Electrocardiographic Signal Compression using Multiscale Recurrent Patterns

Eddie B. L. Filho, Eduardo A. B. da Silva, Murilo B. de Carvalho, Waldir S. S. Júnior, José Koiller

***Abstract*— In this work, we use the *Multidimensional Multiscale Parser* (MMP) algorithm, a recently developed universal lossy compression method, to compress data from ECG signals. The MMP is based on *approximate multiscale pattern matching*, encoding segments of an input signal using expanded and contracted versions of patterns stored in a dictionary. The dictionary is updated using concatenated and displaced versions of previously encoded segments, therefore MMP builds its own dictionary while the input data is being encoded. The MMP can be easily adapted to compress signals of any number of dimensions, and has been successfully applied to compress two-dimensional image data. The quasi-periodic nature of ECG signals makes them suitable for compression using recurrent patterns, like MMP does. However, in order for MMP to be able to efficiently compress ECG signals, several adaptations had to be performed, such as the use of a continuity criterion among segments and the adoption of a prune-join strategy for segmentation. The rate-distortion performance achieved was very good. We show simulation results were MMP performs as well as some of the best encoders in the literature, although at the expense of a high computational complexity.**

***Index Terms*— Recurrent Pattern Matching, Multiscale Decomposition, Vector Quantization, Electrocardiogram.**

I. Introduction

In this paper, we report the results of the application of a universal lossy data compression scheme referred to as MMP (multidimensional Multiscale Parser) [1] to the compression of ECG (Electrocardiogram) data. In the last years, digital storage media is getting progressively cheaper and steadily growing in capacity. However, ECG compression is still important for applications where data transmission through telephone lines or telecommunication networks is required. Therefore, considerable effort has been made to achieve high compression rates with the distortion being low enough so that the diagnostic accuracy is not compromised.

The compression algorithms reported in the literature can be classified in three groups: direct, parametric and transform-based. In the direct class, the algorithms employ prediction techniques to estimate a sample or group of samples from previously encoded data. A residual signal is then generated by subtracting the actual sample values from the predicted ones and the difference is quantized and encoded [2]. In the parametric class, the algorithms attempt to extract features from the input signal which will be used later to drive a model-based synthesizer in order to reconstruct the signal [3]. In the transform-based class, the input signal is first transformed to another domain by a (usually) linear transformation. The transformed signal is then compressed using some combination of quantization and entropy coding. One advantage of this approach is that it is usually easier to compress the signal in the transformed domain, as long as the transformation is conveniently chosen. Some of the best known encoders are in this class [4], [5], [19], [20].

In this work, we propose a new technique to compress ECG signals. In order to make the comparisons to other algorithms easier, we use ECG signals taken from the MIT/BIH arrhythmia database. This database contains parts of ECG exams of 48 subjects, with two derivations (signals); each derivation approximately 30 minutes long. The signals were sampled at 360 Hz and quantized at 11 bits of resolution. One of these signals (the first three seconds) is illustrated in Fig. 1. The results are assessed using the PRD (*percent-root-mean-square-difference*) distortion metric and the CDR (*compressed-data-rate*) rate, defined as:

$$PRD = 100\sqrt{\frac{\sum_{n=0}^{N-1}\left(x(n)-\hat{x}(n)\right)^2}{\sum_{n=0}^{N-1}\left(x(n)-\mu\right)^2}} \quad (1)$$

$$CDR = \frac{B}{T} \quad (2)$$

where μ is the baseline value of the analog-to-digital conversion used for the acquisition of the data $x(n)$ (in the MIT/BIH arrhythmia database $\mu = 1024$), B is the total number of bits spent in the compressed representation of $x(n)$ and T is the duration of the original input signal in seconds.

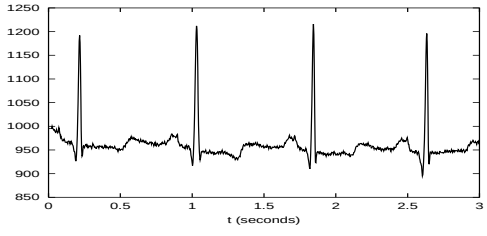

Fig. 1. A typical ECG signal.

Many of the state-of-the-art compression schemes developed so far are transform-based and rely on the three-step

Manuscript received February 1, 2005; revised July 5, 2005. This work was accomplished through the partnership between UFAM (Universidade Federal do Amazonas) and UFRJ/COPPE (Universidade Federal do Rio de Janeiro), with the financial support provided by SUFRAMA (Superintendencia da Zona Franca de Manaus).

Eddie B. L. Filho is with the Genius Institute of Technology, Manaus, AM, Brazil (e-mail: efilho@genius.org.br).

Eduardo A. B. da Silva is with the Department of Electronics Engineering and the Department of Electrical Engineering, COPPE, Universidade Federal do Rio de Janeiro, Rio de Janeiro, RJ, Brazil (e-mail: eduardo@lps.ufrj.br).

Murilo B. de Carvalho is with the Department of Telecommunications Engineering, Universidade Federal Fluminense, Niterói, RJ, Brazil (e-mail: murilo@telecom.uff.br).

Waldir S. S. Júnior is with the Fundação Centro de análise, Pesquisa e Inovação Tecnológica, Manaus, AM, Brazil (e-mail: waldirjr@fucapi.br).

José Koiller is with the Courant Institute of Mathematical Sciences, New York, USA (e-mail: koiller@cims.nyu.edu).

encoding paradigm: a first step of transformation is followed by a quantization step and an entropy coding step. Unlike these methods, our proposed scheme is based on MMP, a universal lossy compression method that merges all the steps of the three-step approach in one. The MMP algorithm is based on *multiscale pattern matching* [1]. In ordinary pattern matching, two vectors $\mathbf{v}$ and $\mathbf{u}$ of equal lengths are said to match if they are closer, according to some metric, than a predefined threshold. On the other hand, in multiscale pattern matching the two vectors to be matched can have different lengths, that is $\ell(u) \neq \ell(v)$. In order to match them, we first change the size of the vectors using a *scale transformation* $T_N(\mathbf{x}) : \mathbb{R}^{\ell(\mathbf{x})} \mapsto \mathbb{R}^N$. For example, if we want to match vector $\mathbf{u}$ of size 2 to vector $\mathbf{v}$ of size 5 we first evaluate the *scaled version* $\mathbf{u}^s = T_5(\mathbf{u})$ and after that we proceed by performing an ordinary match between $\mathbf{u}^s$ an $\mathbf{v}$. In this case, the scale transformation can be implemented using classical interpolation methods [6]. Alternatively we could match a scaled down version of $\mathbf{v}$, that is $\mathbf{v}^s = T_2(\mathbf{v})$, to the vector $\mathbf{u}$, in which case the transformation could be implemented using downsampling techniques. It should be noted that sometimes $\mathbf{u}$ can match $\mathbf{v}^s$ while $\mathbf{u}^s$ does not match $\mathbf{v}$, depending on the choice of the vectors and the particular scale transformations chosen. In our application, however, this is not a problem because the choice of which transformation should be used is implicit in the structure of the MMP algorithm. The matching with scales is graphically illustrated in Fig. 2.

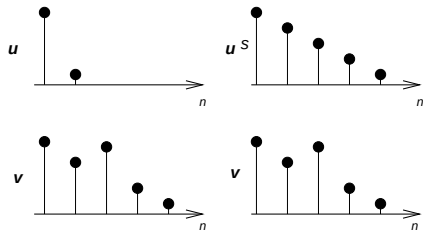

Fig. 2. Approximate matching with scales.

The MMP algorithm can also be regarded as a kind of variable block-size vector quantization (VBVQ). Indeed, it has points in common with other works on variable block-size VQ (see, for example [14]–[17]), as it quantizes regions of the signal with blocks of different dimensions taken from a dictionary. However, the MMP algorithm when seen as a variable block-size VQ presents unique characteristics: an adaptive dictionary (which obviates the need for previous training), a binary segmentation tree (even for multidimensional data) and the multiscale pattern matching. For example, unlike MMP, neither [16] nor [17] use the concept of multiscale pattern matching, and they use a quadtree segmentation strategy with fixed codebooks at each different block dimension.

This paper is organized as follows. In section II, the basic segmentation procedure used in the MMP algorithm is presented. In section III, a version of MMP using a rate-distortion optimized segmentation tree is discussed. In section IV, an improved MMP, with a more general segmentation, is described. In section V, a method to improve the performance of MMP for smooth signals is presented. In section VI, some modifications to the MMP algorithm are made in order to further improve the performance for ECG signals. Next, in section VII, we present experimental results and comparisons to other state-of-the-art ECG encoders. Finally, in section IX we present our conclusions. Appendix I contains details of the particular implementation of MMP used in our simulations. Appendix II contains an analysis of the computational complexity of the algorithm.

II. The MMP algorithm

The MMP is a lossy compression scheme based on multiscale pattern matching. It has a dictionary $\mathcal{D} = \{\mathbf{v}_0, \mathbf{v}_1, \ldots, \mathbf{v}_{L-1}\}$ of L fixed-length vectors $\mathbf{v}_i$ that it uses to encode variable-sized segments of an input vector $\mathbf{X}^0 = \begin{pmatrix} x(0) & x(1) & \ldots & x(N-1) \end{pmatrix}$, where the dimension N is a power of two. When attempting to encode $\mathbf{X}^0$, MMP searches in the dictionary $\mathcal{D}$ for the best vector $\mathbf{v}_{i_0}$ that can be used to replace $\mathbf{X}^0$. In the spirit of multiscale pattern matching, the search procedure tests all vectors in $\mathcal{D}$, no matter the sizes of each $\mathbf{v}_i$. The choice of the best vector is based on the minimization of some performance criterion. In one of its simpler versions, MMP chooses the vector that minimizes the squared error $\xi_0 = \|\mathbf{X}^0 - \mathbf{v}_{i_0}\|^2$. If the squared error ξ_0 is smaller than or equal to a predefined distortion threshold d^*, then the encoding of $\mathbf{X}^0$ is done and MMP outputs a single bit flag $'1'$, indicating that a match has been found, followed by the dictionary index i_0. If the matching attempt fails, which means that the squared error ξ_0 is above the threshold d^*, then MMP splits the input vector in two segments, $\mathbf{X}^1 = \begin{pmatrix} x(0) & x(1) & \ldots & x(N/2-1) \end{pmatrix}$ and $\mathbf{X}^2 = \begin{pmatrix} x(N/2) & x(N/2+1) & \ldots & x(N-1) \end{pmatrix}$. MMP then outputs a one bit flag $'0'$ indicating splitting, and after that it repeats the matching attempt for $\mathbf{X}^1$. If it succeeds, it outputs $'1'$ followed by the dictionary index i_1. MMP then proceeds to encode the second segment $\mathbf{X}^2$. If the attempt to match $\mathbf{X}^1$ fails instead, it outputs the flag $'0'$ and split $\mathbf{X}^1$ in two smaller segments, $\mathbf{X}^3 = \begin{pmatrix} x(0) & x(1) & \ldots & x(N/4-1) \end{pmatrix}$ and $\mathbf{X}^4 = \begin{pmatrix} x(N/4) & x(N/4+1) & \ldots & x(N/2-1) \end{pmatrix}$, before it attempts to encode $\mathbf{X}^2$. In fact, the splitting is recursively repeated until a match is successful, before the recursive procedure returns. Fig. 3 illustrates this segmentation procedure.

