EEG Data Compression Techniques

Giuliano Antoniol,* *Member, IEEE*, and Paolo Tonella

***Abstract*— In this paper, electroencephalograph (EEG) and Holter EEG data compression techniques which allow perfect reconstruction of the recorded waveform from the compressed one are presented and discussed. Data compression permits one to achieve significant reduction in the space required to store signals and in transmission time. The Huffman coding technique in conjunction with derivative computation reaches high compression ratios (on average 49% on Holter and 58% on EEG signals) with low computational complexity. By exploiting this result a simple and fast encoder/decoder scheme capable of real-time performance on a PC was implemented. This simple technique is compared with other predictive transformations, vector quantization, discrete cosine transform (DCT), and repetition count compression methods. Finally, it is shown that the adoption of a collapsed Huffman tree for the encoding/decoding operations allows one to choose the maximum codeword length without significantly affecting the compression ratio. Therefore, low cost commercial microcontrollers and storage devices can be effectively used to store long Holter EEG's in a compressed format.**

***Index Terms*—Data compression, EEG signal, Huffman code.**

I. Introduction

INFORMATION theory is applied to data compression in many fields, to efficiently store or transmit texts, sounds, images, and signals. In this paper, different techniques for electroencephalograph (EEG) and dynamic or Holter EEG data compression will be discussed, with the requirement that compression should not prevent perfect reconstruction of the original information from the compressed one (such compression techniques are called "lossless").

The present work was performed in cooperation with the Neurological Department of the Santa Chiara Hospital in Trento where the hardware acquires up to 32 channels, with 8-b accuracy, at a maximum sampling rate of 1 kHz. However, in everyday practice, a minor number of channels and a lower sampling rate suffice. All results reported are referred to 128-Hz sampling rate per channel, 8-b accuracy, 20 channels (20 480-bps data stream), which is considered sufficient to achieve a good EEG signal quality. Lossy compressions can preserve relevant signal information to ensure that diagnostic errors are avoided. Nevertheless, due to the present lack of legislation and approved standards, in clinical practice physicians consider exact EEG reconstruction an essential requirement superior to better compression performance.

Manuscript received September 28, 1995; revised September 13, 1996. *Asterisk indicates corresponding author.*

*G. Antoniol is with the IRST-Istituto per la Ricerca Scientifica e Tecnologica, via Sommarive, Povo (Trento) 38050 Italy (e-mail: antoniol@irst.itc.it).

P. Tonella is with the IRST-Istituto per la Ricerca Scientifica e Tecnologica, Povo (Trento) 38050 Italy.

Publisher Item Identifier S 0018-9294(97)00287-5.

EEG data compression is desirable for a number of reasons. Primarily it decreases transmission time, archival storage space, and, in portable systems, it decreases memory requirements or increases channels and bandwidth. One of the first goals of this work was to automatically collect EEG's data acquired with the above characteristics (20 480 bps) from peripheral hospitals or patients' homes, through low speed transmission media such as switched telephone lines or cellular telephones, with low cost hardware, without the need of a physician's presence.

Data compression algorithms allowed the authors to realize a laptop system to send EEG's (20 channels, 128 Hz, 8 b), in real time over telephone lines with a 14 400-bps modem. A nurse is in charge to acquire the signal, and during acquisition physicians need only to communicate with him/her by phone. Hence, there is no longer the need for the patients to see a physician in person. Data are collected from the centralized Neurological Department, and diagnosis and treatment advice are successively sent back. This also results in a global reduction of cost, since patient transportation is no longer required.

Another motivation for data compression, as proposed in this paper, is for cases in which the amount of data to be stored overcomes the capabilities of commercial storage supports. Holters are long-term (e.g., 24 h) EEG signals, the acquisition and storage of which require small, low power devices that can be easily worn by patients. Holters are necessary when symptoms rarely appear, and clinical signs are difficult to detect from a short in-office recording. In our case, costs and technological limits of the available mass storage devices (40-Mb PCMCIA) impose to reduce the sampling rate from 128 to 64 Hz and the number of recorded channels from 20 to 12; nevertheless, the reduced signal quality results still acceptable for patients with pathologies requiring Holter examinations. Even with this reduced setup, 66 Mb are required to store a Holter of 24 h; therefore, compression techniques are useful and economically convenient.

The microcontroller (Hitachi H8-532, in our case) supervising the Holter acquisition device can be dedicated to data compression only during a fraction of the time between two consecutive input samples. Codeword length generated by compression algorithms could be very long (for infrequent inputs), causing loss of data due to the microcontroller limited computation capability. To cope with the analog-to-digital (A/D) converter data rate and low computational requirement, a compression technique based on a fixed maximum codeword length has been adopted.

To our knowledge, EEG lossless data compression has not been deeply investigated. Thus, widely employed compres-

sion algorithms (repetition count, Huffman coding), vector quantization, and techniques based on signal predictors (linear predictors, maximum likelihood, neural networks) were evaluated and are presented in this paper. Data compressors are compared with two widely used compression programs: *gzip* and *lharc*. The first is based on Lempel–Ziv code [1], [18], [19] and was developed under the GNU Project, by the Free Software Foundation; the latter is based on adaptive Huffman code [1], [19] and was developed by Tagawa *et al*. as freeware.

In the following, reference is made to a measure of *data compression* defined as

$$C = 100 \cdot \frac{L_{\text{orig}} - L_{\text{comp}}}{L_{\text{orig}}} \% \tag{1}$$

where L_{orig} and L_{comp} are the original and compressed file length.

Signal compression is performed by removing redundancy, which exhibits itself in terms of statistical dependence between samples. Prediction methods exploit time dependence and estimate the next sample from the previous ones. Spatial dependence between input channels is captured by lossless methods based on vector quantization which perform better than other compression strategies (62%). However, with a very simple prediction scheme, a 58% compression was achieved, allowing a real-time compressor to be implemented.

To cope with the strict time constraints a fixed maximum word length code was designed. It results that 16 b suffice to effectively compress both EEG's and Holter EEG's with a limited loss in performance.

Furthermore, experimental evidence is demonstrated that knowledge about EEG biologically relevant cues, e.g., alpha-waves, can only slightly improve the compression ratio.

The paper is organized as follows. Section II reviews Huffman coding, collapsed Huffman coding, and repetition count compression; Section III surveys the main predictive compression techniques; Section IV introduces transformation compression; Section V analyzes techniques to exploit EEG peculiarities; Section VI discusses experimental results on compressing 154 EEG's and four Holter EEG's; Section VII is devoted to conclusions and future work.

