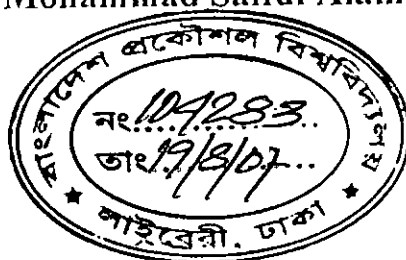


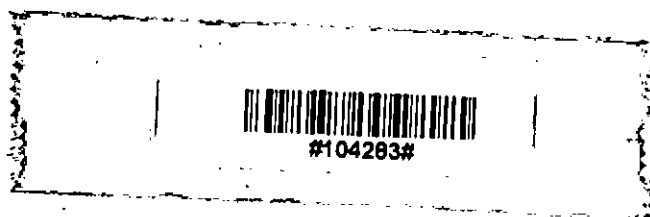
Compression of ECG Signal Based on Its Deviation from a Standard Reference Using Discrete Cosine Transform

by
Mohammad Saiful Alam



A Thesis
Submitted to the department of
Electrical and Electronic Engineering, BUET,
in partial fulfilment of the requirements for the degree of

MASTER OF SCIENCE IN
ELECTRICAL AND ELECTRONIC ENGINEERING


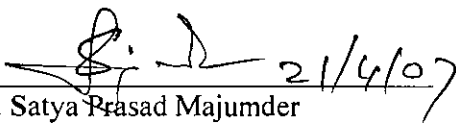
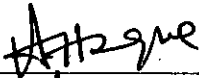
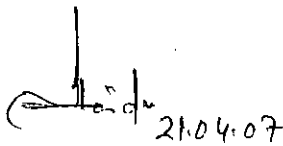


DEPARTMENT OF ELECTRICAL AND ELECTRONIC ENGINEERING,
BANGLADESH UNIVERSITY OF ENGINEERING AND TECHNOLOGY

APRIL 2007

The thesis titled **Compression of ECG Signal Based on Its Deviation from a Standard Reference Using Discrete Cosine Transform** Submitted by **Mohammad Saiful Alam** Roll No: **040406209F**, Session **April 2004** has been accepted as satisfactory in partial fulfillment of the requirement for the degree of **Master of Science in Electrical and Electronic Engineering** on **April 21, 2007**.

BOARD OF EXAMINERS

1. 
Dr. Newaz Muhammad Syfur Rahim
Associate Professor
Dept. of EEE, BUET, Dhaka
Chairman
(Supervisor)
2. 
Dr. Satya Prasad Majumder
Professor & Head
Dept. of EEE, BUET, Dhaka
Member
(Ex- officio)
3. 
Dr. Md. Aynal Haque
Professor
Dept. of EEE, BUET, Dhaka
Member
4. 
Wg Cdr Dr Md Hossam-e-Haider
Instructor, Class-A
Dept. of EECE, MIST,
Mirpur cantonment, Dhaka
Member
(External)

CANDIDATE'S DECLARATION

It is hereby declared that this thesis or any part of it has not been submitted elsewhere for the award of any degree or diploma.

Mohammad Saiful Alam

Mohammad Saiful Alam

ACKNOWLEDGEMENTS

The author would like to express his gratitude to his supervisor Dr. Newaz Muhammad Syfur Rahim for his kind supervision, patience and valuable support in making a difficult task to a pleasant one.

The author is grateful to Dr. Satya Prasad Majumder, head of the Department of EEE, for his kind assistance in this research.

The author wishes to express his thanks and regards to Dr. Md. Aynal Haque for providing the CD of ECG signals and Dr. Md. Kamrul Hasan for giving opportunity to work in the DSP laboratory.

Sincerest thanks to my family and friends for their encouragement, advice and support to complete the thesis.

CONTENTS

Acknowledgements	iv
Abstract	vii
List of Tables	viii
List of Figures	ix
1 Introduction	1
1.1 Background	1
1.2 The Objective of This Research	5
1.3 Organization of the Thesis	6
2 The Heart and the ECG Signal	7
2.1 The Heart	7
2.1.1 Anatomy and Physiology of the Heart	7
2.1.2 Electrical Conduction System of the Heart	9
2.2 The Electrocardiogram	11
2.2.1 Electrical Basis of the Electrocardiogram	11
2.2.2 Components of the Electrocardiogram	12
2.2.3 ECG paper	13
2.2.4 ECG Leads	14
3 Review of Some ECG Compression Methods	18
3.1 Introduction	18
3.2 Distortion Measures	18
3.3 Compression Measures	20
3.4 Direct Methods	21
3.4.1 Tolerance-Comparison Data Compression Techniques	22
3.4.1.1 The Amplitude Zone Time Epoch Coding Technique	22
3.4.1.2 The Turning Point Technique	23
3.4.1.3 The Coordinate Reduction Time Encoding System	23
3.4.1.4 Fan and SAPA Techniques	24
3.4.1.5 Slope Adaptive Interpolation Encoding Scheme	24
3.4.1.6 The SLOPE Algorithm	24
3.4.1.7 The CORNER Algorithm	25
3.4.2 Data Compression by Differential Pulse Code Modulation	25
3.4.3 Entropy Coding and its Application in ECG Compression	27
3.4.3.1 Entropy Coding	27
3.4.3.1.1 Huffman Coding	28
3.4.3.1.2 Arithmetic Coding	29
3.4.3.2 Entropy Coding of ECG's	34
3.5 Parametric Methods	34
3.5.1 Beat Codebook	34

3.5.2	Analysis by Synthesis ECG Compressor (ASEC)	35
3.6	Transformation Methods	37
3.6.1	Wavelet Transform Based Compression	37
3.6.2	DCT and DCT-Based Compression	39
3.6.2.1	The One-Dimensional DCT	39
3.6.2.2	DCT Based Compression	41
4	The Proposed Method	44
4.1	Introduction	44
4.2	Compression Algorithm	44
4.2.1	The First Pre-processing Stage	46
4.2.2	The Second Pre-processing Stage	49
4.2.3	Encoding Stage	55
4.3	Bit Allocation and Compression Measure	57
4.4	Decompression Algorithm	58
5	Results	60
5.1	Database Used for Compression	60
5.2	Results of Simulation	61
5.3	Comparison with Other Methods	76
6	Conclusion	78
6.1	Findings	78
6.2	Future Perspectives	80
Appendix - A QRS Detection Algorithm		81
Appendix - B MATLAB Programs		88
References		109

Abstract

Modern clinical systems require the storage and transmission of large amount of ECG signals. Efficient data compression is needed in order to reduce the amount of data. In ECG signal compression algorithms, the aim is to reach maximum compression ratio, while keeping the relevant diagnostic information in the reconstructed signal.

In this work, an ECG compression algorithm is presented which is based on optimum quantization of Discrete Cosine Transform coefficients. DCT is applied on the residual signals that are obtained after subtracting the standard reference signal from period normalized and dc removed beats. For optimum quantization of the DCT coefficients, the optimum quantization and threshold vectors are generated using Lagrange multipliers so that the entropy of the quantized coefficients are minimized at a target distortion. The quantized coefficients are lossless encoded by arithmetic encoding after following some steps. The compressed signal consists of arithmetic encoded bits along with some additional bits that are necessary for the decompression process. For measuring the distortion introduced by lossy compression, Percent Root mean square Distortion (PRD and PRD2) have been used. The amount of compression is measured by Compression Ratio (CR) and Compressed Data Rate (CDR).

The proposed compressor is designed for low values of PRD / PRD2. For low PRD / PRD2 high compression ratio and low compressed data rate have been achieved. The compression algorithm have been applied to channel MLII of 40 records of MIT-BIH Arrhythmia Database (360 Hz sampling frequency and 11 bits/sample quantizer resolution), which consists of large and varied ECG signals. A compression rate of approximately 291 bits/second and compression ratio of 13.7:1 has been achieved with a very good reconstructed signal quality (PRD=3.0%). For 6.5% PRD2, 20.1:1 compression ratio and 196 bits/sec compressed data rate has been found. The proposed compression algorithm has been found to have the best performances at any PRD / PRD2. These performances are better than other compression algorithms found in the literature.

List of Tables

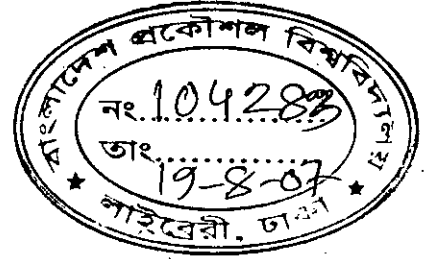
<u>Table</u>	<u>Title</u>	<u>page</u>
3.1	Frequencies and Probabilities of Five Symbols	31
3.2	The Process of Arithmetic Encoding	33
3.3	The Process of Arithmetic Decoding	33
5.1	CR and CDR of the test signals at 1.5%, 2.0%, 2.5% and 3.0% PRD	61
5.2	CR and CDR of the test signals at 2.0%, 3.5%, 5.0% and 6.5% PRD2	64
5.3	Comparison of the proposed method with other methods for fixed compression ratio	76
5.4	Comparison of the proposed method with other methods for fixed PRD	77

List of Figures

<u>Figure</u>	<u>Title</u>	<u>Page</u>
Figure 2.1	- Anatomy of the Heart	8
Figure 2.2	- Electrical conduction system	10
Figure 2.3	- Electrical basis of the ECG	12
Figure 2.4	- Components of the ECG	13
Figure 2.5	- ECG paper	14
Figure 2.6	- The standard (bipolar) limb leads I, II, and III	16
Figure 2.7	- The augmented (unipolar) leads aVR, aVL, and aVF	16
Figure 2.8	- Precordial (unipolar) leads	17
Figure 3.1	- Compression by two-step DPCM	26
Figure 3.2	- Compression by Average Beat Subtraction	27
Figure 3.3	- Huffman Codes	28
Figure 3.4	- ECG Compression based on Long Term Prediction	35
Figure 3.5	- General scheme of the ASEC	36
Figure 3.6	- One dimensional cosine basis function	40
Figure 4.1	- The first pre-processing stage	45
Figure 4.2	- The second pre-processing stage	45
Figure 4.3	- The encoding stage	45
Figure 4.4	- The partitioned unnormalized beats of ECG signal	46
Figure 4.5	- The period normalized beats of ECG signal having zero dc component.	47
Figure 4.6	- The standard reference Signal	47
Figure 4.7	- The vector p (beat durations of the ECG signal)	48
Figure 4.8	- DC coefficients of the beats.	48
Figure 4.9	- The vector m	48
Figure 4.10	- The residual signals	49

Figure 4.11	- Points of D-H found after the test of thresholding and quantization process of the 2nd DCT coefficient.	51
Figure 4.12	- The procedure of getting the value of λ from the points of D-H	52
Figure 4.13	- Quantization and threshold vectors for PRD=1.5%	53
Figure 4.14	- Quantization and threshold vectors for PRD=2.0%	53
Figure 4.15	- Quantization and threshold vectors for PRD=2.5%	54
Figure 4.16	- Quantization and threshold vectors for PRD=3.0%	54
Figure 4.17	- The processing on the quantized coefficients	55
Figure 4.18	- The vector r	56
Figure 4.19	- The vector r'	56
Figure 5.1	- (a) Compression ratios and (b) Compressed Data Rates found at PRD = 3.0%	63
Figure 5.2	- (a) Compression ratios and (b) Compressed Data Rates found at PRD2 = 6.5%	66
Figure 5.3	- (a) Average Compression Ratio and (b) Average Compressed Data Rate at PRD 1.5%, 2.0%, 2.5% and 3.0%, (c) Average Compression Ratio and (d) Average Compressed Data Rate at PRD2 2.0%, 3.5%, 5.0% and 6.5%.	67
Figure 5.4	- Detail of (a) record 100/MLII and (b) reconstructed signal for PRD = 2.5%, CR = 13.1	68
Figure 5.5	- (a) 9 sec part of record 100/MLII, and reconstructed signal for (b) PRD = 1.5%, (c) PRD = 2.0%, (d) PRD = 2.5%, (e) PRD = 3.0%	69
Figure 5.6	- (a) 7 second section of record 112 / MLII, (b) reconstructed signal for PRD2 = 6.5%, (c) residual signal.	70
Figure 5.7	- (a) 7 second section of record 121 / MLII, (b) reconstructed signal for PRD2 = 6.5%, (c) residual signal.	71
Figure 5.8	- (a) 7 second section of record 124 / MLII, (b) reconstructed signal for PRD2 = 6.5%, (c) residual signal.	72
Figure 5.9	- (a) 7 second section of record 205 / MLII, (b) reconstructed signal for PRD2 = 6.5%, (c) residual signal.	73

Figure 5.10	- (a) 7 second section of record 222 / MLII, (b) reconstructed signal for PRD2 = 6.5%, (c) residual signal.	74
Figure 5.11	- (a) 7 second section of record 232 / MLII, (b) reconstructed signal for PRD2 = 6.5%, (c) residual signal.	75
Figure A.1	- Block diagram of QRS detection algorithm	81
Figure A.2	- Impulse response of the bandpass filter	82
Figure A.3	- (a)The magnitude and (b)phase response of the bandpass filter	82
Figure A.4	- The sequences of peak detection of the ECG signal taken from the record 106 (MLII) of MIT-BIH arrhythmia database.	84
Figure A.5	- Detected peaks of the original ECG signal taken from record 107/MLII	85
Figure A.6	- Detected peaks of the original ECG signal taken from record 115/MLII	85
Figure A.7	- Detected peaks of the original ECG signal taken from record 117/MLII	85
Figure A.8	- Detected peaks of the original ECG signal taken from record 121/MLII	86
Figure A.9	- Detected peaks of the original ECG signal taken from record 124/MLII	86
Figure A.10	- Detected peaks of the original ECG signal taken from record 215/MLII	86
Figure A.11	- Detected peaks of the original ECG signal taken from record 220/MLII	87
Figure A.12	- Detected peaks of the original ECG signal taken from record 221/MLII	87
Figure A.13	- Detected peaks of the original ECG signal taken from record 233/MLII	87



Chapter 1

Introduction

1.1 Background

As a diagnosis tool of cardiac diseases, the 12-channel (or leads) electrocardiogram (ECG) is a very important physiological signal. An ECG signal can be captured by putting several electrodes at particular positions on the body. It is used to describe the voltage variations of cardiac rhythm. With different combinations of electrode pairs, different ECG channels can be obtained. The 12-lead ECG, including 3 standard leads and 6 chest leads, are used by a doctor to check the heart problem of a patient. To facilitate the processing of the signal, it is often digitized. Possible processing tasks for digitized ECG signals are compression, transmission, encryption, error correction, etc. In this dissertation, it is aimed at the technique of ECG compression.

A typical electrocardiogram monitoring device generates massive volume of digital data. Depending on the intended application for the data, the sampling rate ranges from 125 to 500 Hz. Each data sample may be digitized to a 8 bit to 12 bit binary number. Even at the lowest sampling rate in the range and assuming just one sensor that generates 8 bit data, we would accumulate ECG data at a rate of 7.5 KB per minute or 450 KB per hour. At the other extreme (12 sensors generating 12 bit values at 500 Hz), data is generated at a rate of 540 KB per minute or more than 30 MB per hour.

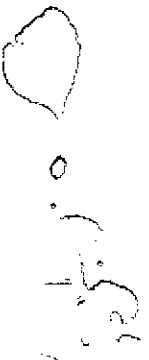
The online storage and transmission of digital ECG signals are useful in many applications, including the Holter recording and telemedicine. The Holter recorder is a mobile ECG online measurement set and a recording medical device widely used in hospitals and clinics. It is usually carried by a patient over 24 hours. Therefore, it must record a long duration of ECG data. The huge amount of data becomes a problem when the storage space is very limited. We can either increase the storage memory, which is costly, or have an ECG compression technique with high compression ratio to solve the

problem. In addition, for the application of telemedicine, ECG compression is also necessary.

Telemedicine is a very important topic in the area of biomedical engineering. It uses modern telecommunication techniques and allows the doctors to monitor a patient's condition or consult with other doctors in a long distance based on wire or wireless systems. In this case, in general, not only the ECG signal but also other biomedical signals have to be transmitted simultaneously, these include electromyogram (EMG), electroencephalogram (EEG), ultrasound, X-ray, computed tomography(CT), magnetic resonance image (MRI), etc. Thus a biomedical signal compression technique, including ECG compression, is needed for reducing the amount of data to meet the constraint of very limited bandwidth.

So an efficient ECG compressor is needed in order to reduce the huge volume data; in practice, this may be done only with lossy compression techniques (which allow reconstruction error). With lossless compressors, high compression ratios can not be achieved. In ECG signal compression algorithms the goal is to achieve a minimum information rate, while retaining the relevant diagnostic information as much as possible in the reconstructed signal.

In the past, many schemes have been presented for compression of ECG data. ECG compression methods have been mainly classified into three major categories[1],[2]: direct data compression (DDC), transformation compression (TC), and parameter extraction compression (PEC). DDC methods are based on their detection of redundancies on direct analysis of the actual signal samples to provide the compression. The DDC schemes are AZTEC, TP, CORTES, Fan, SAPA, SAIES, SLOPE algorithm, CORNER algorithm, DPCM, ADPCM, Entropy Coding such as Huffman coding and Arithmetic coding. In the TC methods, the original samples are subjected to a (linear) transformation and the compression is performed in the new domain. Fourier Transform, Walsh Transform, Cosine Transform, KLT and Wavelet Transform are examples of transformation compression. While, PEC is an irreversible process with which a particular characteristic or parameter of the signal is extracted. The extracted



parameter (e.g. measurement of the probability distribution) is subsequently utilized for the classification based on a priori knowledge of the signal features. ANN, Long-Term Prediction, Vector Quantization etc are included in PEC. The Existing techniques can reduce the bandwidth significantly. A brief summary of the previous significant works in this field has been given here:

The AZTEC algorithm was developed by Cox et al. [8] was a popular ECG compression algorithm. 10:1 compression ratio has been achieved (for 500 Hz sampled ECG with 12 bit resolution) but the distortion is significantly high. The P and T waves are affected very much. The reconstructed signal has poor fidelity mainly because of the discontinuity of the waves (step like quantization). A modified AZTEC algorithm was proposed in [9] where the quality of the reconstruction was improved.

The Turning Point [11] algorithm reduces one data sample from the consecutive two samples. The CR is fixed at 2:1.

CORTES algorithm [12] is a hybrid of the AZTEC and TP algorithms. It applies the AZTEC algorithm to the lower frequency regions and TP to the high frequency regions (QRS complexes). The typical performances are compression ratio of 5:1 and PRD of 7%. The sampling frequency is 200 Hz, and the quantization is 12 bit.

Fan and SAPA [14] algorithms are both based on first order interpolation [1] and have the same performance. In this method, the reconstructed signal looks like a broken line. The fidelity of the reconstructed signal becomes poorer as the CR is increased. Typical performances are a compression ratio of 3:1 and a PRD of 4%. The sampling frequency is 250 Hz, and the quantization is 12 bit.

The SAIES algorithm [15] combines the AZTEC and Fan compression techniques and the reported performances are a CR of 5.9:1 for PRD of 16.3% where the sampling frequency is 166 Hz and quantization resolution is 10 bit.

The SLOPE algorithm [16] delimits linear segments of different lengths and different slopes of the ECG signal.

The CORNER algorithm [17] was applied on MIT-BIH database (sampling frequency 360 Hz and quantizer resolution 11 bit) and average bit rate of 0.79 bits per sample was achieved for SNR of 27 dB.

Several ECG compression algorithms based on DPCM have been presented in the literature [18, 19, 20]. Some of them use the DPCM as minor part of the overall compression scheme. In the method of [19], compression is done by two step DPCM and the quasiperiodic characteristic of the ECG signal has been exploited to reduce the variance of the prediction error. In another important work [20] the compression was performed by average beat subtraction and the average bit rate was 174 bps.

Entropy coding is generally used for lossless data compression. Two popular entropy coding algorithms are Huffman coding and arithmetic coding. They are being implemented for lossless compression at the last step of lossy compression of ECG signals.

Many efficient ECG compression methods based on beat codebook have been presented in the literature. In the method of adaptive compression of ECG [22], the residual signal (which has been found after subtracting the selected beat of the code book from the current beat) is quantized adaptively. The achieved bit rate was 193.3 bps with PRD between 4.33% and 19.3%.

In the Long-Term Prediction (LTP) model [2], the prediction of the n^{th} sample is made using samples of past beats and the LTP residual signal is quantized. The performance of the algorithm are: bit rate 71 and 650 bps with PRD2 between 10% and 1%.

The ASEC algorithm [6] is based on analysis by synthesis coding. The incoming beat is segmented into P, QRS and T sections which are then coded separately. The beat is matched with the code book, LTP coding is done using the chosen codeword to

produce the predicted signal, and the residual signal undergoes STP coding and adaptive quantization to produce the coded signal. Bit rate of approximately 100 bps has been achieved with a good reconstructed signal quality (PRD2 below 8%).

Many orthogonal transform compression algorithms for ECG signals have been presented in the last thirty years, such as the Fourier Transform, Walsh Transform [25], Cosine Transform [26], and Karhunen-Loeve Transform (KLT) [27]. The typical performances of the transform methods are compression ratio between 3:1 to 12:1. In the recent years, many ECG compression algorithms have been proposed based on the Wavelet Transform (WT) [28], [29]. The reported performances are compression ratio from 13.5:1 to 22.9:1 with the corresponding PRD between 5.5% and 13.3%.

In the DCT based ECG compression methods, the signal is portioned into blocks, each block is DCT transformed and the coefficients are thresholded and quantized either by unique threshold and quantization value or by threshold and quantization vector. The quantized coefficients are sometimes lossless encoded by entropy coding. In the work of [34], a constant threshold value is applied for all coefficients where the quantization value is fixed at 1. Varying threshold, CR and distortion is controlled. The CAB/2-D DCT [31] uses a unique quantization step size for all coefficients. In [30], the values of the quantization vector grows linearly. Varying the inclination of the line segment controls the CR and distortion. In the optimized quantization of DCT coefficients method [38], the block size was taken 64, and optimum quantization and threshold vectors were for minimization of entropy and distortion.

However, the seriousness of the problem can be relieved further significantly if we have a more efficient ECG data compression technique that is capable of reducing the amount of data as much as possible while preserving the necessary signal quality for cardiac diagnosis.

1.2 The Objective of This Research

As transform methods usually achieve higher compression ratios and are insensitive to noise existing in original ECG signal, we preferred a transform based compression

method for ECG signal. Discrete Cosine Transform based compression is being used in various fields including the ECG compression for better decorrelation and energy compaction properties. So we intend to propose an efficient ECG compression algorithm in this dissertation which is based on the deviation of ECG signal from a standard reference using DCT. The signal energy concentrated in a few transform coefficients helped us to improve the compression ratio.

1.3 Organization of the Thesis

Chapter 1 is an introductory chapter. It contains the background and motivation of ECG compression, objective and outline of the proposed algorithm and organization of this thesis.

A discussion about the human heart, electrical conduction system of the heart, and electrocardiogram is presented in the chapter 2.

Chapter 3 reviews some ECG data compression methods that have been reported in the literature. This chapter also presents distortion and compression measures that are integral part of the compression methods.

The proposed compression algorithm is described in the chapter 4. It also includes the decompression algorithm, bit allocation and compression measure of the compressed signal.

Chapter 5 includes the results of the proposed compression algorithm and compares the performance with some other methods.

Chapter 6 contains a summary, conclusions, and recommendations for continuation.

Chapter 2

The Heart and the ECG Signal

2.1 The Heart

2.1.1 Anatomy and Physiology of the Heart

The heart, whose sole purpose is to circulate blood through the circulatory system (the blood vessels of the body), consists of four hollow chambers (Figure 2.1) [39]. The upper two chambers, the right and left atria, are thin-walled; the lower two, the right and left ventricles are thick-walled and muscular. The walls of the ventricles are composed of three layers of tissue: the innermost thin layer is called the endocardium; the middle thick, muscular layer, the myocardium; and the outermost thin layer, the epicardium. The walls of the left ventricle are more muscular and about three times thicker than those of the right ventricle.

The atrial walls are also composed of three layers of tissue like those of the ventricles, but the middle muscular layer is much thinner. The two atria form the base of the heart; the ventricles form the apex of the heart.

The interatrial septum (a thin membranous wall) separates the two atria, and a thicker, more muscular wall, the interventricular septum, separates the two ventricles. The two septa, in effect, divide the heart into two pumping systems, the right heart and the left heart, each one consisting of an atrium and a ventricle.

The right heart pumps blood into the pulmonary circulation (the blood vessels within the lungs and those carrying blood to and from the lungs). The left heart pumps blood into the systemic circulation (the blood vessels in the rest of the body and those carrying blood to and from the body).

The right atrium receives unoxygenated blood from the body via two of the body's largest veins (the superior vena cava and inferior vena cava) and from the heart itself by way of the coronary sinus. The blood is delivered to the right ventricle through the tricuspid valve. The right ventricle then pumps the unoxygenated blood through the pulmonic valve and into the lungs via the pulmonary artery. In the lungs, the blood picks up oxygen and releases excess carbon dioxide.

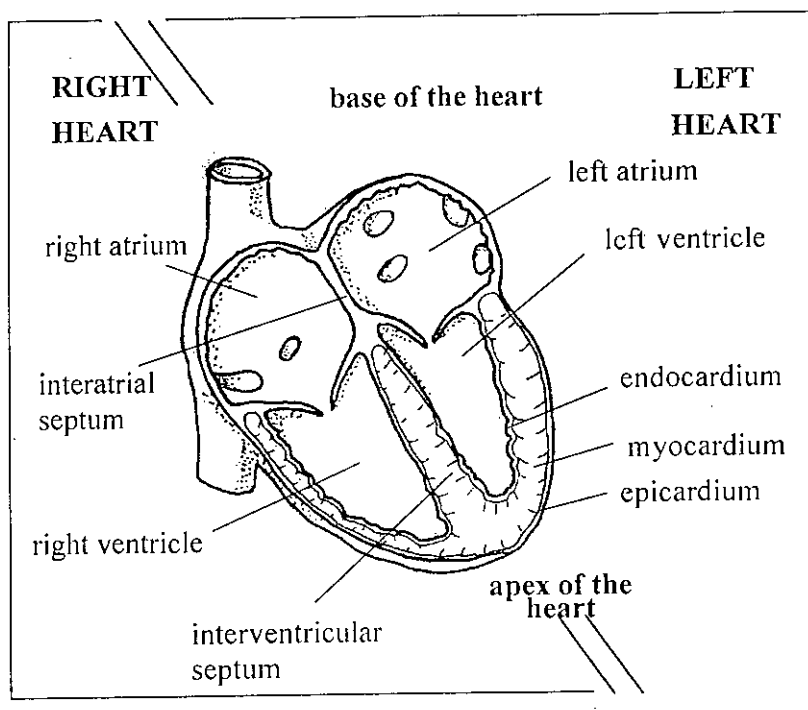


Figure 2.1. Anatomy of the heart

The left atrium receives the newly oxygenated blood from the lungs via the pulmonary veins and delivers it to the left ventricle through the mitral valve. The left ventricle then pumps the oxygenated blood out through the aortic valve and into the aorta, the largest artery in the body. From the aorta, the blood is distributed throughout the body where the blood releases oxygen to the cells and collects carbon dioxide from them.

The heart performs its pumping action over and over in a rhythmic sequence. First, the atria relax (atrial diastole), allowing the blood to pour in from the body and lungs. As the atria fill with blood, the atrial pressure rises above that in the ventricles, forcing the

tricuspid and mitral valves to open and allowing the blood to empty rapidly into the relaxed ventricles. Then the atria contract (atrial systole), filling the ventricles to capacity.

Following the contraction of the atria, the pressures in the atria and ventricles equalize, and the tricuspid and mitral valves begin to close. Then, the ventricles contract vigorously, causing the ventricular pressure to rise sharply. The tricuspid and mitral valves close completely, and the aortic and pulmonic valves snap open, allowing the blood to be ejected forcefully into the pulmonary and systemic circulations.

Meanwhile, the atria are again relaxing and filling with blood. As soon as the ventricles empty of blood and begin to relax, the ventricular pressure falls, the aortic and pulmonic valves shut tightly, the tricuspid and mitral valves open, and the rhythmic cardiac sequence begins anew.

The period from the opening of the aortic and pulmonic valves to their closing, during which the ventricles contract and empty of blood, is called ventricular systole. The following period from the closure of the aortic and pulmonic valves to their reopening, during which the ventricles relax and fill with blood, is called ventricular diastole. The sequence of one ventricular systole followed by a ventricular diastole is called the cardiac cycle, commonly defined as the period from the beginning of one heart beat to the beginning of the next.

2.1.2 Electrical Conduction System of the Heart

The electrical conduction system of the heart (figure 2.2) is composed of the following structures: • Sinoatrial (SA) node. • Internodal atrial conduction tracts and the interatrial conduction tract (Bachmann's bundle). • Atrioventricular (AV) junction consisting of the atrioventricular (AV) node and bundle of His. • Right bundle branch, left bundle branch, and left anterior and posterior fascicles. • Purkinje network.

The prime function of the electrical conduction system of the heart is to transmit minute electrical impulses from the SA node (where they are normally generated) to the atria and ventricles, causing them to contract (Figure 2.2).

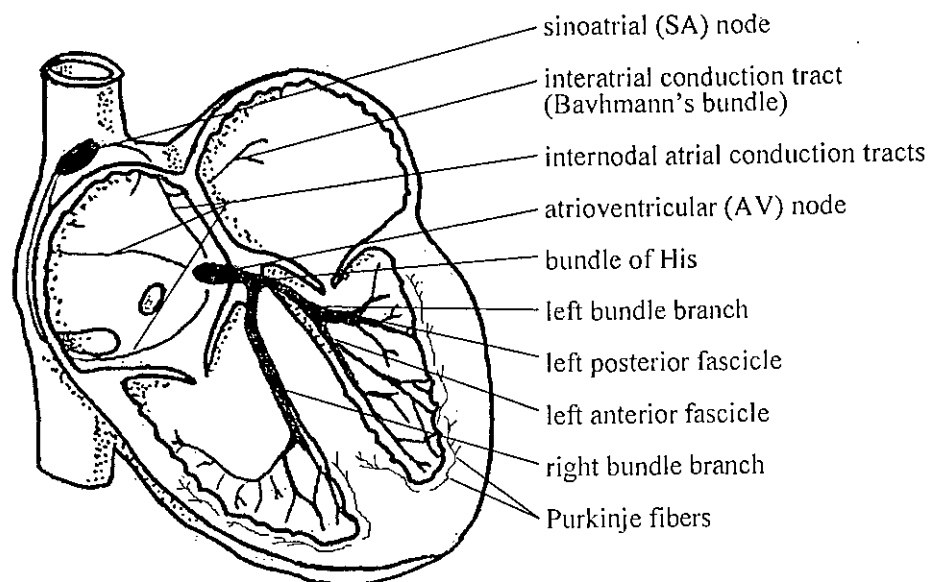


Figure 2.2. Electrical conduction system

The SA node lies in the wall of the right atrium near the inlet of the superior vena cava. It consists of pacemaker cells that generate electrical impulses automatically and regularly.

The three internodal atrial conduction tracts, running through the walls of the right atrium between the SA node and the AV node, conduct the electrical impulses rapidly from the SA node to the AV node in about 0.03 second. The interatrial conduction tract (Bachmann's bundle), a branch of one of the internodal atrial conduction tracts, extends across the atria, conducting the electrical impulses from the SA node to the left atrium.

The AV node lies partly in the right side of the interatrial septum in front of the opening of the coronary sinus and partly in the upper part of the interventricular septum above the base of the tricuspid valve. The primary function of the AV node is to relay the electrical impulses from the atria into the ventricles in an orderly and timely way. A ring of fibrous tissue insulates the remainder of the atria from the ventricles, preventing electrical impulses from entering the ventricles except through the AV node.

The electrical impulses slow as they travel through the AV node, taking about 0.06 to 0.12 second to reach the bundle of His. The delay is such that the atria can contract and empty, and the ventricles fill before they are stimulated to contract.

The bundle of His lies in the upper part of the interventricular septum, connecting the AV node with the two bundle branches. Once the electrical impulses enter the bundle of His, they travel more rapidly on their way to the bundle branches, taking 0.03 to 0.05 second.

