Submitted to the IEEE Transactions on Biomedical Engineering

Wavelet Compression of ECG Signals by the Set Partitioning in Hierarchical Trees (SPIHT) Algorithm

Zhitao Lu, Dong Youn Kim [*], and William A. Pearlman
Electrical, Computer and Systems Engineering Department
Rensselaer Polytechnic Institute
Troy, NY 12180-3590

January 12, 2000

Abstract

A wavelet ECG data codec based on the Set Partitioning In Hierarchical Trees (SPIHT) compression algorithm is proposed in this paper. The SPIHT algorithm [1] has achieved notable success in still image coding. We modified the algorithm for the one-dimensional (1-D) case and applied it to compression of ECG data. Experiments on selected records from the MIT-BIH arrhythmia database revealed that the proposed codec is significantly more efficient in compression and in computation than previously proposed ECG compression schemes. The coder also attains exact bit rate control and generates a bit stream progressive in quality or rate.
KEYWORDS: ECG signal compression, wavelet signal processing, portable heart monitoring.

1 Introduction

Multichannel ECG data provide cardiologists with essential information to diagnose heart disease in a patient. In an ambulatory monitoring system, the volume of ECG data is necessarily large, as a long period of time is required in order to gather enough information about the patient. As an example, with the sampling rate of 360 Hz, 11 bits/sample data resolution, a 24-hour record requires about 43 Mbytes per channel. Therefore, an effective data compression scheme for ECG signals is required in many practical applications including: (a)ECG data storage; (b) ambulatory recording systems; and (c) ECG data transmission over telephone line or digital telecommunication network. Compression schemes used on ECG data fall into two categories: direct and transform schemes. Examples of direct schemes that attempt to code the signal directly are FAN, AZTEC, CORTES, and ASEC. A good

[*]D.Y. Kim is with the Department of Biomedical Engineering and Research Institute of Medical Engineering, Yonsei University, Wonju, Korea.

review and comparison of some of these methods are presented in [2]. Among transform schemes, the wavelet transform schemes have shown promise because of their good localization properties in the time and frequency domain. Several wavelet and/or wavelet packet based compression algorithms have been proposed in [3] [4] [5].

In this paper, we propose a wavelet coder based on the the Set Partitioning in Hierarchical Trees (SPIHT) compression algorithm. This algorithm is considered the premier state-of-the-art algorithm in image compression and is here modified to suit the special characteristics of ECG signals. The paper is organized as follows: Section 2 is a brief introduction to the wavelet transform and its filter bank implementation. Section 3 presents the coding algorithm based on 1-D SPIHT. Section 4 is a simple example to show how the coding algorithm works. We applied the proposed codec on selected records from the MIT-BIH arrhythmia database and present the results and comparisons with other coders in the literature in Section 5. Section 6 concludes the paper.

2 Wavelet Transform

The wavelet transform comprises the coefficients of the expansion of the original signal x(t) with respect to a basis $\psi_{\omega,n}(t)$, each element of which is a dilated and translated version of a function ψ called the mother wavelet, according to

$$\psi_{\omega,n}(t) = \frac{1}{\sqrt{2^\omega}}\psi(\frac{t-2^\omega n}{2^\omega}),\ \omega, n \in Z, \tag{1}$$

where Z is the set of integers. Depending on the choice of the mother wavelet appropriately, the basis can be orthogonal or biorthogonal. The wavelet transform coefficients, given by the inner product of $x(t)$ and the basis functions,

$$W(\omega, n) =< x(t), \psi_{\omega,n}(t) > \tag{2}$$

comprise the time-frequency representation of the original signal. The wavelet transform has good localization in both frequency and time domains, having fine frequency resolution and coarse time resolution at lower frequency, and coarse frequency resolution and fine time resolution at higher frequency. Since this matches the characteristic of most signals, it makes the wavelet transform suitable for time-frequency analysis. In data compression, the wavelet transform is used to exploit the redundancy in the signal. After the original signal is transformed into the wavelet domain, many coefficients are so small that no significant information is lost in the signal reconstructed by setting these coefficients to zero.

In digital signal processing, the fast forward and inverse wavelet transforms are implemented as tree-structured, perfect-reconstruction filter banks. The input signal is divided into contiguous, nonoverlapping blocks of samples called frames and is transformed frame by frame for the forward transform. Within each frame, the input signal is filtered by the analysis filter pair to generate lowpass and highpass signals,which are then downsampled by a factor of two. Then this analysis filter pair is applied to the downsampled lowpass signal recursively to generate layered wavelet coefficients shown in Figure 1. In different layers, the coefficients have different frequency and time resolution. In layer i, each coefficient corresponds to two coefficients in layer $i+1$ in the time domain. For the inverse transform, the

coefficients in the highest layer are upsampled by a factor of two (zeros are inserted between successive samples), filtered by the low- and high-pass synthesis filter and added together to get the lowpass signal for next layer. This process is repeated for all layers until the full size signal is reached to complete the inverse transform.

The selection of different analysis-synthesis filter pairs, which correspond to different wavelet bases, is very important for obtaining effective data compression. For the design of perfect reconstruction filter pairs, being beyond the scope of this paper, we refer the reader to the literature, such as [6] and many other works.