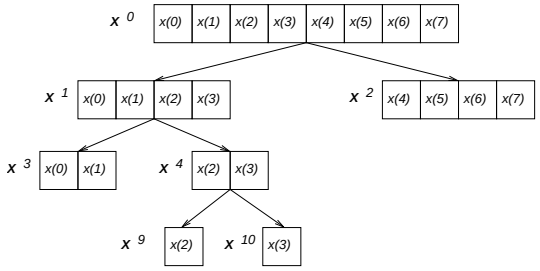

Fig. 3. A segmentation of the input vector $\mathbf{X}^0$.

As can be seen from Fig. 3, the segmentation procedure defines a segmentation tree $\mathcal{S}$, where each node n_j corresponds to a different segment $\mathbf{X}^j$ of the input vector. The segmentation tree associated with the example above is illustrated in

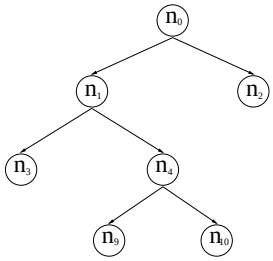

Fig. 4. A segmentation tree $\mathcal{S}$.

Fig. 4. In this example, the output generated by MMP would be the sequence $0, 0, 1, i_3, 0, 1, i_9, 1, i_{10}, 1, i_2$. Note that in an actual implementation, these indexes would be coded using an adaptive arithmetic encoder [18]. The flag sequence 0010111 defines the chosen segmentation tree in a top-down fashion. As long as one knows which are the vectors associated to each index i_j (that is, both the encoder and the decoder have the same dictionary $\mathcal{D}$), the output sequence generated by MMP is easily decoded as follows: the decoder begins by reconstructing the segmentation tree from the segmentation flags. After that, the decoder knows the lengths associated to each index i_j, since each node n_j corresponds to a vector of known size. Then the decoder proceeds by replacing each node n_j by a correctly scaled version of the vector $\mathbf{v}_{i_j}$ in the dictionary $\mathcal{D}$.

The MMP algorithm as described so far operates with a fixed database, that is, the dictionary is static. A much better performance can be achieved if we allow the dictionary to adapt itself to the data being encoded. This is accomplished by updating the dictionary with concatenations of previously encoded segments. This resembles the way the lossless Lempel-Ziv compression scheme [7] works. For example, referring to Fig. 3, as soon as the indexes i_9 and i_{10} are determined, we can build $\hat{\mathbf{X}}^9$ and $\hat{\mathbf{X}}^{10}$, the reconstruction versions of the input segments $\mathbf{X}^9$ and $\mathbf{X}^{10}$, by properly scaling the dictionary vectors $\mathbf{v}_{i_9}$ and $\mathbf{v}_{i_{10}}$. Then we have the reconstructed version of $\mathbf{X}^4$ by the concatenation of $\hat{\mathbf{X}}^9$ and $\hat{\mathbf{X}}^{10}$, that is $\hat{\mathbf{X}}^4 = \begin{pmatrix} \hat{x}(2) & \hat{x}(3) \end{pmatrix}$. We can then safely include $\hat{\mathbf{X}}^4$ in the dictionary, since the information used to generate this new vector is available both to the encoder and to the decoder, ensuring the proper synchronization of the dictionaries at both ends. The next update of the dictionary in this example will be done by including in it the concatenations of $\hat{\mathbf{X}}^3$ and $\hat{\mathbf{X}}^4$, as soon as the two segments are available. When MMP tries to represent an input segment $\mathbf{X}^j$ by one of the vectors in its dictionary $\mathcal{D}$, it first has to apply scale transformations to adjust the length of each vector in $\mathcal{D}$ to equal the length of $\mathbf{X}^j$. If the length of the input vector is N, the segmentation procedure can create vectors of lengths $N/2, N/4, \ldots, 1$. This implies that there are at most $1 + \log_2(N)$ different lengths or scales. Therefore, to save time, we could keep $1 + \log_2(N)$ copies of the dictionary, one at each scale, to avoid the computation of the scale transformation each time a match is attempted. This way, we only need to use the scale transformation when we are including a new vector in the dictionary. We denote a copy of the dictionary at scale $2^{-p}N$ as $\mathcal{D}^p$. If we are using this multiple codebook scheme, to include a new vector $\hat{\mathbf{X}}^j$ in the dictionary we must actually include $T_{2^{-p}N}[\hat{\mathbf{X}}^j]$ in $\mathcal{D}^p$ for $p = 0, 1, \ldots, \log_2(N)$.

III. R-D OPTIMIZATION OF THE SEGMENTATION TREE

The segmentation tree generated by the distortion-controlled version of MMP is created using local decisions based on distortions calculations, and is thus not globally optimum in a rate-distortion sense. We can improve the R-D performance of MMP by optimizing its segmentation tree [1].

Referring to Fig. 3 and 4, each node n_j is associated to a segment of the input vector $\mathbf{X}^j$ that is best represented by a scaled version of a dictionary element $\mathbf{v}_{i_j}$, referred to as $\mathbf{v}^s_{i_j}$. Therefore, we can associate to each node the distortion:

$$D(n_j) = d\left(\mathbf{X}^j, \mathbf{v}^s_{i_j}\right) \tag{3}$$

where $d(\mathbf{u}, \mathbf{v})$ is some distortion metric. For example, the distortion can be the squared error $d(\mathbf{u}, \mathbf{v}) = \|\mathbf{u} - \mathbf{v}\|^2$.

We call $R(n_j)$ the rate needed to specify the index i_j, that is:

$$R(n_j) = -\log_2(\Pr(i_j)) \tag{4}$$

where $\Pr(i_j)$ is the probability of occurrence of index i_j in the dictionary.

The overall distortion is then:

$$D(\mathcal{S}) = \sum_{n_j \in \mathcal{S}_\mathcal{L}} D(n_j) \tag{5}$$

where $\mathcal{S}_\mathcal{L}$ is the set of leaf nodes of $\mathcal{S}$. The amount of bits needed to encode this approximation is the rate $R(\mathcal{S})$, which is given by:

$$R(\mathcal{S}) = R_t(\mathcal{S}) + \sum_{n_j \in \mathcal{S}_\mathcal{L}} R(n_j) \tag{6}$$

where $R_t(\mathcal{S})$ is the rate required to specify the segmentation tree.

The best segmentation $\mathcal{S}^*$, in an R-D sense, leads to the minimum rate $R(\mathcal{S})$ given that the distortion $D(\mathcal{S})$ is no greater than a target distortion D^* or, alternatively, the minimum distortion $D(\mathcal{S})$ at rate R^*. This is a constrained minimization problem stated as:

$$\begin{aligned} \mathcal{S}^* &= \arg \min_{\mathcal{S} \in \mathcal{S}_{R^*}} D(\mathcal{S}), \\ \mathcal{S}_{R^*} &= \{\mathcal{S} : R(\mathcal{S}) = R^*\} \end{aligned} \tag{7}$$

To find $\mathcal{S}^*$ we can find the solution to a related unconstrained problem introducing a Lagrange multiplier λ. It is well known that if we find the minimum of the Lagrangian cost $J(\mathcal{S}) = D(\mathcal{S}) + \lambda R(\mathcal{S})$ [8], we also find the solution to the constrained problem when we choose $R(\lambda) = R^*$. Hence,

we have that

$$\begin{aligned}\mathcal{S}^*=&\arg\min_{\mathcal{S}}\left(J(\mathcal{S})\right)\\ =&\arg\min_{\mathcal{S}}\left(\sum_{n_j\in\mathcal{S}_\mathcal{L}}D(n_j)+\lambda\left(R_t(\mathcal{S})+\sum_{n_j\in\mathcal{S}_\mathcal{L}}R(n_j)\right)\right)\\ =&\arg\min_{\mathcal{S}}\left(\lambda R_t(\mathcal{S})+\sum_{n_j\in\mathcal{S}_\mathcal{L}}\left(D(n_j)+\lambda R(n_j)\right)\right)\\ =&\arg\min_{\mathcal{S}}\left(\lambda R_t(\mathcal{S})+\sum_{n_j\in\mathcal{S}_\mathcal{L}}J(n_j)\right)\end{aligned}\tag{8}$$

where $J(n_j)=D(n_j)+\lambda R(n_j)$.

A sub-tree $\mathcal{S}(n_j)$ of $\mathcal{S}$ at node n_j is the binary tree composed by all the nodes of $\mathcal{S}$ that have n_j as the root node. Fig. 5 illustrates the sub-tree $\mathcal{S}(n_4)$ of the binary tree in Fig. 4. We denote $\mathcal{S}-\mathcal{S}(n_j)$ the tree obtained from $\mathcal{S}$ by pruning the sub-tree $\mathcal{S}(n_j)$.

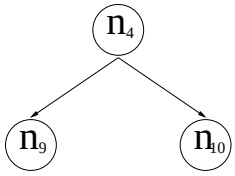

Fig. 5. The sub-tree $\mathcal{S}(n_4)$ of the binary tree in Fig. 4.