The right bundle branch and the left common bundle branch arise from the bundle of His, straddle the interventricular septum, and continue down both sides of the septum. The left common bundle branch further divides into two major divisions: the left anterior fascicle and the left posterior fascicle. The bundle branches and their fascicles subdivide into smaller and smaller branches, the smallest ones connecting with the Purkinje network, an intricate web of tiny Purkinje fibers spread widely throughout the ventricles beneath the endocardium. The ends of the Purkinje fibers finally terminate at the myocardial cells. The bundle of His, the right and left bundle branches, and the Purkinje network are also known as the His-Purkinje system of the ventricles.

The electrical impulses travel very rapidly to the Purkinje network through the bundle branches in less than 0.01 second. All in all, it normally takes the electrical impulses less than 0.2 second to travel from the SA node to the Purkinje network in the ventricles.

2.2 The Electrocardiogram

2.2.1 Electrical Basis of the Electrocardiogram

The electrocardiogram (ECG) is a graphic record of the changes in magnitude and direction of the electrical activity, or, more specifically, the electric current, that is generated by the depolarization and repolarization of the atria and ventricles (Figure 2.3). This electrical activity is readily detected by electrodes attached to the skin. But neither the electrical activity that results from the generation and transmission of

electrical impulses which are too feeble to be detected by skin electrodes nor the mechanical contractions and relaxations of the atria and ventricles (which do not generate electrical activity) appear in the electrocardiogram.

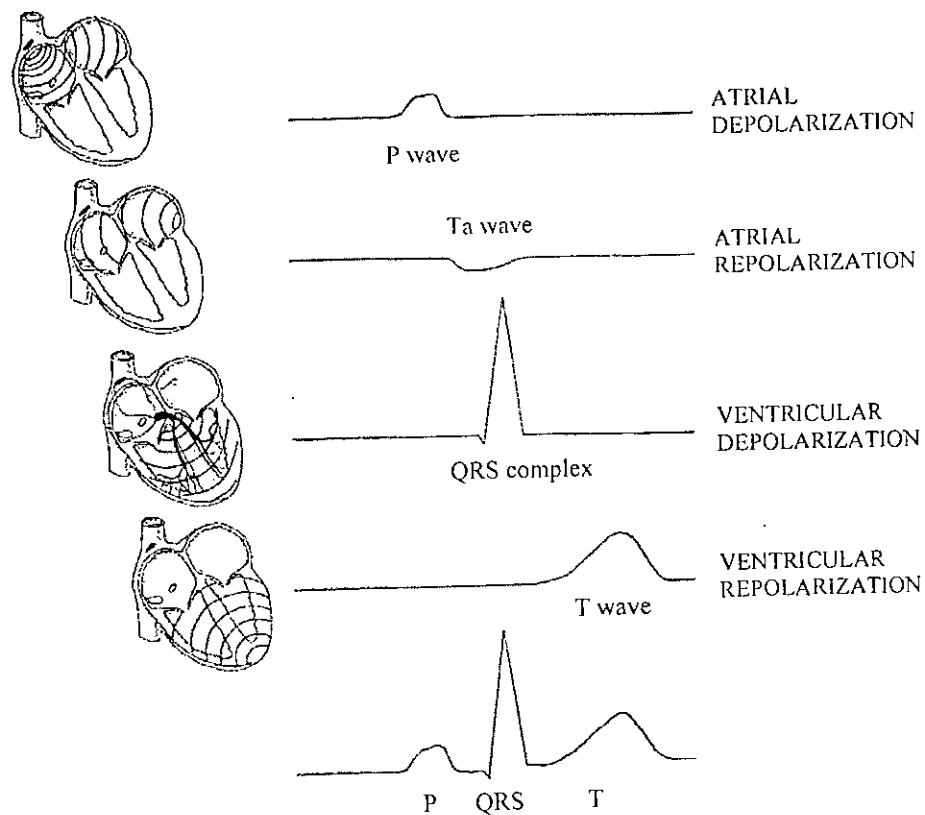


Figure 2.3 - Electrical basis of the ECG

2.2.2 Components of the Electrocardiogram

After the electric current generated by depolarization and repolarization of the atria and ventricles is detected by electrodes, it is amplified, displayed on an oscilloscope, recorded on ECG paper, or stored in memory. The electric current generated by atrial depolarization is recorded as the P wave, and that generated by ventricular depolarization is recorded as the Q, R, and S waves: the QRS complex. Atrial repolarization is recorded as the atrial T wave (Ta), and ventricular repolarization, as the ventricular T wave, or simply, the T wave. Because atrial repolarization normally occurs during ventricular depolarization, the atrial T wave is buried or hidden in the QRS complex.

In a normal cardiac cycle, the P wave occurs first, followed by the QRS complex and the T wave (Figure 2.4).

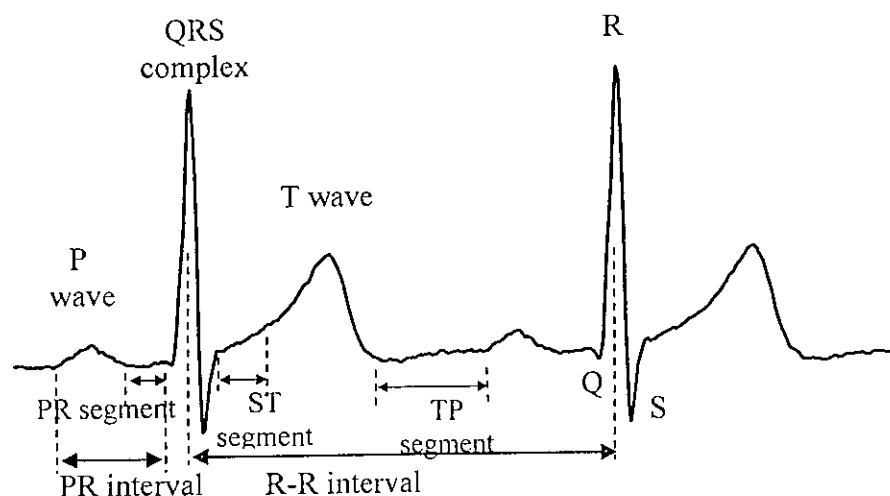


Figure 2.4 - Components of the ECG.

The sections of the ECG between the waves and complexes are called segments and intervals: the PR segment, the ST segment, the TP segment, the PR interval, the QT interval, and the R-R interval. Intervals include waves and complexes, whereas segments do not.

When electrical activity of the heart is not being detected, the ECG is a straight, flat line – the isoelectric line or baseline.

2.2.3 ECG paper

The paper used in recording electrocardiograms has a grid to permit the measurement of time in seconds and amplitude in millimeters along the horizontal lines and voltage (amplitude) in millimeters along the vertical lines (Figure 2.5).

The grid consists of intersecting dark and light vertical and horizontal lines that form large and small squares. When the ECG is recorded at the standard paper speed of 25 mm/sec:

- The dark vertical lines are 0.20 second (5 mm) apart.

- The light vertical lines are 0.04 second (1 mm) apart.
- The dark horizontal lines are 5-mm apart (0.5 mV).
- The light horizontal lines are 1-mm apart (0.1 mV).

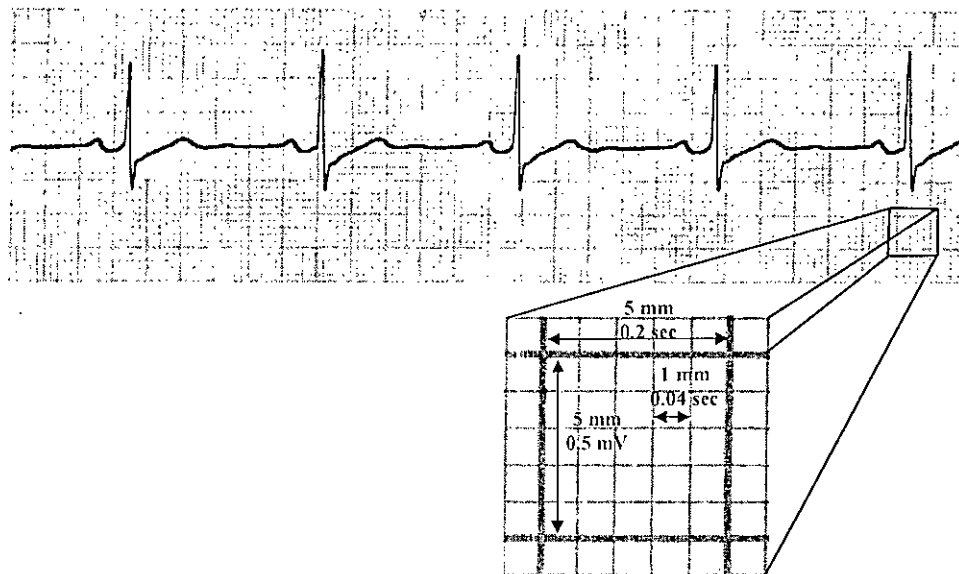


Figure 2.5 - ECG paper

Conventionally, the sensitivity of the ECG machine is adjusted, i.e., calibrated so that 1 mV electrical signal (referred to the electrode) produces a 10-mm deflection on the ECG.

2.2.4 ECG Leads

An ECG lead is a record (spatial sampling) of the electrical activity generated by the heart that is sensed by either one of two ways: (1) two discrete electrodes of opposite polarity or (2) one discrete positive electrode and an “indifferent,” zero reference point. A lead composed of two discrete electrodes of opposite polarity is called a bipolar lead; a lead composed of a single discrete positive electrode and a zero reference point is a unipolar lead.

Depending on the ECG lead being recorded, the positive electrode may be attached to the right or left arm, the left leg, or one of several locations on the anterior chest wall.

The negative electrode is usually attached to an opposite arm or leg or to a reference point made by connecting the limb electrodes together.

For a detailed analysis of the heart's electrical activity, usually in the hospital setting, an ECG recorded from 12 separate leads (the 12-lead ECG) is used. The 12-lead ECG is also used in the prehospital phase of emergency care in certain advanced life support services to diagnose acute myocardial infarction and to help in the identification of certain arrhythmias. A 12-lead ECG consists of three standard (bipolar) limb leads (leads I, II, and III) (Figure 2.6), Three augmented (unipolar) leads (leads aVR, aVL, and aVF) (figure 2.7), and six precordial (unipolar) leads (V₁, V₂, V₃, V₄, V₅, and V₆) (Figure 2.8).

When monitoring the heart solely for arrhythmias, a single ECG lead, such as the standard limb lead II, is commonly used, especially in the prehospital phase of emergency care.

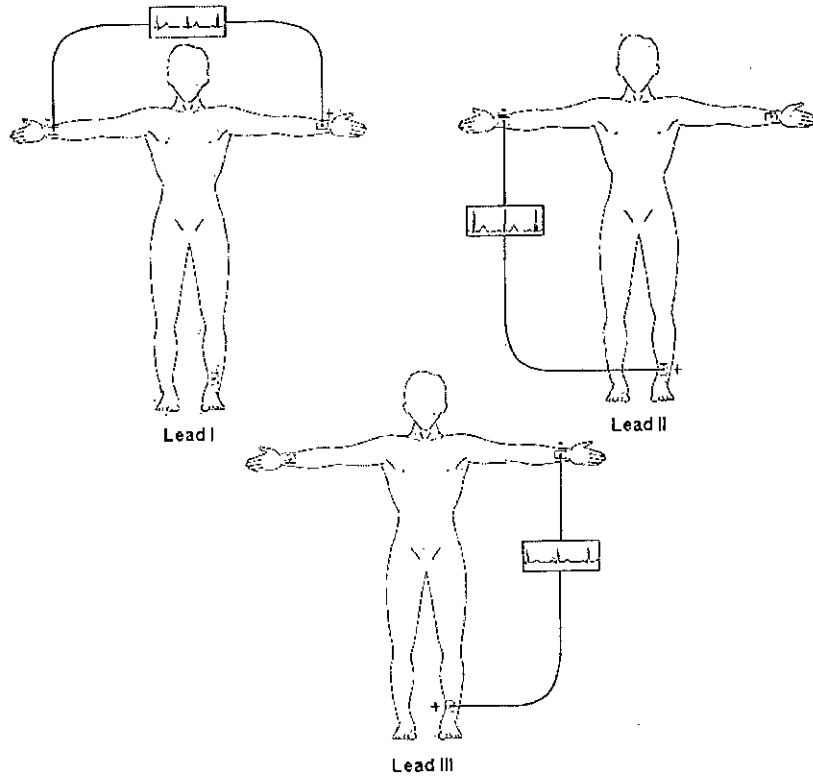


Figure 2.6 - The standard (bipolar) limb leads I, II, and III.

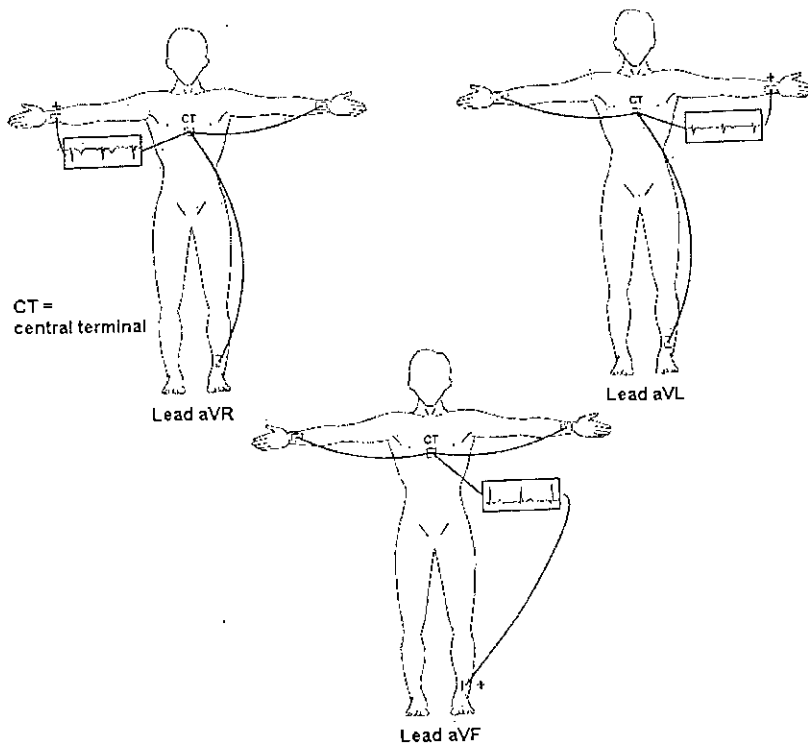


Figure 2.7 - The augmented (unipolar) leads aVR, aVL, and aVF.

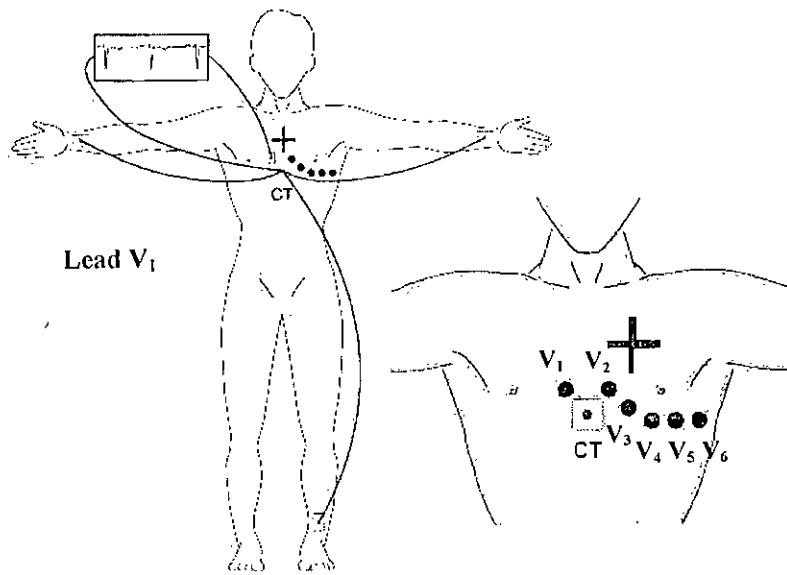


Figure 2.8 - Precordial (unipolar) leads.

Chapter 3

Review of Some ECG Compression Methods

3.1 Introduction

A Large Variety of techniques for ECG compression has been proposed and published over the last thirty years. These techniques have become essential in a large variety of applications, from diagnosis through supervision and monitoring applications.

In general, compression techniques may be divided into two: errorless methods and methods that produce reconstruction errors. In most ECG applications, the errorless methods do not provide sufficient compression, and hence errors are to be expected in practical ECG compression systems. ECG compression methods have been mainly classified into three major categories [1], [2]: direct data compression, transformation methods, and parametric techniques. This classification is not accurate and some compression algorithms may be classified into two or more categories. In the direct methods, the samples of the signal are directly handled to provide the compression. In the transformation methods, the original samples are subjected to a (linear) transformation and the compression is performed in the new domain. In the parametric methods, a preprocessor is employed to extract some features that are later used to reconstruct the signal.

Two important related parts of the compression methods to evaluate the performance are the measurement of quality and compression. So, before reviewing some compression methods, they are discussed below.

3.2 Distortion Measures

One of the most difficult problems in ECG compression applications and reconstruction is defining the error criterion. The purpose of the compression system is to remove redundancy, the irrelevant information (which does not contain diagnostic information

in the ECG case). Consequently the error criterion has to be defined such that it will measure the ability of the reconstructed signal to preserve the relevant information. Such a criterion has been defined in the past as "diagnostability" [3]. A similar problem exists in synthesized speech signals, in which the criterion "intelligibility" has been defined [4]. Today the accepted way to examine diagnostability is to get cardiologists' evaluations of the system's performance. This solution is good for getting evaluations of coders' performances, but it can not be used as a tool for designing ECG coders and certainly, can not be used as an integral part of the compression algorithm.

In most ECG compression algorithms, the Percent Root-mean-square Difference (PRD) measure is employed:

$$PRD = \sqrt{\frac{\sum_{n=1}^N (x(n) - \tilde{x}(n))^2}{\sum_{n=1}^N x^2(n)}} \times 100 \quad (3.1)$$

where $x(n)$ is the original signal, $\tilde{x}(n)$ is the reconstructed signal, and N is the length of the window over which the PRD is calculated. As this measure is very sensitive to the DC level of the original signal, a second definition of the PRD that overcomes this problem is sometimes used

$$PRD2 = \sqrt{\frac{\sum_{n=1}^N (x(n) - \tilde{x}(n))^2}{\sum_{n=1}^N (x(n) - \bar{x})^2}} \times 100 \quad (3.2)$$

where \bar{x} is the average value of the original signal.

Despite their widely accepted use as distortion measures, PRD and PRD2 do not indicate precisely the quality of the reconstruction [5]. In other words, a low value of these measures does not guarantee total preservation of the essential features of the original record and the decompressed signal has also to be evaluated by visual inspection. Recently, a new ECG distortion measure, called Weighted Diagnostic Distortion (WDD), has been introduced [6]. WDD, which is based on PQRST diagnostic features, seems well correlated with cardiologists' perception, but it is expensive to calculate [5].

In the literature, there are some other error measures for comparing original and reconstructed ECG signals [7], such as the Root Mean Square error (RMSE):

$$RMSE = \sqrt{\frac{\sum_{n=1}^N (x(n) - \tilde{x}(n))^2}{N}} \quad (3.3)$$

Another distortion measure is the Signal to Noise Ratio, which is expressed as:

$$SNR = 10 \log \left(\frac{\sum_{n=1}^N (x(n) - \bar{x})^2}{\sum_{n=1}^N (x(n) - \tilde{x}(n))^2} \right) \quad (3.4)$$

where \bar{x} is the average value of the original signal.

A Maximum amplitude error or Peak Error (MAX or PE), is also an error measure, which is expressed as:

$$MAX = \max_n \{ |x(n) - \tilde{x}(n)| \} \quad (3.5)$$

All these error measures have many disadvantages, which all result in poor diagnostic relevance. For example, base line drift in the reconstructed signal causes a non-zero value in all these error measures, but this distortion has no diagnostic meaning. Furthermore, every segment of the beat has a different diagnostic meaning and significance. A given distortion in one segment does not necessarily have the same weight as the same distortion in another segment. For example, in many patients' ECG, the ST segment is much more diagnostically significant than the TP segment.

3.3 Compression Measures

Many problems exist in the definition of compression measure. These problems mostly derive from the lack of uniformity (no standardization) in the test conditions of the various algorithms in respect of sampling frequencies and quantization levels. The size of compression is often measured by the Compression Ratio (CR) which is defined as the ratio between the number of bits to represent the original signal and the number of bits to represent the compressed signal.

$$CR = \frac{\text{total number of bits to represent the original ECG signal}}{\text{total number of bits to represent the compressed ECG signal}} \quad (3.6)$$

The problem is that every algorithm is fed with an ECG signal that has a different sampling frequency and a different number of quantization levels; thus, the bit rate of the original signal is not standard. Some attempts were made in the past to define standards for sampling frequency and quantization, but these standards were not implemented and the algorithms' developers still use rates and quantizers that are convenient to them.

In the literature, some authors use the number of bits transmitted per sample of the compressed signal as a measure of information rate. This measure removes the dependency on the quantizer resolution, but the dependence on the sampling frequency remains.

Another way is using the number of bits transmitted per second which is expressed as compressed data rate (CDR).

$$\begin{aligned} CDR &= \frac{\text{total number of bits to represent the compressed ECG signal}}{\text{duration of original ECG signal in sec}} \\ &= \frac{\text{total number of bits to represent the compressed ECG signal}}{\text{number of samples of original ECG signal / sampling frequency}} \end{aligned} \quad (3.7)$$

This measure removes the dependence on the quantizer resolution as well as the dependence on the sampling frequency.

3.4 Direct Methods

Direct data compression methods rely on prediction or interpolation algorithms which try to diminish redundancy in a sequence of data by looking at successive neighboring samples. Prediction algorithms employ a priori knowledge of previous samples, whereas interpolation algorithms use a priori knowledge of both previous and future samples. In consideration of the algorithmic structure of present ECG data reduction methods, direct data compression schemes can be classified into three categories: tolerance-comparison data compression methods, data compression by a differential pulse code modulation

(DPCM) techniques, and entropy coding techniques. In the first category a present error threshold is utilized to discard data samples; the higher the present error threshold the higher the data compression with lower recovered signal fidelity will result. The DPCM techniques attempt to diminish signal redundancy by using intersample correlation. The entropy coding techniques reduce signal redundancy whenever the quantized signal amplitudes have a nonuniform probability distribution.

3.4.1 Tolerance-Comparison Data Compression Techniques

In this section, some of the known tolerance-comparison ECG compression algorithms will be introduced.

3.4.1.1 The AZTEC (Amplitude Zone Time Epoch Coding) Technique

The AZTEC algorithm was originally developed by Cox et al. [8] for preprocessing real-time ECG's for rhythm analysis. It has become a popular data reduction algorithm for ECG monitors and databases with an achieved compression ratio of 10:1 (500 Hz sampled ECG with 12 bit resolution). However, the reconstructed signal demonstrates significant discontinuities and distortion (PRD of about 28%). In particular, most of the signal distortion occurs in the reconstruction of the P and T waves due to their slowly varying slopes.

The AZTEC algorithm converts raw ECG sample points into plateaus and slopes. The AZTEC plateaus (horizontal lines) are produced by utilizing the zero-order interpolation. The stored values for each plateau are the amplitude value of the line and its length (the number of samples with which the line can be interpolated within aperture ϵ). The production of an AZTEC slope starts when the number of samples needed to form a plateau is less than three. The slope is saved whenever a plateau of three samples or more can be formed. The stored values of the slope are the duration (number of samples of the slope) and the final elevation (amplitude of last sample point). Even though the AZTEC provides a high data reduction ratio, the reconstructed signal has poor fidelity mainly because of the discontinuity (step-like quantization) of the waves. A significant improvement in the shape, while smoothing the discontinuity, is achieved by using a smoothing filter, but this improvement causes higher error.

A modified AZTEC algorithm was proposed in [9], in which the threshold ϵ is not constant and is a function of the temporary changes in the signal properties. A data compression ratio comparable to that of the original AZTEC algorithm was achieved and signal reconstruction was improved (by means of PRD).

In another algorithm [10], vector quantization was used along with the m-AZTEC to produce a multi-lead ECG data compressor. This approach yielded a compression ratio of 8.6:1.

3.4.1.2 The Turning Point Technique

The turning point (TP) data reduction algorithm [11] was developed for the purpose of reducing the sampling frequency of an ECG signal from 200 to 100 Hz .

The algorithm processes three data points at a time: a reference point (X_0) and two consecutive data points (X_1 and X_2). Either X_1 or X_2 is to be retained. This depends on which point preserves the slope of the original three points. In this method, only the amplitudes are to be stored but not their locations. Therefore, local error results.

The compression ratio is a fixed 2:1, the sampling frequency is 200 Hz and the quantization is 12 bit.

3.4.1.3 The Coordinate Reduction Time Encoding System (CORTES)

CORTES algorithm [12] is a hybrid of the AZTEC and TP algorithms. In this algorithm, the ability of the TP is exploited to track the fast changes in the signal, and the ability of the AZTEC is exploited to compress effectively isoelectric regions. CORTES applies the TP algorithm to the high frequency regions (*QRS* complexes), whereas it applies the AZTEC algorithm to the lower frequency regions and to the isoelectric regions of the ECG signal.

The typical performances are compression ratio of 5:1 and PRD of 7%. The sampling frequency is 200 Hz, and the quantization is 12 bit.

3.4.1.4 Fan and SAPA Techniques

Fan and Scan-Along Polygonal Approximation (SAPA) algorithms, are both based on first-order interpolation [1]. The Fan algorithm was tested on ECG signals in the 1960's by Gardenhire, and further description was reported in recent reports [13] of the Fan method. In this method, the compressor searches for the most distant sample (on the time axis), such that if a line is drawn between it and the last stored sample, the local error along the line will be lower than a specific error tolerance - ϵ . The location and the amplitude of this sample are stored, and this process recurs. In this method, the reconstructed signal looks like a broken line, and its fidelity depends on the error threshold (ϵ). The greater the threshold is, the better the compression ratio, but the reconstructed signal has poorer fidelity.

The Scan-Along Polygonal Approximation (SAPA) techniques [14] are based on a similar idea to the Fan algorithm, and have similar performances. The SAPA2 algorithm, one of the three SAPA algorithms, showed the best results.

Typical performances are a compression ratio of 3:1 and a PRD of 4%. The sampling frequency is 250 Hz, and the quantization is 12 bit.

3.4.1.5 SAIES: Slope Adaptive Interpolation Encoding Scheme

The SAIES algorithm [15] combines the AZTEC and Fan compression techniques. It employs the AZTEC's slope compression technique in encoding the QRS-complex, and utilizes the Fan technique for encoding the low-frequency waves of the ECG (the isoelectric, P, and T waves).

The reported performances are a compression ratio of 5.9:1 and a PRD of 16.3%. The sampling frequency is 166 Hz, and the quantization is 10 bit.

3.4.1.6 The SLOPE Algorithm

The SLOPE algorithm [16] attempts to delimit linear segments of different lengths and different slopes in the ECG signal. It considers some adjacent samples as a vector, and this vector is extended if the coming samples fall in a fan spanned by this vector and a

threshold angle; otherwise, it is delimited as a linear segment. Similar to the SAPA and Fan algorithms, the SLOPE reconstructed signal looked like as continuous broken line.

The reported performances were: average bit rate of 190 bps “while still maintaining clinically significant information”. The results were given for a database sampled at 120 Hz and quantized at 8 bits.

3.4.1.7 The CORNER Algorithm

The CORNER algorithm [17] selects “corner points” by using the curvature of a sample and its displacement from an encoded linear segment as criteria. The curvature is estimated using the second-order difference signal.

The reported performances were average bit rate of 0.79 bits per sample with SNR of 27 dB. The database used to evaluate the method was the MIT-BIH database, which has a sampling frequency of 360 Hz and 11 bit quantizer resolution.

3.4.2 Data Compression by Differential Pulse Code Modulation (DPCM)

Some algorithms for ECG compression based on DPCM have been presented in the literature. Some of them use the DPCM as minor part of the over-all compression scheme. The basic idea behind the DPCM is that the error (residual) between the actual sample and the estimated sample value:

$$r(n) = x(n) - \hat{x}(n) \quad (3.8)$$

is quantized and transmitted or stored. The reconstruction error is mainly caused by the amplitude quantization noise of the quantized residual.

The performances of DPCM coders as a linear predictors for a compression system for ECG signals were tested by Ruttiman and Pipberger [18]. In their research some important conclusions were reached: (a) increasing the predictor order beyond 2 does not improve performance, (b) the prediction coefficients are barely changed as a function of time and, therefore, there is no use of Adaptive DPCM (ADPCM). Huffman

coding was combined with this compressor, and the reported performances were not significantly different from the performances of other direct compression methods. The database used has a sampling frequency of 500 Hz and 8 bit quantization. The compression ratio is about 7.8:1 with PRD of 3.5%.

In other work which was performed by Hsia [19], an attempt was made to exploit the quasi-periodic characteristic of the ECG signal to reduce the variance of the prediction error. The algorithm processes every cycle (beat) of the heart separately with two-stage DPCM. In the first stage, the prediction error (residual) of the current heartbeat is calculated $r(n)$ by DPCM with a third order linear predictor. In the second stage, the residual of the previous beat is subtracted from the residual of the current one, and the difference $e(n)$ is encoded with entropy code. Figure 3.1 shows this compressor.

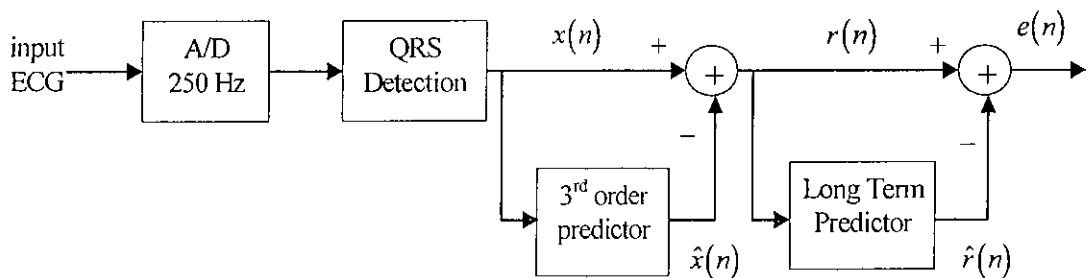


Figure 3.1 : Compression by two-step DPCM

The performances of the algorithm are compression ratio of 2:1 without any reconstruction error.

Another important work was made by Hamilton PS & Tompkins WJ [20]. In their compression algorithm, the current heartbeat is subtracted from an average beat, the residual is first differenced and then Huffman encoded (Figure 3.2).

Using quantization step sizes of 35 μV and a sampling frequency of 100 Hz, the compressor is reported to have produced average data rates of 174 bits per second for the 24 hr MIT-BIH arrhythmia database.

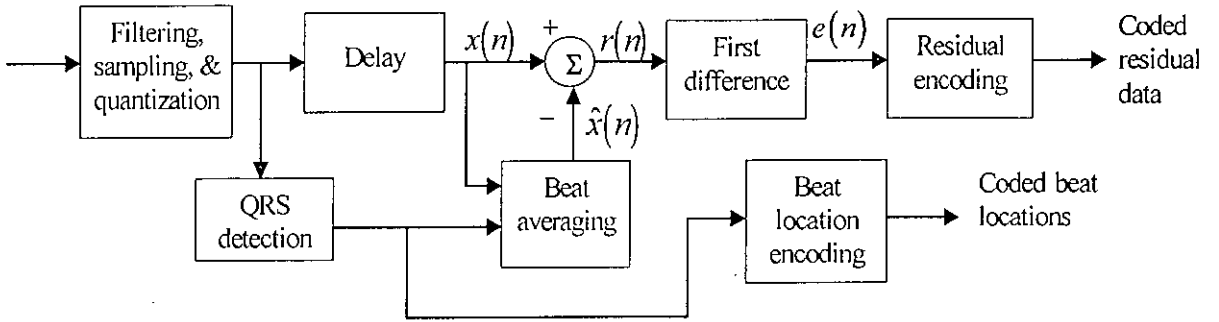


Figure 3.2 : Compression by Average Beat Subtraction

3.4.3 Entropy Coding and its Application in ECG Compression

3.4.3.1 Entropy Coding

A Discrete Memoryless Source (DMS) coding system produces a symbol every τ_s second. Each symbol is selected from a finite alphabet of symbols x_i ; $i=1,2,\dots,L$, occurring with probabilities $p(x_i)$, $i=1,2,\dots,L$. The entropy of the DMS in bits per source symbol is

$$H(x) = -\sum_{i=1}^L p(x_i) \log_2 p(x_i) \leq \log_2 L \quad (3.9)$$

where equality holds when the symbols are equally probable. The average number of bits per source symbol is $H(x)$ and the source rate in bits per second is defined as

$$R = \frac{H(x)}{\tau_s} \quad (3.10)$$

In a coder that fits one set of N bits for every symbol (fixed-length code words), the number of bits required for symbol coding is

$$N = \lceil \log_2 L \rceil \quad (3.11)$$

When the source symbols are not equally probable, a more efficient encoding method is to use variable-length codewords. An example of such encoding is the Morse code. In the Morse code, the letters that occur more frequently are assigned short code words and those that occur infrequently are assigned long code words. Following this general philosophy, we may use the probabilities of occurrence of the different source letters in

the selection of the code words. The problem is to devise a method for selecting and assigning the code words to source letters. This type of encoding is called *entropy coding*.