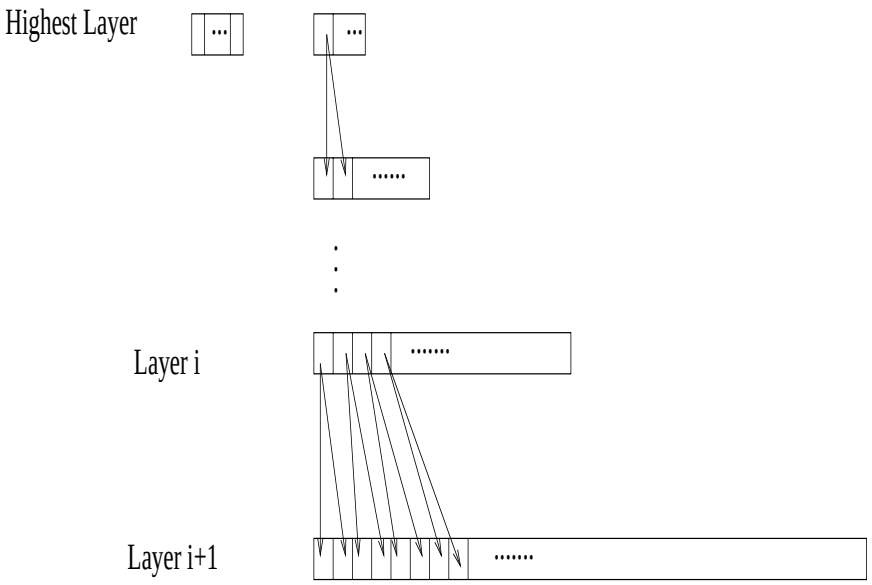

Figure 1: The temporal orientation tree

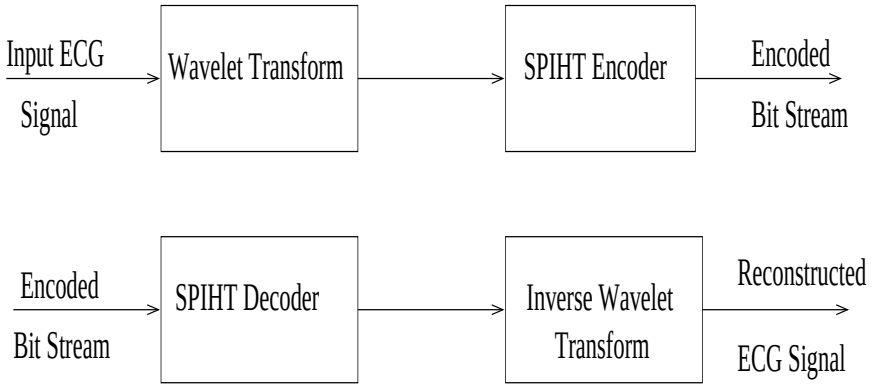

Figure 2: The diagram of the proposed encoder and decoder

In implementation, the frame size, number of layers of the wavelet transform and the filter pair need to be appropriately selected. The number of layers determined the coarsest frequency resolution of the transform and should be at least four for adequate compression. The frame size is taken to be a power of 2 that exceeds the number of layers. The frame should contain several periods of the ECG signal, but should still be short enough for acceptable

Table 1: The Coefficients of the Biorthogonal 9/7 Tap Filters

Lowpass	0.852699	0.377403	-0.11062	-0.023849	0.037829
Highpass	0.788485	0.418092	-0.04069	-0.064539	

coding delay and memory usage. Our choices of 6 layers of wavelet decomposition and 1024 sample frames fulfill the requirements. Among the potential perfect reconstruction filter pairs, we seek a good tradeoff between compression performance, generally better for longer filters, and computational complexity, obviously smaller for the shorter filters. The biorthogonal 9/7 tap filters [9], whose coefficients are in Table 1, have been chosen, because they have proved to offer the best such tradeoff for images and have shown to give the best compression performance for wavelet coding of ECG signals among all filters tested in [4]. Since these filters are symmetric, we employ a symmetric(reflective) data extension scheme at the boundaries of the frames to obtain perfect reconstruction at the boundaries in the absence of coding.

3 Coding Algorithm

After the wavelet transform, we use the SPIHT algorithm to encode the wavelet coefficients. The SPIHT algorithm has received widespread recognition for its notable success in image coding [1]. We have also implemented it in the case of one dimension (1-D) for coding wavelet packet transforms of audio signals and obtained very good compression performance [8]. Here we apply the SPIHT algorithm to the wavelet (purely dyadic) transform of ECG signals. The diagram of the encoder and decoder is shown as in Figure 2.

The principles of the SPIHT algorithm are partial ordering of the transform coefficients by magnitude with a set partitioning sorting algorithm, ordered bit plane transmission and exploitation of self-similarity across different layers. By following these principles, the encoder always transmits the most significant bit to the decoder.

3.1 Temporal Orientation Trees

As shown in Figure 1, a tree structure, called "temporal orientation tree", defines the temporal relationship in the wavelet domain. Every point in layer i corresponds to 2 points in the next layer $i+1$, with the arrow indicating the parent-offspring relation. This definition is analogous to that of spatial orientation trees in [1]. Each node either has no offspring or 2 offspring. In a typical 1-D signal, most of the energy is concentrated in low frequency bands, so that the coefficients are expected to be better magnitude-ordered as we move downward following the temporal orientation tree to the leaves (terminal nodes).