If the Lagrangian costs $J(n_l)$ associated with the approximation of each segment $\mathbf{X}^l$ are independent, then the Lagrangian cost of two sub-trees $J(\mathcal{S}(n_l))$ and $J(\mathcal{S}(n_m))$ are also independent, as long as all nodes of both sub-trees are different. Then a fast search algorithm, similar to the one in [9], can be implemented considering that if $J(n_l)\leq J(\mathcal{S}(n_{2l+1}))+J(\mathcal{S}(n_{2l+2}))$, then the sub-trees $\mathcal{S}(n_{2l+1})$ and $\mathcal{S}(n_{2l+2})$ must be pruned from $\mathcal{S}$ in order to decrease the cost. Unfortunately, this is not the case with MMP, since the costs $J(n_l)$ are coupled by the dictionary updating procedure. That is, the computation of the overall variation of the cost obtained by pruning subtree $\mathcal{S}(n_l)$ has to take into account the variation in the cost of the other nodes $J(n_k)$ due to the variation of the dictionary (that was caused by the pruning of subtree $\mathcal{S}(n_l)$). However, if the initial dictionaries are large enough, one can argue that the effect of the dictionary update on the minimization of $J(n_l)$ can be negligible. In practical MMP implementations, we usually impose an upper limit to the size M of the input vector $\mathbf{X}^0$ due to the finite amount of memory available. This forces the input data to be broken in blocks of size M that are sequentially processed by MMP algorithm (Note that the initial dictionary of one block is the final dictionary of the previous block). Although the assumption of a large initial dictionary is not true for the first blocks, the dictionary eventually grows large enough so that the Lagrangian costs $J(n_l)$ are almost decoupled. In this sense, one could use the algorithm in [9] to find an approximate R-D optimal solution.

However, if we want to use relatively large blocklengths or if the dictionary is too small (as happens at very low rates), we should modify the algorithm to take into account the impact of the dictionary-updating procedure. We know that the dictionary is updated by the inclusion of the concatenation of previously encoded segments. Therefore, if we choose to prune a sub-tree, the impact in the cost is not restricted to that sub-tree, but can affect all nodes that are to the right of the node which is being analyzed. That is, if we prune a sub-tree, we might remove from the dictionary an element that would otherwise be used later to approximate an input segment, therefore increasing the cost. In [10], an algorithm is proposed that, although sub-optimal, takes into account the impact of the pruning of a node in all other dependent nodes, yielding better performance than the direct application of the algorithm in [9].

In this paper we use a slightly different approach that has a smaller computational cost. The optimization begins by initializing the segmentation tree as the full binary tree of depth $\log_2(N)$, where N is the length of the input vector $\mathbf{X}^0$. The dictionary is temporarily updated following the updating rule described in section II. Then, we perform an analysis at each pair of nodes n_{2k+1} and n_{2k+2} sharing the same parent node n_k to decide whether to prune or not, in order to lower the Lagrangian cost. The decision must be made considering the Lagrangian cost of the subtree $\mathcal{S}(n_{2k+1})$, the cost of the subtree $\mathcal{S}(n_{2k+2})$ and the cost of the single node n_k. Whenever we decide to prune, the dictionary must be reset to its previous state, since it contains an invalid entry (resulting from the concatenation of $\hat{\mathbf{X}}^{2k+1}$ and $\hat{\mathbf{X}}^{2k+2}$). The same procedure is repeated for each pair of nodes until no more nodes are pruned. In order to evaluate the Lagrangian costs, we need to know the probabilities of occurrence of each vector in the dictionary, as well as the probability of each segmentation flag. In our implementation of MMP, these probabilities are estimated by keeping a record of the number of times each vector has been used, as well as the number of occurrences of each flag. Therefore, whenever we do a temporary update of the dictionary, we must also perform a temporary update of the statistics record.

IV. Further optimization of the segmentation tree: the prune-join algorithm

In section III we showed how to perform a rate distortion optimization of the segmentation tree. At first glance, this might suggest that there is no more room for enhancement, regarding the segmentation, but that is not the case at all. Fig. 6(a) contains all options of segmentation for an input vector of size 8 associated to a binary tree. In Fig. 6(b-c), we show some of the other possibilities of segmentation. This illustrates that the binary tree structure constrains the options for segmentation.

In [11], an algorithm to extend the segmentation options of a binary tree, called the prune-join algorithm, is presented. In that framework, an input vector is firstly segmented, according to an R-D optimized segmentation tree and each segment is approximated by a polynomial function. The optimization is performed by iteratively pruning a full binary tree, in the spirit of [9]. This first step is the *prune step*. Then a second

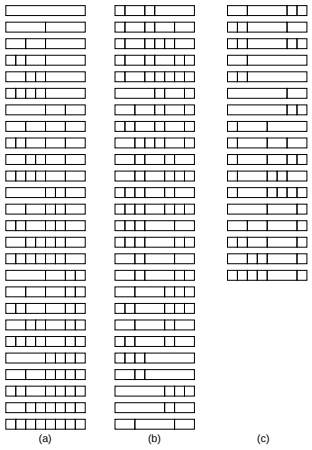

Fig. 6. Segmentation restriction on MMP: (a) Allowed, (b-c) Not allowed.

step, the *join step*, is performed where all neighbor nodes who are not descendants from the same parent node are tested to verify if they can be joined together to improve the overall performance, that is, lowering the Lagrangian cost. The join step can be repeated recursively allowing junctions of junctions of nodes to be made. After that, the coefficients of the polynomials of the remaining nodes are scalar quantized and encoded. In [11], it was shown that the prune join algorithm outperforms the single prune-only algorithm and, unlike the prune-only algorithm, it provides a nearly optimum R-D performance, at least in the case presented in [11], where the signals to be encoded are assumed to be piecewise polynomial. Those theoretical results cannot be directly applied to the MMP case, because the use of the dictionary and the multiscale pattern matching approach are radically different from the scalar coding of the polynomial coefficients used in the original prune-join algorithm. However, we can expect an improvement in performance as we extend the possibilities of segmentation. The options of segmentation illustrated in Fig. 6(b-c) can all be obtained from a binary tree segmentation followed by a single join step.

The prune-join idea can be incorporated to MMP. In order to do that, we firstly apply the original MMP segmentation procedure to an input vector $\mathbf{X}^0$, obtaining a rate-distortion-optimized segmentation tree $\mathcal{S}$. The rate-distortion optimization procedure described in section III performs the prune part of the algorithm. After that, we carry on an analysis to verify if any two neighbor nodes not sharing the same parent node can be joined together to lower the Lagrangian cost. Fig. 7(a) illustrates an example of a segmentation tree after the pruning step. Fig. 7(b) illustrates a possible join operation.

We denote $U(n_j, n_l)$ the union of the two neighbor nodes n_j and n_l. At each pair of neighbor leaf nodes (n_l, n_l) that descend from distinct parent nodes, we test to see whether the cost of encoding both independently, (that is $J(n_j) + J(n_l) + \lambda R_{nu}(i,j)$) is greater than the cost to encode the union ($R_{nu}(i,j)$ is the rate associated to the encoding of the flags indicating that the nodes should not be joined). The

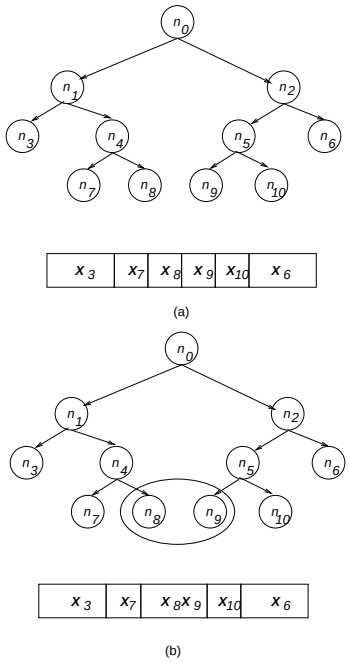

Fig. 7. The join step.

Lagrangian cost of the union is given by:

$$J(U(n_j, n_l)) = d\left(\begin{pmatrix} \mathbf{X}^j & \mathbf{X}^l \end{pmatrix}, \mathbf{v}^s_{i_{jl}}\right) + \lambda\left(R_u(j,l) - \log_2(\Pr(i_{jl}))\right) \quad (9)$$

where $R_u(j,l)$ is the rate spent to encode the join flag, $\mathbf{v}^s_{i_{jl}}$ is the best vector in the dictionary scaled to the sum of the lengths of $\mathbf{X}^j$ and $\mathbf{X}^l$ and i_{jl} denotes the index of the vector chosen.

Similarly to the case of the R-D optimization of the segmentation tree, some care should be taken when evaluating (9). Whenever we choose to join two nodes n_j and n_l, the dictionary will no longer be updated with the result of the concatenation of the reconstruction vectors associated to each of them. This can affect the evaluation of the costs for all subsequent nodes. To solve this problem, whenever a union occurs the dictionary update due to each block is removed and the statistics record is corrected, adjusting the results for the other blocks. This procedure is similar to that used during the prune step. Besides, the cost of the union must be computed together with the first block being analyzed, to prevent the use of an element that would be pruned later. It is also interesting to point out that if we are using multiple copies of the codebook to speed up the computation of the multiscale pattern matching, we no longer have only $1+\log_2(N)$ different scales, since the join procedure will add some extra scales to the ones available due to the binary tree subdivision scheme.

V. MMP USING SMOOTHNESS CRITERIA BY SIDE-MATCH VQ

The representation $\hat{\mathbf{X}}^0$ that MMP produces can have severe discontinuities at the boundaries of the segments $\hat{\mathbf{X}}^j$ generated by the segmentation procedure, even if the original input vector $\mathbf{X}^0$ is smooth. This is illustrated in Fig. 8.

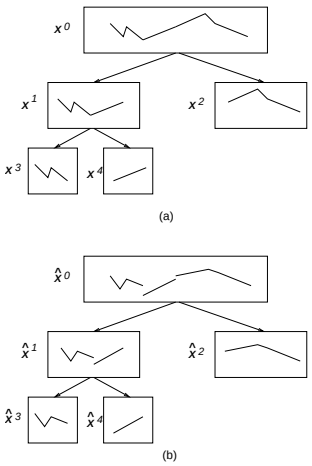

Fig. 8. Segmentation-induced discontinuity: (a) original signal; (b) reconstructed signal.

This happens because the distortion definition in (3), used to evaluate the cost of a node n_j, is independent of the representations chosen for other neighbor segments. Therefore, MMP has no explicit control over the smoothness at the boundaries of the segments. For example, referring to Fig. 8, $\mathbf{X}^3$ and $\mathbf{X}^4$ are independently approximated by $\hat{\mathbf{X}}^3$ and $\hat{\mathbf{X}}^4$, respectively. Although the original segments have no discontinuities, the concatenation of $\hat{\mathbf{X}}^3$ and $\hat{\mathbf{X}}^4$ generates $\hat{\mathbf{X}}^1$, the representation for $\mathbf{X}^1$ created by MMP, that has a discontinuity at the point where $\hat{\mathbf{X}}^3$ and $\hat{\mathbf{X}}^4$ are concatenated (see Fig. 8). In [12], it was proposed an efficient method to control the smoothness of the representations generated by a vector quantization (VQ) scheme called side-match vector quantization (SM-VQ). SM-VQ can be incorporated to MMP in order to improve its performance for smooth input vectors [13]. The idea is to choose a subset of the dictionary $\mathcal{D}$, called the *state dictionary* $\mathcal{D}_S$, composed of the N_S best vectors $\mathbf{v}_k \in \mathcal{D}$ according to some smoothness criterion. In order to better understand the side-match MMP, one should first make the definitions that follow.