II. Data Compression

Lossless data compression can be achieved by assigning *short descriptions* to the most frequent symbols of a data source and necessarily longer descriptions to the less frequent symbols. For our purposes, the *descriptions* are binary strings represented by a variable length binary code.

It can be shown that there is no loss of generality in considering only *prefix codes*, in which a codeword cannot be a prefix of some other codeword. In fact, given an optimal data compression achievable with a variable length code, there exists a prefix code that achieves the same result [1].

Prefix codes are of special interest because they greatly simplify encoding (i.e., data compression) and decoding operation. In fact, they permit one to recognize a codeword without prior knowledge of its length, as a left-to-right scan of its bits can never simultaneously satisfy one codeword and the prefix of another codeword.

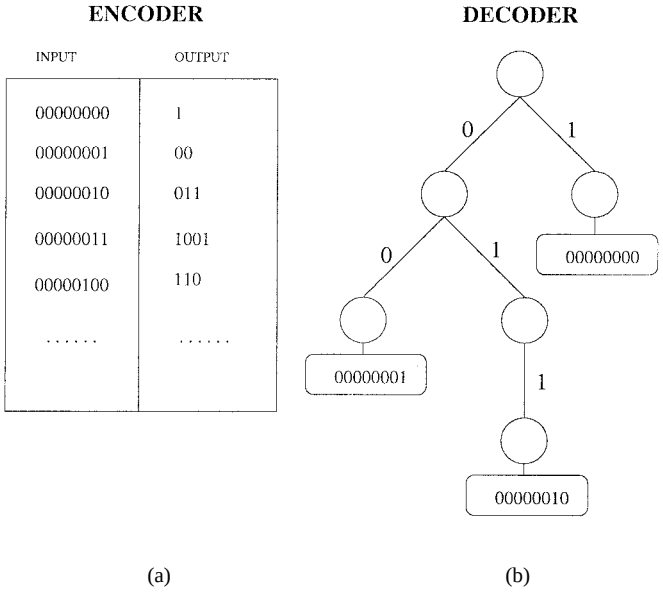

Fig. 1. Encoding/decoding scheme.

The basic operation of the compression algorithm follows the data structures shown in Fig. 1. Encoding is performed by concatenating the codewords corresponding to each symbol in the file and retrieved with a lookup table access. Decoding with a prefix code is also simple: a binary tree, the leaves of which are the given symbols, is a suitable prefix code representation for the decoding algorithm. The decoding steps are a sequence of left or right moves, according to zero or one inputs, until a leaf is reached.

Shannon theorem [1] expresses the upper bound of data compression in terms of *entropy*, a quantity based on the information source probability distribution, defined as

$$H = -\sum_{i=1}^{M} p_i \log_2 (p_i) \tag{2}$$

where p_i is the probability of the ith symbol of the source alphabet $\mathcal{A} = \{a_1, \cdots, a_M\}$. The maximum achievable compression is equal to

$$C_{\text{lim}} = 1 - \frac{H}{b} \tag{3}$$

where b is the original fixed number of bits per symbol.

A. Huffman Codes

The Huffman coding scheme provides optimal prefix codes through a greedy algorithm exploiting a binary tree (Huffman tree) [3]. As its compression ratio is very close to the limit expressed by (3), Huffman encoding is also called entropy encoding (see [1] for a more formal discussion of this topic).

Classical Huffman coding schemes cannot be used to obtain a code with a fixed maximum word length, because the maximum depth of the Huffman tree simply results from the code design algorithm itself, without any possibility of control. Hence, a technique based on the collapsed Huffman tree (CHT) is proposed. In a CHT, each leaf is associated with a variable

length bit string encoding the symbol on that leaf, and there exists exactly one leaf associated with a set of symbols instead of a single symbol. The idea is to make the tree leaves collapse so that long paths from the root are shortened, thus limiting the maximum bit string length of the code-words.

Let the alphabet of the source symbols be $A = \{a_1, \cdots, a_M\}$ and let $I = \{1, 2, \cdots, M\}$ be its set of indexes. With no loss of generality, one can assume that symbols are ordered according to their probabilities, i.e., if p_i is the probability of a_i, then

$$p_1 > p_2 > \cdots > p_M. \tag{4}$$

Let us assume that A is partitioned into two disjoint sets A_1 and A_2, with indexes in I_1, I_2. The CHT compression algorithm considers A_2 as a single symbol s to be encoded, with an associated probability

$$p_s = \sum_{i \in I_2} p_i. \tag{5}$$

Let us consider the problem of obtaining the exact reconstruction of the samples from a symbol a_k, belonging to A_2, and sharing the same prefix code generated for s with all the other elements of A_2. In such a case, the s codeword followed by the original, fixed length, b bits of the symbol a_k is emitted. In the decoding step, when the bit string associated with s is found, the successive b bits are interpreted as the original symbol.

The compression ratio achieved by this technique is

$$C = 1 - \frac{H}{b} - p_s \tag{6}$$

where H is the entropy of symbols $A_1 \bigcup \{s\} = \{a_{i_1}, \cdots, a_{i_m}, s\}$. The term p_s accounts for the probability of symbols belonging to A_2.

The maximum codewords length constraint is embedded in the value chosen for m (i.e., the cardinally of A_1), because the final Huffman tree has $m+1$ leaves, hence the number of bits of a code-word is computed by summing up the depth of the CHT leading to the leaf associated to that code-word plus, if necessary, b bits due to the encoding of the original bit sequence, when the leaf represents A_2. Since a code could be regarded as a partition of A into the two sets A_1, A_2, in order to design the optimum code a set of indexes I_1^* has to be chosen such that

$$\min_{I_1} \left\{ -\frac{1}{b} \sum_{i \in I_1} p_i \log_2(p_i) - \frac{1}{b} p_s \log_2(p_s) + p_s \right\}. \tag{7}$$

Unfortunately, it is possible to choose the set I_1 of m indexes from $\{1, \cdots, M\}$ in a number of ways equal to the binomial of m over M. It is obviously impossible to explore the whole search space, therefore we have chosen to determine the sets I_1, I_2 with the following simple heuristics:

$$\begin{aligned} I_1 &= \{1, \cdots, m\} \\ I_2 &= \{m+1, \cdots, M\}, \end{aligned} \tag{8}$$

As probabilities are ordered, this partition ensures that symbols with high probability have their own codeword, while infrequent symbols are collapsed into one single leaf of the Huffman tree. It is not excluded that this solution corresponds to a local minimum, because the function to be minimized is composed by elementary terms that are not monotonic ($-x \log_2(x)$ with $x \in [0, 1]$), so that it is not simple to derive the behavior of the cost function (7) when terms p_i, p_j are exchanged between I_1, I_2.