3.2 Set Partitioning Sorting Algorithm

The same set partitioning rule is defined in the encoder and decoder. The subset of subband coefficients c_i in the subset $\mathcal{T}$ is said to be significant for bit depth n if $\max_{i\in\mathcal{T}}\{|c_i|\} \geq 2^n$, otherwise it is said to be insignificant. If the subset is insignificant, a 0 is sent to the decoder. If it is significant, a 1 is sent to the decoder and then the subset is further split according to the temporal orientation tree until all the significant sets are a single significant point. In this stage of coding, called the *sorting pass*, the indices of the coefficients are put onto three lists, the list of insignificant points (LIP), the list of insignificant sets (LIS), and the list of significant points (LSP). In this pass, only bits related to the LSP entries and binary outcomes of the magnitude tests are transmitted to the decoder. In implementation, we grouped together the entries in the LIP and LIS which have the same parent into an entry atom. For each entry atom in LIP, we estimated a pattern in both encoder and decoder to describe the significance status of each entry in the current sorting pass. If the result of the significance test of the entry atom is the same as the specified pattern, we can use one bit to represent the status of the whole entry atom which otherwise had two entries and representation of significance by two bits. If the significance test result does not match the pattern, we transmitted the result of the significance test for each entry in the atom. Since the ECG signal has periodic characteristics, we correctly estimated the pattern with high probability, so were able to save one bit frequently enough to give noticeable improvement in compression performance.

3.3 Refinement Pass

After each sorting pass, we get the significant coefficients for the threshold 2^n, and then send to the decoder the nth most significant bit of every coefficient found significant at a higher threshold. By transmitting the bit stream in this ordered bit plane fashion, we always transmit the most valuable (significant) remaining bits to the decoder.

The outline of the full coding algorithm is as follows:

1. Initialization.
   Set the list of significant points (LSP) as empty. Set the roots of similarity trees in the lists of insignificant points (LIP) and insignificant sets (LIS). Set the significance threshold 2^n with $n = \lfloor \log_2(max_{(i)}|c_i|) \rfloor$

2. Sorting pass.
   Using the set partitioning algorithm distribute the appropriate indices of the coefficients to the LIP, LIS,and LSP.

3. Refinement pass:
   For each entry in the LSP significant for higher n,
   send the nth most significant bit to the decoder.

4. Decrement n by one and return to step 2
   until the specified bit rate is reached.

4 An Example to Show the Coding Process

In this section, we use a simple example to show how the coding algorithm works. A four level wavelet decomposition of an input signal of length 32 produces the 32 wavelet coefficients distributed among the subbands as shown in Figure 3, with the arrows indicating the parent-offspring relationships in the temporal trees. The number in each cell is the value of the integer-rounded wavelet coefficient. The actions of the coding process are shown in Table 2. Following are some of the important definitions and explanations in Table 2.

LIS List of insignificant sets: contains sets of wavelet coefficients which are defined by tree structures, and which had been found to have magnitude smaller than a threshold (are insignificant). The sets are designated by, but exclude the coefficient corresponding to the tree or all subtree roots, and have at least two elements.

LIP List of insignificant points: contains individual coefficients that have magnitude smaller than the threshold.

LSP List of significant points: points found to have magnitude larger than the threshold (are significant).

$\mathcal{O}(c_i)$ in the tree structures, the set of offspring (direct descendants) of a tree node defined by point location (i).

$\mathcal{D}(c_i)$ set of descendants of node defined by point location (i).

$\mathcal{L}(c_i)$ set defined by $\mathcal{L}(c_i) = \mathcal{D}(c_i) - \mathcal{O}(c_i)$.

Type A entry in LIS: the entry i represents $\mathcal{D}(c_i)$.

Type B entry in LIS: the entry i represents $\mathcal{L}(c_i)$.

1. The largest coefficient magnitude is 59, so the threshold is 32.
   The LSP set is empty, the initial LIP are coefficients $\{0, 1, 2, 3\}$ and initial LIS are coefficients $\{2, 3\}$.

2. Sorting pass in LIP:
   SPIHT begins to code the significance of individual coefficients in LIP. c0 is significant: 1 is sent followed by a positive sign bit , and c0 is moved to the LSP. c1 is significant: 1 is sent followed by a negative sign bit , and c1 is moved to the LSP. (1+ represents positive significant, 1- represents negative significant). c2 and c3 are both insignificant, so 0 is sent for each.

3. Sorting pass in LIS:
   After finishing the LIP, SPIHT begins to test the LIS(active entry indicated by bold letter). For type A entry, When an entry in LIS is significant, 1 is sent. Then its two offspring are checked like an entry in the LIP. If $\mathcal{L}(c_i)$ is not empty, that entry is moved to the end of the LIS and changed to type B. If $\mathcal{L}(c_i)$ is empty, that entry is removed from the LIS. When an entry in the LIS is insignificant, 0 is sent. In this case, the type

A $\mathcal{D}(c2)$ is found significant, and split into offspring c4, c5, and $\mathcal{L}(c2)$, which goes to the end of the LIS as type B. c4 and c5 are found to be insignificant, they are moved to the LIP and two 0's are sent. $\mathcal{D}(c3)$ is insignificant, so a 0 is sent.