The position of the first sample of $\mathbf{X}^j$ inside $\mathbf{X}^0$ is given by:

$$FP(j) = N\left((j+1)\,2^{-\lfloor \log_2(j+1) \rfloor} - 1\right) \tag{10}$$

where N is the length of $\mathbf{X}^0$. For example, referring to Fig. 3 we have: $FP(0) = 0$ (meaning that the first sample of $\mathbf{X}^0$ is $x(0)$), $FP(1) = 0$ (meaning that the first sample of $\mathbf{X}^1$ is $x(0)$), $FP(2) = 4$ (meaning that the first sample of $\mathbf{X}^2$ is $x(4)$) and so on.

The length N^j of $\mathbf{X}^j$, can be evaluated as:

$$N^j = N2^{-\lfloor \log_2(j+1) \rfloor} \tag{11}$$

We define the *left-neighbor* of $\mathbf{X}^j$ as the vector:

$$\begin{aligned}\mathbf{L}^j &= \begin{pmatrix} L^j(0) & L^j(1) & \dots & L^j(N^j-1) \end{pmatrix} \\ &= \begin{pmatrix} \hat{x}(Fx) & \dots & \hat{x}(Fy) \end{pmatrix}, FP(j) \geq 1\end{aligned} \tag{12}$$

where $Fx = FP(j) - N^j$ and $Fy = FP(j) - 1$. According to (12), the left-neighbor $\mathbf{L}^j$ is a vector of the same length of $\mathbf{X}^j$ whose components, denoted by $L^j(n)$, are the reconstructed samples at the left side of $\mathbf{X}^j$.

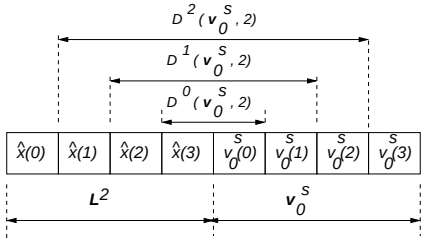

Fig. 9. Evaluation of the rugosity.

In order to measure the smoothness of the approximation, we define three parameters:

i) The *zero-order discontinuity*:

$$D^0\left(\mathbf{v}_k^s, j\right) = \left|L^j(N^j - 1) - v_k^s(0)\right| \tag{13}$$

ii) The *first-order discontinuity*:

$$\begin{aligned}D^1\left(\mathbf{v}_k^s, j\right) = \big|&L^j(N^j-2) - \\ &L^j(N^j-1) - v_k^s(0) + v_k^s(1)\big|\end{aligned} \tag{14}$$

iii) The *second-order discontinuity*:

$$\begin{aligned}D^2\left(\mathbf{v}_k^s, j\right) = \big|&L^j(N^j-3) - 2L^j(N^j-2) + \\ &L^j(N^j-1) - v_k^s(0) + 2v_k^s(1) - v_k^s(2)\big|\end{aligned} \tag{15}$$

These definitions are based on finite-differences approximations to zero-order, first-order and second-order derivatives, respectively.

We then define a *rugosity* metric as:

$$R(\mathbf{v}_k^s, j) = \alpha D^0\left(\mathbf{v}_k^s, j\right) + \beta D^1\left(\mathbf{v}_k^s, j\right) + \gamma D^2\left(\mathbf{v}_k^s, j\right) \tag{16}$$

where $\mathbf{v}_k^s$ is a properly scaled vector of the dictionary. For example, referring to Fig. 3, when MMP attempts to encode $\mathbf{X}^2$, it has available the representations $\hat{\mathbf{X}}^3$, $\hat{\mathbf{X}}^9$ and $\hat{\mathbf{X}}^{10}$. Therefore, it knows $\hat{\mathbf{X}}^0$ up to the fourth sample ($FP(2) = 4$), that is $\hat{\mathbf{X}}^0 = \begin{pmatrix} \hat{x}(0) & \hat{x}(1) & \hat{x}(2) & \hat{x}(3) & ? & ? & ? & ? \end{pmatrix}$. The left-neighbor of $\mathbf{X}^2$ is determined as $\mathbf{L}^2 = \begin{pmatrix} \hat{x}(0) & \hat{x}(1) & \hat{x}(2) & \hat{x}(3) \end{pmatrix}$. In this example, MMP scales all vectors of $\mathcal{D}$ to length 4, and, in the spirit of SM-VQ, builds a state dictionary $\mathcal{D}_S$ containing the N_S least rugose vectors, according to (16). Fig. 9 illustrates which data are used to compute $R\left(\mathbf{v}_0, 2\right)$.

In other words, we use the last samples of the left-neighbor $\mathbf{L}^j$ of $\mathbf{X}^j$ to evaluate which are the best vectors in $\mathcal{D}$ to encode $\mathbf{X}^j$, according to our smoothness criterion. It should be noted that for $j = 2^k - 1$, with k an integer, the left-neighbor is undefined (there are no samples at the left of $\mathbf{X}^j$). In that case, the state dictionary is the full dictionary, that is $\mathcal{D}_S = \mathcal{D}$.

The state dictionary is updated according to these rules and prior to each matching attempt. The size N_S of the state dictionary is also adapted. Based on the assumption that simpler signals require less vectors in the dictionary in order to be well represented, we use an *activity* metric, as defined in (17), to estimate the number of vectors N_S required in $\mathcal{D}_S$ to encode the input segment. The activity was chosen because it accounts for the number of transitions through the vector,

being a reasonably good measure of the complexity of the signal.

$$A\left(\mathbf{L}^j\right) = \sum_{n=1}^{N^j-1} \left|L^J(n-1) - L^j(n)\right| \tag{17}$$

The actual size of the state dictionary is proportional to the activity of the left-neighbor, evaluated by (17), as defined in Appendix I.

One could argue that the use of the state dictionary introduces a distortion-only optimization criterion in a framework that is rate-distortion optimized, which could lead to suboptimal performance. However, the strategy of filling the state dictionary with the N_S least rugose elements can be interpreted as a simple statistical modeling of the source. It is equivalent to say that the dictionary remains the same, but the probabilities of occurrence of the $|\mathcal{D}| - N_S$ most rugose elements of $\mathcal{D}$ are zero ($|\mathcal{D}|$ is the number of elements in $\mathcal{D}$). Therefore, the solution still performs a rate-distortion optimization, at least for the class of sources that conform to our simple statistical model.

VI. The displacement dictionary

As can be seen from Fig. 1, a typical ECG signal is almost periodic. The MMP algorithm adapts its dictionary to the input signal and has the potential to learn the pattern in one period. However, since the length of the segments are not multiples of the basic period, MMP has to learn several shifted versions of the period to efficiently represent the input. To improve the performance for periodic or quasi-periodic signals, we can use a *displacement dictionary* $\mathcal{D}^D$, where we include displaced versions of the approximations for the already encoded segments. The displacement dictionary helps to speed up the learning rate of the algorithm in this case. It can be implemented by keeping the M last samples of the reconstructed signal in a vector $\mathbf{V}_D^j$. That is:

$$\begin{aligned}\mathbf{V}_D^j = \big(\ &\hat{x}(FP(j)-M) \quad \hat{x}(FP(j)-M+1) \quad \dots \\ &\hat{x}(FP(j)-1)\ \big)\end{aligned} \tag{18}$$

The displacement dictionary is defined as:

$$\begin{aligned}\mathbf{u}(p) &= \big(\ V_D^j(p) \quad V_D^j(p+1) \quad \dots \quad V_D^j(p+N^j-1)\ \big) \\ \mathcal{D}^D &= \{\mathbf{u}(p)\}, p = 0, 1, \dots, M - N^j\end{aligned} \tag{19}$$

In the spirit of side-match MMP, a displacement-state dictionary $\mathcal{D}_S^D$ composed of the N_S^D least rugose elements in $\mathcal{D}^D$ (as defined in (16)) is also defined for the displacement dictionary. When MMP attempts to encode the segment $\mathbf{X}^j$ it searches in $\mathcal{D}_S^D$ for the element that is best suited to encode $\mathbf{X}^j$ in a rate-distortion sense. If we define:

$$\begin{aligned}J_D^j(p) &= d\left(\mathbf{u}(p), \mathbf{X}^j\right) + \lambda R(p), \mathbf{u}(p) \in \mathcal{D}_S^D \\ p_j &= \min_p J_D^j(p) \\ \mathbf{v}_D^j &= \mathbf{u}(p_j)\end{aligned} \tag{20}$$

where $R(p)$ is the rate needed to encode the position p. In other words, MMP searches for the element in $\mathcal{D}_S^D$ that minimizes the Lagrangian cost $J_D^j(p)$, as defined in (20).

During the optimization procedure of sections III, IV and V, the minimum Lagrangian cost of the displacement-state dictionary $J_D^j(p_j)$ (plus λ times the rate needed to encode a flag indicating that the displacement-state dictionary is chosen) is compared to the minimum Lagrangian cost found using the state dictionary $\mathcal{D}_S$ (plus λ times the rate needed to encode a flag indicating that the displacement dictionary is not chosen), and the overall minimum is selected. When the optimization is complete, the flags that describe the segmentation tree, the flags indicating the joins and the dictionary indexes (each index with an additional flag indicating if the index corresponds to the state dictionary or to the displacement-state dictionary) are encoded.

VII. Experimental results

We have implemented MMP in software and applied it to compress ECG data from the MIT/BIH arrhythmia database. In order to make comparisons to the works in [4], [5] and [19] easier, we used both channels of the following 11 records: 100, 101, 102, 103, 107, 109, 111, 115, 117, 118 and 119, referred to as dataset A. Records 201, 208, 212, 213, 228, 231 and 232 were also compressed for further evaluation, referred to as dataset B.

Fig. 10 shows the simulation results (PRD versus bit rate) for the 11 records of the dataset A (full-length).