B. Repetition Count Compression

If a file contains long sequences of repeated symbols it is possible to compress it, reducing symbol repetitions by means of run-length compression techniques [20]. In this paper we adopt a run-length compression variant, the *repetition count compression*, in which each symbol is followed by the *repetition counter*, representing the number of subsequent occurrences of that symbol. This technique is advantageous when the overhead introduced by the counter bits is compensated by the repeated symbols deletion. On the repetition count compressed data stream Huffman coding is applied to further reduce it. However, the entire process does not guarantee a higher compression ratio compared to the original data entropy encoding.

More formally, if H is the entropy of the source and N is the number of samples, $L = NH$ is the file length produced by an entropy code. By using a repetition counter, the number of symbols to be stored in the compressed file is lowered to $N/(R+1)$, where R is the repetition counter average value. In such a case, the number of bits per symbol has to be augmented by the repetition counter bits (say B), and the length of the file produced by the repetition count compression becomes $L' = N(H+B)/(R+1)$, under the assumption that the entropy of the symbols remains unchanged, which is an optimistic hypothesis because, in general, entropy is supposed to increase when repeated symbols are deleted from the data stream. Consequently, L' is supposed to increase with respect to this approximated value. A compression improvement requires $L' < L$, which implies the following inequality:

$$HR > B. \tag{9}$$

Therefore, a compression advantage is achieved when Huffman encoding of the repeated symbols is larger than the number of bits required to represent the repetition counter.

III. Predictive Compression Techniques

Fig. 2 shows a predictor scheme for performing an invertible transformation widely used to reduce the entropy of the signal: the lower the entropy of the transformed signal, the greater the compression ratio. A prediction of a signal sample based on some past values is made by a unit called *predictor*, and the error between the actual value and the predicted one is coded with a variable length code (e.g., the Huffman code). Such error can be expressed by the following difference:

$$e_n = x_n - f(x_{n-1}, \cdots, x_{n-N}) \tag{10}$$

where x_n is the nth input sample (a shorthand for $x(nT)$ with $1/T$ the sampling rate) and e_n is the nth value of the discrepancy between the input sample and the predicted one.

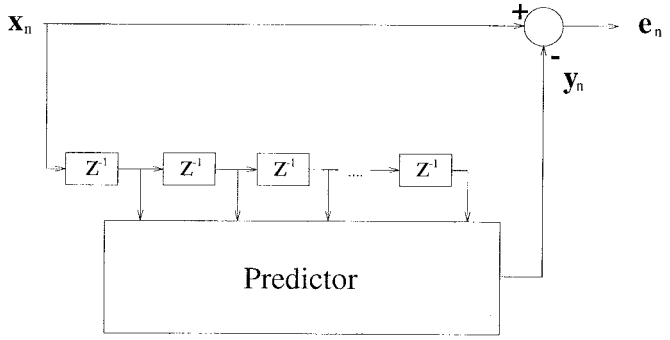

Fig. 2. Signal transformation based on a predictive scheme.

If the prediction is accurate most of the time, prediction errors are close to zero, and thus short bit strings can be used for encoding them and long bit strings for other values of e_n.

Five different kinds of predictors were tested. They are, respectively, based on Markov chains, digital filtering, linear predictive coding (LPC), adaptive linear prediction, and artificial neural networks (ANN's).

A. Markov Predictor

The predictor shown in Fig. 2 can be implemented under the assumption that the signal has Markovian properties and is generated by a source modeled as a Markov chain.

A first-order Markov chain was used to model the signal; it requires the estimation of all conditional probabilities of the type $P[\mathcal{X}_n = a_i|\mathcal{X}_{n-1} = a_j]$, where $\mathcal{X}_n$ is a discrete random variable taking values on the finite alphabet $\mathcal{A} = \{a_1, \cdots, a_M\}$. These probabilities can be approximated with frequencies if a big enough training set is available. A frequency matrix, whose elements F_{ij} count the occurrences of $\mathcal{X}_n = a_i$ preceded by $\mathcal{X}_{n-1} = a_j$, was derived from a training set (or train set) consisting of 96 EEG's; a different test set consisting of 58 EEG's was used for evaluation. Matrix F_{ij} corresponds to the joint probability distribution of $\mathcal{X}_n$ and $\mathcal{X}_{n-1}$ and is used in (11) to estimate the conditioned probability $\Pi_{ij} = P[\mathcal{X}_n = a_i|\mathcal{X}_{n-1} = a_j]$

$$\prod_{ij} \simeq \frac{F_{ij}}{\sum_k F_{kj}}. \tag{11}$$

As reported in [1], $\mathcal{X}_n$ can be predicted from $E[\mathcal{X}_n|\mathcal{X}_{n-1}]$. Therefore, the following estimate for the successor of the symbol a_j minimizing the mean squared error can be used

$$\text{succ}\ (a_j) = \sum_k k \prod_{kj} \qquad \forall a_j \in \mathcal{A}. \tag{12}$$

As data are quantized with 8 b, there are 256 symbols generating a 256×256 frequency matrix; experimental evidence was found, on the test set, that the differences between the identity function and the most probable successors of symbols $0 \cdots 255$, computed according to (12) range from -3 to $+3$, denoting that the Markovian estimate is generally close to identity.

B. Digital Filtering Predictor

It is possible to design a digital linear predictive (LP) filter for the EEG signals and to use it according to the predictive scheme of Fig. 2. Let us consider the digital filter described by the difference equation

$$e_n = x_n - b_1 x_{n-1} - b_2 x_{n-2} - \cdots - b_N x_{n-N}. \tag{13}$$

The predictor of Fig. 2 is represented by the following equation using the same set of coefficients $\{b_1 \cdots b_N\}$:

$$y_n = b_1 x_{n-1} + b_2 x_{n-2} + \cdots + b_N x_{n-N}. \tag{14}$$

It is possible to obtain a low entropy error signal e_n by minimizing the total square error

$$E = \sum_n e_n^2. \tag{15}$$

Under the following conventions:

$$\begin{aligned} \overline{b}^t &= [b_1 \cdots b_N] \\ \overline{x}^t &= [x_{n-1} \cdots x_{n-N}] \end{aligned} \tag{16}$$

the minimum mean squared error linear predictor of the form $\hat{x} = \overline{b}^t \cdot \overline{x}$ is given by a solution $\overline{b}$ of the equation

$$\overline{b}^t \cdot \overline{R}_x = E[x_n \overline{x}^t] \tag{17}$$

where $\overline{R}_x$ is the correlation matrix of process x_n. In practice, the true autocorrelation function is not known and an estimate obtained from a set of samples is used (see [1] and [2] for more details).