4. For a type B LIS entry, if it is significant, 1 is sent, add its two offspring to the LIS as type A, and remove that entry from LIS. If it is insignificant, 0 is sent. In this case, $\mathcal{L}(c2)$ is significant, so a 1 is sent and the offspring of c2, c4 and c5, become roots of type A sets in the LIS and $\mathcal{L}(c2)$ (2B) is removed from the LIS. $\mathcal{D}(c4)$ and $\mathcal{D}(c5)$ are then tested as above with the actions given in the table.

5. Refinement Pass: After the sorting pass. SPIHT begins the refinement pass. We check each old entry of LSP (the coefficients which became significant under the last threshold). Send 1 if it is significant under this threshold and reduce its magnitude by the current threshold. Since this is the first refinement pass, there are no old LSP entries. These new entries of LSP, c0, c1 and c8, are reduced in magnitude by the current threshold of 32, so that their values become c0(27), c1(16), c8(11).

6. Reduce the threshold to 16.

7. Sorting Pass in LIP: Check the significance for LIP entries under threshold 16. c2 and c3 are significant and moved to the LSP, while c4, c5, and c9 remain insignificant.

8. Sorting Pass in LIS: Check the significance for LIS entries under threshold 16.

9. Refinement Pass: check old LSP members c0, c1, c8, send their significance information, reduce the magnitude of significant old LSP entry and all new entry in LSP. Their value become c0(11), c1(0), c8(11), c2(9), c3(5), c16(6).

10. Reduce the threshold to 8 and repeat sorting pass and refinement pass until the bit budget or quality requirement is reached.

In the decoder side, the same process is running. The only difference is that the significant/insignificant decisions found in the encoder by comparing the coefficients to a threshold are input to the decoder. Since the lists are initialized identically, they are formed in the decoder exactly as in the encoder. In the refinement pass, the threshold is added to the significant coefficients, instead of subtracted. (The addition or subtraction of threshold is equivalent to adding or removing a bit in a bit plane representation of the coefficient's magnitude.)

Note that the encoding and decoding are comprised of simple operations: comparison to threshold, movement of co-ordinates to lists, and bit manipulations. There are no complex calculations needed for modeling and training prior to coding. The only search is the single search for the initial threshold. The method is completely self-adaptive, always finding the most significant bits of the largest coefficients and sending them before those bits of smaller coefficients. The method is also extremely efficient, as it has the capability to locate large descendent sets with maximum magnitude smaller the final threshold and representing them with a single 0.

Step	Point or Set Tested	Output Bit	Action	Control Lists	code bits accumulated
(1)				LIS = {2A, 3A} LIP = {0, 1, 2, 3} LSP = $\emptyset$	
(2)	$c0$	1 +	c0 to LSP	LIP = {1, 2, 3} LSP = {0}	1 2
	$c1$	1 -	c1 to LSP	LIP = {2, 3} LSP = {0, 1}	3 4
	$c2$	0	none		5
	$c3$	0	none		6
(3)	$\mathcal{D}(c2)$	1	test offspring	LIS = {**2A**, 3A}	7
	$c4$	0	c4 to LIP	LIP = {2, 3, 4}	8
	$c5$	0	c5 to LIP	LIP = {2, 3, 4, 5}	9
			type changes	LIS = {3A, 2B}	
	$\mathcal{D}(c3)$	0	none	LIS = {**3A**, 2B}	10
(4)	$\mathcal{L}(c2)$	1	add new sets	LIS = {3A, 4A, 5A}	11
	$\mathcal{D}(c4)$	1	test offspring	LIS = {3A, **4A**, 5A}	12
	c8	1+	c8 to LSP	LSP = {0, 1, 8}	13,14
	c9	0	c9 to LIP	LIP = {2, 3, 4, 5, 9}	15
			type changes	LIS = {3A, 5A, 4B }	
	$\mathcal{D}(c5)$	0	none	LIS = {3A, **5A**, 4B }	16
	$\mathcal{L}(c4)$	0	none	LIS = {3A, 5A, **4B** }	17
(5)				LIS = {3A, 5A, 4B} LIP = {2, 3, 4, 5, 9} LSP = {0, 1, 8}	
(6)		reduce threshold			
(7)	c2	1 -	c2 to LSP	LSP = {0, 1, 8, 2} LIP = {3, 4, 5, 9}	18 19
	c3	1 +	c3 to LSP	LSP = {0, 1, 8, 2, 3} LIP = {4, 5, 9}	20 21
	c4	0	none	LIP = {4, 5, 9}	22
	c5	0	none	LIP = {4, 5, 9}	23
	c9	0	none	LIP = {4, 5, 9}	24
(8)	$\mathcal{D}(c3)$	0	none	LIS = {**3A**, 5A, 4B}	25
	$\mathcal{D}(c5)$	0	none	LIS = { 3A, **5A**, 4B}	26
	$\mathcal{L}(c4)$	1	add new sets	LIS = {3A, 5A, 8A, 9A}	27
	$\mathcal{D}(c8)$	1	test offspring	LIS = {3A,5A,**8A**, 9A}	28
	c16	1+	c16 to LSP	LSP = {0, 1, 8, 2, 3, 16}	29, 30
	c17	0	c17 to LIP	LIP = {4, 5, 9, 17}	31
			remove c8 from LIS	LIS = {3A, 5A, 9A}	
	$\mathcal{D}(c9)$	0	test offspring	LIS = {3A,5A, **9A**}	32
(9)	c0	1			33
	c1	1			34
	c8	0			35
(10)		reduce threshold			
...					