Two popular algorithms for this kind of coding are Huffman and Arithmetic coding.

3.4.3.1.1 Huffman Coding

A commonly used method for data compression is Huffman coding [21]. It produces best code when the probabilities of the symbols are negative powers of 2. The method starts by building a list of all the alphabet symbols in descending order of their probabilities. It then constructs a tree, with a symbol at every leaf, from the bottom up. This is done in steps, where at each step the two symbols with smallest probabilities are selected, added to the top of the partial tree, deleted from the list, and replaced with an auxiliary symbol representing both of them. When the list is reduced to just one auxiliary symbol (representing the entire alphabet), the tree is complete. The tree is then traversed to determine the codes of the symbols. This is best illustrated by an example.

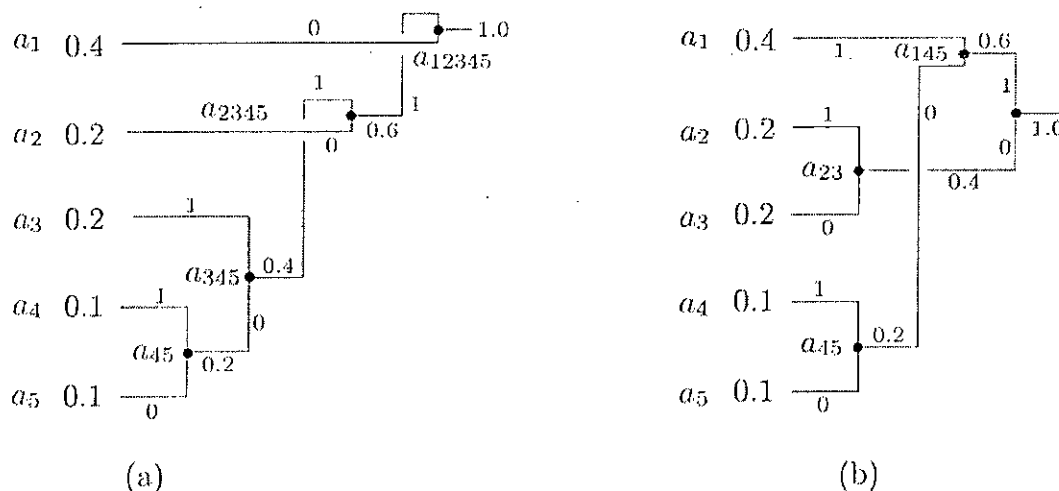


Figure 3.3: Huffman Codes

Given five symbols with probabilities as shown in Figure 3.3a, they are paired in the following order:

1. a_4 is combined with a_5 and both are replaced by the combined symbol a_{45} , whose probability is 0.2.

2. There are now four symbols left, a_1 , with probability 0.4, and a_2 , a_3 , and a_4 , with probabilities 0.2 each. We arbitrarily select a_3 and a_4 , combine them and replace them with the auxiliary symbol a_{34} , whose probability is 0.4.
3. Three symbols are now left, a_1 , a_2 , and a_{34} , with probabilities 0.4, 0.2, and 0.4, respectively. We arbitrarily select a_2 and a_{34} , combine them and replace them with the auxiliary symbol a_{234} , whose probability is 0.6.
4. Finally, we combine the two remaining symbols, a_1 and a_{234} , and replace them with a_{1234} with probability 1.

The tree is now complete. It is shown in Figure 3.3a “lying on its side” with the root on the right and the five leaves on the left. To assign the codes, we arbitrarily assign a bit of 1 to the top edge, and a bit of 0 to the bottom edge, of every pair of edges. This results in the codes 0, 10, 111, 1101, and 1100. The assignments of bits to the edges is arbitrary. The average size of this code is $0.4 \times 1 + 0.2 \times 2 + 0.2 \times 3 + 0.1 \times 4 + 0.1 \times 4 = 2.2$ bits/symbol, but even more importantly, the Huffman code is not unique. Some of the steps above were chosen arbitrarily, since there were more than two symbols with smallest probabilities. Figure 3.3b shows how the same five symbols can be combined differently to obtain a different Huffman code (11, 01, 00, 101, and 100). The average size of this code is $0.4 \times 2 + 0.2 \times 2 + 0.2 \times 2 + 0.1 \times 3 + 0.1 \times 3 = 2.2$ bits/symbol, the same as the previous code.

3.4.3.1.2 Arithmetic Coding

The Huffman method is simple, efficient, and produces the best codes for the individual data symbols. However, it produces ideal variable-size codes (codes whose average size equals the entropy) is when the symbols have probabilities of occurrence that are negative powers of 2 (i.e., numbers such as $1/2$, $1/4$, or $1/8$). This is because the Huffman method assigns a code with an integral number of bits to each symbol in the alphabet. Information theory shows that a symbol with probability 0.4 should ideally be assigned a 1.32-bit code, since $-\log_2 0.4 \approx 1.32$. The Huffman method, however, normally assigns such a symbol a code of 1 or 2 bits. Arithmetic coding overcomes the

problem of assigning integer codes to the individual symbols by assigning one (normally long) code to the entire input file. The method starts with a certain interval, it reads the input file symbol by symbol, and uses the probability of each symbol to narrow the interval. Specifying a narrower interval requires more bits, so the number constructed by the algorithm grows continuously. To achieve compression, the algorithm is designed such that a high-probability symbol narrows the interval less than a low-probability symbol, with the result that high-probability symbols contribute fewer bits to the output. An interval can be specified by its lower and upper limits or by one limit and width. The output of arithmetic coding is interpreted as a number in the range $[0, 1)$. [The notation $[a, b)$ means the range of real numbers from a to b , including a but not including b . The range is “closed” at a and “open” at b .] Thus the code 9746509 is be interpreted as 0.9746509, although the 0. part is not included in the output file.

The first step is to calculate, or at least to estimate, the frequencies of occurrence of each symbol. For best results, the exact frequencies are calculated by reading the entire input file in the first pass of a two-pass compression job. However, if the program can get good estimates of the frequencies from a different source, the first pass may be omitted.

With this example in mind, it should be easy to understand the following rules, which summarize the main steps of arithmetic coding:

1. Start by defining the “current interval” as $[0, 1)$.
2. Repeat the following two steps for each symbol s in the input stream:
 - 2.1. Divide the current interval into subintervals whose sizes are proportional to the symbols’ probabilities.
 - 2.2. Select the subinterval for s and define it as the new current interval.
3. When the entire input stream has been processed in this way, the output should be any number that uniquely identifies the current interval (i.e., any number inside the current interval).

For each symbol processed, the current interval gets smaller, so it takes more bits to express it, but the point is that the final output is a single number and does not consist of

codes for the individual symbols. The average code size can be obtained by dividing the size of the output (in bits) by the size of the input (in symbols).

The following example shows the compression steps for the short string "SWISS_MISS". Table 3.1 shows the information prepared in the first step (the *statistical model* of the data). The five symbols appearing in the input may be arranged in any order. For each symbol, its frequency is first counted, followed by its probability of occurrence (the frequency divided by the string size, 10). The range [0, 1) is then divided among the symbols, in any order, with each symbol getting a chunk, or a subrange, equal in size to its probability. Thus "S" gets the subrange [0.5, 1.0) (of size 0.5), whereas the subrange of "I" is of size 0.2 [0.2, 0.4). The cumulative frequencies column is used by the decoding algorithm.

Char	Freq	Prob.	Range	CumFreq
		Total CumFreq=		10
S	5	5/10 = 0.5	[0.5, 1.0)	5
W	1	1/10 = 0.1	[0.4, 0.5)	4
I	2	2/10 = 0.2	[0.2, 0.4)	2
M	1	1/10 = 0.1	[0.1, 0.2)	1
_	1	1/10 = 0.1	[0.0, 0.1)	0

Table 3.1: Frequencies and Probabilities of Five Symbols.

The symbols and frequencies in Table 3.1 are written on the output stream before any of the bits of the compressed code. This table will be the first thing input by the decoder.

The encoding process starts by defining two variables, Low and High, and setting them to 0 and 1, respectively. They define an interval [Low, High). As symbols are being input and processed, the values of Low and High are moved closer together, to narrow the interval. After processing the first symbol "S", Low and High are updated to 0.5 and 1, respectively. The resulting code for the entire input stream will be a number in this range ($0.5 \leq \text{Code} < 1.0$). The rest of the input stream will determine precisely where, in the interval [0.5, 1), the final code will lie. A good way to understand the process is to imagine that the new interval [0.5, 1) is divided among the five symbols of our alphabet using the same proportions as for the original interval [0, 1). The result is the five subintervals [0.5, 0.55), [0.55, 0.60), [0.60, 0.70), [0.70, 0.75), and [0.75, 1.0). When

the next symbol W is input, the third of those subintervals is selected, and is again divided into five subsubintervals. As more symbols are being input and processed, Low and $High$ are being updated according to

$$NewHigh := OldLow + Range \times HighRange(X);$$

$$NewLow := OldLow + Range \times LowRange(X);$$

where $Range = OldHigh - OldLow$ and $LowRange(X)$, $HighRange(X)$ indicate the low and high limits of the range of symbol X , respectively. In the example above, the second input symbol is W , so we update $Low := 0.5 + (1.0 - 0.5) \times 0.4 = 0.70$, $High := 0.5 + (1.0 - 0.5) \times 0.5 = 0.75$. The new interval $[0.70, 0.75)$ covers the stretch [40%, 50%) of the subrange of S . Table 3.2 shows all the steps involved in coding the string "SWISS_MISS". The final code is the final value of Low , 0.71753375, of which only the eight digits 71753375 need be written on the output stream.

The decoder works in the opposite way. It starts by inputting the symbols and their ranges, and reconstructing Table 3.1. It then inputs the rest of the code. The first digit is "7", so the decoder immediately knows that the entire code is a number of the form 0.7..... This number is inside the subrange $[0.5, 1)$ of S , so the first symbol is S . The decoder then eliminates the effect of symbol S from the code by subtracting the lower limit 0.5 of S and dividing by the width of the subrange of S (0.5). The result is 0.4350675, which tells the decoder that the next symbol is W (since the subrange of W is $[0.4, 0.5)$).

To eliminate the effect of symbol X from the code, the decoder performs the operation $Code := (Code - LowRange(X)) / Range$, where $Range$ is the width of the subrange of X . Table 3.3 summarizes the steps for decoding our example string.

Char.	The calculation of low and high	
S	L	$0.0 + (1.0 - 0.0) \times 0.5 = 0.5$
	H	$0.0 + (1.0 - 0.0) \times 1.0 = 1.0$
W	L	$0.5 + (1.0 - 0.5) \times 0.4 = 0.70$
	H	$0.5 + (1.0 - 0.5) \times 0.5 = 0.75$
I	L	$0.7 + (0.75 - 0.70) \times 0.2 = 0.71$
	H	$0.7 + (0.75 - 0.70) \times 0.4 = 0.72$
S	L	$0.71 + (0.72 - 0.71) \times 0.5 = 0.715$
	H	$0.71 + (0.72 - 0.71) \times 1.0 = 0.72$
S	L	$0.715 + (0.72 - 0.715) \times 0.5 = 0.7175$
	H	$0.715 + (0.72 - 0.715) \times 1.0 = 0.72$
-	L	$0.7175 + (0.72 - 0.7175) \times 0.0 = 0.7175$
	H	$0.7175 + (0.72 - 0.7175) \times 0.1 = 0.71775$
M	L	$0.7175 + (0.71775 - 0.7175) \times 0.1 = 0.717525$
	H	$0.7175 + (0.71775 - 0.7175) \times 0.2 = 0.717550$
I	L	$0.717525 + (0.71755 - 0.717525) \times 0.2 = 0.717530$
	H	$0.717525 + (0.71755 - 0.717525) \times 0.4 = 0.717535$
S	L	$0.717530 + (0.717535 - 0.717530) \times 0.5 = 0.7175325$
	H	$0.717530 + (0.717535 - 0.717530) \times 1.0 = 0.717535$
S	L	$0.7175325 + (0.717535 - 0.7175325) \times 0.5 = 0.71753375$
	H	$0.7175325 + (0.717535 - 0.7175325) \times 1.0 = 0.717535$

Table 3.2: The Process of Arithmetic Encoding.

Char.	Code - low	Range		
S	$0.71753375 - 0.5$	$= 0.21753375$	$/ 0.5$	$= 0.4350675$
W	$0.4350675 - 0.4$	$= 0.0350675$	$/ 0.1$	$= 0.350675$
I	$0.350675 - 0.2$	$= 0.150675$	$/ 0.2$	$= 0.753375$
S	$0.753375 - 0.5$	$= 0.253375$	$/ 0.5$	$= 0.50675$
S	$0.50675 - 0.5$	$= 0.00675$	$/ 0.5$	$= 0.0135$
	$0.0135 - 0$	$= 0.0135$	$/ 0.1$	$= 0.135$
M	$0.135 - 0.1$	$= 0.035$	$/ 0.1$	$= 0.35$
I	$0.35 - 0.2$	$= 0.15$	$/ 0.2$	$= 0.75$
S	$0.75 - 0.5$	$= 0.25$	$/ 0.5$	$= 0.5$
S	$0.5 - 0.5$	$= 0$	$/ 0.5$	$= 0$

Table 3.3: The Process of Arithmetic Decoding.

3.4.3.2 Entropy Coding of ECG's

Huffman coding has been implemented as part of some ECG DPCM coders and other coders. In the DPCM coders, like those discussed in the previous section, the residual was mapped into variable length codewords instead of fixed length ones. The residual in those DPCM coders, has a non-uniform distribution and therefore, a better compression ratio could be achieved. Recently Arithmetic coding is also being used for lossless compression at the last step of lossy compression.

3.5 Parametric Methods

Although most of the reported ECG compression algorithms belong to the direct data techniques and transformation techniques, more and more ECG compression algorithms based on parametric techniques have been proposed in recent years. Some of these algorithms are hybrids of direct and parametric techniques or transformation and parametric techniques. The compression algorithms based on parametric techniques require a preprocessing stage, which is sometimes heavy in the sense of calculation, but this is not a problem for computers today.

3.5.1 Beat Codebook

In the recent years, many ECG compression algorithms based on a Beat Codebook have been presented. This group of algorithms is found to be very efficient in ECG compression because it exploits the quasi-periodic nature of ECG signals. In this method, the redundancy, which exists in the form of correlation between beats (complexes), is reduced by matching a beat from a beat codebook to the currently processed beat. All algorithms belong to this group have a QRS detector stage to locate and segment every beat.

In the work of [22], average-beat templates are subtracted from the ECG signal. The residual (which has reduced variance) is quantized adaptively, first differenced, and Huffman encoded. The coded residual signal is stored along with the beat type (two bits) and the beat arrival time. This compression algorithm was tested with the MIT-BIH database, and the achieved bit rate was 193.3 bps, with PRD between 4.33% and 19.3%, depends on the tested signal.

Nave and Cohen [2] used a Long-Term Prediction (LTP) model, where the prediction of the n th sample is made using samples of past beats. The LTP residual signal was quantized and further compressed using the Huffman code (Figure 3.4)

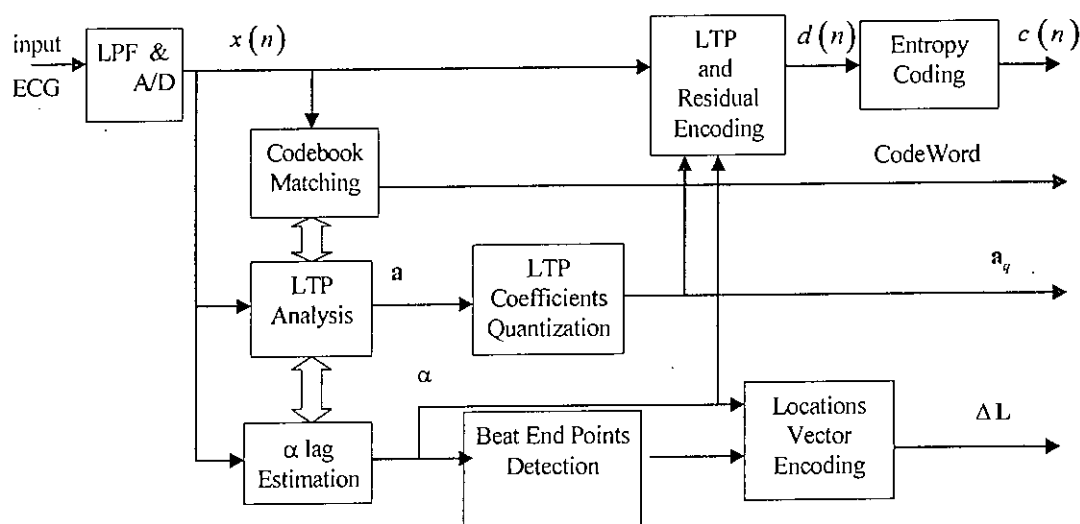


Figure 3.4: ECG Compression based on Long Term Prediction

The compression ratio depends on the number of the residual quantizer levels, which is determined prior to compression execution. For each cycle (beat) a number of parameters are to be stored (transmitted): the index of the chosen beat codeword, the quantized LTP coefficients, the beat locations vector, the quantizer range, and the coded residual (optionally).

The algorithm was tested on a local ECG database, which has a sampling frequency of 250 Hz and quantization of 10 bits/sample. The performances of the algorithm were: bit rate between 71 and 650 bps with PRD between 10% and 1% respectively.

3.5.2 Analysis by Synthesis ECG Compressor (ASEC)

The ASEC algorithm [6] is based on analysis by synthesis coding, and consists of a beat codebook, long and short-term predictors, and an adaptive residual quantizer. Fig. 3.5 shows the general scheme of the ASEC.

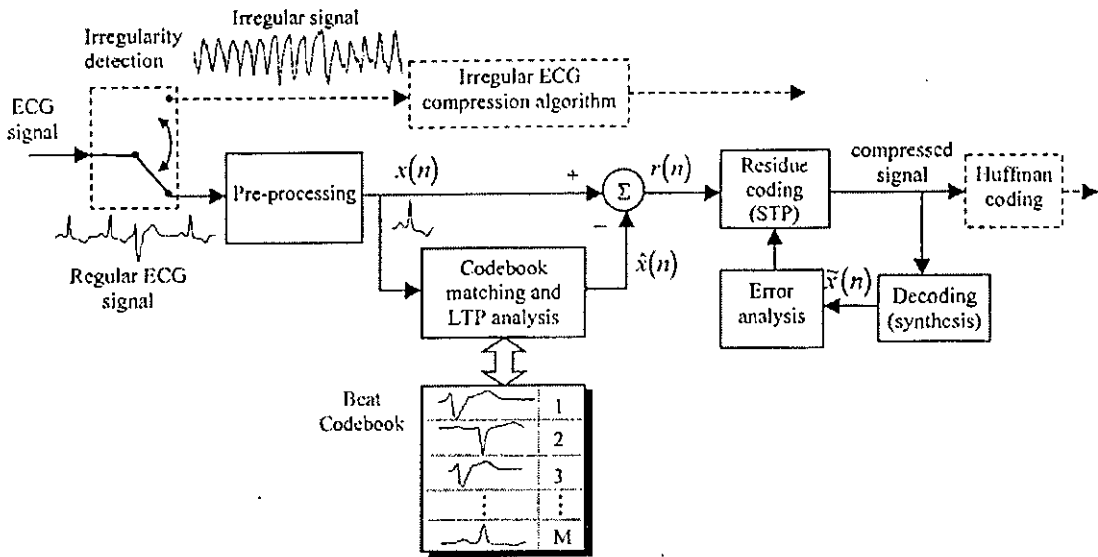


Figure 3.5: General scheme of the ASEC. Huffman coding (which was not implemented) can improve the results by approximately 10%

The ECG signal is first classified into one of two types: 1. Regular PQRST complex ECG signal (the lower branch), or to 2. Irregular ECG signal (the upper branch), such as ventricular fibrillation (VF) and ventricular tachycardia (VT). These irregular signals, in general are less probable than the regular PQRST signal. Because the irregular signals have no PQRST elements, they are not encoded like the regular ECG signal. Here, only the compression algorithm of regular PQRST ECG signals is described. The ASEC algorithm consists of three main subsystems: 1) preprocessing, 2) coding: codebook matching and long-term prediction (LTP), residue coding, error analysis, and 3) decoding. The ECG signal is processed beat by beat. The incoming beat is segmented into three time regions (P, QRS and T sections), which are then coded separately. The beat is matched with the codebook to find the best matching stored beat (“codeword”). LTP coding is performed using the chosen codeword to produce the LTP estimated (predicted) signal $\hat{x}(n)$. The difference between the original signal $x(n)$ and the LTP estimated signal $\hat{x}(n)$ is defined as the residue. The residue undergoes STP coding and adaptive quantization to produce the coded signal. Prior to transmission, the signal to be transmitted is decoded, and the quality of the reconstructed signal is tested (by means of WDD or PRD measure). The residual signal is re-encoded with higher bit rate till the

quality of the reconstructed signal is satisfied (below a predetermined distortion threshold).

A mean compression rate of approximately 100 bits/s (compression ratio of about 30:1) has been achieved with a good reconstructed signal quality (PRD2 below 8%).

3.6 Transformation Methods

Transformation techniques have generally been used in vector cardiography or multilead ECG compression and require preprocessing of the input signal by a linear orthogonal transformation and encoding of the output (expansion coefficients) using an appropriate error criterion. For signal reconstruction, an inverse transformation is carried out and the ECG signal is recovered with some error.

Many orthogonal transform compression algorithms for ECG signals have been presented in the last thirty years, such as the Fourier Transform, Walsh Transform [25], Cosine Transform [26], and Karhunen-Loeve Transform (KLT) [27]. The typical performances of the transform methods are compression ratio between 3:1 to 12:1.

In the recent years, many ECG compression algorithms have been proposed based on the Wavelet Transform (WT) [28], [29]. The reported performances are compression ratio from 13.5:1 to 22.9:1 with the corresponding PRD between 5.5% and 13.3%.

3.6.1 Wavelet Transform Based Compression

The wavelet transform is a time scale representation that has been used successfully in a broad range of applications, including ECG, audio, still and moving picture compression. The wavelet decomposes a signal $f(t)$ into a weighted sum of basis functions, where the basis functions are dilated and translated versions of a prototype function ψ called the mother wavelet. Dilation is accomplished by multiplying t by some scaling factor s^v where $v \in Z$. Translation is accomplished by considering all the integral shifts of ψ . Putting this together gives the wavelet decomposition of a signal,

$$f(t) = \sum_{v \text{ finite}} \sum_{k \text{ finite}} c_{v,k} \psi(s^v t - k), \quad k \in Z \quad (3.12)$$

With s fixed, the value ν controls the scale at which the signal is analyzed by ψ . If ν is large, low-frequency characteristics of the signal are analyzed, conversely, if ν is small, high-frequency characteristics of the signal are analyzed. Thus the wavelet decomposition allows us to view the signal at various frequency bands. s is fixed to 2, so the wavelet decomposition acts like a cascaded octave bandpass filter.

A wide variety of functions can be chosen as the mother wavelet ψ , provided that

$$\int_{-\infty}^{\infty} \psi(t) dt = 0. \text{ Here for example } \psi \text{ is chosen to be Daubechie's } W_6 \text{ wavelet.}$$

The coefficients $c_{\nu k}$ are computed by means of the forward wavelet transform, which can be accomplished by convolving the signal $f(t)$ with the basis function $\psi_{\nu k}(t)$, where

$$\psi_{\nu k}(t) = 2^{\nu/2} \psi(2^\nu t - k) \quad (3.13)$$

The multiplication $2^{\nu/2}$ is needed to make the bases orthonormal.

The compression algorithm is as follows. The ECG signal is split into blocks of size N , where N is a power of 2 (this is a requirement imposed by the W_6 wavelet). The blocking allows compression to be done in real time, as the ECG data is collected. Each block is compressed separately, by the following steps [45]:

1. The forward wavelet transform is applied to each block, generating N coefficients $c_{\nu k}$.
2. All coefficients with magnitude less than some threshold T are set to zero.
3. The remaining nonzero coefficients are quantized using the operation

$$\hat{c}_{\nu k} = \lfloor c_{\nu k} / 2^{\nu/2} + 0.5 \rfloor \quad (3.14)$$

4. The block of quantized coefficients $\hat{c}_{\nu k}$ is then runlength encoded.
5. The runlength encoded block is entropy encoded using an adaptive arithmetic encoder.

The amount of compression achieved is controlled by the threshold value T – the higher the threshold, the higher the compression ratio.

The ECG data is recovered by inverting steps 5, 4, 3 and 1 of the compression algorithm.

3.6.2 DCT and DCT-Based Compression

Due to better decorrelation and energy compaction properties and to the existence of efficient algorithms for computation, Cosine [32] transform has been widely investigated for data compression. The DCT, for example, has been used for ECG [30,31,34,35], image [32,36,37], video [32], and audio [32] compression.

3.6.2.1 The One-Dimensional DCT

The most common DCT definition of a 1-D sequence $b[n]$ of length N is

$$B[m] = \left(\frac{2}{N}\right)^{1/2} c_m \sum_{n=0}^{N-1} b[n] \cos\left[\frac{(2n+1)m\pi}{2N}\right], \quad (3.15)$$

$$m = 0, 1, \dots, N-1$$

Similarly, the inverse transformation is defined as

$$b[n] = \left(\frac{2}{N}\right)^{1/2} \sum_{m=0}^{N-1} c_m B[m] \cos\left[\frac{(2n+1)m\pi}{2N}\right], \quad (3.16)$$

$$n = 0, 1, \dots, N-1$$

In both equations (1) and (2) c_m is defined as

$$c_m = \begin{cases} (1/2)^{1/2} & \text{for } m = 0 \\ 1 & \text{for } m \neq 0 \end{cases}$$

The coefficient $B[0]$, which is directly related to the average value of the time-domain block, is often called the *DC coefficient*, and the remaining coefficients are called *AC coefficients*.

The plot of $\sum_{n=0}^{N-1} \cos\left[\frac{(2n+1)m\pi}{2N}\right]$ for $N = 8$ and varying values of m is

shown in Figure 3.6.

The first the top-left waveform ($m = 0$) renders a constant (DC) value, whereas, all other waveforms ($m = 1, 2, \dots, 7$) give waveforms at progressively increasing frequencies. These waveforms are called the *cosine basis function*. These basis functions are orthogonal. Orthogonal waveforms are independent, that is, none of the basis functions can be represented as a combination of other basis functions.

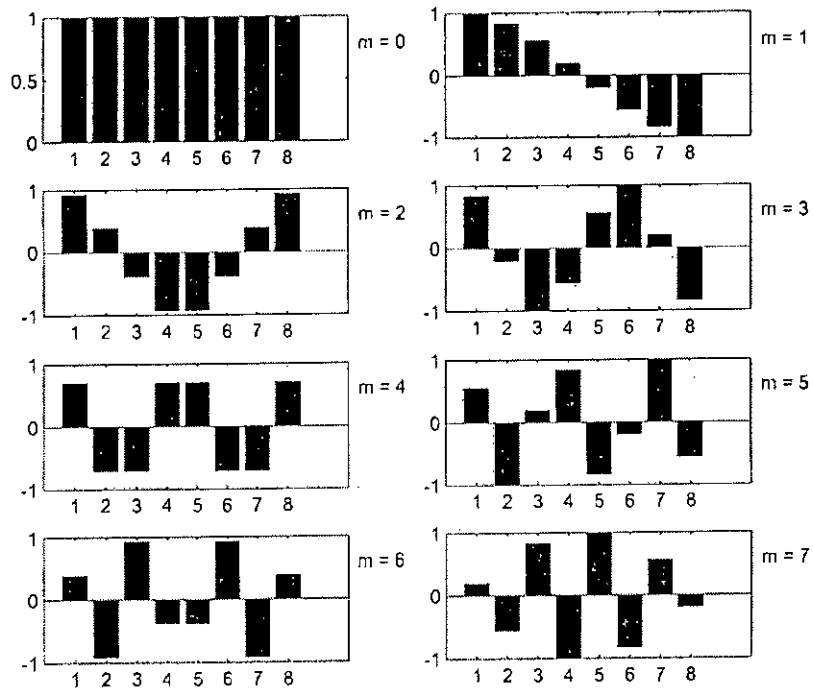


Figure 3.6: One dimensional cosine basis function

If the input sequence has more than N sample points then it can be divided into sub sequences of length N and DCT can be applied to these chunks independently. Here, a very important point to note is that in each such computation, the values of the basis function points will not change. Only the values of $b[n]$ will change in each sub-sequence. This is a very important property, since it shows that the basis functions can be pre-computed offline and then multiplied with the sub-sequences. This reduces the number of mathematical operations (i.e., multiplications and additions) thereby rendering computation efficiency.

3.6.2.2 DCT Based Compression

There are normally 4 general steps in a DCT-based compression of a data sequence x :

1. Partition of x in N_b consecutive blocks b_i , $i=0, 1, \dots, N_b-1$, each one with L_b samples;
2. DCT computation for each block;
3. Quantization of the DCT coefficients;
4. Lossless encoding of the quantized DCT coefficients.