As tangible improvements were not observed with a high value of N, only the results of a second-order filter will be reported in Section VI.

C. Linear Predictive Coding

LP relies on the hypothesis that data are realizations of a discrete-time stationary real-valued random process the mathematical description of which is known. In practice, the mathematical description is rarely available. Furthermore, the stationarity hypothesis is hardly ever valid for the whole signal but it can be reasonably assumed for a short signal frame [7]. Thus, the usually adopted approach is to find the best predictor on a short signal frame of the available data by using (13)–(17). The technique that encodes the original waveform with the computed coefficients on a sequence of frames is called LPC.

To reconstruct the waveform, for each frame, the parameters of the LP and the initial values of the signal are used. By minimizing the square error on a frame, LPC cannot allow perfect waveform reconstruction. However, if the prediction errors are also transmitted, exact reconstruction is possible. Thus, in a simple form, the transmitted information contains, for each frame: the LP coefficients $\{b_1 \cdots b_N\}$ (or their quantized representations), the first N samples and, for each subsequent point, the prediction errors.

D. Adaptive Linear Prediction

Two adaptive linear predictors, the *least mean square* and the *sign adaptive* linear predictor, were tested.

Predictor coefficients in (14) could be thought of as time dependent functions, capable of adapting to signal behavior. The least mean square algorithm updates predictor coefficients according to the equation

$$b_i(n) = b_i(n-1) + \beta x_{n-1} e_n \tag{18}$$

where β weights the adaptation rate, and e_n is the prediction error at time n.

The predictor coefficients in the sign adaptive algorithm are updated according to the equation

$$b_i(n) = b_i(n-1) + \Delta x_{n-1} \text{ sgn } (e_n) \tag{19}$$

where Δ weights the adaptation rate, and sgn (e_n) takes the value $+1$ or -1 according to the prediction error sign. As reported in [12], the selection of β and Δ values is critical: while low values do not lead to a sufficient adaptation, high value could cause system instability. In the present work these values were experimentally determined (see Section VI) to ensure system stability and high compression ratios.

E. Artificial Neural Network Predictor

A predictor can be implemented by an ANN, which uses a certain number of past samples to predict a future event trying to *track* the signal.

To implement the tracking ANN, a multilayer perception architecture was chosen. The ANN was trained with resilient propagation [10] with N input, H hidden nodes and one output $(N-H-1)$. The symbols to be encoded are given by

$$e_n = x_n - g(x_{n-1}, \cdots, x_{n-N}) \tag{20}$$

where $g(x_{n-1}, \cdots, x_{n-N})$ represents the output of the ANN.

The ANN was trained with a set of patterns comprising N past samples of the signal as inputs and the actual value as output. More details of this ANN application with different network topologies can be found in [8].

IV. Transformation Compression

A sequence of N signal samples can be considered as being a point X in an N-dimensional space. A more efficient representation of X can be obtained by applying an orthogonal transformation $Y = TX$, where Y denotes the transformed vector and T denotes the transformation matrix [16]. The objective is to select a subsequence of Y containing M components, where M is substantially less than N [the remaining $(N-M)$ components are discarded] leading to compression. By coding with Huffman the difference between the original signal and the signal reconstructed from the M components, these techniques turn out to be lossless, because the exact signal can be recovered from the M components and the differences. It is possible to show [16] that the Karhunen–Loeve transformation (KLT) is the *optimum* one for signal representation with respect to the mean square error criterion. The main disadvantage in using KLT is a substantial computation time.

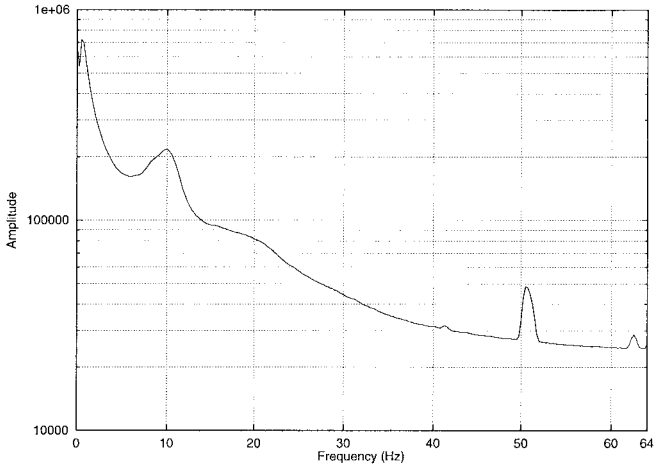

Fig. 3. Average (see text) EEG spectrum as computed by the DCT. Peak at 10 Hz is due to alpha rhythm, and peak at 50 Hz is due to power lines.

The discrete cosine transform (DCT) is a suboptimal solution with computational advantages, in fact, fast algorithms [16] exist. This purely real transform for a real input vector makes it appealing for EEG compression. DCT is also used for electrocardiogram (ECG) data compression [12] and, furthermore, it has frequently been reported that DCT achieves performance close to KLT.

Fig. 3 shows the average EEG spectrum as computed by the DCT on the whole corpus. A considerable amount of spectral power is located at low frequencies, but the rate of spectrum decrease is not very high. Therefore, DCT based compression ratio is not expected to be high since deleted components are likely to introduce significant errors. It is interesting to note a peak located at about 10 Hz, corresponding to the most important EEG waveforms, the alpha waves, ranging from 8 to 13 Hz. A second peak at about 50 Hz represents the electrical network interference and is usually removed by a digital notch filter before visualization. As biologically relevant components lie under 20 Hz, corresponding to 50 of 256 components, the first 50 DCT components were retained in the compression process.

V. Exploiting EEG Peculiarities

The data compression techniques presented so far make little or no use of EEG knowledge. In previous works related to ECG signal compression, it was shown that the use of domain dependent knowledge is extremely important and has produced, as a result, the best compression strategies (e.g., the *beat subtraction* compression) [11]–[13]. EEG data have a temporal correlation and a spatial correlation that can be exploited to design compression strategies.