Fig. 11 shows a comparison with the mean results reported in [5] for SPIHT [4] and WT-DCCR-TV-VQ [5]. It is important to note that we have chosen to present the results for SPIHT in [5] instead of the ones in [4]. This was so because the results in [5] correspond to full-length (around 30 minutes) records, unlike the ones in [4], that correspond to just the first 10 minutes of the records (this is a fair comparison since the results for 30 minutes are in general slightly better than the ones for 10 minutes). The data for Fig. 11 was obtained by averaging the actual PRD values obtained for both channels of the 11 records at each bit rate. From this figure, one can see that MMP outperforms the SPIHT and WT-DCCR-TV-VQ for all data rates. Table I contains the same data of Fig. 11 in tabular form. In order to make a fair comparison of the proposed method's performance with the one in [19], Table II contains results for the first 10 minutes of the records considered in Table I. This was necessary because the results available in [19] correspond to 10 minutes and not to the full-length ECG signals. Table III contains MMP results for the first 10 minutes of the dataset B.

Although MMP outperforms SPIHT [4] and WT-DCCR-TV-VQ [5], this is not true for the JPEG2000-based method in [19]. A distinctive feature of this method is the period normalization that is carried out in order to assemble an image with each period being one line. This provides a very effective exploitation of the redundancy among periods of the quasi-periodic ECG waveform. One should note that a similar period normalization technique can be easily incorporated into MMP. In fact, with period normalization, the displacement dictionary could be used very effectively, since there would be a tendency to use with high probability the index of this dictionary corresponding to a displacement equal to the period.

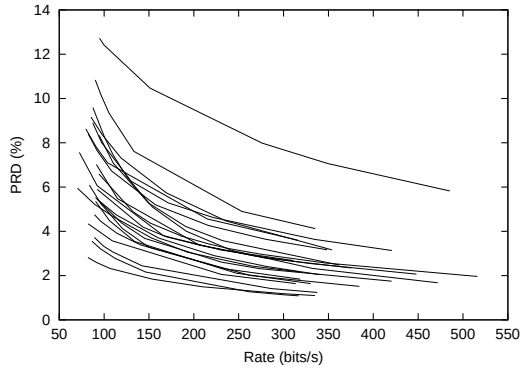

Fig. 10. Performance of MMP with the MIT/BIH database, full-length dataset A ($PRD \times CDR$).

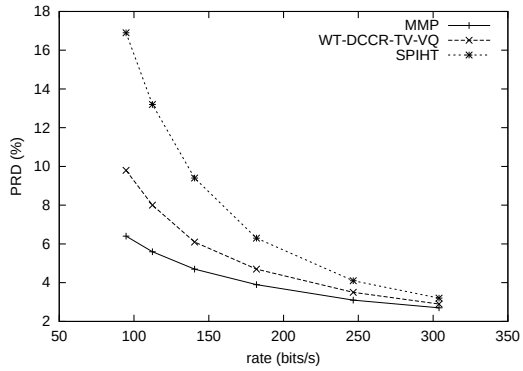

Fig. 11. Mean performance of MMP compared to SPIHT and WT-DCCR-TV-VQ for the dataset A (full-length).

Indeed, this is a promising area for further developments in ECG encoding using the MMP algorithm.

Fig. 12, 13, 14 and 15 contain 1 second of the channel 0 of the registers 100, 107, 117 and 119 respectively, at different PRD values and rates.

VIII. ANALYSIS OF THE RECONSTRUCTED SIGNALS

As presented in the last section, the performance of the proposed algorithm is good in a PRD sense. However, the usefulness of the reconstructed signals for diagnosis is yet to be assessed. In order to do so, it is desirable that we start with a brief discussion about the morphology of an ECG signal [21].

The heart is essentially a muscular pump. It pumps blood continuously through the circulatory system and consists of four chambers: right and left atria (upper chambers) and right and left ventricles (lower chambers). Each atrium connects to a ventricle. The right atrium receives blood carrying carbon dioxide from the body and moves it to the right ventricle. The right ventricle sends this blood to the lungs, where it is exchanged for oxygen-rich blood. This blood is received again by the heart in the left atrium. From the left atrium, the oxygen-rich blood goes to the left ventricle, which pumps it throughout the body. For an efficient pumping, these four chambers have to contract and relax in a co-ordinated fashion, which is carried out by electrical stimuli originated from a group of cells called sinus node.

TABLE I
PERFORMANCE COMPARISON OF MMP, SPIHT AND WT-DCCR-TV-VQ, FULL-LENGTH DATASET A ($PRD \pm \sigma$).

$CDR(bits/s)$	MMP	$WT-DCCR$ [5]	$SPIHT$ [5]
94.7	6.4 ± 2.45	9.8 ± 3.60	16.9 ± 3.68
112.3	5.6 ± 2.22	8.0 ± 2.97	13.2 ± 2.95
140.5	4.7 ± 1.99	6.1 ± 2.33	9.4 ± 2.32
181.9	3.9 ± 1.79	4.7 ± 1.86	6.3 ± 1.59
246.6	3.1 ± 1.57	3.5 ± 1.44	4.1 ± 1.09
304.0	2.7 ± 1.41	2.9 ± 1.23	3.2 ± 0.94

TABLE II
PERFORMANCE COMPARISON OF MMP AND JPEG2000, FIRST 10 MINUTES OF DATASET A (PRD).

CR	$JPEG2000$ [19]	MMP
8:1	1.52	1.96
10:1	1.86	2.34
16:1	2.74	3.29
20:1	3.26	3.86

The ECG has the objective of measuring the time variation of these electrical stimuli (voltage differences on the surface of the body due to the electrical fields). Fig. 16 shows a portion of the ECG signal corresponding to a cycle of the heart beating. The waves labeled with the letters **P**, **Q**, **R**, **S** and **T** correspond to special portions of the cycle. The **P** wave represents the depolarization of the atria and is usually 80 to 100ms in duration. The period of time from the beginning of the **P** wave to the beginning of the **QRS** complex is called the **P-R** interval, which is normally from 120 to 200ms in duration. This interval represents the time between the onset of atrial depolarization and the onset of ventricular depolarization. If the **P-R** interval is greater than 200ms, a conduction defect is present (first-degree heart block). The **QRS** complex represents ventricular depolarization. The duration of the **QRS** complex is normally 60 to 100ms, which indicates that ventricular depolarization is normally very fast. If the duration of **QRS** complex is greater than 100ms, conduction is probably impaired within the ventricles. The isoelectric period (**ST** segment) following the **QRS** is the time at which the entire ventricle is depolarized and corresponds to the plateau phase of the ventricular action potential. The **ST** segment is important in the diagnosis of ventricular ischemia or hypoxia, because under those conditions, the **ST** segment can become either depressed or elevated. The **T** wave represents ventricular repolarization and is longer than depolarization. The **Q-T** interval represents the time for both ventricular depolarization and repolarization to occur, and therefore is an estimation of the duration of an average ventricular action potential. This interval can range from 200 to 400ms, depending upon heart rate. At high heart rates, the duration of the ventricular action potentials decreases, which shortens the **Q-T** interval. In addition, prolonged **Q-T** intervals can be a diagnostic for susceptibility to certain types of tachyarrhythmias, being important to determine if a given **Q-T** interval is excessively long.

In what follows, using the basic concepts summarized in the last paragraph, we discuss the diagnostic information present in the reconstructed signals shown in Figs. 12 through 15.

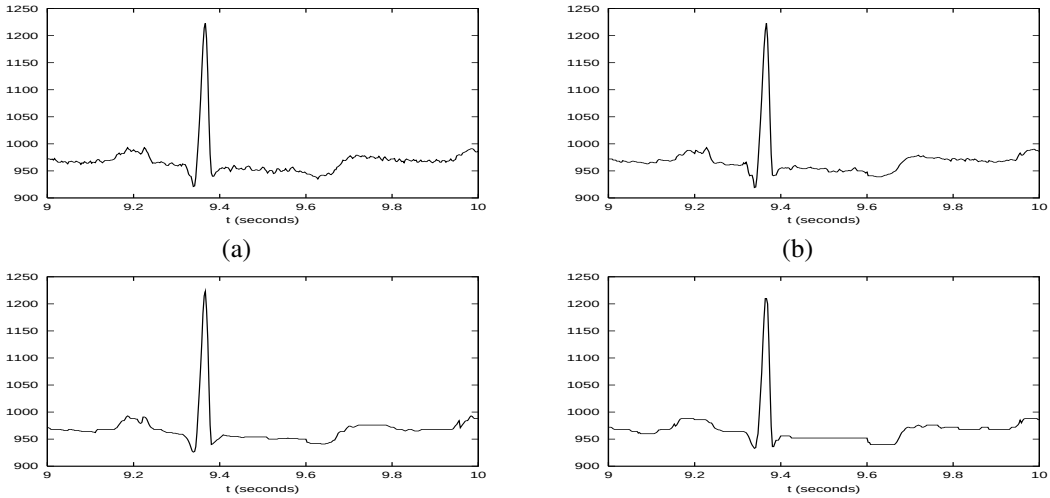

Fig. 12. Register 100: (a) original; (b) reconstructed ($PRD = 2.67$ and $CDR = 305.41$); (c) reconstructed ($PRD = 3.84$ and $CDR = 164.97$); (d) reconstructed ($PRD = 6.04$ and $CDR = 89.57$).

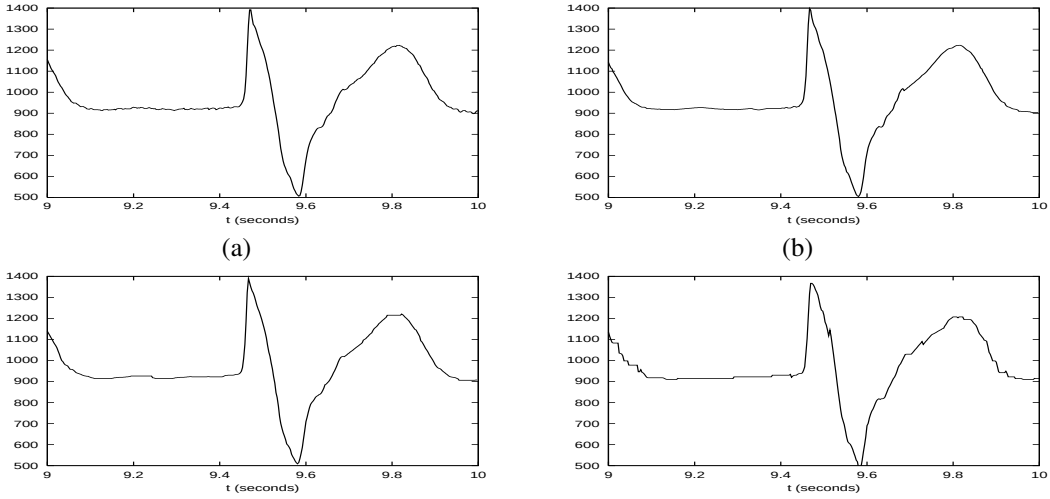

Fig. 13. Register 107: (a) original; (b) reconstructed ($PRD = 1.68$ and $CDR = 471.84$) ; (c) reconstructed ($PRD = 3.15$ and $CDR = 242.10$); (d) reconstructed ($PRD = 8.25$ and $CDR = 100.25$).