The type-II DCT (DCT-II) [32] is commonly used for data compression due to its greater capacity to concentrate the signal energy in few transform coefficients. Let $b_i[n]$, $n=0, 1, \dots, L_b-1$, represent the L_b values in block b_i ; the one-dimensional DCT-II of this block generates a transformed block B_i constituted by a sequence of L_b coefficients $B_i[m]$, $m=0, 1, \dots, L_b-1$, given by:

$$B_i[m] = \left(\frac{2}{L_b}\right)^{1/2} c_m \sum_{n=0}^{L_b-1} b_i[n] \cos\left[\frac{(2n+1)m\pi}{2L_b}\right], \quad (3.17)$$

$$m=0, 1, \dots, L_b-1$$

where

$$c_m = 1 \text{ for } 1 \leq m \leq L_b - 1 \text{ and } c_0 = (1/2)^{1/2}$$

The DCT can be seen as a one-to-one mapping for N point vectors between the time and the frequency domains [37]. Given B_i , b_i can be recovered by applying the inverse DCT-II:

$$b_i[n] = \left(\frac{2}{L_b}\right)^{1/2} \sum_{m=0}^{L_b-1} c_m B_i[m] \cos\left[\frac{(2n+1)m\pi}{2L_b}\right], \quad (3.18)$$

$$n=0, 1, \dots, L_b-1$$

To quantize B_i , one can use a *quantization vector*, q . Each element $q[n]$, $n=0, 1, \dots, L_b-1$, of q is a positive integer in a specified interval and represents the quantization step size for the coefficient $B_i[n]$. The elements $\hat{B}_i[n]$ of the quantized DCT block \hat{B}_i are obtained by the following operation:

$$\hat{B}_i[n] = B_i[n] / q[n], \quad (3.19)$$

$$n=0, 1, \dots, L_b-1$$

$$i=0, 1, \dots, N_b-1$$

where // represents division followed by rounding to the nearest integer. In a work about image compression, Ratnakar [36] showed that it is possible to achieve a considerable gain in the CR, for a fixed distortion, by using thresholding. If $t[n]$, $n=0, 1, \dots, L_b-1$ are the elements of the *threshold vector*, \mathbf{t} , the elements of $\hat{\mathbf{B}}_i$ are now given by:

$$\hat{B}_i[n] = \begin{cases} 0, & \text{if } |B_i[n]| < t[n] \\ B_i[n] // q[n], & \text{otherwise} \end{cases} \quad (3.20)$$

$$n=0, 1, \dots, L_b-1$$

$$i=0, 1, \dots, N_b-1$$

With or without thresholding, the dequantization, performed during the decompression process to find an approximation to the original coefficients, consists simply in the multiplication of each quantized coefficient by the correspondent component of \mathbf{q} . For most DCT-based compressors, the quantization is the only lossy operation involved. The definition of \mathbf{q} and \mathbf{t} has a strong impact in CR and distortion [36]. A low quality quantization can lead to low compression ratios associated with high distortions. The intrinsic difficulties to define \mathbf{q} and \mathbf{t} , though, have led to the utilization of very simple quantization strategies in the DCT based ECG compressors reported in the literature. Ahmed et al. [34], for example, uses a unique threshold value t_0 for all coefficients. Coefficients with estimated variances less than t_0 are quantized to zero. All elements of the quantization vector are equal to 1. Varying t_0 controls the CR and the distortion. The CAB/2-D DCT [31] uses a unique quantization step size for all coefficients. This value is defined to minimize the squared mean error between the original and the reconstructed signal, for a given CR, under the condition of having the same quantization step size for all coefficients. As pointed out by Lee and Buckley [31], the good resulting compression ratios are principally due to a 2-D approach, that simultaneously explores the correlation between consecutive samples and consecutive beats of the signal, rather than to the quantization strategy. Poel [30] uses a \mathbf{q} vector whose components are values from a line segment. The value of $q[0]$ is fixed at 1 and the next values grow linearly up to the value of $q[L_b-1]$. Varying the inclination of the line segment controls the CR and the distortion. The lossless encoding of the quantized

DCT coefficients generally involves run length encoding, because the quantization normally generates many null values, followed by an entropy encoder [31].

Batista, Melcher and Carvalho [38] developed a method where the ECG signal was partitioned into blocks of 64 samples and DCT was applied to those blocks. The coefficients were quantized and thresholded by the optimum q and t vectors which were defined in a way so that the entropy of the quantized coefficients were minimized for a given distortion. The dc coefficients were differentially encoded. Then all the coefficients were resized in the limit of -128 to 127 and encoded by arithmetic encoding. The method was applied on the first two minutes of both channels of the 48 records of the MIT-BIH Arrhythmia Database. Average compression ratios of 6.2, 7.9, 9.3 and 10.9 were achieved for PRDs equal to 1.5%, 2.0%, 2.5% and 3.0% respectively.

In our present work, we followed their method [38] of defining the optimum q and t vectors. So the scheme of generation of these vectors is described in the next chapter.

Chapter 4

The Proposed Method

4.1 Introduction

This paper presents an ECG compressor based on optimized quantization of Discrete Cosine Transform (DCT) coefficients. The ECG to be compressed is partitioned into beats and a standard reference signal is subtracted from the period normalized beats. Then DCT is applied on the blocks of residual signal and each DCT block is quantized using a quantization and a threshold vector that are specifically defined for that signal. These vectors are defined, via Lagrange multipliers, so that the estimated entropy is minimized for a given distortion in the reconstructed residual signals. The optimization method presented in this paper is an adaptation for ECG of a technique previously used for image compression [36]. The quantized coefficients are clipped within a specified range, and the zeros are coded by run length encoding. In the last step of the compressor, the quantized coefficients are coded by arithmetic coder. The Percent Root-Mean-Square Difference (PRD) has been adopted as a measure of the distortion introduced by the compressor.

The ECG signals on which the proposed compression algorithm has been applied for evaluation are taken from the MIT-BIH Arrhythmia Database where the sampling rate is 360 Hz and sample resolution is 11 bits/sample. The figures 4.4 to 4.19 presented in the description of the compression algorithm are related to the steps of compression of the first 3.5 minutes of the ECG signal of record 100, channel MLII.

4.2 Compression Algorithm:

The proposed compression algorithm can be divided into pre-processing stage and encoding stage. The pre-processing stage is also divided into the first and second pre-processing stages. The figures 4.1 to 4.3 shows the block diagrams of the compression algorithm.

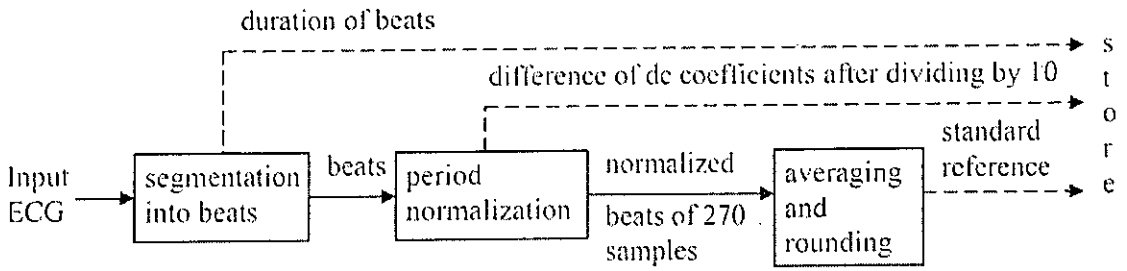


Figure 4.1 : The first pre-processing stage

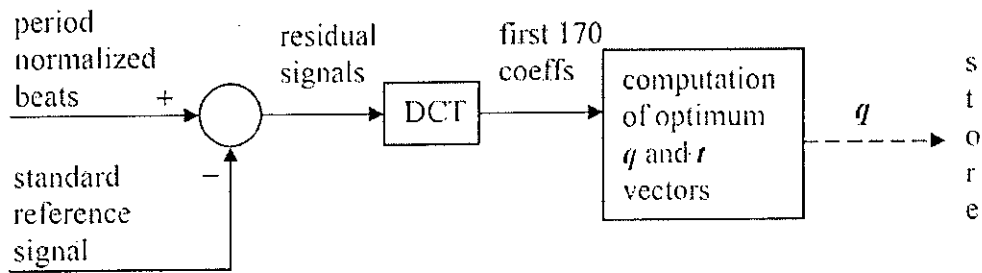


Figure 4.2 : The second pre-processing stage

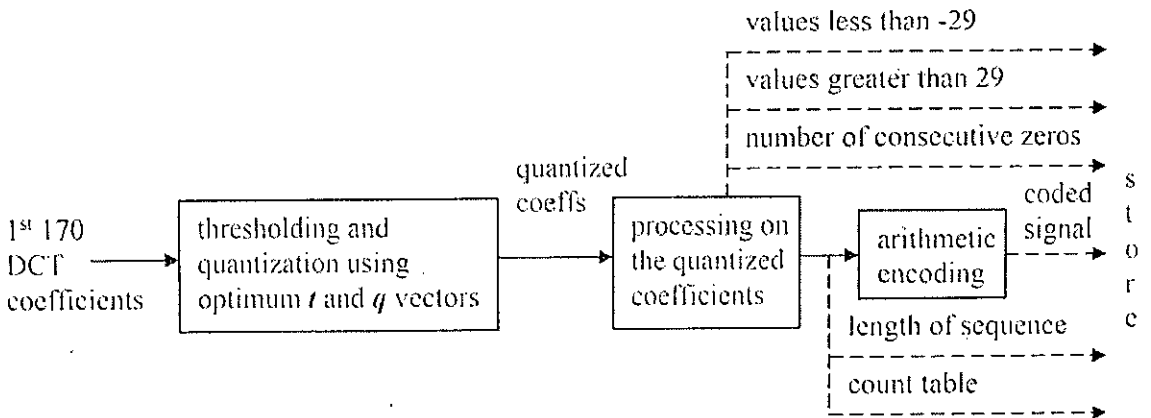


Figure 4.3 : The encoding stage

4.2.1 The First Pre-processing Stage

At first, the ECG signal is partitioned into its periods or beats. A beat is defined as the signal between two R waves. The R waves are detected by the algorithm described in the appendix-A. As ECG signal is quasiperiodic, the lengths of the partitioned blocks (beats) are not equal. The dc components of the beats are also different. The partitioned beats with unequal beat durations are shown in the figure 4.4.

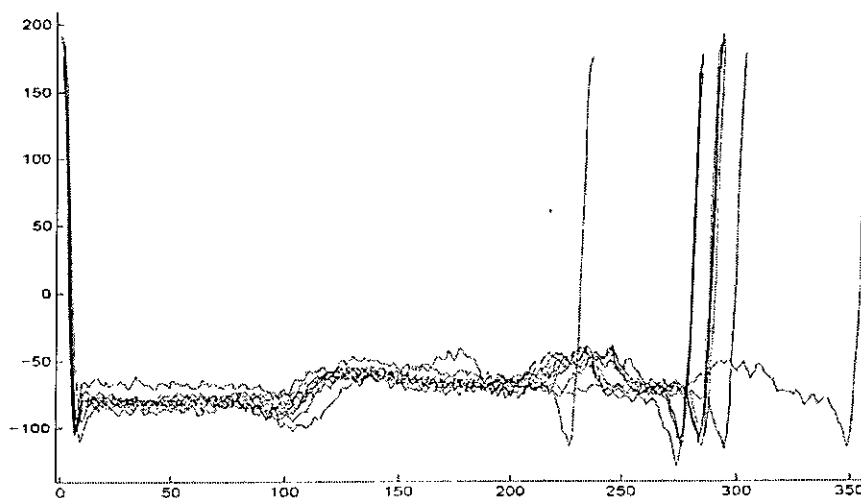


Figure 4.4 : The partitioned unnormalized beats of ECG signal

These partitioned beats are then period normalized. The dc variations are also removed. For this purpose the partitioned blocks are DCT transformed. The dc coefficients are removed by assigning zeros. Then the first 270 DCT coefficients are taken from the DCT coefficient blocks. If any DCT coefficient block contains less than 270 coefficients, then the blank coefficients are filled with zeros. Then the blocks of 270 DCT coefficients are IDCT transformed. These operations make the beats of ECG signal of equal length (270 samples which is the standard period) with zero dc components. This is shown in figure 4.5.

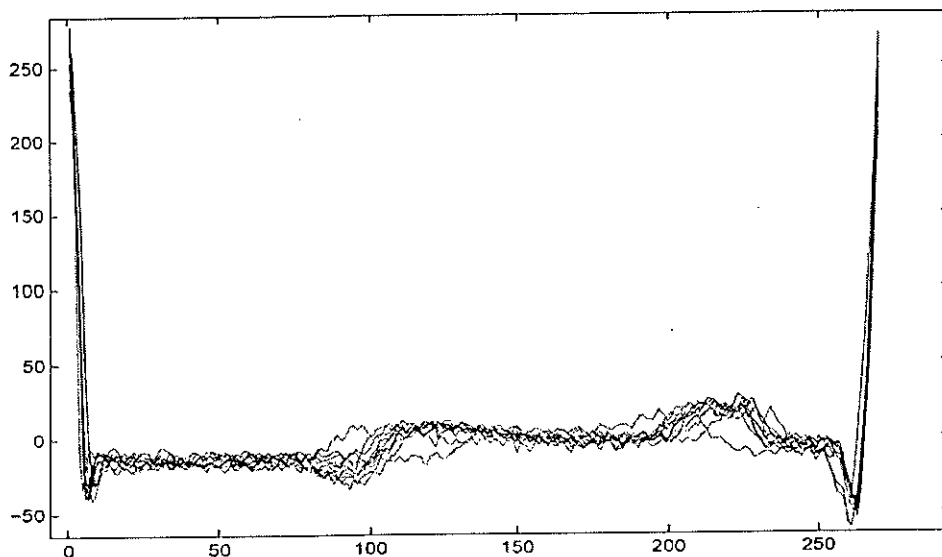


Figure 4.5 : The period normalized beats of ECG signal having zero dc component.

After averaging and rounding these normalized periods, we get the standard reference signal (figure 4.6). This is stored in the vector R .

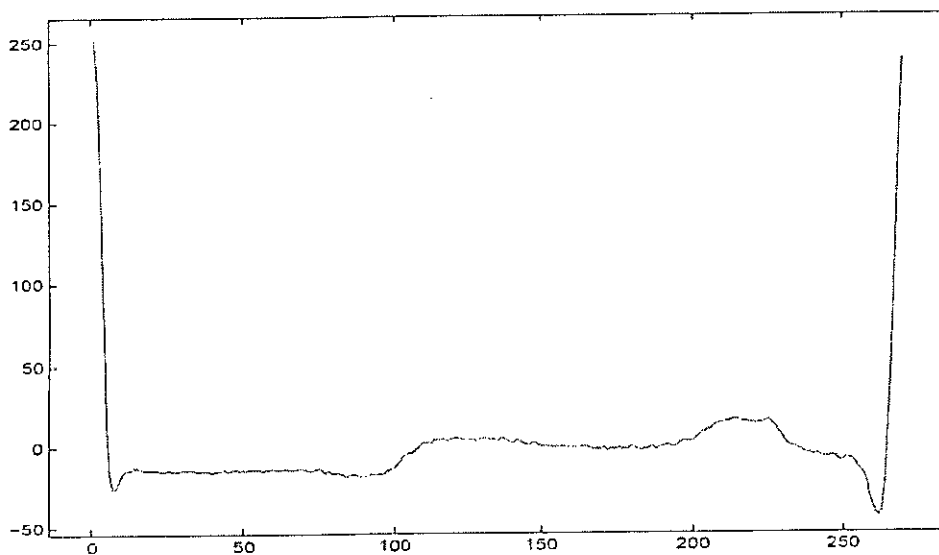


Figure 4.6 : The standard reference Signal

The duration of the beats that are segmented from the original ECG signal are stored in the vector p , which is shown in the figure 4.7. The removed dc coefficients are shown in the figure 4.8. These dc coefficients are rounded after dividing by 10, and the difference of them are stored in the vector m . It is shown in the figure 4.9.

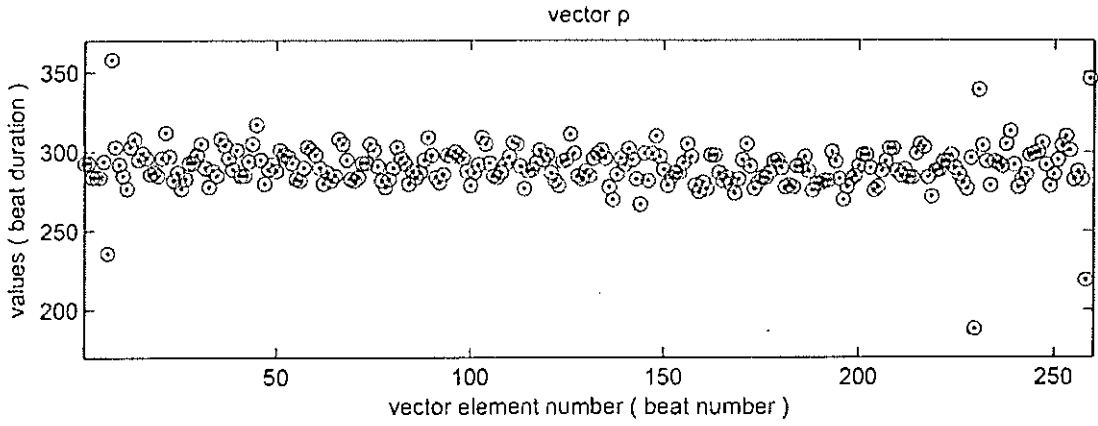


Figure 4.7 : The vector p (beat durations of the ECG signal)

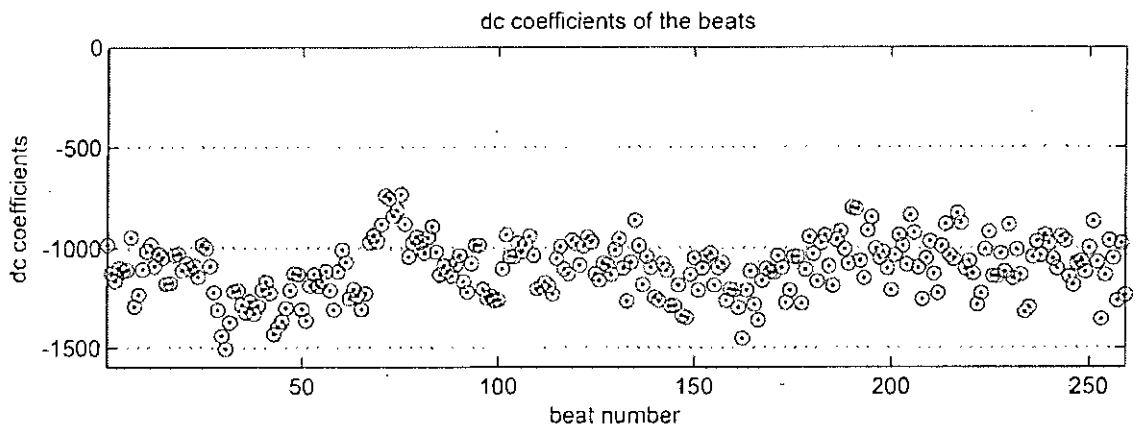


Figure 4.8 : DC coefficients of the beats.

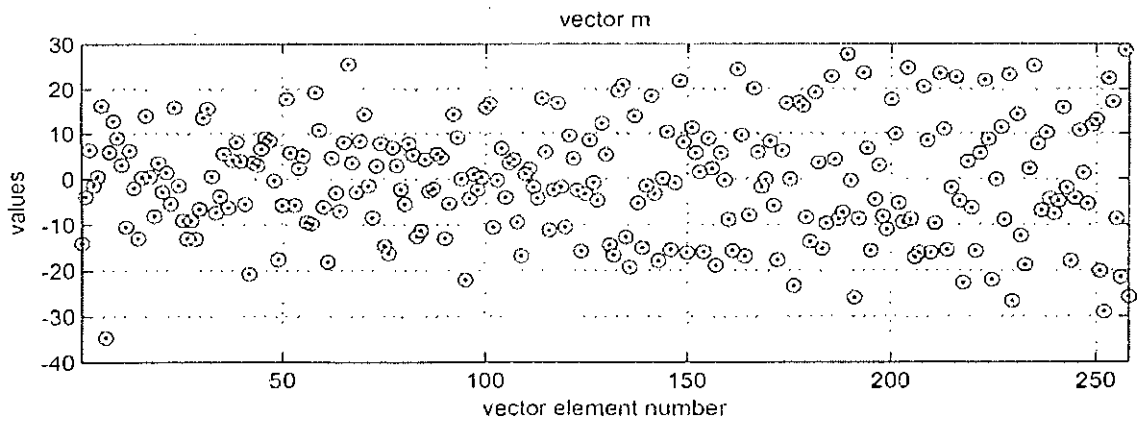


Figure 4.9 : The vector m

4.2.2 The Second Pre-processing Stage

Figure 4.2 shows the second pre-processing stage where the optimum quantization and threshold vectors are generated. The reference signal is subtracted from the period normalized beats to get blocks of residual signal of equal length (270 samples). The residual signals are shown in figure 4.10.

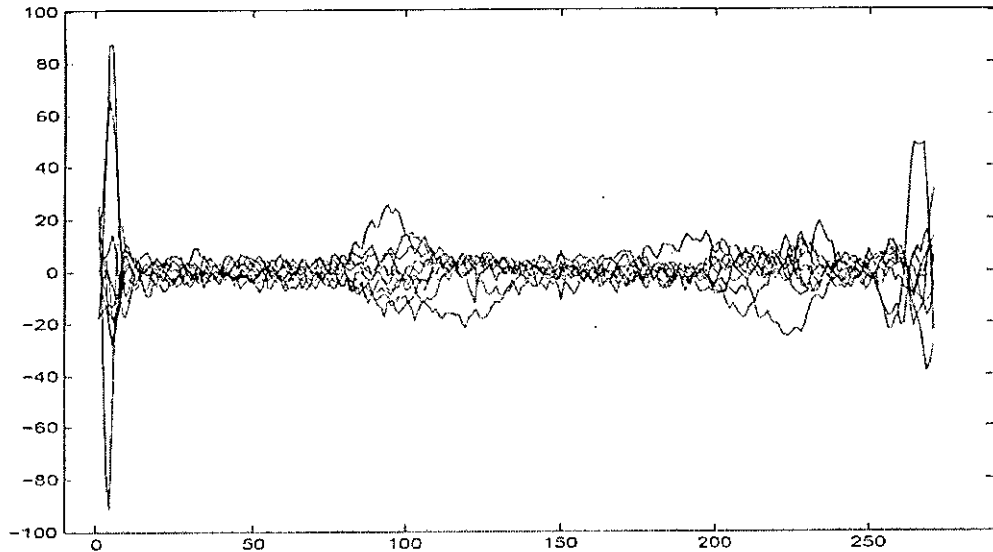


Figure 4.10 : The residual signals

DCT is then applied to these residual signals. The first 170 DCT coefficients are stored for the encoding stage.

The first 170 DCT coefficients are considered for quantization and thresholding operation. Rest of the coefficients are assumed to be zero. Let N_b be the total number of beats, B_i be the i^{th} vector of 170 DCT coefficients (i.e, first 170 DCT coefficients corresponding to the residual signal of i^{th} beat).

B is to be thresholded and quantized using the optimum threshold vector t and quantization vector q . Thresholding and quantization are necessary for the minimization of entropy and reducing allocation of bits for the values. Thresholding generates a lot of zeros and quantization limits the values in a shorter range. To get the optimum t and q vectors, the following tasks are done. For all the n^{th} coefficients, $q[n] = 1, 2, 3, \dots, 32$ and $t[n] = q[n]/2, q[n]/2+0.25, q[n]/2+0.5, \dots, 32$ are tested for quantization and thresholding operation, where $n = 1, 2, 3, \dots, 170$. Let \hat{B} be the quantized coefficients.

$$\hat{B}_i[n] = \begin{cases} 0, & \text{if } |B_i[n]| < t[n] \\ \text{round}(B_i[n]/q[n]), & \text{otherwise} \end{cases} \quad (4.1)$$

$$n = 1, 2, 3, \dots, 170$$

$$i = 1, 2, 3, \dots, N_b$$

The mean squared error introduced by the quantization of coefficient number n is given by,

$$D_n(q[n], t[n]) = \frac{1}{N_b} \sum_{i=1}^{N_b} (B_i[n] - q[n]\hat{B}_i[n])^2 \quad (4.2)$$

$$n = 1, 2, 3, \dots, 170$$

Considering that, due to the quantization and thresholding process, a value v arises $N_n(v)$ times in the coefficient number n of the N_b quantized blocks. Then the entropy $H_n(q[n], t[n])$ of the coefficient number n , measured over all quantized DCT blocks is given by,

$$H_n(q[n], t[n]) = - \sum_v p_n(v) \log_2 p_n(v) \quad (4.3)$$

where

$$p_n(v) = \frac{N_n(v)}{N_b} \quad (4.4)$$

As a result, for thresholding and quantization of the n^{th} coefficient, we get D-H curves. D-H curves for the 2nd DCT coefficient are shown in the figure 4.11.

Then for a target distortion, we get the desired optimum $q[n]$ and $t[n]$ by minimization of Lagrangian J_n ,

$$J_n = H_n(q[n], t[n]) + \lambda_n D_n(q[n], t[n]), \quad (4.5)$$

$$n = 1, 2, 3, \dots, 170$$

where λ is the Lagrange multiplier. The value of λ is given by the negative slope of the line joining the two points on the convex hull supporting the D-H graph at target distortion. Figure 4.12 shows how λ is found from the D-H curve. q is a part of the compressed signal. Figure 4.13, 4.14, 4.15 and 4.16 shows the q and t vectors for PRDs 1.5%, 2.0%, 2.5% and 3.0% respectively.

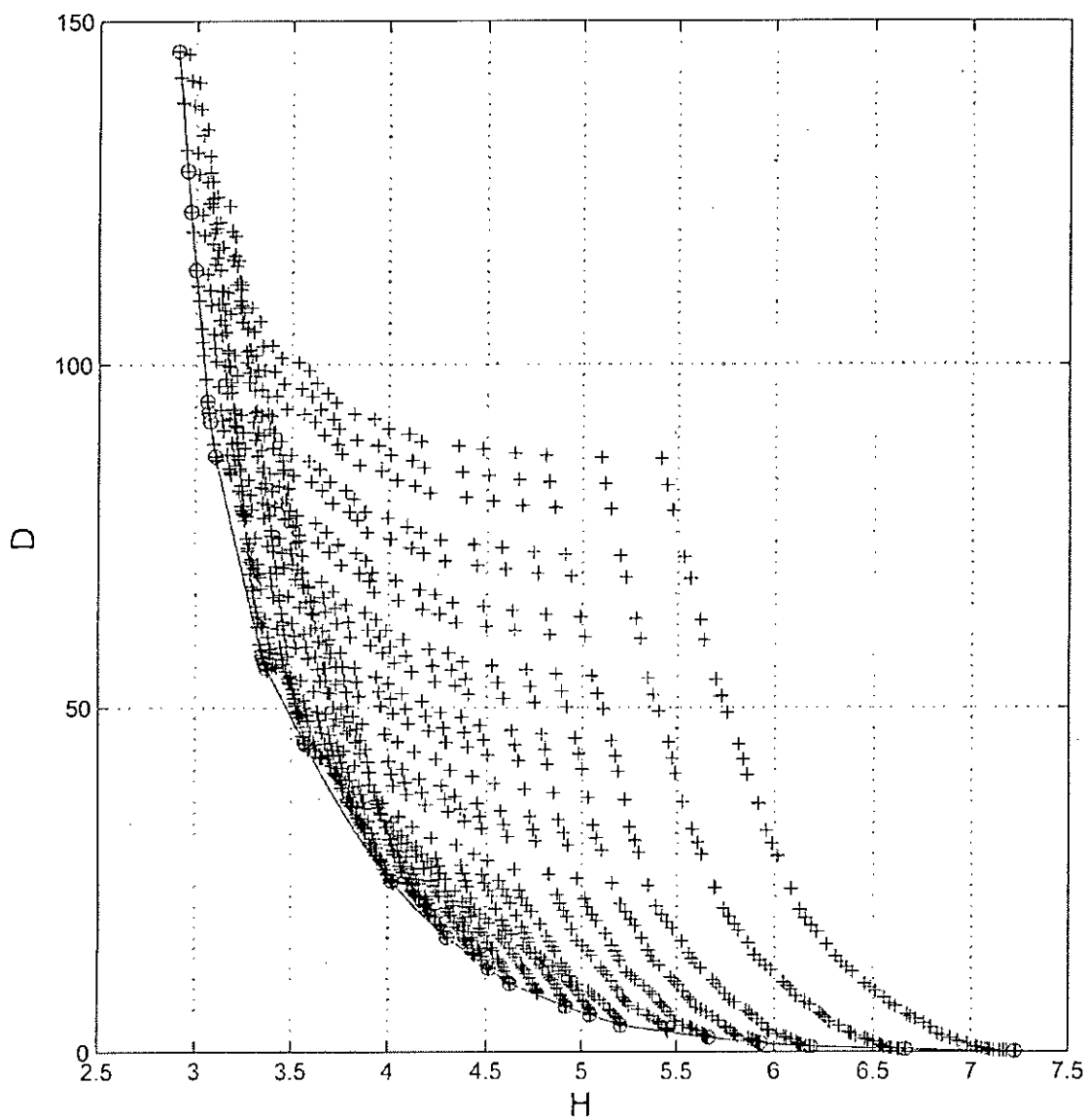


Figure 4.11 : Points of D-H found after the test of thresholding and quantization process of the 2nd DCT coefficient. The circles are the points on the inner convex hull.