As far as spatial correlation is concerned, viable techniques are *vector quantization* and *multivariate time series analysis* [17]. In this paper vector quantization is exploited by mapping a vector comprising the input channels (or derivatives) samples to a code vector, and encoding, in addition, the errors vector.

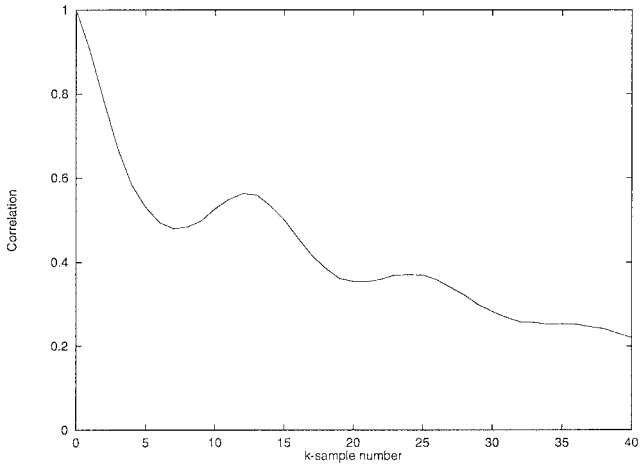

Fig. 4. Average time correlation for the EEG signal (sampling frequency = 128 Hz).

Time correlation can in turn be taken into account by modifying the general approach to predictive compression of Fig. 2 and adding some delayed inputs to the predictor.

A. Predictive Compression with Delayed Inputs

Fig. 4 shows the average time correlation computed on an EEG training set corpus. It is clear that some time correlation exists not only for the samples that are close to each other, but also for samples with a time lag of about 12. The maximum achieved for a time delay of 12 samples corresponds to 10.6 Hz. This result has a biological explanation; in fact, alpha-waves, typical waveforms for normal EEG's, range from 8 to 13 Hz. To take advantage of the alpha-waves correlation peak we propose to modify (10) to account for longer time lag. The approach, applied by [11] to linear filter for ECG's data compression, is generalized to the previously described signal predictors.

Let $\mathcal{N} = \{k | \alpha \leq k \leq \beta\}$ determine the neighborhood of the time lag to be considered allowing (10) to be rewritten as

$$e_n = x_n - f(x_{n-1}, \cdots, x_{n-N}, x_{n-\alpha}, \cdots, x_{n-\beta}). \quad (21)$$

Correlation values of Fig. 4 suggest that $N = 5$, $\alpha = 10$, and $\beta = 15$ allow the predictor to be fed with the most correlated past samples. Actually, experimental tests show that N and $\beta - \alpha$ can be lowered without decreasing the performance of the predictor, while making it simpler and faster. For this reason only the results with $N = 2$, $\alpha = 12$, and $\beta = 13$ are reported.

B. Vector Quantization of the EEG Signal

A Vector Quantizer (VQ) of dimension k and size N is a mapping Q from a k-dimensional vector, in an Euclidean space R^k, into a finite set C containing N output or reproduction points, called *code vectors* or *codewords*

$$Q: R^k \rightarrow C \quad (22)$$

where $C = \{y^{(1)}, y^{(2)}, \cdots, y^{(N)}\}$ and $y^{(i)} \in R^k$ for each $i \in \{1, 2, \cdots, N\}$. The set C is called *codebook* or, simply, the *code*.

Vector quantization can be applied in a natural fashion to EEG signals by considering the vector of the input channels, and by building a codebook for this set of vectors [1]. A slight variation is to consider the derivative instead of the channels' values since EEG derivatives are likely to be close to zero. Thus, with the same codebook size, a lower average distortion can be achieved.

For lossless signal compression, the label of the code vector must be stored with the errors vector, that is the difference between input and code vectors.

VI. Results

This section presents results of the different compression techniques applied to a corpus of EEG examinations randomly extracted from the Hospital archives of the last two years. Results on the compression of Holter EEG's are presented as well. The compression performance degradation associated with the use of a fixed maximum word length code is then discussed.

A. Unbounded Maximum Word Length Compression

EEG's were recorded at S. Chiara Hospital from 20 input channels, with 8-b accuracy, at a sampling rate of 128 Hz. The database of 154 EEG's (global data set) was split into two disjoint subsets, the train set (96 files) and the test set (58 files). The train set was used to compute an average frequency distribution that has allowed the construction of an "average" Huffman tree. Effectiveness of this tree was measured on the test set.

The use of a file-independent "average" Huffman tree simplifies the operations made by the encoder and the decoder, resulting in a reduction of the compression/decompression times and system complexity. The availability of the "average" Huffman tree also makes it possible to encode samples on-line while they are acquired, because it is not necessary to wait for the entire sample sequence to compute frequencies, before starting the encoding process.

Whenever a signal is stored in a file its probability distribution is exactly known. In this case a file-dependent code is designed, and compression begins only after this preliminary analysis on the file. Data referred to as Huf* in Table I (and, in general, marked with an * in the other tables) are obtained under this assumption (i.e., the perfect knowledge of the probability distribution), while Huf data are file independent. Parenthesized values for train and test set denote that the distinction between these two sets is only due to the computation of an average probability distribution on the train set and its use on the test set. Unparenthesised values refer to techniques using the train set to determine their internal parameters (besides the average probability distribution) and the test set for the generalization capability.

As the derivative of the EEG signal has generally low values, it can be coded more effectively than the signal itself. A discretized value of the first derivative can be obtained with

TABLE I
COMPRESSION PERCENTAGE (1) ON TRAIN SET (96 EEG'S), TEST SET (58 EEG'S), AND GLOBAL SET (ALL 154 EEG'S). DATA ARE SAMPLED AT 20 480 BPS (20 CHANNELS, 128 HZ, 8 B). ASTERISK INDICATES THE USE OF A FILE-DEPENDENT CODE (INSTEAD OF AN AVERAGE CODE). PARENTHESIZED RESULTS USE THE TRAIN SET ONLY TO DETERMINE THE AVERAGE CODE, WHILE UNPARENTHESIZED RESULTS USE IT (ALSO) TO DETERMINE THE INTERNAL PARAMETERS OF THE TECHNIQUE

Technique	Global	Train Set	Test Set
Gzip	38.8	-	-
Lharc	38.9	-	-
Huf	39.5	(39.4)	(39.7)
Huf*	40.9	-	-
Deriv	57.6	(57.7)	(57.4)
Deriv*	58.4	-	-
Markov	57.8	57.9	57.6
Markov*	58.6	58.7	58.4
Filter	57.0	(57.3)	(56.7)
Filter*	58.0	-	-
ANN	55.0	55.0	54.9
ANN*	55.9	55.9	55.7
LPC	59.7	(59.9)	(59.5)
LPC*	60.2	-	-
Adapt-LMS	58.1	(58.3)	(57.9)
Adapt-LMS*	58.8	-	-
Adapt-SGN	58.1	(58.3)	(57.9)
Adapt-SGN*	58.9	-	-
DCT	47.3	(47.2)	(47.3)
DCT*	47.9	-	-
VQ	59.0	59.3	58.7
VQ*	62.0	62.2	61.6

a simple, reversible computation corresponding to the trivial predictor $f(x_{n-1}, \cdots, x_{n-N}) = x_{n-1}$ (see Fig. 2).