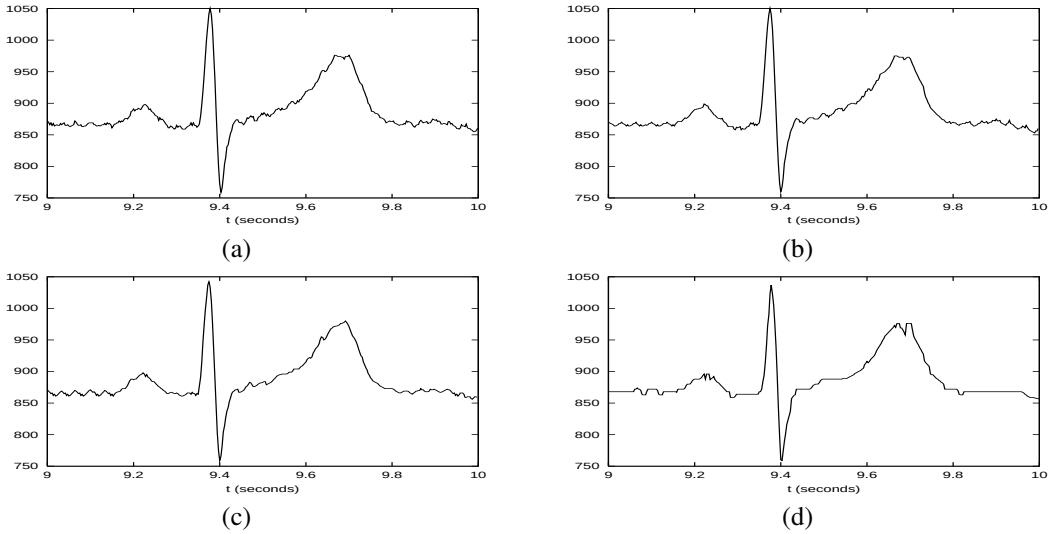

Fig. 14. Register 117: (a) original; (b) reconstructed ($PRD = 0.88$ and $CDR = 469.91$); (c) reconstructed ($PRD = 1.38$ and $CDR = 236.15$); (d) reconstructed ($PRD = 2.40$ and $CDR = 100.65$).

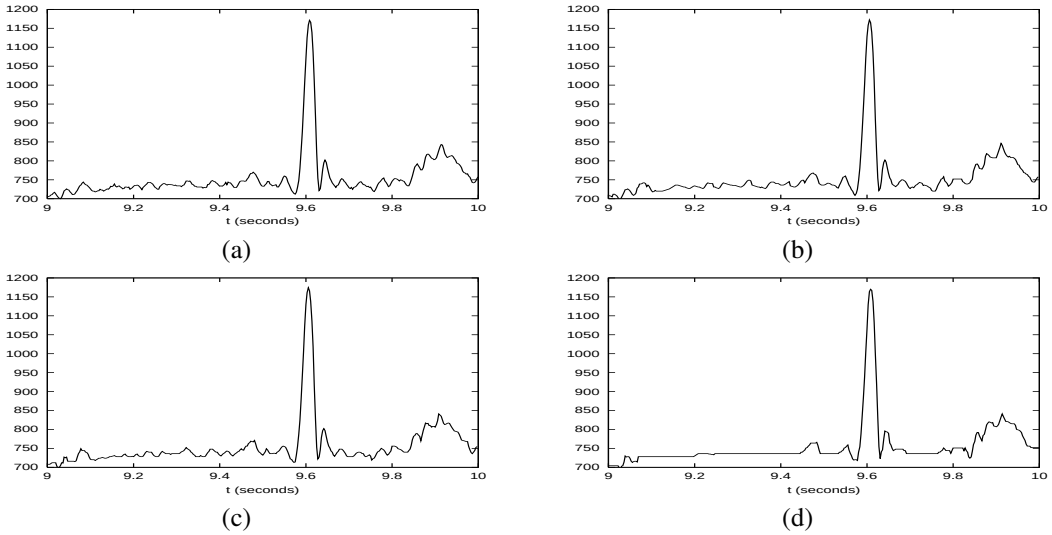

Fig. 15. Register 119: (a) original; (b) reconstructed ($PRD = 1.10$ and $CDR = 390.30$); (c) reconstructed ($PRD = 1.41$ and $CDR = 286.77$); (d) reconstructed ($PRD = 2.36$ and $CDR = 147.79$).

TABLE III
PERFORMANCE OF MMP FOR THE FIRST 10 MINUTES OF DATASET B (PRD).

CR	MMP
8:1	2.82
10:1	3.42
16:1	5.00
20:1	5.96

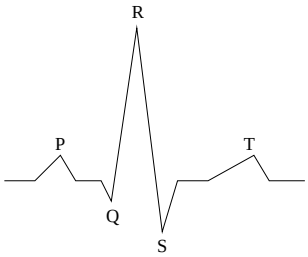

Fig. 16. ECG of a cycle of the heart beating.

Figs. 12(b) to 12(d) depict an ECG suitable for diagnostic analysis and compressed at 305.41, 164.97 and 89.57bps, respectively. The first two reconstructed signals present only moderate distortion in the **P** and **T** waves, being thus suitable for diagnosis. However, the reconstructed signal at 89.57bps (Fig. 12(d)) shows a great deal of distortion on its **T** wave and **ST** segment; this could compromise the diagnosis of coronary artery diseases like ischemia and angina (if present). A point worth noticing is that, in medical practice, an ECG signal is usually filtered prior to analysis (in order to remove noise from the electrical network and/or muscular tremor) and would look much like the compressed ones in Figs. 12(b) and 12(c).

The ECG in Fig. 13 presents a conduction impairment within the ventricles. This can be deduced from the longer than usual duration of the **QRS** complex. This feature is preserved even in the reconstructed signal at the lowest rate (Fig. 13(d)), which means that all the compressed signals can be used for effective diagnosis.

Another ECG tracing is shown in Fig. 14. Although the reconstructed signals are not very deformed, the ones at the two smaller rates (Figs. 14(c) – 236.15bps and 14(d) – 100.65bps) present impairments on the top of the **T** wave. However, such impairments would not affect their diagnosis.

The last signal, shown in Fig. 15, is very noisy. Therefore, it is difficult to detect the **P** wave even on the original signal. Indeed, each segment in a typical ECG is more related to one kind of derivation (channel). For a safe analysis of the **P** wave in this signal, another derivation would have to be considered. The first two reconstructed signals (Figs. 15(b) – 390.30bps and 15(c) – 286.77bps) preserve most of the features of the original signal and could be used without problem for diagnosis. The last reconstructed signal (Fig. 15(d) – 147.79bps) presents a moderate deformation in all the segments, but could still be used for diagnosis.

As can be seen from the above analysis, MMP-encoded ECG signals preserve most of the diagnostically useful information, even at very low rates. However, for a safe diagnosis, rates over 150bps should be preferred. A point worth noticing is that MMP always preserves, with high accuracy, the **QRS** complex.

IX. CONCLUSIONS

We have applied a recently developed universal lossy compression algorithm called MMP to the problem of ECG data encoding. The MMP algorithm is based on approximate multiscale pattern matching, an extension of ordinary pattern matching. It uses a dictionary of patterns, which is adaptively built while the data is being encoded, and a simple segmentation procedure that can be trivially extended to operate on multidimensional data. We incorporated some ideas reported on the literature to the basic algorithm, obtaining relevant improvements in performance, namely, the use of a prune-join strategy and of a side-match criterion. We also added a

displacement dictionary to help MMP to explore the quasi-periodicity of ECG signals. Originally used as an image data encoder, MMP performed very well to encode ECG data, having performance as good as the ones the best encoders known, although at the expense of a higher computational complexity (see appendix II). It also preserves most of the diagnostically useful information, even at low rates. Therefore, we have that MMP is an effective method for ECG compression and opens new avenues for the development of ECG encoders.

APPENDIX I
IMPLEMENTATION DETAILS

MMP is based in approximate pattern matching with scales. Therefore, we must define the specific scale transformation used. In our simulations we chose:

i) When changing from $\mathbf{v}$ of size N_0 to $\mathbf{v}^S$ of size $N > N_0$, $v^S(n)$ $(n = 0, 1, \ldots, N-1)$ is given by:

$$\begin{aligned}
m^0(n) &= \left\lfloor \frac{n(N_0-1)}{N-1} \right\rfloor \\
m^1(n) &= (n\,(N_0-1)) \mod (N-1) \\
m^2 &= (N-N_0) \mod (N_0-1) \\
r &= \frac{N-N_0}{N_0-1} \\
h &= v^S(n-1) \\
v^s(n) &= \begin{cases} v\left(m^0(n)\right), & for\ cd1 \\ \left\lfloor \frac{v\left(m^0(n)+1\right)-v\left(m^0(n)\right)}{r+1} \right\rfloor + h, & for\ cd2 \\ \left\lfloor \frac{v\left(m^0(n)+1\right)-v\left(m^0(n)\right)}{r+m^2+1} \right\rfloor + h, & for\ cd3 \end{cases} \\
cd1 &: m^1(n) = 0 \\
cd2 &: cd4\ and\ cd5 \\
cd3 &: otherwise \\
cd4 &: \left(m^1(n) > 0\right) \\
cd5 &: \left(\left(m^2 = 0\right)\ or\ \left(m^0(n) < (N_0-2)\right)\right)
\end{aligned} \tag{21}$$

ii) When changing from N_0 to $N < N_0$, $v^S(n)$ $(n = 0, 1, \ldots, N-1)$ is given by:

$$\begin{aligned}
u(k,n) &= \begin{cases} v\left(k + n\left\lfloor \frac{N_0}{N} \right\rfloor + 1\right), & for\ cd1 \\ v\left(k + n\left\lfloor \frac{N_0}{N} \right\rfloor - 1\right), & for\ cd2 \\ v\left(k + n\left\lfloor \frac{N_0}{N} \right\rfloor\right), & for\ cd3 \end{cases} \\
v^S(n) &= \sum_{k=-1}^{\left\lfloor \frac{N_0}{N} \right\rfloor} u(k,n) \\
cd1 &: k + n\left\lfloor \frac{N_0}{N} \right\rfloor < 0 \\
cd2 &: k + n\left\lfloor \frac{N_0}{N} \right\rfloor > (N_0-1) \\
cd3 &: otherwise
\end{aligned} \tag{22}$$

