Data Compression in a Half-Inch Reel-to-Reel Tape Drive," Hewlett-Packard J., 40(3), 1989

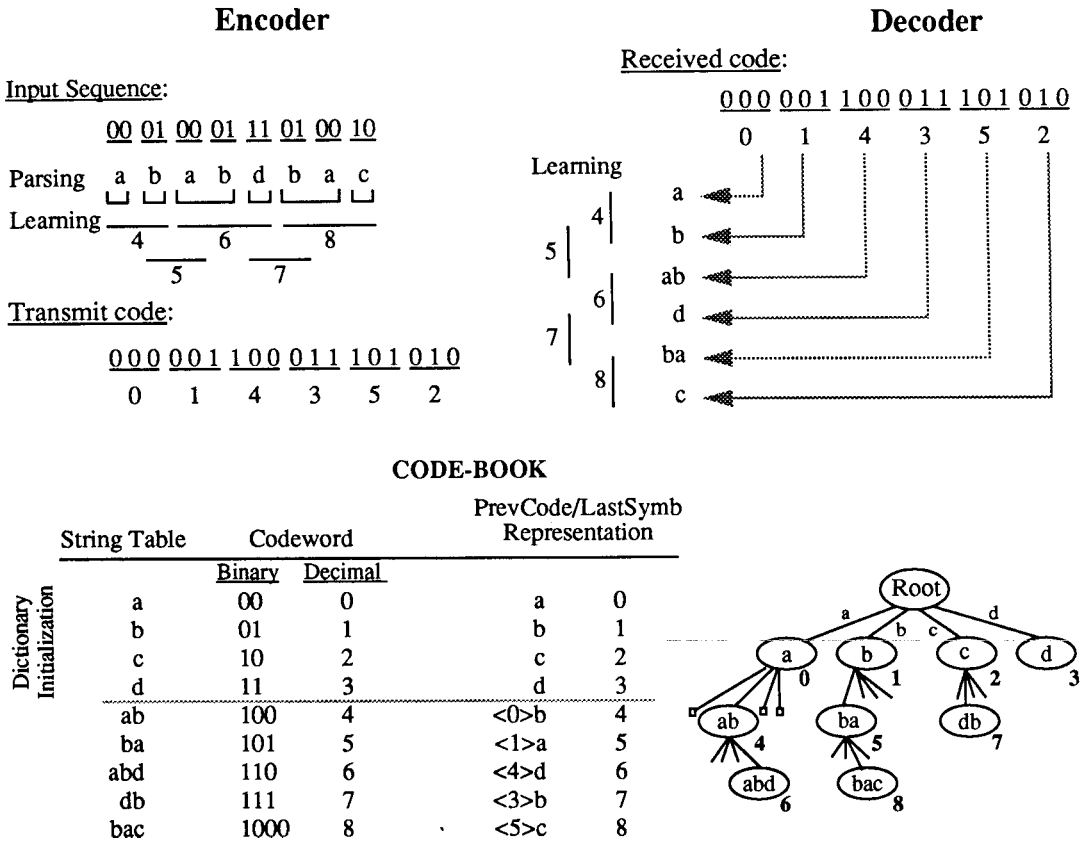


Figure (1) The Lempel-Ziv coding scheme (LZ78): principle of operation. Both the encoder and the decoder maintain a copy of the dictionary that changes in lock-step.

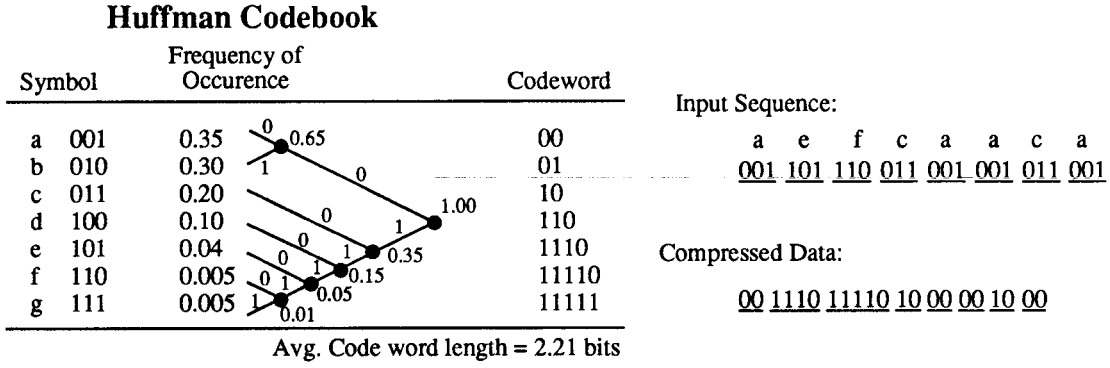


Figure (2) The Huffman compression/decompression process applied to a sample input sequence of three-bit symbols.