By encoding the first derivative with Huffman code the results shown in Table I were obtained. Other different predictors (Markov, digital filter, LPC, ANN, etc.) were tested and compared with the first derivative (see Table I for a comparison).

The digital filter uses two past values (x_{n-1}, x_{n-2}) to make the prediction (second-order filter). Higher-order filters were tested, but they did not produce any significant improvement. This result is similar to the one obtained for the ECG's and reported in [11] and [13], showing that no substantial improvement is achieved by increasing the order of the filter from two.

The ANN predictor is a multilayer perception with architecture 4-2-1, and was trained with resilient propagation [10] with a learning rate equal to 0.1. The inputs to the network are the four past values of the signal (x_{n-1}, x_{n-2}, x_{n-3}, x_{n-4}), while what the network is trained to produce is the actual value (x_n). As the EEG's available for training provide a huge number of patterns, they were subsampled in order to build a tractable training set. A subset of 2000 training patterns was randomly extracted for each file in the train set, giving rise to a total of 192 000 training patterns. The train set has been presented to the network 1000 times (epochs = 1000), resulting in a mean square error of 0.000 22 (data were normalized in the range [0, 1]).

Results on adaptive linear prediction refer to a second-order filter (x_{n-1}, x_{n-2} are used to predict x_n). Compression percentages reported in Table I were obtained with $\beta = 0.000\,000\,5$ and $\Delta = 0.000\,01$, values experimentally determined to grant one order of magnitude as stability range.

The LPC technique gives interesting results. Several window widths (from 128 to 1024) and prediction orders (from 2 to 16) were tested, and no substantial variation in the final compression factor was achieved. Therefore, since there was no substantial loss in performance, Table I outcomes refer to a Hamming window of 128 samples with a prediction order of four, corresponding to the most correlated samples, as resulting from Fig. 4.

DCT compression ratios are poor compared to the others; this is not an unexpected result, since the discarded spectral components were shown to contain significant spectral power. Results presented are based on the first 50 components (with a transform of 256 points), corresponding to the biologically relevant part of the spectrum.

Finally, the VQ technique gives very good results, corresponding to the expected spatial correlation between channels. The codebook used in the present experiment consists of 256 code vectors, addressable with 8 b (representing the overhead with respect to error encoding). The derivative (instead of the pure value) was quantized since its values are concentrated around zero, due to the short term temporal autocorrelation. Tests with different codebook sizes were performed (512, 1024), with negligible compression improvements (less than 2%) compared to the increase in quantization time.

Table II summarizes results exploiting the EEG alpha-waves autocorrelation, i.e., predictors also make use of long delayed samples. Many experiments with different short and long delays were conducted. Results reported here refer to the choice of the best performing delays. The digital filter is substantially a *subset auto regressive* predictor as described in [11], and uses x_{n-1}, x_{n-12}.

The ANN predictor was trained with the same parameters as in Table I, except for the inputs, which become x_{n-1}, x_{n-2}, x_{n-12}, x_{n-13}.

LPC with both short and long time delay is a *subset auto regressive* predictor [11]; Table II results refer to the same parameters as Table I, augmented with $\mathcal{N} = \{12, 13\}$ according to (21).

Adaptive linear predictors were simply modified by increasing the delay of the second term from 2 to 12.

ANN predictors, LPC, and adaptive linear predictors show an increase in performance when long delay samples are

TABLE II
TECHNIQUES EXPLOITING ALPHA-RHYTHM TIME AUTOCORRELATION. COMPRESSION PERCENTAGE (1) ON TRAIN SET (96 EEG'S), TEST SET (58 EEG'S), AND GLOBAL SET (ALL 154 EEG'S). DATA ARE SAMPLED AT 20 480 BPS (20 CHANNELS, 128 HZ, 8 B). ASTERISK INDICATES THE USE OF A FILE-DEPENDENT CODE (INSTEAD OF AN AVERAGE CODE). PARENTHESIZED RESULTS USE THE TRAIN SET ONLY TO DETERMINE THE AVERAGE CODE, WHILE UNPARENTHESIZED RESULTS USE IT (ALSO) TO DETERMINE THE INTERNAL PARAMETERS OF THE TECHNIQUE

Technique	Global	Train Set	Test Set
Filter	58.1	(58.3)	(57.8)
Filter*	58.9	-	-
ANN	56.0	56.1	55.8
ANN*	56.7	56.7	56.6
LPC	60.9	(61.0)	(60.6)
LPC*	61.4	-	-
Adapt-LMS	58.1	(58.3)	(57.9)
Adapt-LMS*	58.8	-	-
Adapt-SGN	58.5	(58.6)	(58.4)
Adapt-SGN*	59.2	-	-

used, but the increase does not always justify the complexity overhead of the modified algorithm. The poor improvement in compression factors of Table II can be explained by the low correlation of the twelfth and thirteenth samples and the nonstationarity of the EEG signal (correlation of Fig. 4 is an average correlation). Therefore, our experiments indicate that the alpha peak cannot be efficiently exploited.

B. Repetition Count Compression

The repetition count compression technique was tested both with the original sequence of symbols and with the sequence produced by the derivative predictor.

Figs. 5 and 6 show the plot of maximum, minimum, and average value for the product HR versus B computed on the 154 file corpus. The useful region lies over the line $HR = B$ [see (9)], where some benefit can be obtained with this technique. It is clear from the plots that the repetition count compressor is not convenient, with the derivative data stream.