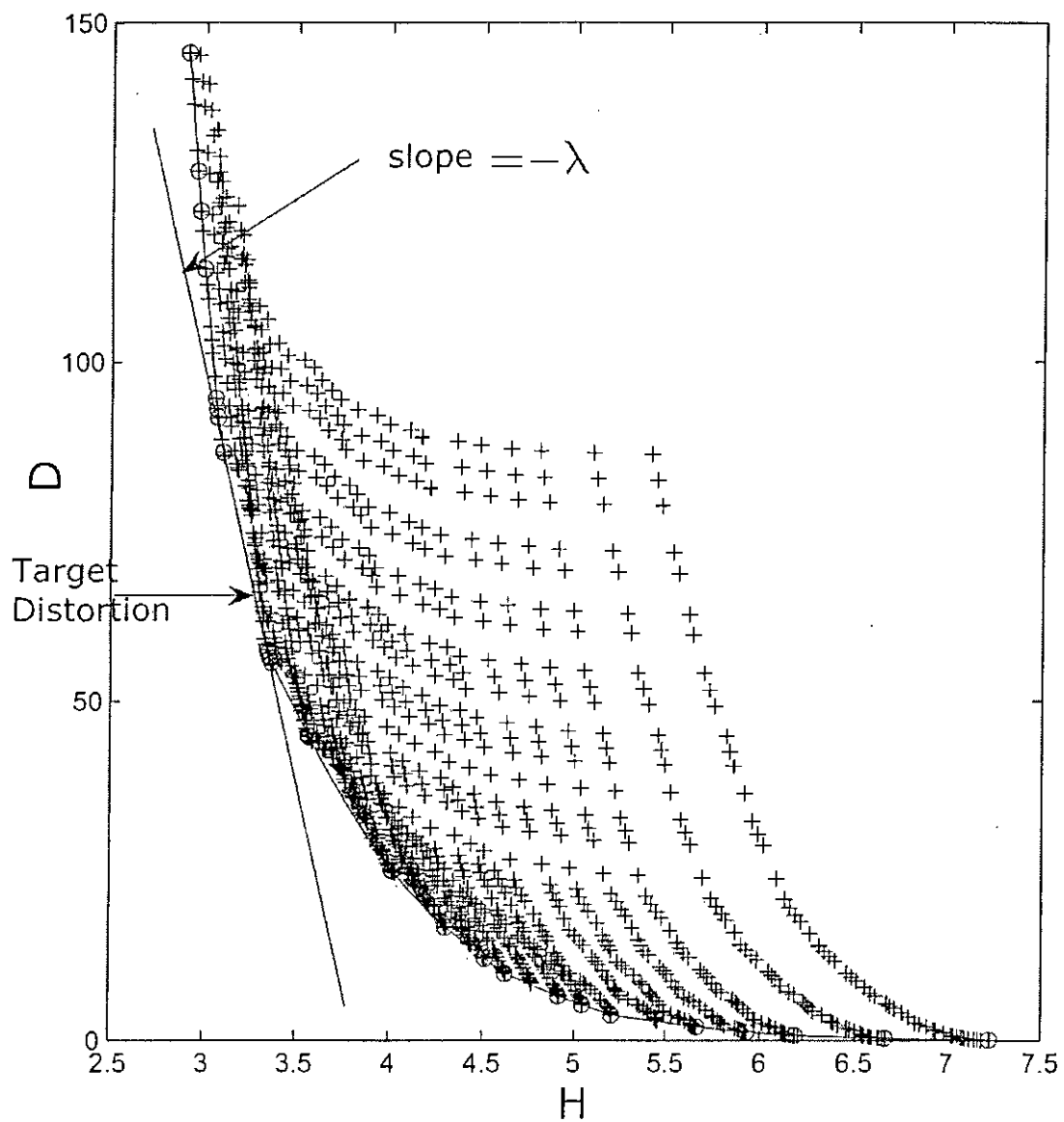


Figure 4.12 : The procedure of getting the value of λ from the points of D-H

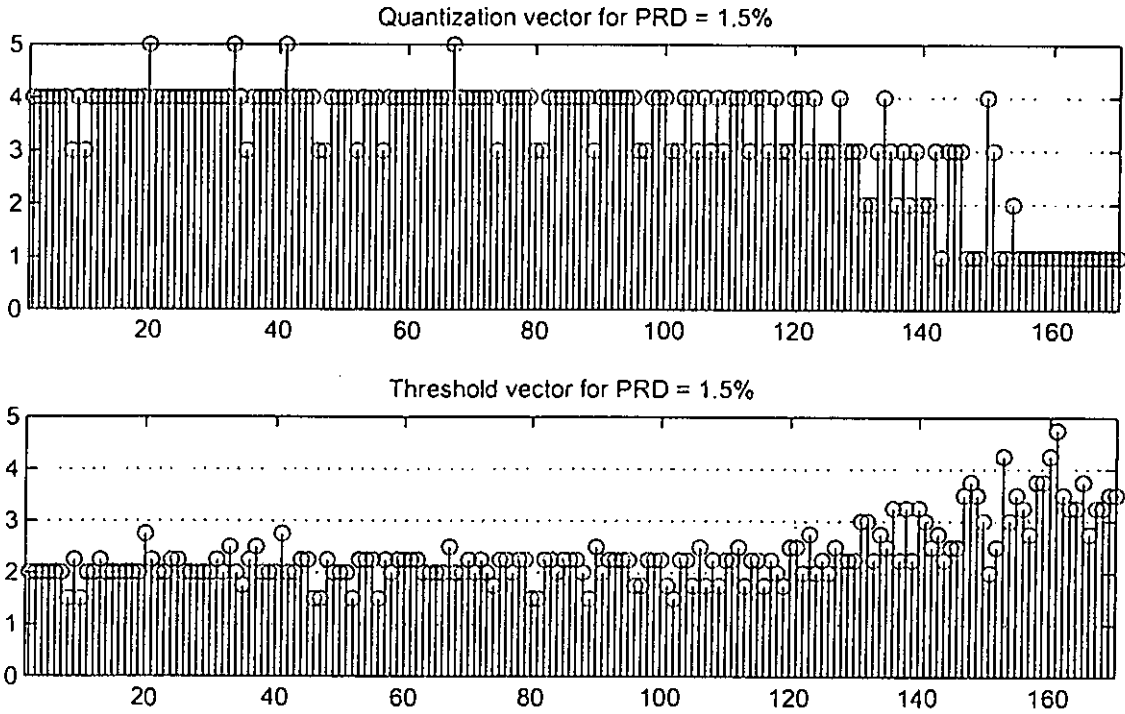


Figure 4.13 : Quantization and threshold vectors for PRD=1.5%, record 100/MLII

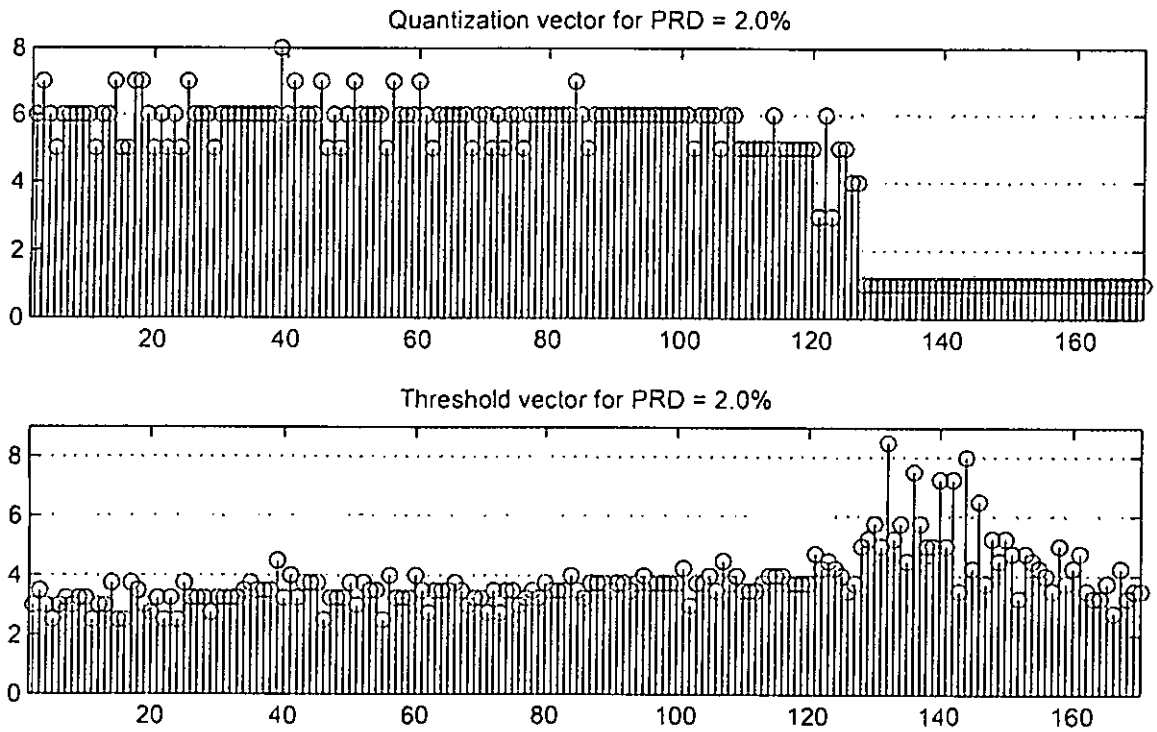


Figure 4.14 : Quantization and threshold vectors for PRD=2.0%, record 100/MLII

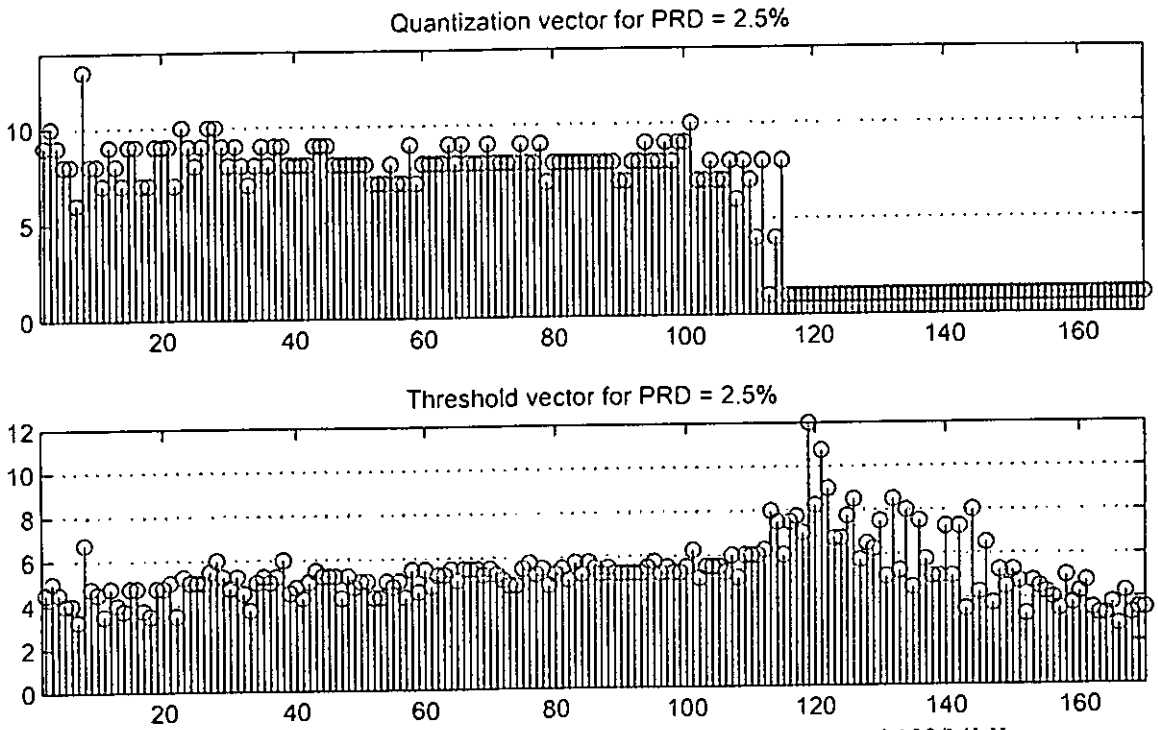


Figure 4.15 : Quantization and threshold vectors for PRD=2.5%, record 100/MLII

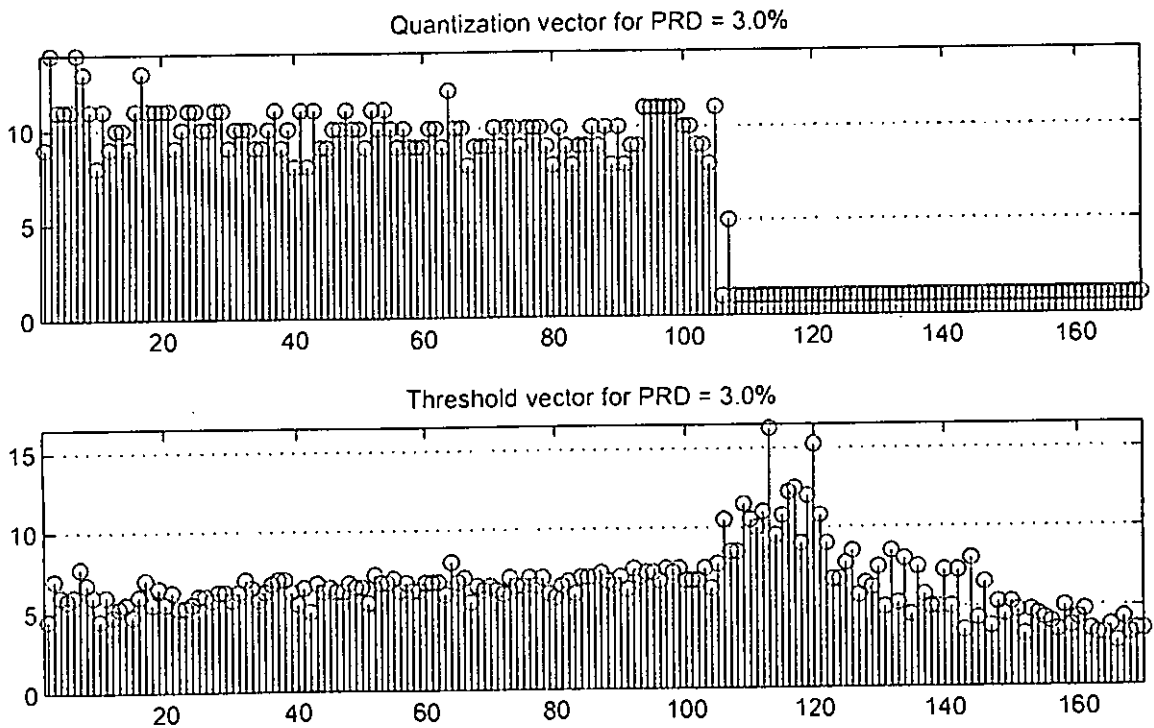


Figure 4.16 : Quantization and threshold vectors for PRD=3.0%, record 100/MLII

4.2.3 Encoding Stage

After defining the optimum quantization and threshold vectors q and t , the quantization of the first 170 DCT coefficients B is performed using q and t , and the quantized coefficients \hat{B} is produced. Some simple processing is done on the quantized coefficient blocks and the vector s is generated containing values -31 to 31. s is then lossless compressed by arithmetic encoding and *comp* is produced which is a stream of binary numbers. Two overheads are also generated before arithmetic coding. They are *count table* of the values and *length of the sequence* (i.e, length of the s vector).

The processing on the quantized DCT coefficients are shown in the figure 4.17 and explained here. Maximum values in \hat{B} are between -29 to 29. For a value $v < -29$, v is replaced by -30 in \hat{B} , and the value $-(v+30)$ is stored in the vector $u1$. Similarly a value $v > 29$ is replaced by 30 in \hat{B} , and $(v-30)$ is stored in vector $u2$.

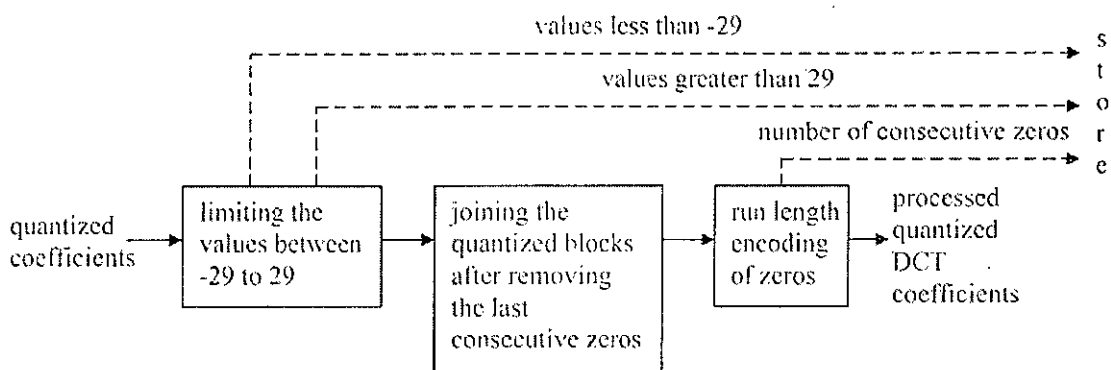
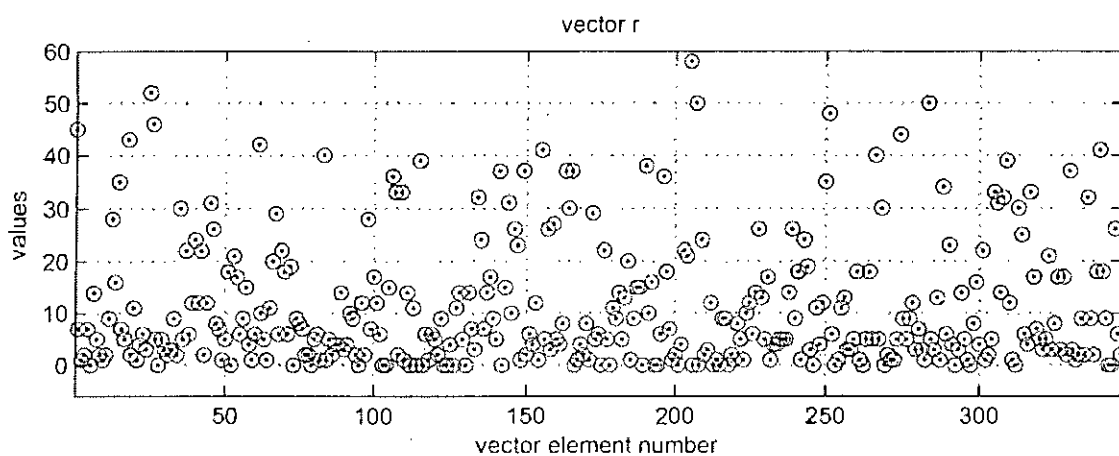
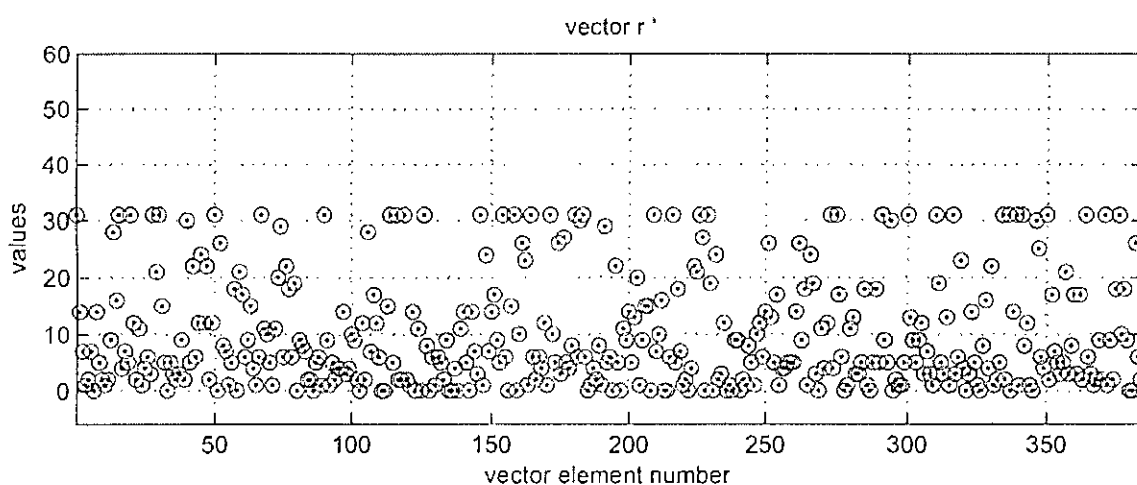


Figure 4.17 : The processing on the quantized coefficients

After every last nonzero quantized DCT coefficient of \hat{B}_i , a value -31 takes place which indicates the end of the i^{th} block. Omitting the last consecutive zeros (zeros after -31), all the blocks are joined together in a single vector s .

There are many zeros in s for threshold and quantization process. These zeros are reduced by run length encoding. If there are more than 9 consecutive zeros, then a value 31 takes place for them in the s vector, and the value (number of consecutive zeros -10) is stored in a vector r .

All the values of $u1$, $u2$ and r vectors are positive numbers. The vector u is rewritten in the vector $u1'$, where 5 bits are allocated for each vector element. So the values of $u1$ are broken down into 5 bits as follows. A value $v \geq 31$ is represented as a sequence of c numbers of 31, where c is the greatest integer such that $31 \times c \leq v$, followed by the value equal to $v - 31 \times c$. For example, the value 66 in $u1$ would be represented as (31, 31, 4) in $u1'$. Similarly the vectors $u2$ and r are rewritten in the vectors $u2'$ and r' where 5 bits are allocated for each vector element. Figures 4.18 and 4.19 show the vectors r and r' respectively.

Figure 4.18 : The vector r Figure 4.19 : The vector r'

4.3 Bit Allocation and Compression Measure

The compressed signal is composed of the vectors $comp$, $u1'$, $u2'$, r' , m , p , reference signal (the R vector), quantization vector q , dc coefficient corresponding to the first beat, count table of the symbols for arithmetic coding, length of the sequence which is to be coded by arithmetic encoding.

As the elements of q are limited to integer values between 1 and 32, allocating 5 bits for each element of the optimum q vector, it occupies $170 \times 5 = 850$ bits in the compressed file. Using 16 bits to represent the number of occurrences of the symbols in s , the count table occupies $63 \times 16 = 1008$ bits. Allocating 10 bits for each element, R vector takes $270 \times 10 = 2700$ bits. 10 and 30 bits are allocated for first dc coefficient and signal length respectively. Thus 4598 bits are fixed irrespective to the duration of the ECG signal. If the length of the ECG signal is large, then these fixed bits do not change the compression ratio. The compression ratio actually depends on the vectors $comp$, $u1'$, $u2'$, r' , m and p only.

The vector m contains the information about the dc coefficients. 8 bits are allocated for each element of m where the values may range from -128 to 127.

The vector p contains the beat durations. 10 bits are allocated for each element of p so that the values may range from 0 to 1023.

5 bits are allocated for each element of the vectors $u1'$, $u2'$ and r' to range the values from 0 to 31.

So the compression ratio can be represented as

$$\text{Compression Ratio (CR)} = \frac{b_{org\ ECG}}{b_{comp} + b_m + b_p + b_r + b_{u1'} + b_{u2'}} \quad (4.6)$$

and the compressed data rate can be measured as

$$\text{Compressed Data Rate (CDR)} = \frac{b_{comp} + b_m + b_p + b_r + b_{u1'} + b_{u2'}}{t_{org\ ECG}} \quad (4.7)$$

where, $b_{org\ ECG}$ = bits required to represent the original ECG signal
 $= \text{length}(\text{ ECG signal}) \times 11$
 b_{comp} = $\text{length}(\text{ comp}) \times 1$
 b_m = $\text{length}(m) \times 8$
 b_p = $\text{length}(p) \times 10$
 $b_{r'}$ = $\text{length}(r') \times 5$
 $b_{u1'}$ = $\text{length}(u1') \times 5$
 $b_{u2'}$ = $\text{length}(u2') \times 5$
 $t_{org\ ECG}$ = duration of ECG signal in sec
 $= \text{length}(\text{ ECG signal}) / 360$

4.4 Decompression Algorithm

Decompression is performed in the reverse order.

Using the count table and signal length, the arithmetic coded signal *comp* is arithmetic decoded and the vector *s* is produced.

From the vectors $u1'$, $u2'$ and r' , $u1$, $u2$ and r vectors are reproduced respectively.

In the place of 31, consecutive zeros are replaced. The number of consecutive zeros are found from the vector r where the numbers are added with 10.

The numbers greater than 29 and less than -29 in s are represented by the numbers 30 and -30 respectively, and are recovered with the help of the vectors $u1$ and $u2$.

The last consecutive zeros of each quantized DCT coefficient block were replaced by the number -31. So these numbers (-31) are replaced by zeros such that the length of the quantized DCT blocks become 170. Then the blocks are separated and \hat{B} is reproduced.

The quantized coefficients of \hat{B} are multiplied by the quantization vector q and we get C .

$$C_i[n] = \hat{B}_i[n] \times q[n] \quad (4.8)$$

$$n = 1, 2, \dots, 170$$

$$i = 1, 2, \dots, N_b$$

The coefficients of C are increased to 270 where the first 170 coefficients are unchanged and the remaining coefficients are zeros.

C is then IDCT transformed and the residual signals are created. These residual signals are added with the standard reference signal (the R vector) and we get the period normalized beats of 270 samples.

Again these normalized beats are DCT transformed. The dc coefficients are replaced that are reproduced from the vector m . The numbers of coefficients are changed according to the vector p . Then these DCT coefficient blocks are IDCT transformed and finally the decoded / reconstructed signal is generated.

Chapter 5

Results

5.1 Database Used for Compression

The database used in this work is a collection of files from the MIT-BIH Arrhythmia Database CD-ROM (third edition). The CD-ROM contains several hundred ECG recordings, over two hundred hours in all. Individual recordings contain one to three signals and range from 20 seconds to nearly 24 hours in length; most have two signals, are about 30 minutes long, and are annotated beat-by-beat. About one-sixth of the disk is occupied by the MIT-BIH Arrhythmia Database, which is fully annotated. The disk also contains nine additional ECG databases, and samples of several other databases of ECGs and other physiologic signals. The disk is written in ISO 9660 format.

The recordings (ECG signals) are found in eleven directories on the disk. One of them (the 'mitdb' directory) is the MIT-BIH Arrhythmia Database, which has been used in this work. The directory consists of 48 annotated records, obtained from 47 subjects studied by the Arrhythmia Laboratory of Beth Israel Hospital in Boston between 1975 and 1979. About 60% of the records were obtained from inpatients. The database contains 23 records (the '100 series') chosen at random from a set of over 4000 24-hour Holter tapes, and 25 records (the '200 series') selected from the same set to include a variety of rare but clinically important phenomena that would not be well-represented by a small random sample. Each record in this directory is slightly over 30 minutes in length. Each signal file contains two channels of ECG signals. These ECG signals were sampled at 360 Hz and quantizer resolution is 11 bits/sample. To each signal file a header file and a reference annotation file are attached. The header files include information about the leads used, the patient's age, sex, and medications. The reference annotation files include beat, rhythm, and signal quality annotations.

5.2 Results of Simulation

To assess the performance of the proposed method, we used the beats of the first two minutes of MLII channel of 40 records of the MIT-BIH Arrhythmia Database. The records are 100, 101, 103, 105, 106, 107, 111, 112, 113, 114, 115, 116, 117, 118, 119, 121, 122, 123, 124, 201, 202, 205, 208, 209, 210, 212, 213, 214, 215, 217, 219, 220, 221, 222, 223, 230, 231, 232, 233 and 234. We determined the optimum q and t vectors for each test signal and for target PRD (equation 3.1) levels of 1.5%, 2.0%, 2.5% and 3.0% and PRD2 (equation 3.2) levels of 2.0%, 3.5%, 5.0% and 6.5% within 0.04 tolerance. In all cases, the real PRD or PRD2 resulting from the compression / decompression process was within the desired range. After determining the optimum q and t vectors, the signals were compressed and the compressions are measured in CR (equation 4.6) and CDR (equation 4.7).

Table 5.1 shows the resulting compression ratio and compressed data rate (in bit per sec) obtained for the test set for each PRD value. In the cases where PRD could not reach to the desired range, the spaces are kept blank. The average CR and CDR achieved is very much convincing.

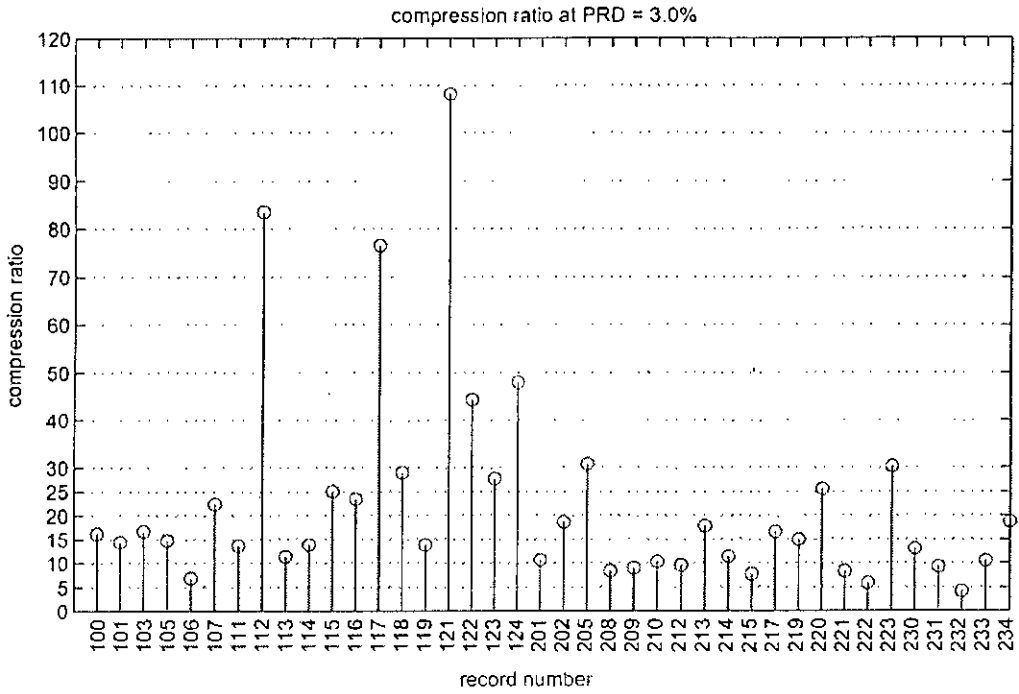
Table 5.1 : CR and CDR of the test signals at 1.5%, 2.0%, 2.5% and 3.0% PRD

Record	PRD=1.5%		PRD=2.0%		PRD=2.5%		PRD=3.0%	
	CR	CDR	CR	CDR	CR	CDR	CR	CDR
100/MLII	7.3	539	10.4	382	13.1	302	16.2	244
101/MLII	6.9	576	9.1	437	11.6	343	14.5	273
103/MLII	7.4	536	10.5	1377	13.8	285	16.7	238
105/MLII	6.8	583	9.4	421	12.0	329	14.8	269
106/MLII	3.5	1136	5.0	800	6.0	665	6.9	567
107/MLII	10.5	376	13.9	285	18.7	211	22.4	177
111/MLII	5.3	748	8.1	489	10.7	370	13.7	290
112/MLII	29.1	136	42.4	93	60.3	66	83.5	47
113/MLII	5.3	742	7.5	527	9.7	409	11.4	348
114/MLII	–	–	8.2	484	11.2	353	13.9	285
115/MLII	12.7	313	16.9	235	21.3	87	25.0	158

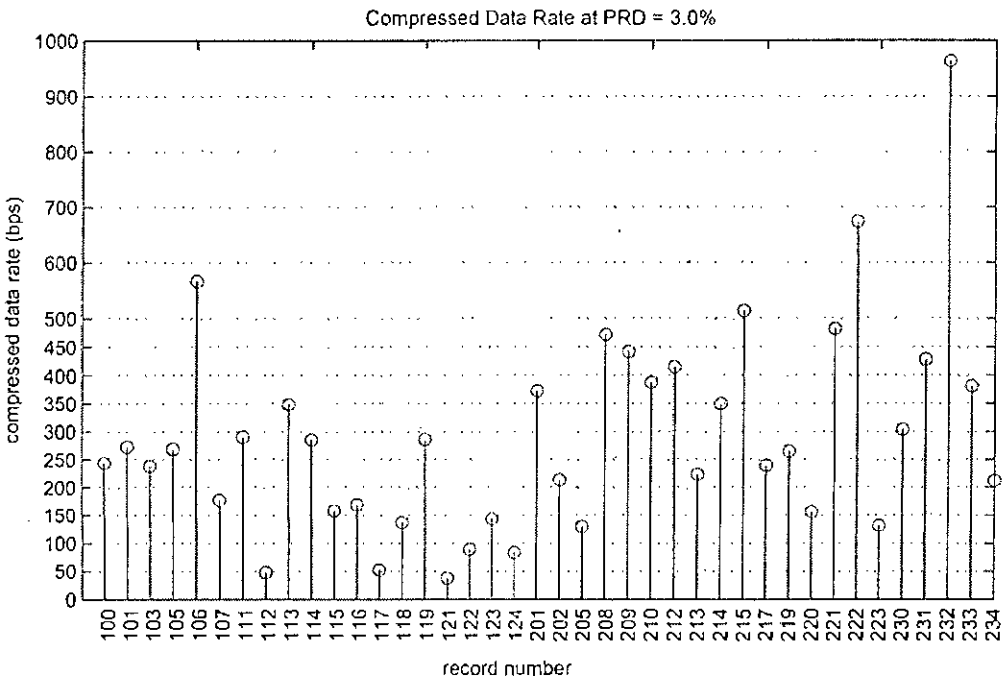
116/MLII	12.2	324	16.2	244	20.4	194	23.5	169
117/MLII	29	137	44.3	90	60.2	66	76.5	52
118/MLII	15.1	262	19.8	200	24.2	164	29.0	137
119/MLII	9.1	435	11.2	355	12.7	311	13.8	286
121/MLII	35.9	110	55.1	72	99.0	40	108.2	37
122/MLII	16.7	237	23.7	167	33.4	119	44.3	89
123/MLII	14.9	267	19.8	200	23.3	170	27.7	143
124/MLII	22.8	173	30.8	129	38.6	103	47.9	83
201/MLII	4.8	831	6.5	611	8.6	460	10.6	372
202/MLII	–	–	8.7	453	13.7	290	18.6	213
205/MLII	12.7	313	17.5	226	24.1	164	30.7	129
208/MLII	5.2	759	6.2	631	7.4	531	8.3	472
209/MLII	4.0	983	5.5	725	7.1	558	9.0	442
210/MLII	4.4	892	6.6	604	8.6	458	10.3	387
212/MLII	4.6	861	6.6	598	8.3	473	9.5	415
213/MLII	10.6	372	12.8	309	15.1	262	17.8	223
214/MLII	6.6	602	8.3	476	9.7	408	11.4	349
215/MLII	4.0	992	5.4	728	6.7	595	7.7	514
217/MLII	8.6	460	11.6	343	14.1	280	16.6	239
219/MLII	9.0	441	10.9	364	12.7	311	14.9	265
220/MLII	10.8	366	15.2	260	20.5	193	25.5	156
221/MLII	4.3	925	5.7	696	7.2	544	8.3	482
222/MLII	–	–	–	–	4.6	867	5.9	674
223/MLII	15.7	253	19.8	200	24.8	159	30.3	131
230/MLII	7.1	558	9.4	420	11.4	349	13.0	304
231/MLII	4.9	813	6.9	577	8.2	485	9.3	428
232/MLII	–	–	–	–	–	–	4.1	963
233/MLII	6.7	592	7.9	501	9.4	423	10.5	381
234/MLII	7.3	538	11.3	351	14.9	267	18.7	212
Average	7.4	533	9.4	423	12.2	325	13.7	291

Figures 5.1 (a) shows the compression ratios and (b) shows the compressed data rates of the test records at PRD = 3.0%.

1042-83



(a)



(b)

Figure 5.1 : (a) Compression Ratios and (b) Compressed Data Rates found at PRD = 3.0%

Table 5.2 shows the compression ratio and compressed data rate obtained for the test set for each PRD2 value. The spaces are kept blank in the cases where PRD2 could not reach to the desired range.