The input data $x(n), n = 0, 1, \ldots$ was segmented in blocks of size $N = 64$. That is, we initially parse $x(n)$ as a sequence of 64-dimensional vectors $x(n) = \{\mathbf{X}_0^0, \mathbf{X}_1^0, \ldots\}$, where $\mathbf{X}_m^0 = \begin{pmatrix} x(64m) & x(64m+1) & \ldots & x(64m+63) \end{pmatrix}$. Then, each input vector $\mathbf{X}_m^0, m = 0, 1, \ldots$ is independently encoded using one or more vectors in the dictionary. Before we start to encode $\mathbf{X}_m^0$, the dictionary $\mathcal{D}$ is initialized to $\mathcal{D}_0 = \{x_{min}, x_{min}+4, \ldots, x_{max}\}$, where x_{min} and x_{max} are respectively the minimum and the maximum values of the samples of $x(n)$. After the encoding of the segment $\mathbf{X}_m^0$, we keep the resulting dictionary $\mathcal{D}_m$ and use it as the initial dictionary to encode the segment $\mathbf{X}_{m+1}^0$. If $x(n)$ is too long, the dictionary can become too large. In order to prevent this, the dictionary can have at most 400000 elements. When the number of elements in the dictionary reaches this limit, we discard the oldest element in the dictionary whenever we include a new one (by oldest we mean that element that has not been used for the longest time). We also used a displacement dictionary with size $M = 1024$. In this case, the length of the vector containing the reconstructed samples for the displacement dictionary is greater than the length of the block, but the way the algorithm works is the same as described in section VI. Due to the side-match implementation, when attempting to encode the input vector $\mathbf{X}_m^0$, the previous encoded vector $\hat{\mathbf{X}}_{m-1}^0$ may be used in the evaluation of the state dictionary for the segments $\mathbf{X}_m^{2^p-1}, p = 0, 1, \ldots$, since the left-neighbors of these blocks are in the previous input vector. In that case, we can generalize the left-neighbor definition to:

$$\mathbf{L}_m^j = \begin{cases} \begin{pmatrix} \hat{x}(Fx) & \ldots & \hat{x}(Fy) \end{pmatrix}, & FP(j) > 0 \\ \begin{pmatrix} \hat{x}(Fz) & \ldots & \hat{x}(Fw) \end{pmatrix}, & FP(j) = 0 \end{cases} \tag{23}$$

where $Fx = Nm + FP(j) - N^j$, $Fy = Nm + FP(j) - 1$, $Fz = Nm - N^j$ and $Fw = Nm - 1$.

The specific rugosity metric used was:

$$\begin{aligned}
R(\mathbf{v}_k^s, j) = \Big| & \left|L_m^j(N^j-3) - L_m^j(N^j-1) + v_k^s(0) - \right. \\
& \left. v_k^s(2)\right| - \left\lfloor \frac{4}{3}\left|L_m^j(N^j-2) - v_k^s(1)\right| \right\rfloor \Big|
\end{aligned} \tag{24}$$

The number of elements N_S of the state dictionary $\mathcal{D}_S$ was calculated as:

$$N_S = \max\left(16\left\lfloor A\left(\mathbf{L}^j\right)\right\rfloor, 200\right) \tag{25}$$

where $A\left(\mathbf{L}^j\right)$ is given by (17).

The number of elements N_S^D of the displacement-state dictionary $\mathcal{D}_S^D$ was equal to $\frac{M}{4} = 256$.

MMP uses its dictionary as a codebook of a VQ. Therefore its performance should improve as the dictionary becomes larger. In order to speed up the rate of increase of the dictionary we add four new vectors at each update, instead of just one. When the segment $\hat{\mathbf{X}}^j$ is determined we include in the dictionary:

i) $\hat{\mathbf{X}}^j$. This is the main update, as described in section II.
ii) $\begin{pmatrix} \hat{x}(Fx) & \hat{x}(Fx+1) & \ldots & \hat{x}\left(Fx+N^j-1\right) \end{pmatrix}$ and $\begin{pmatrix} \hat{x}(Fy) & \hat{x}(Fy+1) & \ldots & \hat{x}\left(Fy+N^j-1\right) \end{pmatrix}$, where $Fx = FP(j) - \frac{N^j}{4}$ and $Fy = FP(j) - \frac{N^j}{2}$. These vectors were included with the side-match criterion in mind. The idea is to help to create a better state dictionary, since shifted versions of the least rugose elements tend to have low rugosity too.

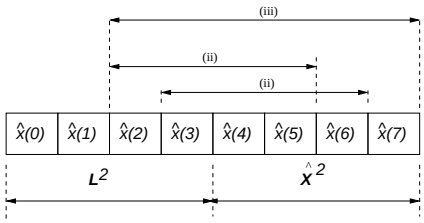

Fig. 17. Three new updates ($j = 2$).

iii) $\left(\hat{x}\left(Fx\right) \quad \hat{x}\left(Fx+1\right) \quad \ldots \quad \hat{x}\left(Fx+\frac{3Nj}{2}-1\right) \right)$, where $Fx = FP(j) - \frac{N^j}{2}$. This vector was added to increase the number of occurrences of joins in the optimization described in section IV.

These additional vectors are illustrated in Fig. 17

The MMP was implemented using multiple copies of the dictionaries, one at each scale. The indexes output by the algorithm were encoded by an adaptive arithmetic coder, with an independent context for each scale. The flags of the segmentation tree, the join flags and the flags used for the selection of the displacement dictionary were also encoded by the arithmetic encoder, using independent contexts at each scale. The contexts of the arithmetic coder were used by the R-D optimization procedure to estimate the probabilities needed to evaluate the rates associated to each symbol.

APPENDIX II
COMPUTATIONAL COMPLEXITY

In this appendix we make a simplified analysis of the computational complexity of MMP.

Considering that:

i) N_{seg} is the number of segments generated by the segmentation procedure of MMP. In other words, it is the number of leaves of the R-D optimized segmentation tree.

ii) The dictionary size increases of one unity for each two leaves, due to the updating procedure.

iii) The number of bits required to encode the segmentation tree is small when compared to the number of bits used for the dictionary indexes.

Then, the total number of bits B spent to encode $\mathbf{X}^0$ is:

$$B = \sum_{k=0}^{N_{seg}-1} \log_2 \left(|\mathcal{D}_0| + \left\lfloor \frac{k}{2} \right\rfloor \right) \tag{26}$$

The maximum size of the dictionary M, that is the size after the encoding of the last segment, is related to the number of segments as:

$$M = |\mathcal{D}_0| + \left\lfloor \frac{N_{seg}}{2} \right\rfloor \tag{27}$$

Equations (26) and (27) define a function $M(B)$ in parametric form (N_{seg} is the parameter). It should be noted that they are still valid, even when we perform a pre-segmentation of the input signal $x(n)$ in a sequence of vectors $\mathbf{X}_m^0$, as described in Appendix I. In this case, the number of segments N_{seg} can be greater than the length N of each $\mathbf{X}_m^0$. Fig. 18

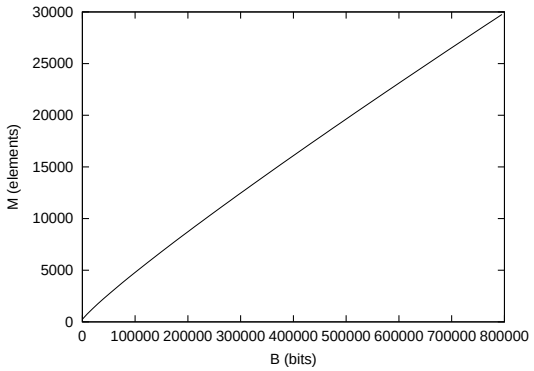

Fig. 18. Maximum dictionary size as a function of the number of bits B ($|\mathcal{D}_0| = 256$).

illustrates the variation of the maximum size of the dictionary as a function of B. From this figure, we can see that the storage requirement is nearly linearly related to the number of bits in the compressed file.

Also, considering that:

i) The computational complexity of a full search VQ is proportional to nS multiplications, where n is the dimension and S is the number of vectors in the codebook.

ii) The R-D optimization described in section III performs $(\log_2(N)+1)$ vector quantization operations on each input vector $\mathbf{X}_m^0$.

iii) The size of the dictionary is fixed during the R-D optimization of the segmentation tree of each input vector $\mathbf{X}_m^0$ (this corresponds to the simplified optimization solution, as in [9]).

Then, the size of the dictionary for $\mathbf{X}_m^0$ is given by:

$$S(m) = |\mathcal{D}_0| + \left\lfloor \frac{N_{seg} m N}{2\ell\left(x(n)\right)} \right\rfloor \tag{28}$$

and the complexity (number of multiplications per input sample) to encode $x(n)$ is:

$$C = \frac{N\left(\log_2(N)+1\right)}{\ell\left(x(n)\right)} \sum_{m=0}^{\left\lfloor \frac{\ell(x(n))}{N} \right\rfloor - 1} S(m) \tag{29}$$

Fig. 19 shows the variation of the complexity as a function of the total number of bits, for $x(n)$ pre-segmented in 10000 blocks of $N = 64$ samples each.

It is clear from Fig. 19 that the complexity increases nearly linearly with the size of the compressed file.

A detailed analysis of the complexity of the complete algorithm has yet to be done, but some comments are:

i) The inclusion of three additional vectors at each update of the dictionary, as described in appendix I, increases the complexity four times.

ii) The use of the side match criterion has the potential to decrease the complexity, since it reduces the size of the dictionary effectively used in the vector quantizations performed during the optimization, but it is signal dependent (smoother signals are encoded with smaller state dictionaries). We also have to take into account the additional calculations needed to built the state dictionary

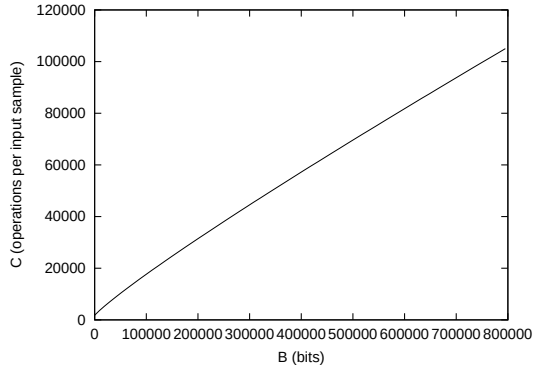

Fig. 19. Complexity as a function of the number of bits B ($|\mathcal{D}_0| = 256$, $N = 64$, $\ell(x(n)) = 640000$).

at each step of the R-D optimization, which increases the complexity.

iii) The complexity of the join step of the prune-join optimization is not addressed in our simplified analysis.

iv) The R-D optimization we actually use, as described in section III, may restart the dictionary updating procedure several times during the optimization. This increases the complexity since the Lagrangian costs of the nodes must be recalculated whenever the dictionary changes.