Table 2: Coding example using the SPIHT method.

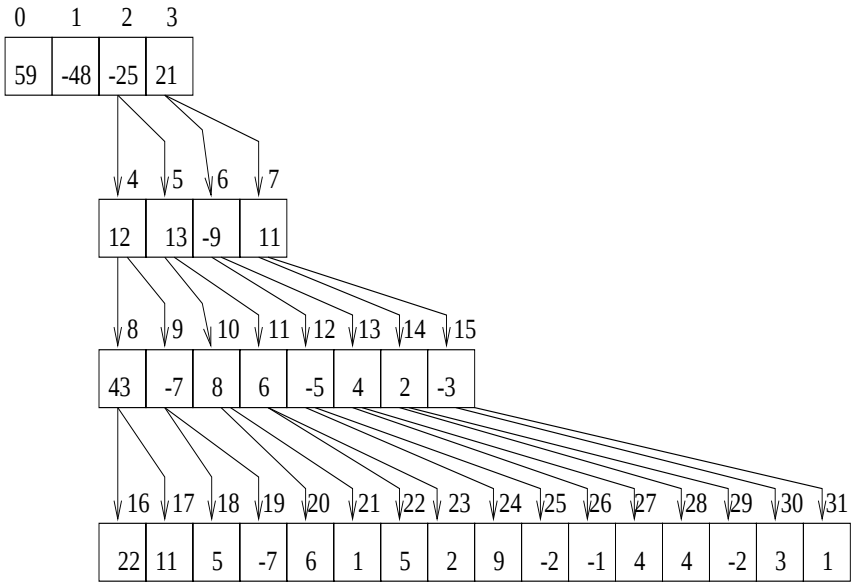

Figure 3: The subbands and temporal trees of the example.

5 Results

We used data in the MIT-BIH arrhythmia database to test the performance of our coding schemes. All ECG data used here are sampled at 360Hz, and the resolution of each sample is 11 bits/sample, so that total bitrate of these data is 3960 bps. The distortion between the original and reconstructed signal is measured by Percent Root mean square Difference(PRD). Although PRD does not exactly correspond to the result of a clinical subjective test, it is easy to calculate and compare, so is widely used in the ECG data compression literature. The formula used to calculate the PRD is as follows:

$$PRD = \sqrt{\frac{\sum_{i=1}^{n}[x_{ori}(i) - x_{rec}(i)]^2}{\sum_{i=1}^{n} x_{ori}(i)^2}} \times 100 \tag{3}$$

where x_{ori} denotes the original data [1], x_{rec} denotes the reconstructed data, and n, the number of samples within one data frame.

We compare the performance of our encoder with two kind of encoders in the literatures– wavelet based codec and direct ECG signal codec. Since the data used in the literatures are usually different in sampling frequency, and sample resolution, exact quantitative comparisons are inconclusive. Nonetheless, we compared the PRD results in similar compression ratio.

5.1 Comparison with Wavelet Codecs

The test datasets are taken from the MIT-BIH arrhythmia database. The record numbers for the first dataset are 100, 101, 102, 103, 107, 109, 111, 115, 117, 118, 119, which consist

[1]The test data included a baseline of 1024 added for storage purposes. In the PRD formula, a level of 1024 is subtracted from each data sample to give $x_{ori}(i)$.

Table 3: Average Test Results for the First Dataset

CR	4:1	5:1	8:1	10:1	12:1	16:1	20:1
PRD	1.19	1.56	2.46	2.96	3.57	4.85	6.49
Time(encoder,sec)	8.05	7.62	7.04	6.86	6.72	6.58	6.48
Time(decoder,sec)	4.79	4.48	4.08	3.95	3.85	3.75	3.68

Times recorded in an SGI Indy Workstation with 133 MHz CPU.

of different rhythms, QRS complex morphologies and ectopic beats. We encoded 10 minutes of data from each of these records. We report compression ratios from actual compressed file sizes and PRD's from decompressing these compressed files. Figure 4 shows the PRD value versus compression ratio for each record of data. We remark that these results are obtained by decoding each record's compressed file at different truncation lengths to obtain the different PRD vs. CR (compression ratio) points. The PRD vs. CR curves are shown aggregately in Figure 4, and the average PRD values of this dataset are presented in Table 3.