Table 5.2: CR and CDR of the test signals at 2.0%, 3.5%, 5.0% and 6.5% PRD2

Record	PRD2=2.0%		PRD2=3.5%		PRD2=5.0%		PRD2=6.5%	
	CR	CDR (bps)	CR	CDR (bps)	CR	CDR (bps)	CR	CDR (bps)
100/MLII	3.9	1010	8.2	483	12.1	328	16.6	239
101/MLII	6.7	594	11.5	345	17.2	230	25.0	161
103/MLII	8.3	477	15.7	253	25.1	158	40.8	97
105/MLII	7.4	542	13.8	288	22.0	180	37.6	105
106/MLII	3.5	1136	5.0	800	6.0	665	6.9	567
107/MLII	13.2	299	24.6	161	–	–	–	–
111/MLII	6.0	659	13.4	296	19.9	199	26.6	149
112/MLII	5.5	719	14.3	278	22.7	174	30.3	131
113/MLII	7.3	545	12.4	320	18.5	214	26.5	149
114/MLII	8.2	486	16.6	238	25.5	155	40.7	97
115/MLII	9.1	434	16.2	245	22.9	172	30.9	128
116/MLII	8.3	479	15.0	263	21.7	183	29.0	136
117/MLII	–	–	13.8	287	24.3	163	35.1	113
118/MLII	8.3	477	14.6	271	20.0	195	24.9	158
119/MLII	7.2	544	9.8	404	12.7	312	14.8	270
121/MLII	8.3	476	20.9	189	33.3	119	49.5	80
122/MLII	8.7	455	14.3	277	21.5	184	32.3	123
123/MLII	7.2	547	13.2	299	18.6	213	22.6	176
124/MLII	12.3	321	21.7	182	32.2	123	40.4	98
201/MLII	4.7	834	9.0	440	12.8	310	16.9	234
202/MLII	5.5	716	16.8	236	29.0	136	44.4	89
205/MLII	5.8	679	14.1	280	21.2	187	30.5	130
208/MLII	6.2	635	9.0	439	11.9	333	14.1	281

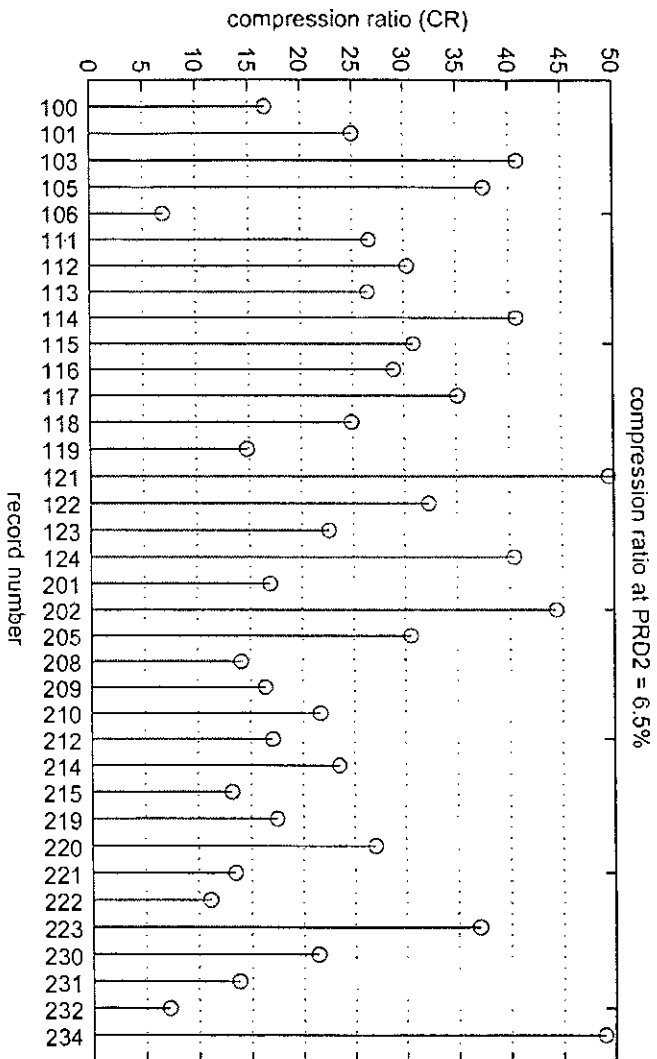
209/MLII	4.4	890	8.7	454	12.6	314	16.4	241
210/MLII	5.7	694	10.4	384	15.7	252	21.7	183
212/MLII	6.5	606	10.0	395	13.3	298	17.1	231
213/MLII	11.7	338	17.5	227	23.5	168	–	–
214/MLII	8.1	485	11.9	334	17.4	227	23.5	169
215/MLII	4.9	813	7.9	500	10.6	372	13.2	299
217/MLII	11.2	353	19.3	205	29.5	134	–	–
219/MLII	7.7	514	10.9	363	14.7	270	17.5	226
220/MLII	6.3	626	12.4	320	19.6	202	27.0	146
221/MLII	5.0	790	8.1	491	10.8	367	13.5	294
222/MLII	–	–	5.3	743	8.4	471	11.1	357
223/MLII	12.2	325	20.1	197	30.0	132	37.0	107
230/MLII	8.9	441	13.8	286	18.2	217	21.4	185
231/MLII	5.4	731	9.0	440	11.3	350	13.8	287
232/MLII	–	–	–	–	3.9	1004	7.2	551
233/MLII	7.7	515	11.0	360	13.5	294	–	–
234/MLII	10.7	368	19.6	202	31.1	127	49.0	80
Average	6.8	583	11.7	338	15.0	260	20.1	196

Figures 5.2 (a) shows the compression ratios and (b) shows the compressed data rates of the test records at PRD2 = 6.5%.

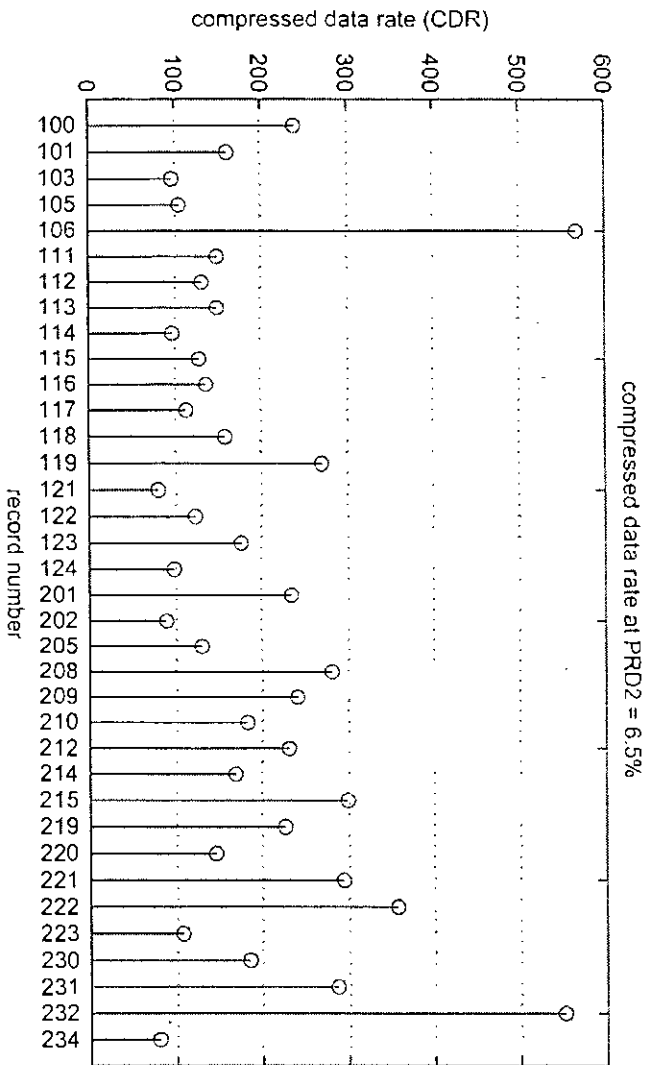
Figure 5.3 (a) and (b) presents the plot of average compression ratio and average compressed data rate at PRDs 1.5%, 2.0%, 2.5% and 3.0%, (c) and (d) presents the plot of average compression ratio and average compressed data rate at PRDs 2.0%, 3.5%, 5.0% and 6.5%.

We also tested the performance of the compressor on the full duration (approximately 30 minutes) of record 100/MLII for PRD = 7.0%. The resulting compression ratio and compressed data rate are 54.8:1 and 79 bps respectively. For a given target PRD, our implementation of the proposed method took approximately 25 min to compress 30 min signal.





(a)



(b)

Figure 5.2 : (a) Compression Ratios and (b) Compressed Data Rates found at PRD2 = 6.5%

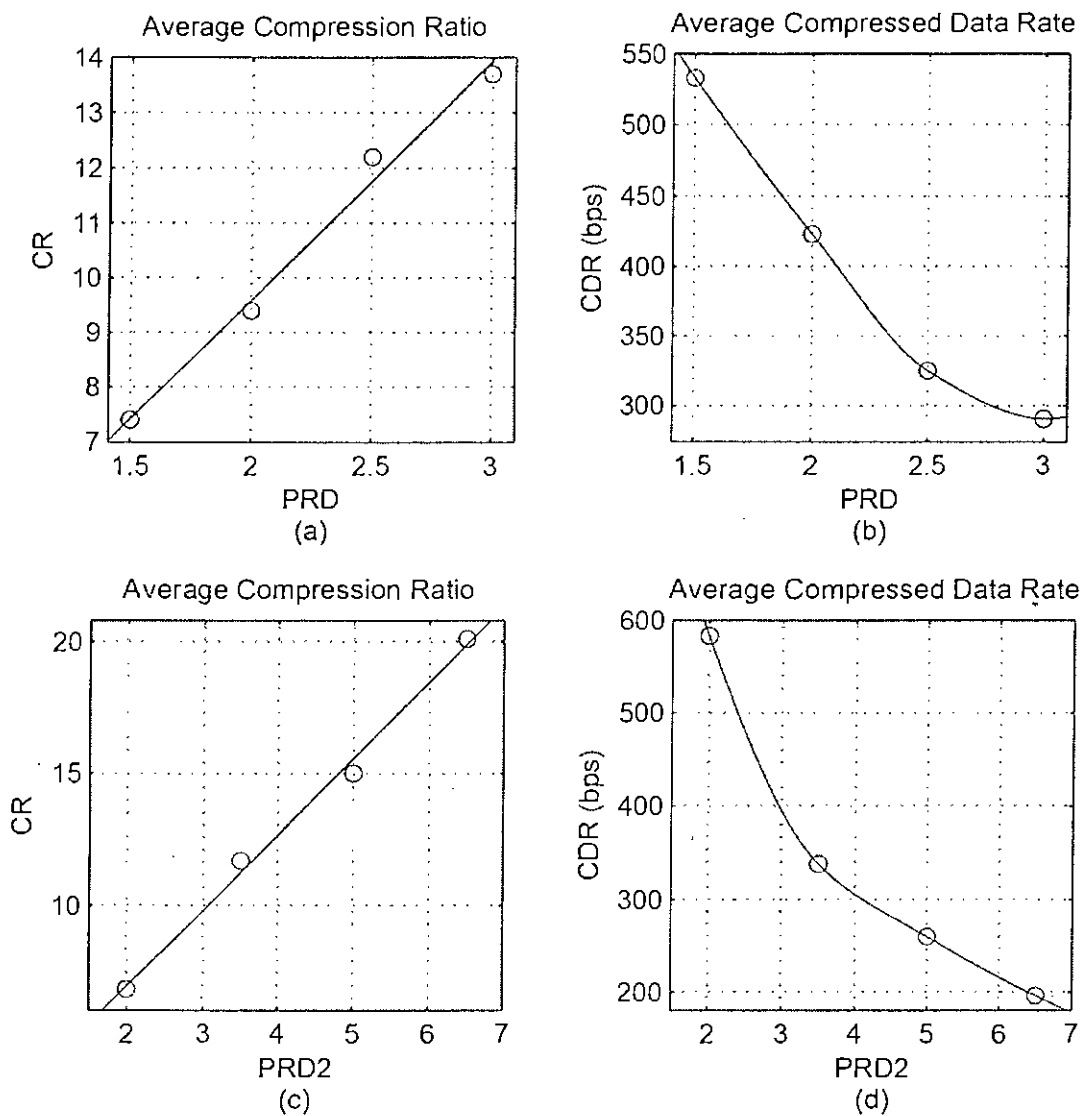


Figure 5.3 : (a) Average Compression Ratio and (b) Average Compressed Data Rate at PRD 1.5%, 2.0%, 2.5% and 3.0%, (c) Average Compression Ratio and (d) Average Compressed Data Rate at PRD2 2.0%, 3.5%, 5.0% and 6.5%.

The processing was performed by the software MATLAB on a 2.4 GHz Intel Celeron running Windows XP. This time includes the reading of the signals from the hard disk, the definition of the optimum q and t vectors for the target PRD, and the compression of the signal.

Figure 5.4 presents a single cycle of record 100/MLII from the MIT/BIH Arrhythmia Database, and the reconstructed signal for PRD equal to 2.5% and CR equal to 13.1, allowing detailed analysis of the distortions in important regions of the signal.

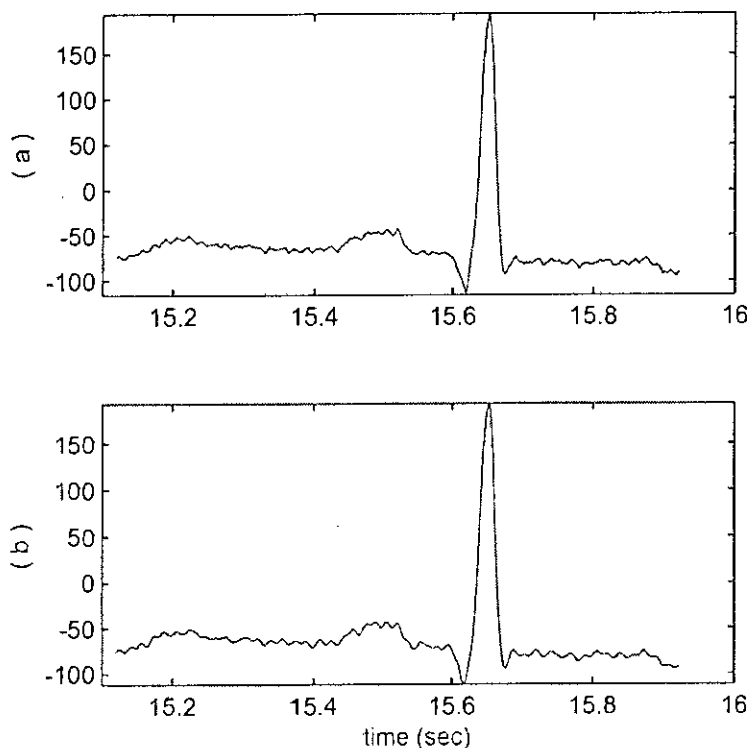


Figure 5.4 : Detail of (a) record 100/MLII and (b) reconstructed signal for PRD =2.5%, CR =13.1

Figures 5.5-5.11 allow visual assessment of the quality of the reconstruction at various distortion levels for some larger sections of ECG traces with different characteristics. Figure 5.5 presents 9 seconds of record 100/MLII from the MIT/BIH Arrhythmia Database, and the reconstructed signal for PRD equal to 1.5%, 2.0%, 2.5% and 3.0%. Figures 5.6–5.11 show 7-sec sections from channel MLII of records 112, 121, 124, 205, 222 and 232 respectively, and the reconstructed signals and reconstructed errors for PRD equal to 6.5%. These traces indicate an excellent preservation of QRS complexes and of all important signal features. Figure 5.10 shows the performance of the compressor in the presence of severe noise.

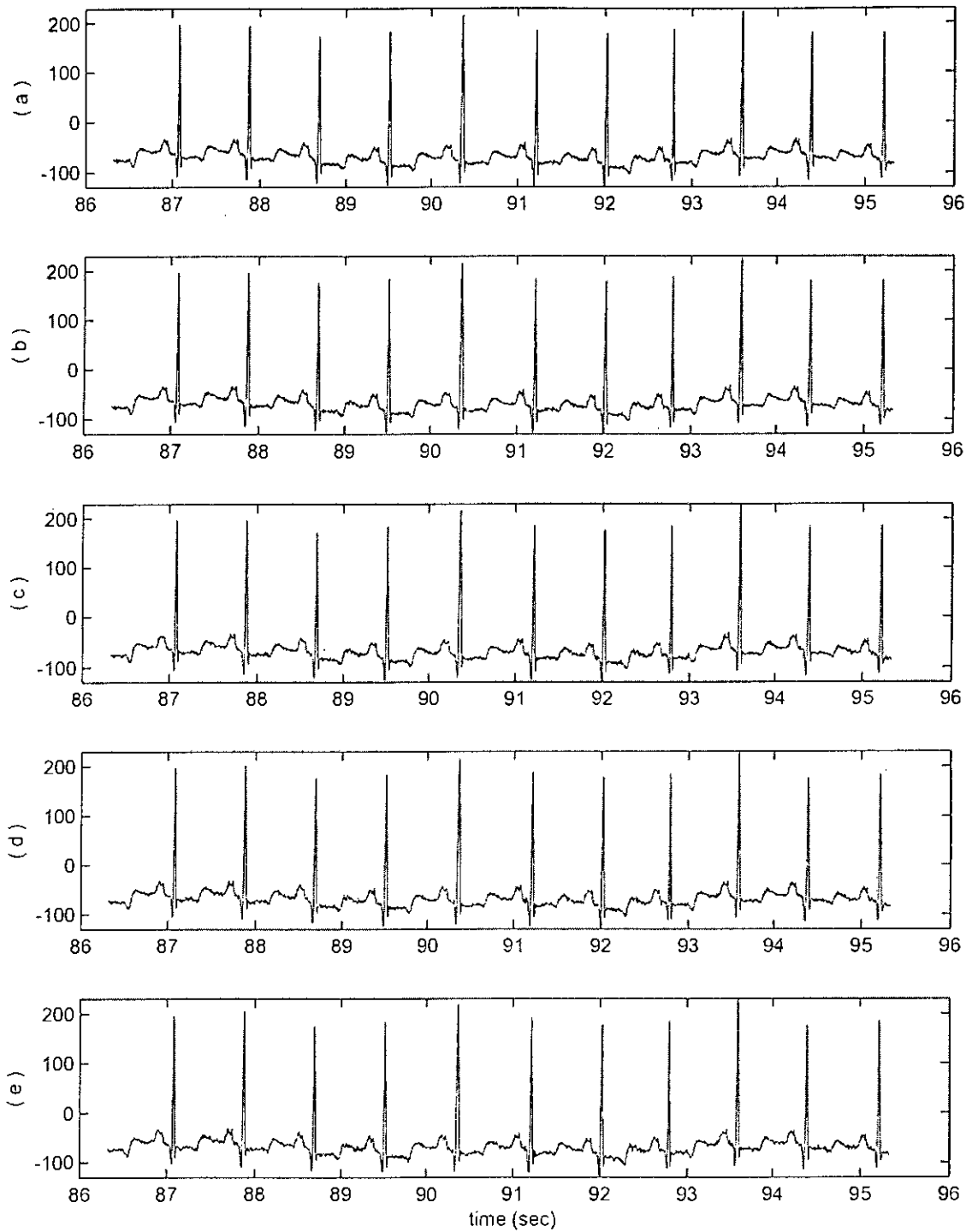


Figure 5.5 : (a) 9 sec part of record 100/MLII, and reconstructed signal for (b) PRD = 1.5%, CR = 7.3; (c) PRD = 2.0%, CR = 10.4; (d) PRD = 2.5%, CR = 13.1; (e) PRD = 3.0%, CR = 16.2

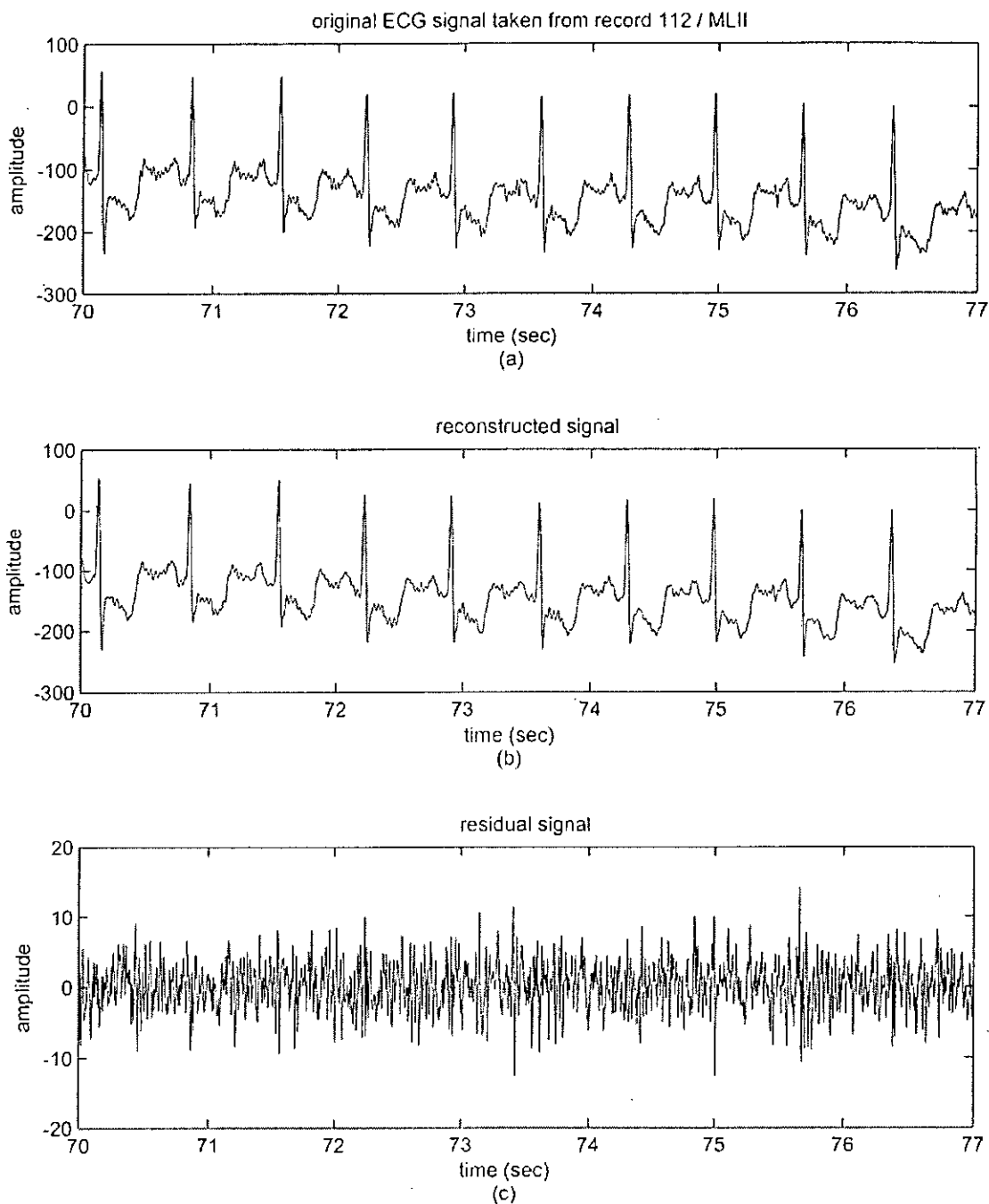


Figure 5.6 : (a) 7 second section of record 112 / MLII, (b) reconstructed signal for PRD2 = 6.5%, PRD = 1.6%, CR = 30.3, CDR = 131 bps, (c) residual signal.

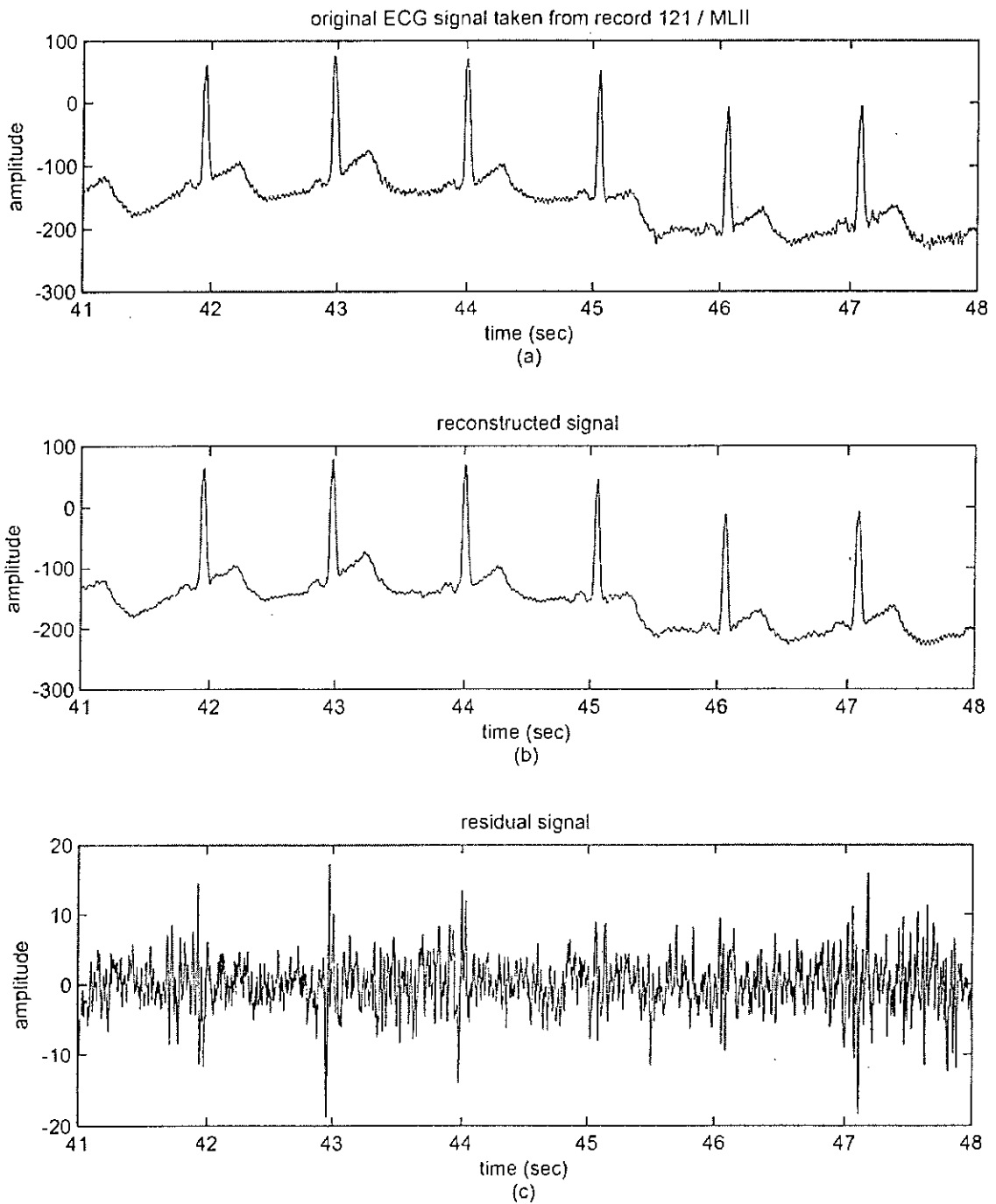


Figure 5.7 : (a) 7 second section of record 121 / MLII, (b) reconstructed signal for $PRD2 = 6.5\%$, $PRD = 1.9\%$, $CR = 49.5$, $CDR = 80$ bps, (c) residual signal.

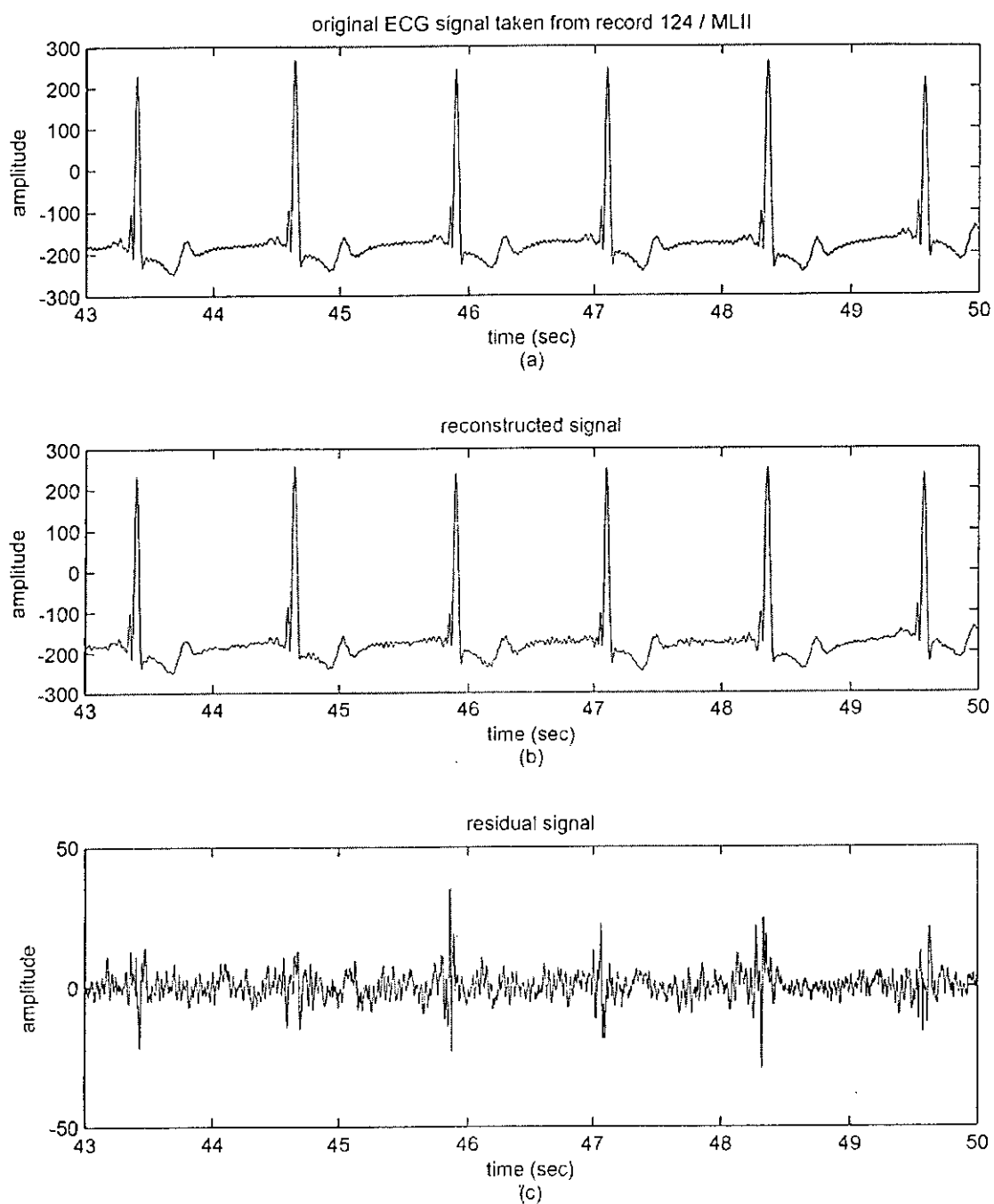


Figure 5.8 : (a) 7 second section of record 124 / MLII, (b) reconstructed signal for PRD2 = 6.5%, PRD = 2.7%, CR = 40.4, CDR = 98 bps, (c) residual signal.