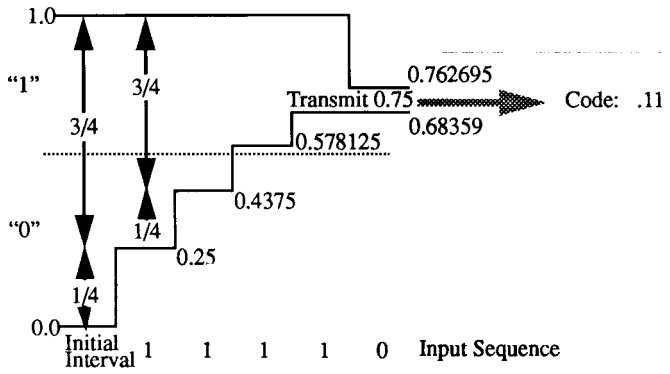


Figure (3) Binary Arithmetic Coding: An illustrative example.
(Fixed probability distribution: pr(0) = .25, pr(l) = .75)

Table 1 Lossless compression of sequential (byte-stream, 8 bit/byte) files

FILE	Uncompr. Size (Bytes)	[1]Entropy 0th-order	[2]Huffman	[3]LZ78	[4]WNC Fixed	[5]WNC Adaptive	[6]Markov Q-coder
Dictionary	206,672	4.42	4.45	4.10	4.99	4.34	**2.95**
English-text	558,261	4.71	4.76	**3.51**	4.84	4.71	3.96
Manual	230,400	5.40	5.45	2.77	(*) 10.51	5.08	**2.75**
Report	217,088	6.09	6.12	5.23	(*) 10.35	5.71	**4.44**
Pinlist	721,167	3.78	3.83	2.05	(*) 8.51	3.69	**2.01**
c_code	39,614	5.20	5.26	**3.86**	6.14	5.23	4.10
Executable	155,648	6.18	6.22	4.88	(*) 10.90	5.92	4.37
AVERAGE	304,121	5.11	5.16	3.77	(*) 8.03	4.95	**3.51**

(*): Expansion

Table 2 Lossless compression of raster-scan images (MR medical images)

FILE	Uncompr. Size (Bytes)	Entropy 0th-order	Huffman	LZ78	WNC Adaptive	Markov Q-coder	[7]Vert. DPCM LZ-9	[8]JPEG (DPCM+QC)
MRI-1	65,536	4.14	4.20	3.72	4.09	3.03	3.00	**2.78**
MRI-2	65,536	4.61	4.67	4.30	4.57	3.39	3.46	**3.15**
MRI-3	65,536	5.29	5.34	4.88	5.24	3.76	3.70	**3.40**
MRI-4	65,536	5.62	5.66	5.16	5.54	3.94	3.79	**3.49**
MRI-5	65,536	5.36	5.40	4.85	5.31	3.77	3.60	**3.32**
MRI-6	65,536	4.32	4.36	3.79	4.26	2.93	3.09	**2.77**
AVG	65,536	4.89	4.94	4.45	4.84	3.47	3.44	**3.15**

1: 0th-order Entropy in bits-per-symbol (bps)
2: Two-pass (non-adaptive) Huffman code
3: LZ78, 8-bit per input-symbol
4: Witten-Neal-Cleary code (fixed prob. distribution)
5: WNC code (Adaptive prob. distribution based on cumulative counts)
6: Full 1rst-order Markov model & Q-coder
7: Lossless Vert. 1rst-order DPCM & LZ78 with 8-bit per input-symbol
8: Two-dimensional, 2nd-order DPCM & Q-coder (lossless JPEG)