For the original data stream, there are some files for which the technique offers some improvement, because the maximum values curve of HR is over the identity for B ranging from 1 to 2 b. For this reason the repetition count compression was applied only to the original data stream, with 1–3 b dedicated to repetition counting. Results on the application of this technique to the whole corpus are reported in Table III.

C. Fixed Maximum Word Length Compression

The solution proposed makes use of the CHT in conjunction with the signal derivative, limiting in this way the maximum word length to a fixed bit number.

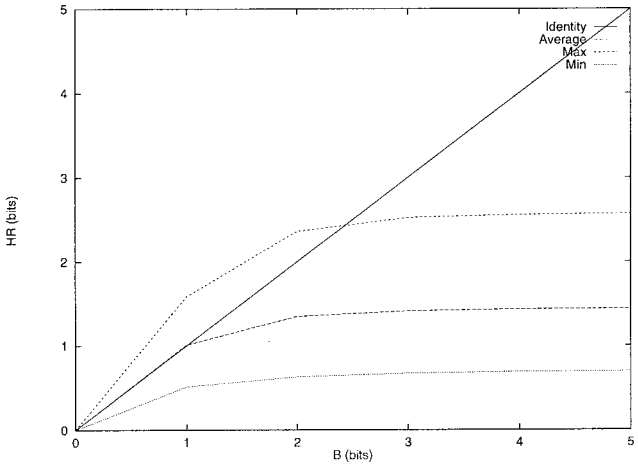

Fig. 5. Original data stream: entropy encoding of the repeated symbols (entropy H times repetition count R) versus repetition counter bits (B). $HR > B$ makes repetition counting convenient respect to entropy encoding.

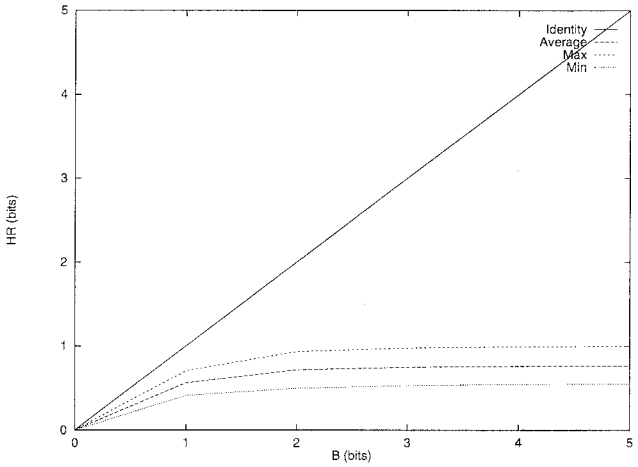

Fig. 6. Derivative data stream: entropy encoding of the repeated symbols (entropy H times repetition count R) versus repetition counter bits (B). $HR > B$ makes repetition counting convenient respect to entropy encoding.

TABLE III
REPETITION COUNT COMPRESSION [PERCENTAGE VALUES, (1)] ON THE ORIGINAL DATA STREAM FOR REPETITION COUNTER BITS B RANGING FROM 1 TO 3. THE GLOBAL SET, COMPRISING 154 EEG'S SAMPLED AT 20 480 BPS (20 CHANNELS, 128 HZ, 8 B), HAS BEEN USED

Data Set	Rep(1)	Rep(2)	Rep(3)
Global	41.6	35.1	26.1

The EEG signal is sampled on 8 b, thus its derivative ranges from −255 to 255, and accordingly 511 symbols are to be encoded. As no viable algorithm exists for building the optimal CHT, the simple heuristics introduced in Section II were adopted.

It is not possible to compare the result of the previously mentioned heuristics with all possible CHT assignments, because, in our case, M is 511 and a reasonable value for m could be eight so that about 10^{17} different possible combinations should be considered. Therefore, we do not

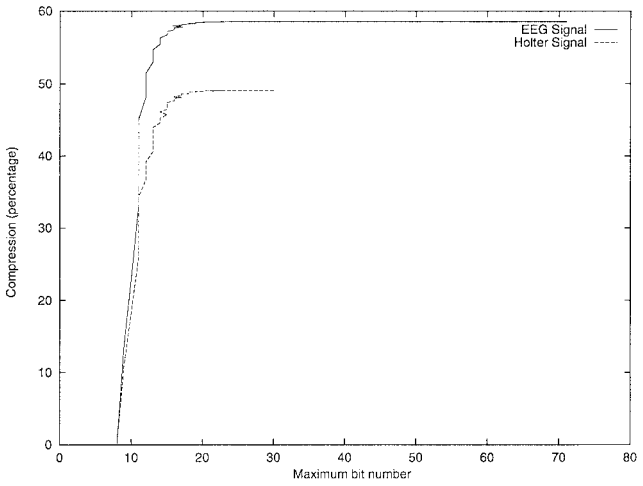

Fig. 7. Compression ratio versus maximum bit number in a CHT for EEG and Holter signals. 16-b lead close to the unbounded code length compression.

actually know how far our solution is from the optimal one. Nevertheless, some tests with the proposed heuristics were performed; all possible exchanges of couples of symbols, respectively, belonging to I_1, I_2 were tried, without any further reduction of the cost function. A Random Walk comprising about 10^7 moves (each move is an exchange of a couple of symbols in I_1, I_2) was made in the search space but no better configuration was reached by the trajectory.

For the test phase, four Holter EEG's were used, with a total number of samples equal to about $175 \cdot 10^6$ (so that both train and test sets contain about 10^7 items). Collapsed Huffman trees were built for m ranging from 511 to 1. Each value of m determines a maximum number of bits required for the code-words.

The same technique was applied to the corpus of the EEG signals previously described.

Fig. 7 shows the compression ratio obtained for different maximum allowable codeword lengths, including exact Huffman code, for both Holter and EEG signals. The greatest compression ratio obtainable on Holter's is equal to 49% and is achieved by the exact Huffman coding when the probability distribution is known. It is inferior to the derivative compression ratio obtained for EEG signals (58%) because the lower sampling rate of Holter increases the speed of signal changes, making it more difficult to compress. Muscular artifacts (i.e., high amplitude signal variations due to muscle movements or stimulations) are also more frequent in Holters. It is worth noting that a small number of bits (16) leads close to the upper bound for the compression factor.

VII. Conclusions

We have presented and discussed several methods to lossless compress EEG's data. We have shown that simple signal transformations, like the derivative, greatly enhance system performance, and in fact, the results are comparable or better than those previously reported [5], with the advantage of exact signal reconstruction, which is an essential requirement for physicians.