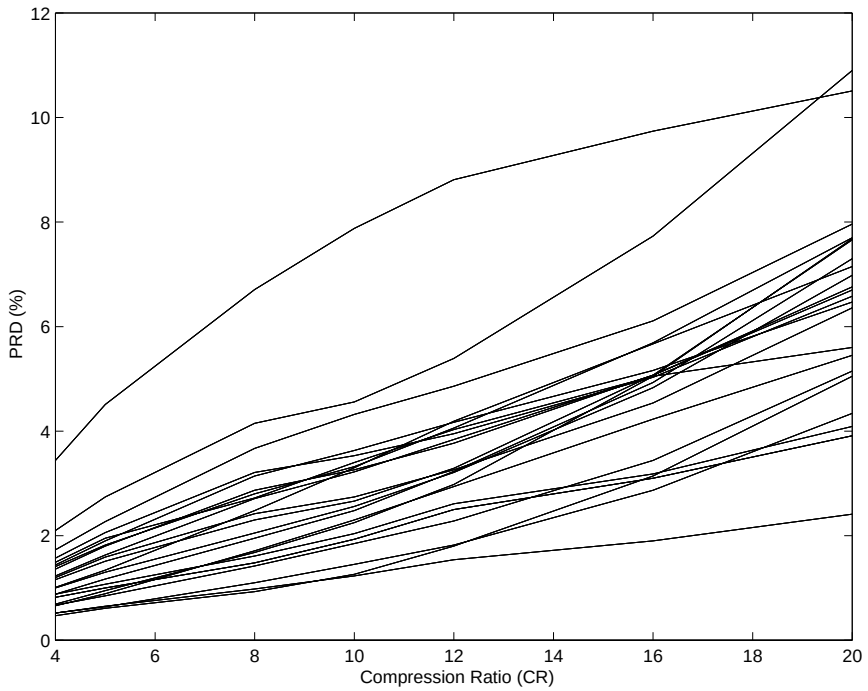

Figure 4: The PRD results of MIT-BIH ECG data

From Figure 4, we see that the results for all data are close to each other, except for the two with the highest PRD's. That means the coder presented here is suitable for a variety of ECG data. Also the performance of the coder degrades only gradually when the compression ratio becomes larger. To exhibit the effect of compression on the reconstructed signal, we reproduce in Figure 5 5.69 seconds (2048 samples) of the original signal from Record 117A

and reconstructed signals decoded at different bitrates of 1000 bps, 400 bps and 200 bps. The chief effect of compression, especially noticeable at 200 bps, is the smoothing of the low-level background noise. Otherwise, the characteristic features of the waveform appear to be faithfully preserved.

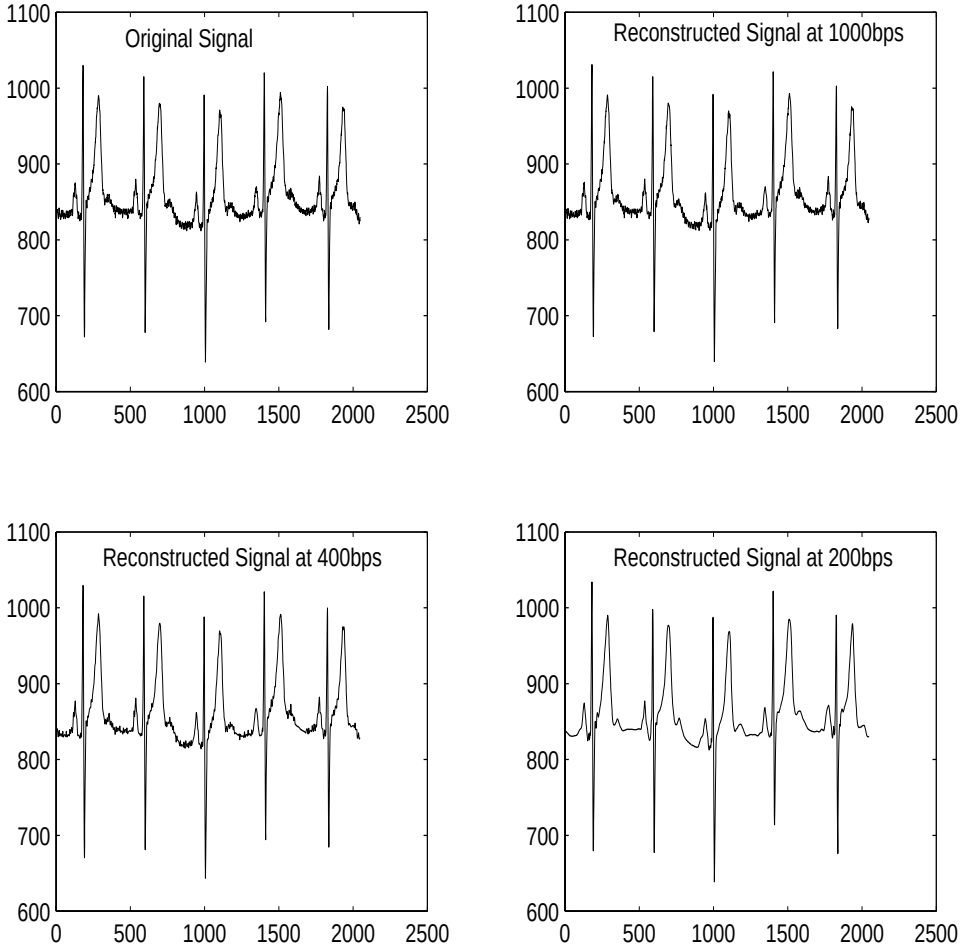

Figure 5: The Original and Reconstructed ECG Signal of MIT-BIH Record 117A

There are other wavelet based ECG codecs in the literature. Hilton presented a wavelet and wavelet packet based EZW encoder [4]. He reported the PRD value of 2.6% with compression ratio 8:1 for record 117 and compared it with the best previous effort for the same data and compression ratio of 3.9% reported in [5]. The PRD value of the coder proposed here is 1.18% for the same record and compression ratio, which is considerably better than the coders in [4] and [5]. The summary of this comparison appears in Table 4. The average PRD values of our coder shown in Table 3 are smaller than those in [4] by factors ranging from about 0.5 to 0.6, depending on the compression ratio.