According to (29), a hardware solution using MMP would have to perform more than 18 million multiplications per second to compress in real time an ECG signal from the MIT/BIH database at $CDR = 200$ bits/s (which corresponds to a expected $PRD = 4$ %). This is not a low cost solution, considering the technology available today, but this can change as the hardware costs tends to continuously decrease over the years. Also, apart from the computational complexity, the state-of-the-art compression performance provided by MMP can be used as a reference to future works.

ACKNOWLEDGMENT

The authors would like to thank Dr. Katia do Nascimento Couceiro, associate professor at the Universidade Federal do Amazonas, for the valuable help in analyzing the ECG waveforms, and Aguinaldo Silva, team leader at Genius Institute of technology, for his continuous interest in this work.

REFERENCES

[1] M. B. de Carvalho, E. A. B. da Silva and W. A. Finamore, "Multidimensional signal compression using multiscale recurrent patterns", *Signal Processing: Image and Video Coding beyond Standards*, No. 82, pp. 1559-1580, Elsevier Science B. V., 2002.

[2] S. Jalaleddine, C. Hutchens, R. Strattan and W. Coberly, "ECG data compression techniques-A unified approach", *IEEE transactions on Biomedical Engineering*, Vol. 37, No. 4, pp. 329-343, April 1990.

[3] G. Nave and A. Cohen, "ECG compression using long term prediction", *IEEE transactions on Biomedical Engineering*, Vol. 40, No. 9, pp. 877-885, September 1993.

[4] Z. Lu, D. Y. Kim and W. A. Pearlman, "Wavelet compression of ECG signals by the set partitioning in hierarchical trees algorithm", *IEEE transactions on Biomedical Engineering*, Vol. 47, No. 7, pp. 849-856, July 2000.

[5] S. G. Miaou, H. L. Yen and C. L. Lin, "Wavelet-based ECG compression using dynamic vector quantization with tree codevectors in single codebook", *IEEE transactions on Biomedical Engineering*, Vol. 49, No. 7, pp. 671-680, July 2002.

[6] P. P. Vaidyanathan, *Multirate Systems and Filter Banks*, Englewood Cliffs, New Jersey: Prentice-Hall Inc., 1993.

[7] J. Ziv and A. Lempel, "Compression of individual sequences via variable-rate coding", *IEEE Transactions on Information Theory*, Vol. it-24, No. 5, pp. 530-536, September 1978.

[8] R. A. Blahut, *Principles and Practice of Information theory*, Cambridge, Massachusetts: Addison-Wesley publishing Company, 1988.

[9] G. J. Sullivan and R. L. Baker, "Efficient quadtree coding of images and video", *IEEE Transactions on Image Processing*, vol.3, No. 3, pp. 327-331, May 1994.

[10] M. B. de Carvalho, E. A. B. da Silva and W. A. Finamore, "Rate Distortion Optimized Adaptive Multiscale Vector Quantization", *IEEE International Conference on Image Processing*, Thessaloniki, October 2001.

[11] R. Shukla, P. L. Dragotti, M. N. Do and M. Vetterli, "Rate-Distortion Optimized Tree-Structured Compression Algorithms for Piecewise Polynomial Images," *IEEE Transactions on Image Processing*, Vol.14, No. 3, pp. 343-359, March 2005.

[12] T. Kim, "Side Match and Overlap Match Vector Quantizers for Images", *IEEE Transactions on Image Processing*, Vol. 1, No. 2, pp. 170-185, February 1992.

[13] E. B. L. Filho, M. B. Carvalho and E. A. B. da Silva, "Multidimensional signal compression using multi-scale recurrent patterns with smooth side-match criterion", *IEEE International Conference on Image Processing*, Singapore, October, 2004.

[14] J. Vaisey and A. Gersho, "Variable block-size image coding", *IEEE International Conference Acoustics, Speech and Signal Processing*, Dallas, TX, April 1987, pp. 1051-1054.

[15] P. A. Chou, T. Lookabaugh and R. M. Gray, "Optimal pruning with applications to tree-structure source coding and modeling", *IEEE Transactions on Information Theory*, vol. 35, No. 2, pp. 299-315, March 1989.

[16] C. Y. Wang, S. J. Liao and Long Wen Chang, "Wavelet image coding using variable blocksize vector quantization with optimal quadtree segmentation", *Signal Processing: Image Communication*, No. 15, pp. 879-890, Elsevier Science B. V., 2000.

[17] S. B. Yang and L. Y. Tseng, "Smooth Side-Match Classified Vector Quantizer with Variable Block Size", *IEEE Transactions on Image Processing*, Vol. 10, No. 5, pp. 677-685, May 2001.

[18] I. H. Witten, R. M. Neal and J. G. Cleary, "Arithmetic Coding for Data Compression", *Communications of the ACM*, Vol. 30, No. 6, pp. 520-540, June 1987.

[19] A. Bilgin, M. W. Marcellin and M. I. Altbach, "Compression of Electrocardiogram Signals using JPEG2000", *IEEE Transactions on Consumer Electronics*, Vol. 49, No. 4, pp. 833-840, Nov. 2003.

[20] S. G. Miaou and S. N. Chao, "Wavelet-Based Lossy-to-Lossless ECG Compression in a Unified Vector Quantization Framework", *IEEE Transactions on Biomedical Engineering*, Vol. 52, No. 3, pp. 539-543, March 2005.

[21] R. E. Klabunde, *Cardiovascular Physiology Concepts*, Philadelphia, Pennsylvania: Lippincott Williams & Wilkins, 2004.

Eddie Batista de Lima Filho was born in Manaus, AM, Brazil, in 1977. He received his B.S. degree in electrical engineering from the Universidade Federal do Amazonas (UFAM), Manaus, AM, Brazil, in 1999, and his M.S. degree in electrical engineering from the Universidade Federal do Rio de Janeiro (COPPE/UFRJ), Rio de Janeiro, RJ, Brazil, in 2004. Since 2001, he has been with the Genius Institute of Technology, Manaus, AM, Brazil, working with Digital TV and Embedded Systems. He is currently working toward the Ph.D. degree in electrical engineering at the Universidade Federal do Rio de Janeiro (COPPE/UFRJ). His research interests include channel coding, distributed source coding and video/image compression.

Eduardo A. B. da Silva (M'95, SM'05) was born in Rio de Janeiro, Brazil in 1963. He received the Engineering degree in Electronics from Instituto Militar de Engenharia (IME), Brazil, in 1984, the M.Sc. degree in Electrical Engineering from Universidade Federal do Rio de Janeiro (COPPE/UFRJ) in 1990 and the Ph.D. degree in Electronics from the University of Essex, England, in 1995. In 1987 and 1988 he was with the Department of Electrical Engineering at Instituto Militar de Engenharia, Rio de Janeiro, Brazil. Since 1989 he has been with the Department of Electronics Engineering (the undergraduate dept.), UFRJ. He has also been with the Department of Electrical Engineering (the graduate studies dept.), COPPE/UFRJ, since 1996. He has been head of the Department of Electrical Engineering, COPPE/UFRJ, Brazil, for the year 2002. He has published more than 30 journal papers and book chapters, and has more than 40 papers published in international conferences. He won the British Telecom Postgraduate Publication Prize in 1995, for his paper on aliasing cancellation in subband coding. He is also co-author of the book "Digital Signal Processing - System Analysis and Design", published by Cambridge University Press, in 2002, that has also been translated to the Portuguese and Chinese languages. He has served as associate editor of the IEEE Transactions on Circuits and Systems - Part I, in 2002 and 2003. He has been a Distinguished Lecturer of the IEEE Circuits and Systems Society in 2003 and 2004. He has given training and consultancy for several Brazilian cable and satellite television companies on digital television. He is part of the team working towards the development of the Brazilian Digital Television System. His research interests lie in the fields of digital signal and image processing, especially signal compression, digital television, wavelet transforms, mathematical morphology and applications to telecommunications. He is a senior member of the IEEE.

Murilo Bresciani de Carvalho was born in Petrópolis, Brazil in 1964. He received the B. E. degree in Electrical Engineering from Universidade Federal Fluminense (UFF), Brazil in 1986, the M.Sc. degree in Telecommunications from Pontifícia Universidade Católica do Rio de Janeiro (PUC-RJ) in 1994 and the D.Sc. degree in Signal Processing from Universidade Federal do Rio de Janeiro (COPPE/UFRJ) in 2001. Since 1994 he has been with the Department of Telecommunications Engineering of UFF. His main interests include digital image/video processing, source/channel coding and digital signal processing.

Waldir Sabino da Silva Júnior received his B.S. degree in electrical engineering from the Universidade Federal do Amazonas (UFAM), Manaus, AM, Brazil, in 2000, and his M.S. degree in electrical engineering from the Universidade Federal do Rio de Janeiro (COPPE/UFRJ), Rio de Janeiro, RJ, Brazil, in 2004. Since 2002, he has been with the Fundação Centro de análise, Pesquisa e Inovação Tecnológica, Manaus, AM, Brazil, as associate professor. He is currently pursuing a Ph.D. degree in electrical engineering at the Universidade Federal do Rio de Janeiro (COPPE/UFRJ). His research at the Universidade Federal do Rio de Janeiro is in the field of data compression, but he is also interested in mathematical morphology and digital signal processing in general.

José Koiller received his B.S. degree in electrical engineering from the Universidade Federal do Rio de Janeiro (UFRJ), Brazil, in 1999, and his M.S. degree in applied mathematics from the Instituto de Matemática Pura e Aplicada (IMPA), Brazil, in 2001. He is currently a CAPES/Brazil fellow pursuing a Ph.D. degree in mathematics at the Courant Institute of Mathematical Sciences, New York. His research at the Courant Institute is in the field of ergodic theory of differentiable dynamical systems, but he remains interested in digital signal processing, and data compression in particular.