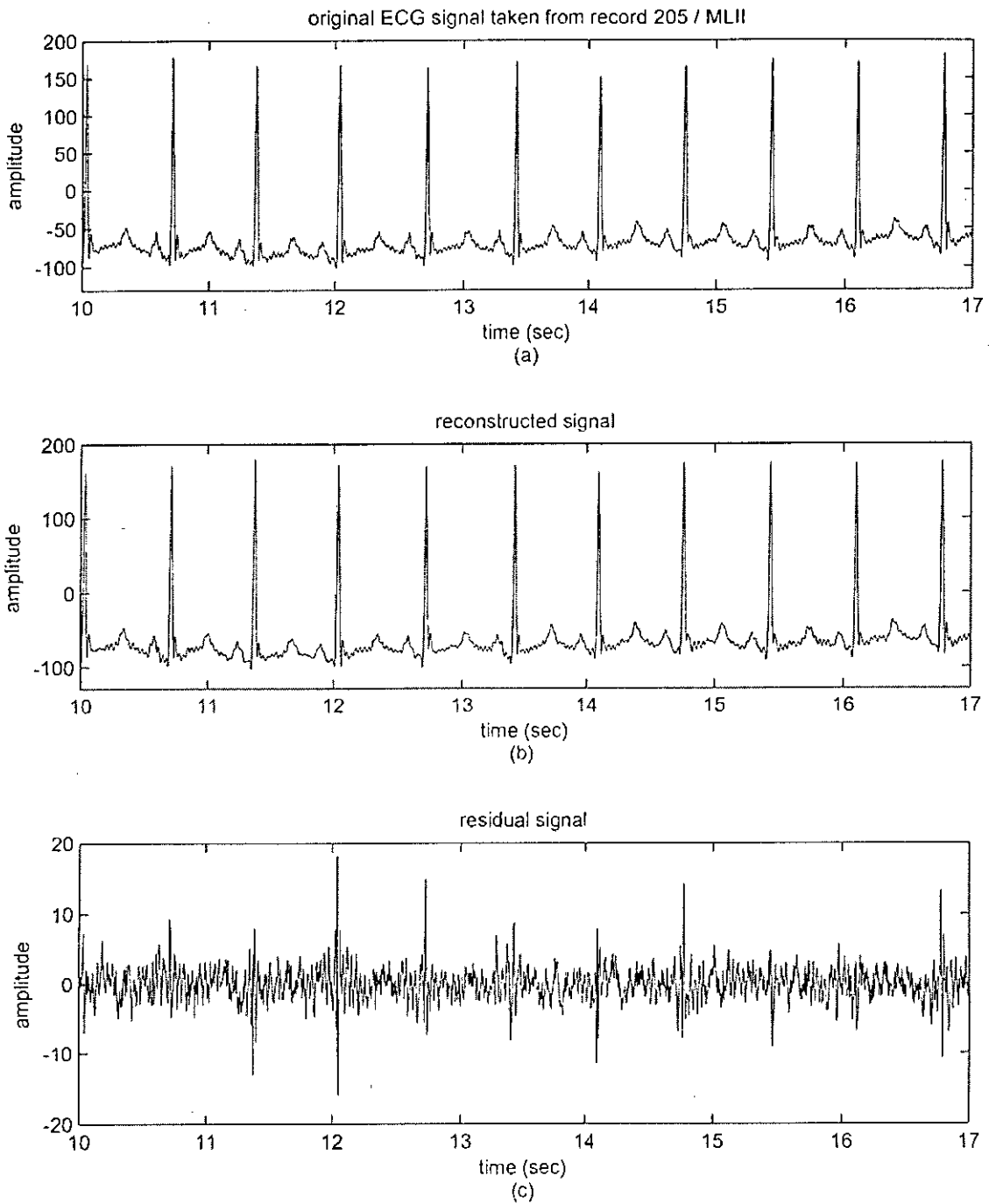


Figure 5.9 : (a) 7 second section of record 205 / MLI, (b) reconstructed signal for PRD2 = 6.5%, PRD = 3.1%, CR = 30.5, CDR = 130 bps, (c) residual signal.

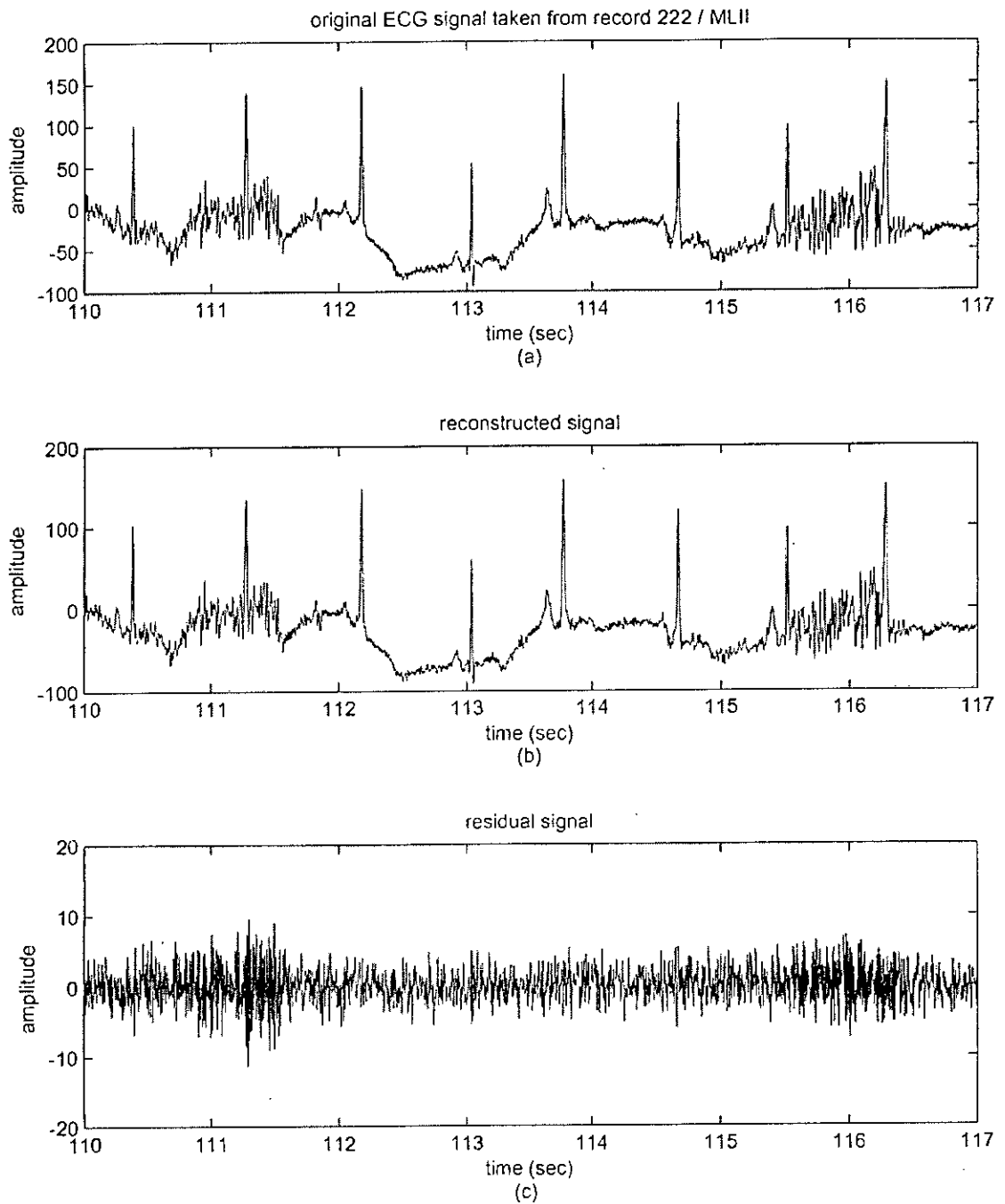


Figure 5.10 : (a) 7 second section of record 222 / MLII, (b) reconstructed signal for $PRD_2 = 6.5\%$, $PRD = 5.1\%$, $CR = 11.1$, $CDR = 357$ bps, (c) residual signal.

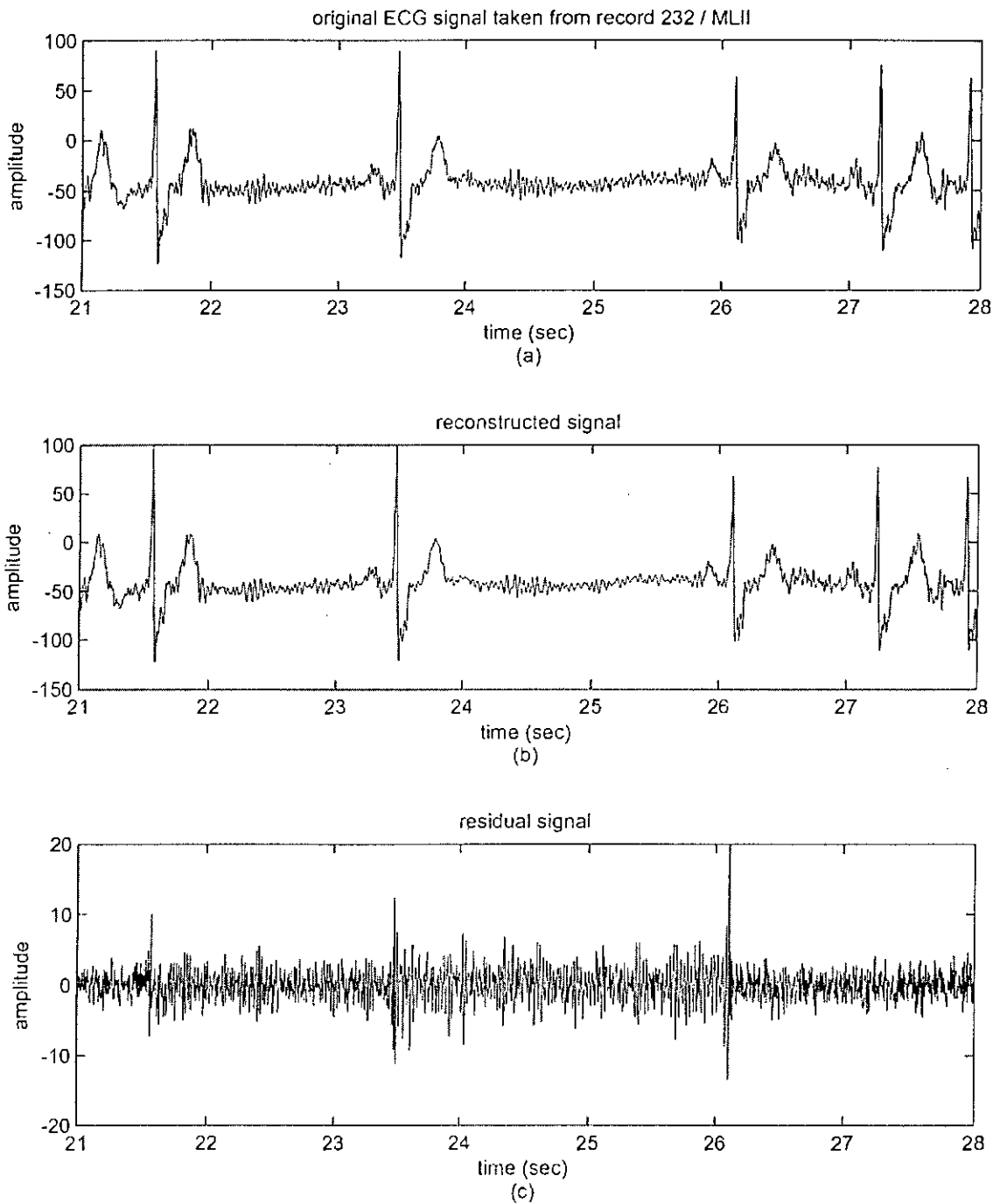


Figure 5.11 : (a) 7 second section of record 232 / MLII, (b) reconstructed signal for PRD2 = 6.5%, PRD = 4.0%, CR = 7.2, CDR = 551 bps, (c) residual signal.

5.3 Comparison with Other Methods

Table 5.3 presents the comparison of the performance of the proposed method with some well known ECG compressors. The PRDs and CRs are obtained from the corresponding literatures. Fixing the CRs, the PRDs of those methods and the proposed method are compared in this table. Table 5.4 compares the CRs for fixed PRDs. We could not compare our method with other methods for high PRDs as we designed our compressor for low PRD. Numbers marked by ^a are PRD2 values. It can be seen from these tables that our proposed method can compress ECG data better than the mentioned methods.

Table 5.3 : Comparison of the proposed method with other methods for fixed compression ratio

From literature Compressor	PRD (%)		Compression Ratio
	Compressor	Proposed Compressor	
Optimized quantization of DCT coefficients [38]	1.5	1.2	6.2
	2.0	1.6	7.9
	2.5	2.0	9.3
	3.0	2.3	10.9
Mean-shape vector quantizer [40]	4.1	2.8	13.1
Wavelet compression by set partitioning in hierarchical trees [42]	1.2	0.8	4.0
	3.0	2.1	10.0
Peak selection and DCT [43]	3.0	1.0	5.3
Sub-band compressor/ F16B FIR filter [41]	2.8 ^a	2.2 ^a	7.3
AZTEC [6]	15.5 ^a	2.0 ^a	6.9
Scan-along polygonal approximation [6]	9.6 ^a	2.0 ^a	6.9
Long term prediction [6]	7.3 ^a	2.0 ^a	6.9
Analysis by synthesis compressor/ASEC _{PRD} [6]	4.0 ^a	2.0 ^a	6.9
Gold washing adaptive vector quantization /wavelet transform [44]	3.3 ^a	1.4 ^a	4.6
	6.3 ^a	2.8 ^a	9.4
	8.2 ^a	3.8 ^a	12.4

Table 5.4 : Comparison of the proposed method with other methods for fixed PRD

From literature Compressor	PRD (%)	Compression Ratio	
		Compressor	Proposed Compressor
Optimized quantization of DCT coefficients [38]	1.5	6.2	7.4
	2.0	7.9	9.4
	2.5	9.3	12.2
	3.0	10.9	13.7
Mean-shape vector quantizer [40]	4.1	13.1	16.7
Wavelet compression by set partitioning in hierarchical trees [42]	1.2	4.0	6.0
	3.0	10.0	13.7
Peak selection and DCT [43]	3.0	5.3	13.7
Sub-band compressor/ F16B FIR filter [41]	2.8 ^a	7.3	9.4
Long term prediction [6]	7.3 ^a	6.9	22.4
Analysis by synthesis compressor/ASEC _{PRD} [6]	4.0 ^a	6.9	12.8
Gold washing adaptive vector quantization /wavelet transform [44]	3.3 ^a	4.6	11.0
	6.3 ^a	9.4	19.4

^a PRD2

Chapter 6

Conclusion

6.1 Findings

In this thesis, we presented an effective ECG compression method based on discrete cosine transform of the deviation of the ECG beats from a standard reference beat. The residual beats contain less variations than the period normalized dc removed beats. So when we applied DCT on those residual beats, the magnitude of the high frequency DCT coefficients became smaller. DCT concentrated the energy of the residual beats in the first few low frequency coefficients. We applied carefully designed quantization strategy to quantize the DCT coefficients. It minimized the entropy of the quantized coefficients at any target distortion. This kind of strategy has been applied before mainly to image compression. Our results show that this quantization strategy has been successfully adopted for ECG signals. The strategy adopted here defines a unique instance of the q and t vectors for each signal. In our work we generated optimum quantization and threshold vectors for small PRD (max 3.0% PRD and 6.5% PRD2).

To keep the quantized coefficients in a defined shorter range, we resized them. It was helpful for lossless arithmetic encoding. After removing the last consecutive zeros from all the quantized blocks, we joined them in a single vector and the zeros inside the vector were reduced by run length encoding. The vector was then arithmetic encoded. These operations helped us to compress the data significantly. The side informations were also kept in some vectors which were also resized. It was useful to allocate less number of bits for the elements of those vectors thereby improve the compression ratio.

For the evaluation of performance of our proposed method, we ran our compressor over the first two minutes of channel MLII of 40 records of the MIT-BIH Arrhythmia Database. Results show that it is capable of achieving good CR values with low distortion. We presented 7-second sections of six records for visual assessment of the

quality of the reconstruction at 6.5% PRD2. These traces present different characteristics and indicate an excellent preservation of all important signal features.

We also compared our method with some other well known ECG compressors. Our method compared favourably with the other compressors, producing considerably small PRD (or PRD2) for a given CR. Since a very simple coding scheme was employed in the last step of our algorithm, it is clear that the good CR×PRD compromise achieved is mainly due to the careful quantization strategy adopted. Taking large number of coefficients into consideration (first 170 from 270) for quantization for the compression process also helped us to get better results.

Although the method performs better, the main disadvantage is, it cannot be implemented for online ECG data compression. The reasons are given here. For generating the standard reference signal and residual signals, the whole ECG signal must be partitioned at first. The optimum quantization and threshold vectors are defined after applying DCT to all the residual signals and then those DCT coefficient blocks are thresholded and quantized. It is impossible to follow this procedure in online compression technique. Resizing the values of the quantized blocks, removing the last consecutive zeros, run length encoding of the inner zeros and then arithmetic encoding – all these tasks must be done in offline operation. Being an offline technique, it requires much time to compress a long duration of ECG signal.

Another disadvantage of this method is that it compress only the signal between the first and last identified QRS complexes. Rest of the signals cannot be compressed as they are not classified as beats. If any QRS complex is not identified, the resulting beat contains more samples of data, and the reconstructed beat becomes smoother than other beats for limited number of DCT coefficients.

6.2 Future Perspective

We can extend our technique in 2-D DCT for future continuation. Combining the quantization strategy used here with a 2-D approach similar to that used in CAB/2-D DCT might result in good compression gains as there exists a good correlation between the adjacent samples and adjacent beats. Taking a fixed number of period normalized beats, 2-D DCT can be applied on it, and after zigzag scanning the DCT coefficients, the optimum quantization and threshold vector can be generated as we have done in this thesis. The quantized coefficients can be encoded in the similar way as shown here.

Appendix-A : QRS Detection Algorithm

An ECG beat is defined as the signal sample from one *R*-wave to the next. Figure A.1 shows the block diagram of the QRS detection algorithm.

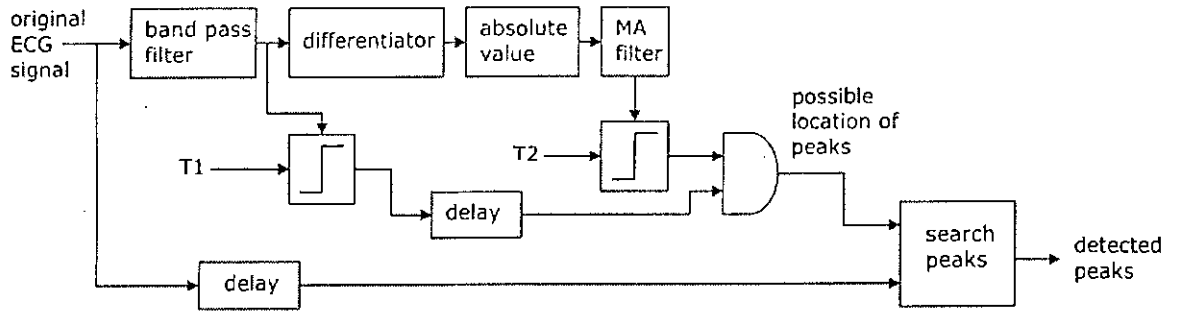


Figure A.1: Block diagram of QRS detection algorithm

At first the ECG signal is passed through a linear phase bandpass filter (4 hz to 40 hz) for smoothing operation and reducing base line shifting. The impulse response, magnitude response and the phase response of the bandpass filter is shown in the figure A.2 and A.3. The group delay of the filter is 150.

Differentiation of the filtered signal provides the slope information of *QRS* complexes. Since there are quick rise and fall times of the *QRS* complex in the ECG signals, the derivative makes it easier to detect the time of occurrence of the *QRS* complexes. The transfer function of the five-point differentiation equation is given by

$$H(z) = \left(\frac{1}{8}\right)(-z^{-2} - 2z^{-1} + 2z^1 + z^2) \quad (\text{A.1})$$

The absolute value of the output of derivative filter can be found by the following operation

$$y(n) = \sqrt{x(n)^2} \quad (\text{A.2})$$

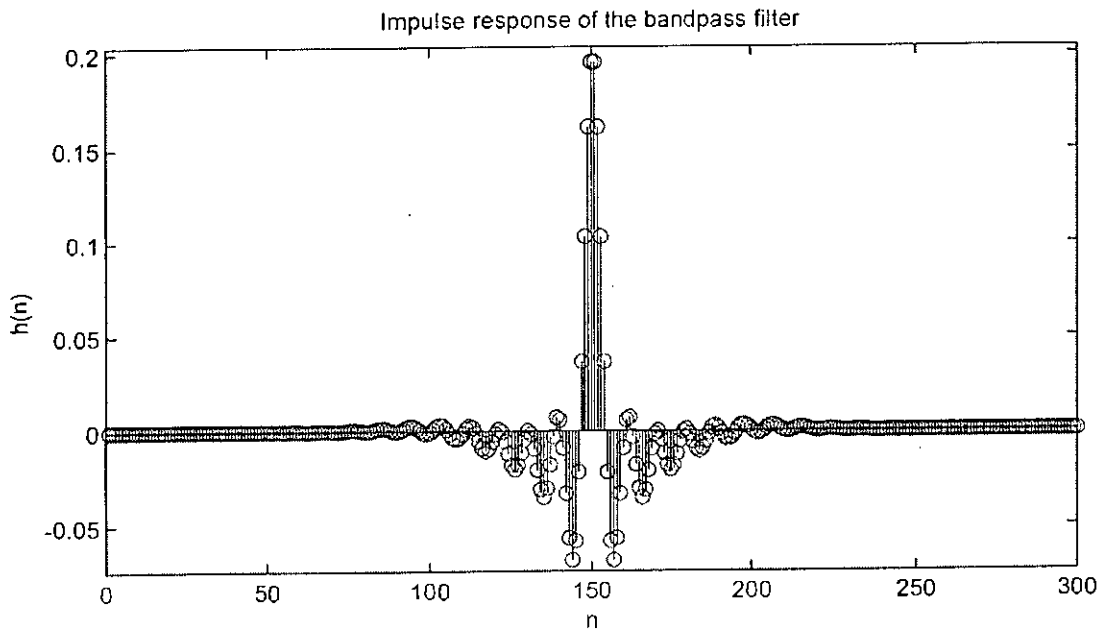


Figure A.2 : Impulse response of the bandpass filter

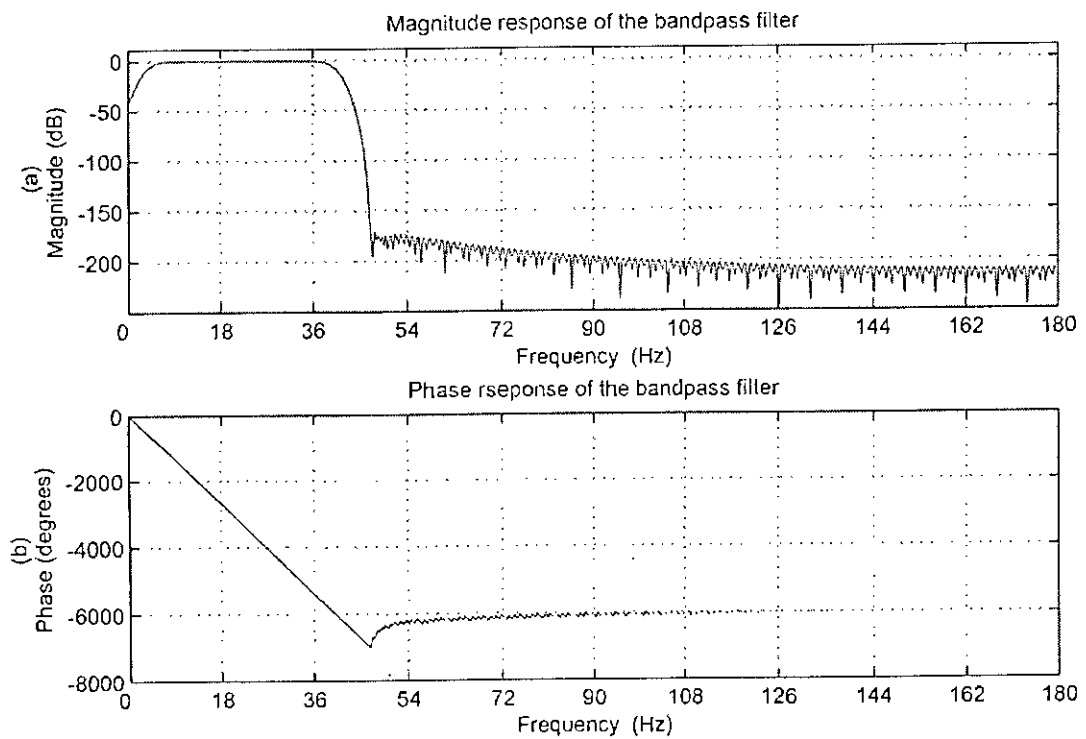


Figure A.3 : (a)The magnitude and (b)phase response of the bandpass filter

This absolute valued signal is then passed through a moving average filter which produces high value at the region of *QRS* complex. The window size has to be taken properly, neither so wide that merges the *QRS* complex and *T* wave together, nor so narrow that produces several peaks in the integration waveform. It is calculated from the equation below

$$y(n) = \left(\frac{1}{N} \right) [x(n-(N-1)) + x(n-(N-2)) + \dots + x(n)] \quad (\text{A.3})$$

where N is the width of the integration window. This thesis takes N as 18.

As seen in figure A.1, the algorithm sets two thresholds T_1 and T_2 to make decisions. T_1 is set 40 for the filtered ECG, and T_2 is set 7 for the signals produced by the moving window integration. The thresholded filtered signal is then delayed by 10 samples and a logical 'and' operation is performed with the thresholded moving squared average signal. As a result, possible location of peaks of the original ECG signal is found. These locations are delayed by 160 samples from the original ECG. Searching in these regions, we get the peaks. If there are more than one peak in the vicinity of 50 samples, the highest peak is considered from those.

The sequences of the *QRS* detection algorithm is shown in figure A.4 where the ECG signal is taken from the record 106/MLII of MIT-BIH arrhythmia database. Figures A.5 to A.13 shows parts of ECG signals and their detected peaks of channel MLII of records 107, 115, 117, 121, 124, 215, 220, 221 and 233 of MIT-BIH arrhythmia database.

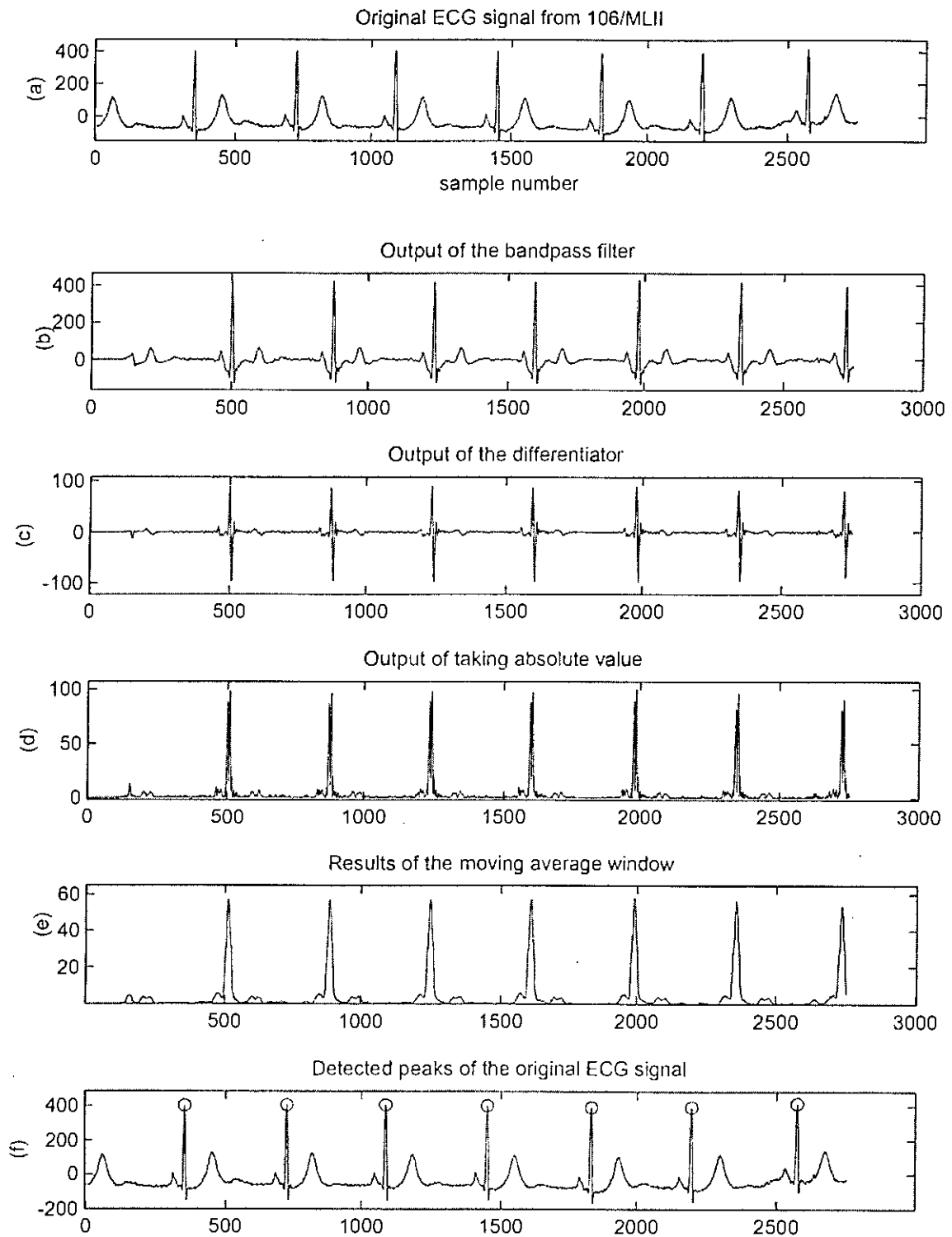


Figure A.4 : The sequences of peak detection of the ECG signal taken from the record 106 (MLII) of MIT-BIH arrhythmia database.