In comparison to other wavelet coders, the PRD ranges of 9.89% to 13.34% for compression ratios of 13:1 to 22:1 in [3] and 3.70% to 6.19% for compression ratios of 6.19 to 7.98 in [7] are significantly higher (inferior) compared to our results. However, the data and sampling rates are different, so these comparisons are inconclusive. Bradie [11], however, used

Table 4: PRD Comparison of Different Coding Algorithms.

Algorithm	PRD(%)	CR	Signal	Sampling R(Hz)	bits/sample
SPIHT	1.18	8:1	MIT-BIH 117	360	11
Hilton	2.6	8:1	MIT-BIH 117	360	11
Diohn	3.9	8:1	MIT-BIH 117	360	11

the same 10-minute long records in the MIT-BIH database and proposed a coding scheme combining wavelet packet expansion and the methodology of the Karhunen-Loeve transform [11]. He reported a single high compression ratio for each record (with average compression ratio 21.4:1) and reported his result in Root-Mean-square error(RMS). For the compression ratio lower than 20:1, the RMS of our algorithm is on the average about equal to his. For the compression ratio higher than 20:1, his result is slightly better, but the clinical utility of the quality of the reconstruction is open to question. Compared to his algorithm, the computation complexity of our algorithm is much less. Also his algorithm appears to be incapable of obtaining a given rate, whereas ours can always achieve precisely any specified bit rate.

5.2 Comparison with Direct ECG Signal Codec

We also compare the performance of our codec with direct ECG signal codecs in the literature. It suffices to compare our codec with Zigel et al. [10], as they reported that the performance of their Analysis by Synthesis ECG Compressor (ASEC) to be superior to AZTEC, SAPA2 and LTP. The ASEC coder is fairly complex as it performs beats segmentation, dependent non-uniform filtering, feature extraction, and minimization in a loop to get a best estimate of the signal according to a model. Then the error residual is vector quantized from a trained codebook.

In order to compare to ASEC, we ran tests of our coder on a second dataset, the same as that used in [10]: 1 minute length of data in record numbers 104, 107, 111, 112, 115, 116, 117, 118, 119, 201, 207, 208, 209, 212, 213, 214, 228, 231 and 232 in the MIT-BIH database. The average PRD results at different compression ratios are listed in Table 5 and shown in Figure 6. (The PRD result of ASEC is from the Figure 6c in [10].) Compared with the Figure 6c in [10], our PRD results are better than those of ASEC algorithm. For record 119, they reported PRD result 5.5 % at bitrate 183 bps, compared to our PRD of 5.0 % at the same bitrate. However, the ASEC algorithm attempts to minimize a different metric called WDD (Weighted Diagnostic Distortion), which is claimed to be better matched to diagnostic distortion than PRD, but requires complex parameter extraction to calculate, as is done within the ASEC procedure.

Besides the good performance in quality vs compression ratio, the codec we proposed here has some other features which are very important in real time environment. First, SPIHT algorithm achieves exact bit usage control and generates an embedded bitstream, meaning that the encoding and decoding process can stop at any pre-specified bitrate or quality requirement. From one available bitstream, the decoder can get different quality