Based on the introduced equations exploiting correlation peaks, we have shown experimental evidence that knowledge about EEG biologically relevant cues, the alpha-waves, produces only a slight improvement in the compression ratio. Therefore it is the authors' conviction that the simplest signal transformation, the derivative, should be actually applied in real systems.

Finally, to cope with the A/D converter data rate and low computational requirement, a compression technique based on CHT to produce a fixed maximum codeword length is described in the paper. The suggested algorithm was applied to encode the prediction errors of various predictors for both EEG and Holter EEG, and experimental evidence is demonstrated in Fig. 7 that it can be effectively applied to compress signals with a limited loss of performance.

All the algorithms written in ANSI C, were developed on Unix workstations, recompiled on Intel 486 under Windows, and the most appealing one (the derivative) was employed in conjunction with a popular transmission package for an in field test. The average frequency table technique for Huffman coding, coupled with derivative computation, allowed the authors to implement a PC-based system to send 20 EEG channels, sampled at 128 Hz with 8-b precision in realtime over switched telephone lines with a low-cost CCITT V.32-b modem. This result, to our knowledge, corresponds to more than doubling the previously transmitted number of channels [4], [6]. The prototype was successfully tested in everyday practice during several months at the Neurological Department of the Santa Chiara Hospital in Trento to automatically collect data from peripheral hospitals.

Acknowledgment

The authors are grateful to Micromed for technical support and for permitting access to their EEG management tools and to S. Chiara Hospital for providing data for the test, design, and validation of the algorithms.

References

[1] A. Gersho and R. M. Gray, *Vector Quantization and Signal Compression.* Norwell, MA: Kluwer, 1992.
[2] S. Lawrence Marple, Jr., *Digital Spectral Analysis with Applications.* Englewood Cliffs, NJ: Prentice-Hall, 1987.
[3] T. H. Cormen, C. E. Leiserson, and R. L. Rivest, *Algorithms.* Cambridge, MA: MIT Press, 1990.
[4] F. Vaz, O. Pacheco, and A. Martins da Silva, "Long distance EEG transmission using the public telephone network," *Bolet. Epileps.*, vol. 1, no. 3, pp. 35–39, 1994.
[5] Y. Ohtaki, K. Toraichi, and Y. Ishiyama, "On compressing method of EEG data for their digital database," in *Proc. ICASSP*, 1992, pp. 581–584.
[6] R. M. Gardner, D. R. Bennet, and R. B. Vorce "Eight channel data set for clinical EEG transmission over dial-up telephone network," *IEEE Trans. Biomed. Eng.*, vol. BME-21, no. 3, pp. 246–249, May 1974.
[7] J. Markel and A. Grey, *Linear Prediction of Speech*. New York: Springer-Verlag, 1976.
[8] R. Battiti, A. Sartori, G. Tecchiolli, P. Tonella, and A. Zorat, "Neural compression: An integrated application to EEG signals," in *Proc. IWANNT*, 1995, pp. 210–217.
[9] G. Antoniol and P. Tonella, "Data compression techniques for EEG signals," IRST, Trento, Italy, Tech. Rep. 9508-03, 1995.
[10] H. Braun and M. Riedmiller, "Rprop: A fast and robust backpropagation learning strategy," in *Proc. ACNN*, 1993.
[11] G. Nave and A. Cohen, "ECG compression using long-term prediction," *IEEE Trans. Biomed. Eng.*, vol. 40, no. 9, pp. 877–885, Sept. 1993.

[12] P. S. Hamilon and W. J. Tompkins, "Theoretical and experimental rate distorsion performance in compression of ambulatory ECG's," *IEEE Trans. Biomed. Eng.*, vol. 38, no. 3, pp. 261–266, Mar. 1991.
[13] S. M. S. Jalaleddine, C. G. Hutchens, R. D. Strattan, and W. A. Coberly, "ECG data compression techniques—A unified approach," *IEEE Trans. Biomed. Eng.*, vol. 37, no. 4, pp. 329–343, Apr. 1990.
[14] S. C. Tai, "An extensive Markov system for ECG exact coding," *IEEE Trans. Biomed. Eng.*, vol. 42, no. 2, pp. 230–232, Feb. 1995.
[15] C. P. Mammen and B. Ramamurthi, "Vector quantization for compression of multichannel ECG," *IEEE Trans. Biomed. Eng.*, vol. 37, no. 9, pp. 821–825, Sept. 1990.
[16] N. Ahmed and K. R. Rao, *Orthogonal Transforms for Digital Signal Processing.* Berlin: Springer-Verlag, 1975.
[17] A. Cohen, F. Flomen, and N. Drori, *EEG Sleep Staging Using Vectorial Autoregressive Models, Advances of Processing and Pattern Analysis of Biological Signals.* New York: Plenum, 1995.
[18] J. Ziv and A. Lempel, "A universal algorithm for sequential data compression," *IEEE Trans. Inform. Theory*, vol. IT-23, no. 3, pp. 337–343, 1977.
[19] T. M. Cover and J. A. Thomas, *Elements of Information Theory.* New York: Wiley, 1991.
[20] G. Held and T. R. Marshall, *Data Compression.* New York: Wiley, 1991.

Giuliano Antoniol (S'80–M'81) was born in Transacqua, Trento, Italy on January 12, 1956. He received the doctoral degree in electronic engineering from the University of Padua, Italy, in 1982.

From 1983 to 1987, he worked at SIP-Italian Telecommunication firm. In 1987, he joined IRST in the Speech Recognition Group as Senior Researcher; during the period 1987–1994, significant results were achieved in the design and development of a multimodal interface for automatic reporting by speech (A.Re.S) to be applied in hospital environment for the dictation of radiological reports. Since 1994 he has lead the IRST Program Understanding and Reverse Engineering Project (PURE) team. His current research interests include software engineering, reverse engineering, distributed systems, and graphical user interfaces.

Paolo Tonella was born in Treviso, Italy, on March 5, 1968. He received the doctoral degree *cum laude* in electronic engineering from the University of Padua, Italy in 1992, with a thesis in the field of neural networks. He is working toward the Ph.D. degree in software engineering at the University of Padua.

In 1994, he joined the Software Engineering group at the Istituto per la Ricerca Scientifica e Tecnologica (IRST), Trento, Italy. He is now a member of the Program Understanding and Reverse Engineering (PURE) Project team at IRST.

His current research interests include software engineering, reverse engineering, and program analysis.