Table 5: Average Test Results for the Second Dataset

CR	4:1	5:1	6.6:1	8:1	10:1	12:1	16:1	20:1
PRD	1.11	1.47	2.04	2.50	3.11	3.82	5.46	7.52

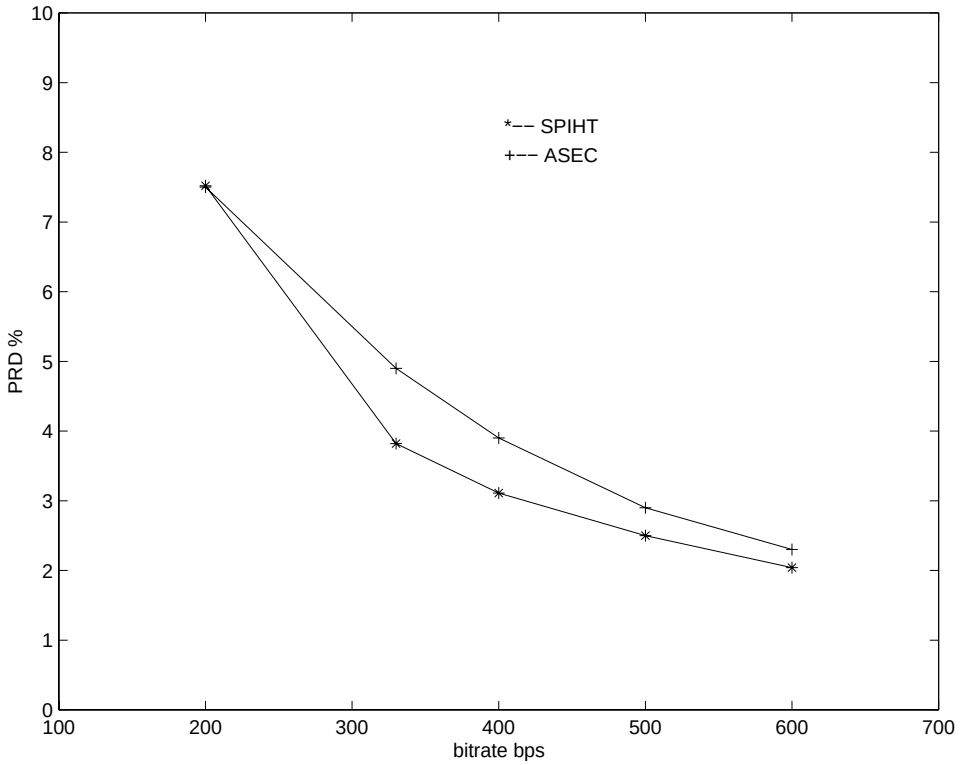

Figure 6: The average PRD results of SPIHT and ASEC

reconstructed signals by decoding subsets of the bitstream. Secondly, the computational complexity of the proposed codec is very low, as witnessed in Table 3 by the average execution times to encode and decode the records of 10 minutes length in the first dataset. (The computer is an SGI Indy workstation with an 133MHz IP22 Processor and 64Mbytes memory). For the lowest compression ratio, where the times are largest, the encoding and decoding times are 8.05 seconds and 4.79 seconds, respectively, far smaller than the data duration of 600 seconds.

6 Conclusions

We proposed a ECG data compression codec based on 1-D SPIHT coding algorithm. We test its performance by coding several records in MIT-BIH ECG arrhythmia database. These records consists of different rhythms, QRS complex morphologies and ectopic beats. The results showed that our coding algorithm has following features:

1. Our algorithm compresses all kinds of ECG data very efficiently, perhaps more efficiently than any previous ECG compression method.

2. Embedded bit stream: The user can truncate the bit stream at any point and obtain the best quality reconstruction for the truncated file size.

3. Exact bit usage control. The coding and decoding process can be stopped at any specified bit rate.

4. The coding and decoding are fast and easy to implement.

The high efficiency, high speed, and simplicity (low complexity) make the algorithm an attractive candidate for use in portable and mobile heart monitoring systems.

References

[1] A. Said and W. A. Pearlman, "A New, Fast and Efficient Image Codec Based on Set Partitioning in Hierarchical Trees,", *IEEE Trans. on Circuits and Systems for Video Technology*, Vol. 6, pp. 243–250, June 1996.

[2] S. Jalaleddine, C. Hutchens, R. Strattan and W. Coberly, "CG data compression techniques - A unified approach", *IEEE Trans. on Biomedical Engineering*, Vol. 37, pp. 329–343, 1990.

[3] A. G. Ramakrishnan, Supratim Saha, "ECG Coding by Wavelet-Based Linear Prediction", *IEEE Trans. on Biomedical Engineering*, Vol. 44, pp. 1253–1261, Dec. 1997.

[4] Michael L. Hilton, "Wavelet and Wavelet Packet Compression of Electrocardiograms", *IEEE Trans. on Biomedical Engineering*, Vol. 44, pp. 394–402, May 1997.

[5] A. Djohan, T. Q. Nguyen, and W. J. Tompkins, "ECG compression using discrete symmetric wavelet transform", *17th Int. Conf. IEEE in Medicine and Biology*, 1995.

[6] S. Mallat, *A Wavelet Tour of Signal Processing*, Academic Press. 1997.

[7] A. Enis Cetin, Hayrettin Koymen and M. Cengiz Aydn, "Multichannel ECG Data Compression by Multirate Signal Processing and Transform Domain Coding Techniques", *IEEE Trans. on Biomedical Engineering*, Vol. 40, pp. 495–499, May 1993

[8] Zhitao Lu and W. A. Pearlman, "An Efficient, Low-Complexity Audio Coder Delivering Multiple Levels of Quality for Interactive Applications", *Proceedings of 1998 IEEE Second Workshop on Multimedia Signal Processing*, Dec. 7-9,1998 Redondo Beach, CA, pp. 529–534.

[9] M. Antonini, M. Barlaud, P. Mathieu, and I. Daubechies, "Image Coding Using Wavelet Transform", *IEEE Trans. Image Processing*, Vol. 1, pp. 205–220, Apr. 1992.

[10] Y. Zigel, A. Cohen, A. Abu-ful, A. Wagshal, A. Katz, "Analysis by Synthesis ECG Signal Compression", *Computers in Cardiology*, Vol. 24, 1997, pp. 279–282.

[11] Brian Bradie, "Wavelet Packet-Based Compression of Single Lead ECG", *IEEE Trans. on Biomedical Engineering*, Vol. 43, May 1996, pp. 493–501.