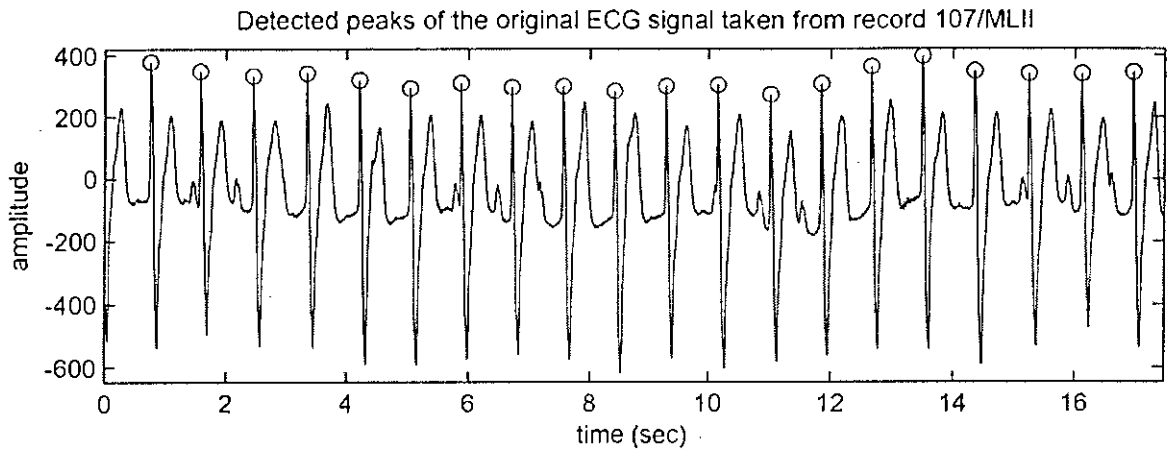


Figure A.5: Detected peaks of the original ECG signal taken from record 107/MLII

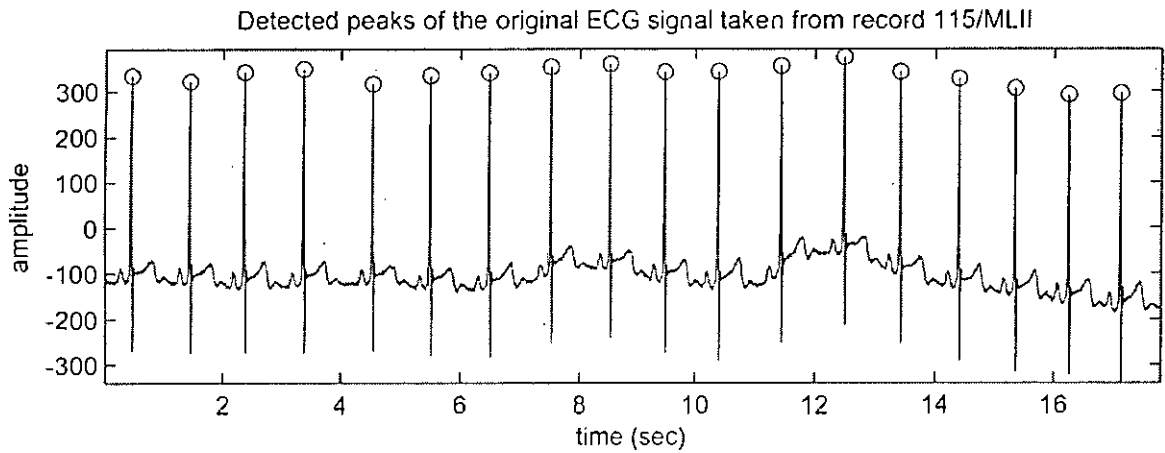


Figure A.6: Detected peaks of the original ECG signal taken from record 115/MLII

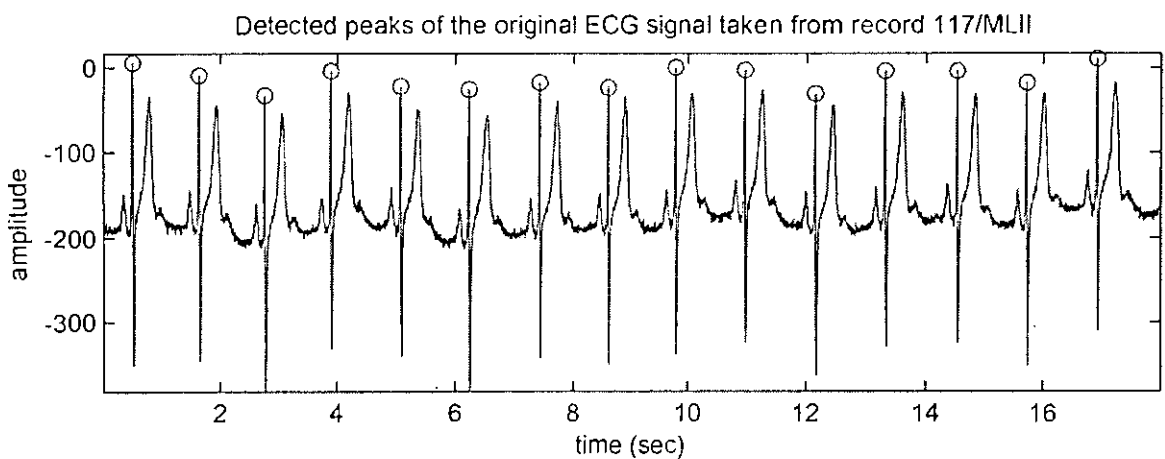


Figure A.7: Detected peaks of the original ECG signal taken from record 117/MLII

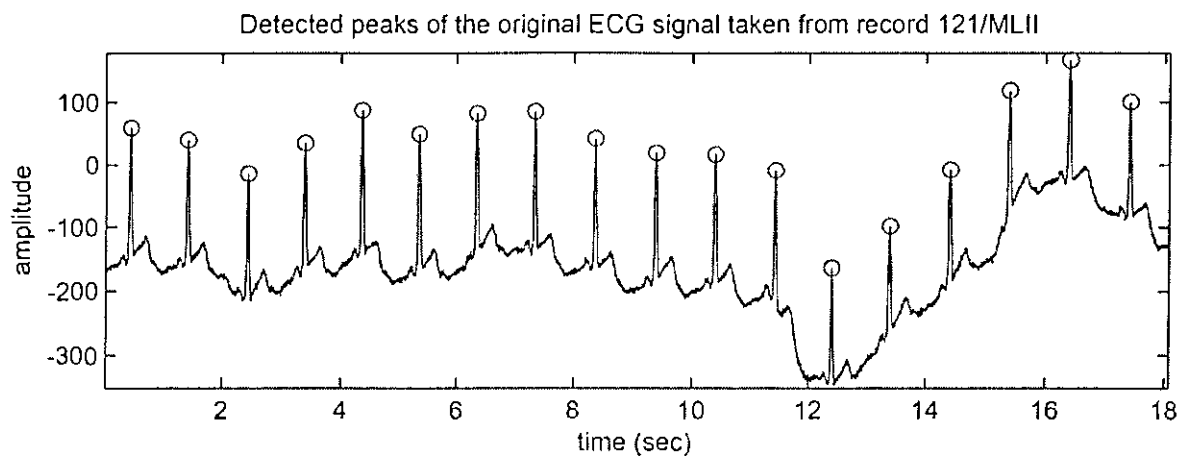


Figure A.8: Detected peaks of the original ECG signal taken from record 121/MLII

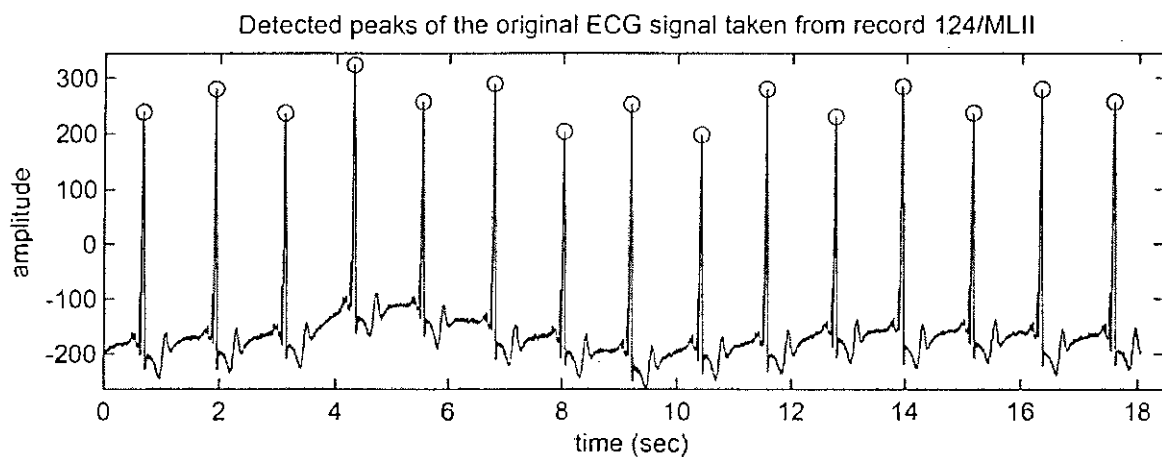


Figure A.9: Detected peaks of the original ECG signal taken from record 124/MLII

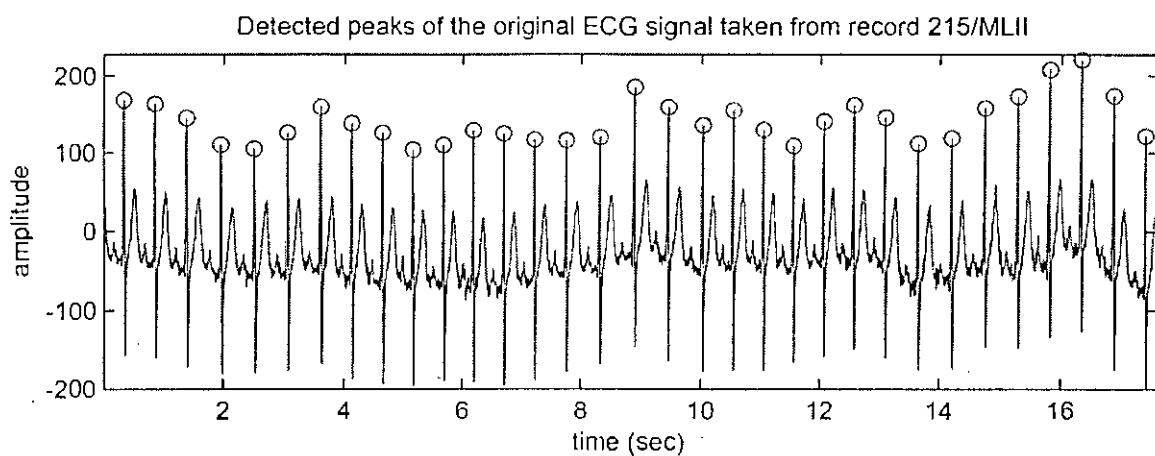


Figure A.10: Detected peaks of the original ECG signal taken from record 215/MLII

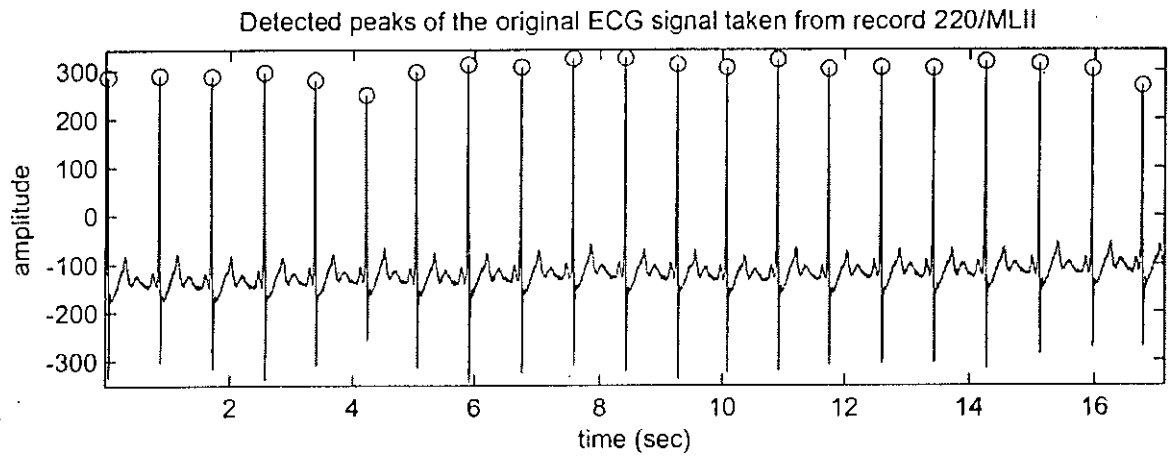


Figure A.11: Detected peaks of the original ECG signal taken from record 220/MLII

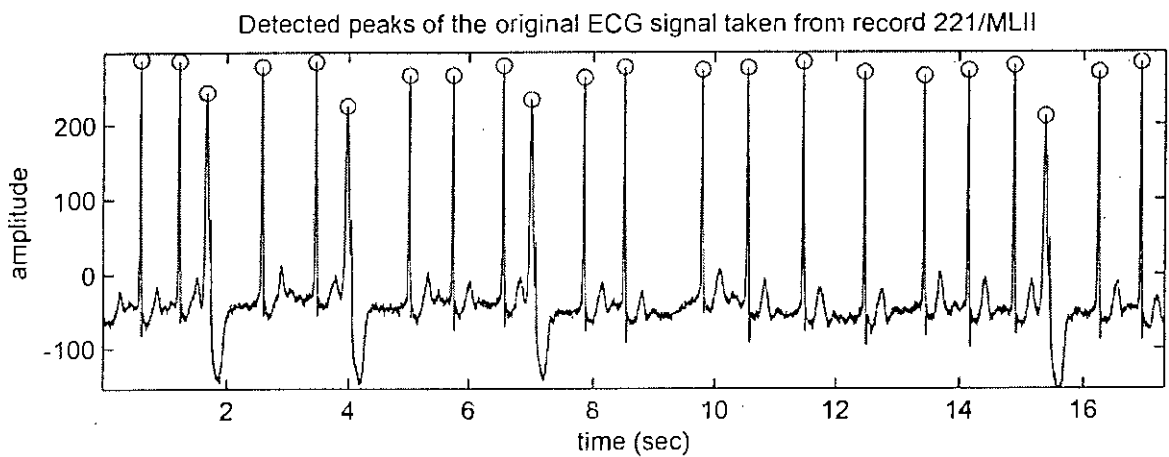


Figure A.12: Detected peaks of the original ECG signal taken from record 221/MLII

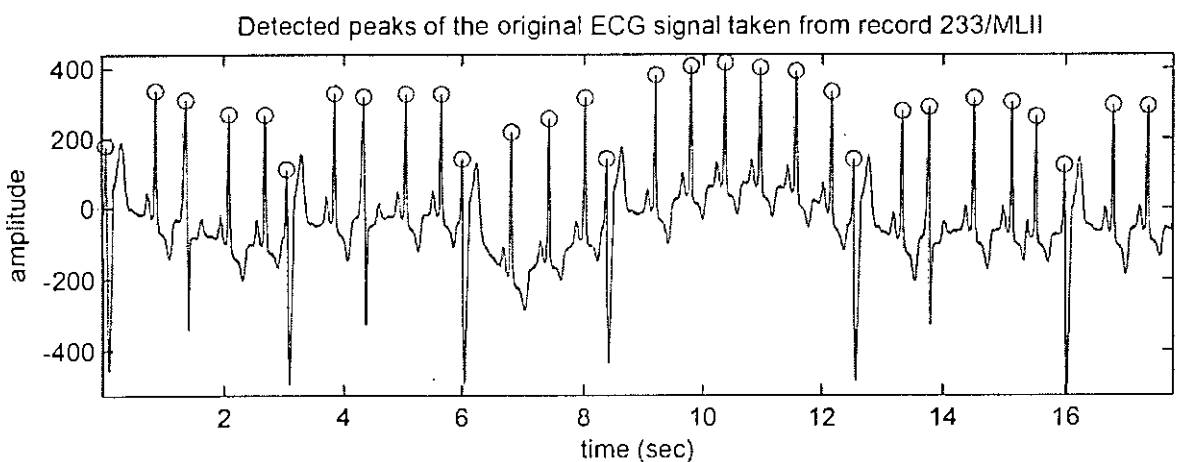


Figure A.13: Detected peaks of the original ECG signal taken from record 233/MLII

Appendix-B: MATLAB Programs

FILE_READ.m

```
clc
clear
PATH= 'D:\Saiful\ECG_CD\cd\mitdb'; % path, where data are saved
ATTRFILE= '100.atr';           % attributes-file in binary format
HEADERFILE= '100.heg';         % header-file in text format
DATAFILE= '100.dat';          % data-file
SAMPLES2READ=75600;           % 2.1 minute
% number of samples to be read% in case of more than one signal: 2*SAMPLES2READ
samples are read

%----- LOAD HEADER DATA -----
fprintf(1, '\n$> WORKING ON %s ...\n', HEADERFILE);
signalh= fullfile(PATH, HEADERFILE);
fid1=fopen(signalh,'r');
z= fgetl(fid1);
A= sscanf(z, '%*s %d %d %d', [1,3]);
nosig= A(1);                   % number of signals
sfreq=A(2);                    % sample rate of data
clear A;
for k=1:nosig
    z= fgetl(fid1);
    A= sscanf(z, '%*s %d %d %d %d %d', [1,5]);
    dformat(k)= A(1);          % format; here only 212 is allowed
    gain(k)= A(2);             % number of integers per mV
    bitres(k)= A(3);          % bitresolution
    zerovalue(k)= A(4);       % integer value of ECG zero point
    firstvalue(k)= A(5);      % first integer value of signal
                                %(to test for errors)
end;
fclose(fid1);
clear A;
```

```

%----- LOAD BINARY DATA -----
if dformat~= [212,212],
    error('this script does not apply binary formats different to 212. ');
end;
signald= fullfile(PATH, DATAFILE);    % data in format 212
fid2=fopen(signald,'r');
A= fread(fid2, [3, SAMPLES2READ], 'uint8');
                                % matrix with 3 rows, each 8 bits long, = 2*12bit
fclose(fid2);
M2H= bitshift(A(:,2), -4);
M1H= bitand(A(:,2), 15);
PRL=bitshift(bitand(A(:,2),8),9);    % sign-bit
PRR=bitshift(bitand(A(:,2),128),5);  % sign-bit
M( : , 1)= bitshift(M1H,8)+ A(:,1)-PRL;
M( : , 2)= bitshift(M2H,8)+ A(:,3)-PRR;
if M(1,:) ~= firstvalue,
    error('inconsistency in the first bit values!');
end;
switch nosig
case 2
    M( : , 1)= (M( : , 1)- zerovalue(1));
    M( : , 2)= (M( : , 2)- zerovalue(2));
    TIME=(0:(SAMPLES2READ-1))/sfreq ;
case 1
    M( : , 1)= (M( : , 1)- zerovalue(1));
    M( : , 2)= (M( : , 2)- zerovalue(1));
    M=M';
    M(1)=[];
    sM=size(M);
    sM=sM(2)+1;
    M(sM)=0;
    M=M';
    TIME=(0:2*(SAMPLES2READ)-1)/sfreq;
otherwise
    % this case did not appear up to now!
    % here M has to be sorted!!!

    disp('Sorting algorithm for more than 2 signals not programmed yet!');
end;
clear A M1H M2H PRR PRL;
fprintf(1,'\n$> LOADING DATA FINISHED \n');
P=M(:,1);
Q=M(:,2);

```

```

%-----
Y=P';      % ECG data
%-----
plot(Y);
clear DATAFILE HEADERFILE M P PATH Q SAMPLES2READ...
    TIME ans bitres dformat fid1 fid2 firstvalue ...
    gain k nosig sfreq signald signalh z zerovaluel
save('w_Y.mat');
%%% NEXT--> s01.m

```

s01.m

```

clear all;
clc;
load w_Y;
Lb=270;
%-----
[b,Nb,avgl]=PARTITIONING(Y,75600);      %75600 samples from Y
%-----
R=round(sum(b)/(Nb));                    %%% R: Standard Reference
clear RR ;
RR= repmat(R,Nb,1);
b1=b-RR ;
B=dct(b1)';
%-----
hold on;
grid on;
plot(b','b') ;
plot(b1','g');
plot (R,'r');
%-----
clear R RR Y avgl b1
save ('w_s01.mat' );
%%% NEXT ---> Entropy_and_Distortion.m

```

Entropy_and_Distortion.m

```

clear all
load w_s01;
Lb1=170;

```

```

Qmax=32;
Tmax=32;
Tincr=0.25;
T=length( 0.5 : Tincr : Tmax );

D=20*ones(Qmax,T,Lb1);
H=20*ones(Qmax,T,Lb1);
%-----
for k=1:Lb1
    for q=1:Qmax
        c(1,:)=B(:,k)';
        l=0;
        %%% D(q,t,k) H(q,t,k)
        for t = q/2 : Tincr : Tmax,
            l=l+1;
            c1=abs(c);
            c2=find(c1<t);
            c(c2)=0;
            ch=round(c/q);
            CC=( B(:,k)-q*ch(1,:) );
            CCC=CC.*CC;
            D(q,l,k)=(ones(1,Nb)*CCC)/Nb;

            ch1=sort(ch);
            H(q,l,k)=0;
            s=1;
            while( s~=Nb+1 )
                a=ch1(s);
                Nv=length(find(ch1(s:Nb)==a));
                Pv=Nv/Nb;
                H(q,l,k)=H(q,l,k)-Pv*log2(Pv);
                s=s+Nv;
            end
        end
    end
end
clear B T fid1 fid2 i k l n r1 c1 sum x Pv flag T a ch ch1 c Pv Nv n s r1 c1 k q t i CC CCC c2;
save ('w_Entropy_and_Distortion.mat');

%NEXT--->s02.m

```

s02.m

```

clc;
clear all
load w_Entropy_and_Distortion;
Arr2=zeros(2,5,Lb1);

for k=2:Lb1,
    x=0;
    for q=1:Qmax,
        l=0;
        for t= q/2 : Tincr : Tmax,
            x=x+1;
            l=l+1;
            Array(1,x)=H(q,l,k); % x
            Array(2,x)=D(q,l,k); % y
        end %%%t
    end %%%q
    j = convhull( Array(1,:), Array(2,:) );
        %%% x=Array(1,:); % H
        %%% y=Array(2,:); % D
        %%% j = convhull( x, y); % convhull( H, D);

    XX(1,:)=Array(1,j); % H
    XX(2,:)=Array(2,j); % D
    plot(Array(1,:),Array(2,:),'b+'); grid on;

    XX=XX';
    XX=sortrows(XX);
    hold on;

    clear Array;
    Array(:,1)=XX(:,2);
    Array(:,2)=XX(:,1);

    M=Array(1,2); %%% lowest entropy.
    L=length(find(Array(:,2)==M)); %%% number of lowest entropy.
    x=length(Array');
    Dis=min(Array(1:L,1)); %%% lowest distortion for lowest entropy.
    P=min(find(Array(1:L,1)==Dis)); %%% lowest position of lowest
        %%% distortion for lowest entropy.

```

```

refslope=0;
change=1;
y=1;                                %% lowest entropy position.
stnumber=y+1;                        %% starting point from which slope will be calculated.
n4=1;
Arr2(1,n4,k)=Dis;                    %% Arr2 is the array of selected points .
Arr2(2,n4,k)=Array(y,2);            %% H

while( (stnumber~=x+1)&(change==1) ),
    change=0;
    for n3=stnumber:x,
        if Array(y,2)~=Array(n3,2),
            slope_yn3=( Array(y,1)-Array(n3,1) )/( Array(y,2)-Array(n3,2) ) ;
            if(slope_yn3<refslope),
                refslope=slope_yn3;
                x1=n3;
                change=1;
            end
        end
    end
    end %n3
    if change==1,
        n4=n4+1;
        y=x1;
        Arr2(1,n4,k)=Array(y,1);      %% D
        Arr2(2,n4,k)=Array(y,2);      %% H
        refslope=0;
        stnumber=y+1;
    end
end %while
plot(Arr2(2,:,k),Arr2(1,:,k),'r-',Arr2(2,:,k),Arr2(1,:,k),'ro');

clear Array XXX XX;
end %k
clear Array Dis L M P change j k l n3 n4 ...
    q refslope slope_yn3 stnumber t x x1 y ;
save ('w_s02.mat');
%%%%%% NEXT--->s03.m

```


s03.m (optimum q, t and Encoding)

```

clear
load w_s02;

%TD=1.37 % PRD=1.5
%TD=3.6 % PRD=2
%TD=6.52 % PRD=2.5
%TD=10.25 % PRD=3
%TD=13.85 % PRD=3.5

TarD=TD*ones(1,Lb1);
clear Arr3;

for k=1:Lb1,
    Arr3(:,k)=Arr2(:,k)';
end
%-----
for k=2:Lb1

    L=length(Arr3(:,1,k));
    P=max(find(Arr3(:,2,k)==max(Arr3(:,2,k))));

    if (TarD(k)>=Arr3(1,1,k))
        lamda(k)=(Arr3(1,1,k)-Arr3(2,1,k))/(Arr3(1,2,k)-Arr3(2,2,k))-50;

    elseif (TarD(k)<Arr3(P,1,k))
        lamda(k)=0;

    else
        for n=1:L-1
            if (Arr3(n,1,k)>TarD(k))&(TarD(k)>=Arr3(n+1,1,k))
                lamda(k)=(Arr3(n,1,k)-Arr3(n+1,1,k))/(Arr3(n,2,k)-Arr3(n+1,2,k));
            end
        end
    end

end

clear n k P L Arr3 ;
%-----
clear q0 t0 a;

```

```

q0=10*ones(1,Lb1);
t0=20*ones(1,Lb1);

tq(1,:)=t0;
tq(2,:)=q0;

for k=2:Lb1
    clear J ;
    x=0;
    for q=1:Qmax,
        l=0;
        for t= q/2 : Tincr : Tmax,
            l=l+1;
            x=x+1;
            J(1,x)=D(q,l,k)-lamda(k)*H(q,l,k);
            J(2,x)=q;
            J(3,x)=t;
        end        %%t
    end            %%q

    p=min(find(J(1,:)==min(J(1,:)))));
    q0(k)=J(2,p);
    t0(k)=J(3,p);
    a(k)=J(1,p);

    tq(1,k)=t0(k);
    tq(2,k)=q0(k);
end            %%k
%-----
clear k x l q t p J a K Q Arr2 Arr3 D H Nb TarD lamda ;
save ('w_q0t0.mat');

%%%% NEXT---> s04.m

```

s04.m

```

clear all;
load w_Y;
load w_q0t0;
%-----

```

```

Y1=Y(1:75600);          %% 3.5 minutes

[b,Nb,s1,C61,y2]=PARTITIONING_1(Y1);
% b :- normalized beats
% s1 :- DC coefficients
% y2 :- beat lengths
% Nb :- number of beats
% C61:- sample number of starting of the first beat
%-----

R=round(sum(b)/(Nb));   %% R :- Reference Signal

RR= repmat(R,Nb,1);
b1=b-RR;               %% b1 :- residual signals
B=dct(b1)';           %% B :- DCT transform of residual signals
%-----

[Bh]=THRESHOLDING_and_QUANTIZATION(B,q0,t0,Lb1,Nb);
%% Bh :- Quantized 170 DCT coefficients
%-----

g=10;                 %% dc coefficients are quantized by 10
s1=round(s1/g);

Lb1=170;
Bh1=Bh(:,2:Lb1);

y6=REMOVING_TAIL_ZEROS(Lb1-1,Nb,Bh1);
%% y6 contains 23.5 , last consecutive 0s are replaced by 23.5

n=find(y6==23.5);

maxall=29;
n1=find(y6<=-maxall); %% n1= find(y6 <= -30)
ZZ1=y6(n1)+maxall;    %% y6(n1) + 29 %% values < -29
n2=find(y6>maxall);  %% find(y6 >= 30) %% also for 23.5
%% n2 includes n
ZZ2=y6(n2)-maxall;   %% y6(n1) - 29 %% values > 29

y6(n2)=(maxall+1);   %% 30
y6(n1)=-(maxall+1); %% -30

```

```

y6(n)=-31;          %%-31 for replacing last consecutive 0s.
ZZ1=-1*ZZ1;
%----- run length encoding of zeros -----
cz=10;              %% cz :- max allowable consecutive zeros
[y6,A11]=RUN_LENGTH_ENCODING(y6,cz);
                    %% A11 :- number of consecutive zeros > 10
[SA11]=SEG( A11-10,31);
[SZZ1]=SEG( ZZ1,31);
[SZZ2]=SEG( ZZ2,31);

%----- arithmetic coding -----
[comp,counts_y6,L_y6]=ARITHMATIC_ENCODING(y6);

clear RR Y b b1 A11 B Bh Bh1 LA11 LPpr LZZ1 LZZ2 Lb Lb1 ...
      Ppr Qmax TD Tincr Tmax YYYYY ZZ1 ZZ2 cz ...
      g maxall n n1 n2 t0 tq y6 y6_1

save w_compression.mat

```

s05.m (Decoding)

```

clear all;
load w_compression;
y6r=arithdeco(comp,counts_y6,L_y6)-32;

A11r=invSEG(SA11,31)+10;          %% num of consecutive zeros > 10
ZZ1r=invSEG(SZZ1,31);            %% values < -29
ZZ2r=invSEG(SZZ2,31);            %% values > 29

cz=10;
[y6r]=RUN_LENGTH_DECODING(y6r,A11r,cz);

ZZ1r= -ZZ1r;

nr=find(y6r==-31);
n1r=find(y6r==-30);
n2r=find(y6r==30);

y6r(n1r)=y6r(n1r)+ZZ1r +1;       %% for -30
y6r(n2r)=y6r(n2r)+ZZ2r -1;       %% for +30

```

```
y6r(nr)=23.5;          %% for -31
```

```
Lb1=170;
Bh1r=JOINING_TAIL_ZEROS(Lb1-1,Nb,y6r);
Bh1r=[zeros(Nb,1),Bh1r];
```

```
Lb=270;
Bhr=zeros(Nb,Lb);
```

```
Bhr(:,1:Lb1)=Bh1r;
```

```
for k=1:Lb1,
    Bhr(:,k)=q0(k)*Bhr(:,k);
end
```

```
Br=Bhr;
RR=repmat(R,Nb,1);
br=idct(Br)';
br=br+RR;
br=dct(br)' ;
```

```
c6r=cumsum([C61,y2]);
```

```
g=10;
s1=s1*g;
s1r=s1;
%-----
```

```
c11=Nb+1;
c10=650 ;
s2r=zeros(c11-1,c10) ;
```

```
s2r(:,1:Lb)=br(:,:);
s2r(:,1)=s1r(1,:)';
```

```
for x=1:c11-1 ,
    Ypr(c6r(x):c6r(x+1)-1)= idct(s2r(x,1:y2(x)));
end
```

```
LENGTH=c6r(length(c6r))-C61;
```

```
[PRD,PRD2]=PERCENT_ROOT_MEAN_SQUARE_DIST(Ypr,C61,LENGTH,Y1)
```

```
Yp1r=Ypr(C61:C61+LENGTH-2);
```

```
Y1r=Y1(C61:C61+LENGTH-2);
```

```
LSZZ1=length(SZZ1);
```

```
LSZZ2=length(SZZ2);
```

```
LSA11=length(SA11);
```

```
Ls1=length(s1);
```

```
Ly2=length(y2);
```

```
%----- Compression Ratio -----
```

```
CR=LENGTH*11/( length(comp) +Ls1*8 +Ly2*10 + LSZZ1*5 +LSZZ2*5 +LSA11*5)
```

```
%----- Compressed Data Rate -----
```

```
CDR= ( length(comp) +Ls1*8 +Ly2*10 + LSZZ1*5 +LSZZ2*5 +LSA11*5)
      /(length(Yp1r)/360)
```

PARTITIONING.m

```
function [b,Nb,avgl]=PARTITIONING(Y, n_samples)
```

```
M1=300;
```

```
As=80;
```

```
n=0:(M1-1);
```

```
wc1=0.0222*pi; % 4 hz
```

```
wc2=0.2222*pi;% 40 hz
```

```
hd=ideal_lp(wc2,M1)-ideal_lp(wc1,M1);
```

```
beta=0.1102*(As-8.7)+10;
```

```
w_kai=kaiser(M1,beta)';
```

```
h=hd.*w_kai;
```

```
Yfilt=filter(h,1,Y);
```

```
clear As beta db grd mag n pha w w_kai wc1 wc2 hd
```

```

%----- Getting the derivatives -----
A=[1 2 0 -2 -1];
A=A./8;
B=[1];
ydev=filter(A,B,Yfilt);

%----- Getting the absolute value -----
ydev_abs = abs(ydev);

%----- moving average filtering -----
N=18;
A=ones(1,N)./N;
B=1;
mov_avrg_sqr=filter(A,B,ydev_abs);

thresh_mov_avgr_sqr=mov_avrg_sqr;
thresh_mov_avgr_sqr(find(thresh_mov_avgr_sqr<7))=0;

thresh_Yfilt=Yfilt;
thresh_Yfilt(find(thresh_Yfilt<40))=0;

A=[zeros(1,10),thresh_Yfilt]&[thresh_mov_avgr_sqr,zeros(1,10)];

possible_peak_positions=[A,zeros(1,150)];
Y1=[zeros(1,10+150),Y];

[a7]= findpeaks(possible_peak_positions,Y1);
c6=a7-10-150;
BB=c6;
if c6(1)<0,
    c6=c6(2:length(BB));
end

%-----
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
                                %%% c6 : time of peaks
y2=diff(c6) ;                    %%% y2 : durations

c10=650 ;                        %%% c10: max duration
c11=length(c6) ;                 %%% c11: number of peaks
avgl=round(sum(y2)/length(y2)) ; %%% avgl: avg length
Lb=270;
b=zeros(c11-1,Lb) ;

```

```

s2=zeros(c11-1,c10) ;
for x=1:c11-1 ,
    s2(x,1:y2(x))=dct(Y(c6(x):c6(x+1)-1) ) ;
end
s2(:,1)=0 ;
b(:,:)=s2(:,1:Lb);

Nb=c11-1;
b=idct(b)';
%plot(b,'b') ;

```

PARTITIONING_1.m

```

function [b,Nb,s1,C61,y2]=PARTITIONING_1(Y)
Lb=270;

M1=300;
As=80;

n=0:(M1-1);
wc1=0.0222*pi; % 4 hz
wc2=0.2222*pi;% 40 hz

hd=ideal_lp(wc2,M1)-ideal_lp(wc1,M1);
beta=0.1102*(As-8.7)+10;

w_kai=kaiser(M1,beta)';
h=hd.*w_kai;

Yfilt=filter(h,1,Y);
clear As beta db grd mag n pha w_w_kai wc1 wc2 hd
%----- Getting the derivatives -----
A=[1 2 0 -2 -1];
A=A./8;
B=[1];
ydev=filter(A,B,Yfilt);

% ----- Getting the absolute value -----
ydev_abs = abs(ydev);

```



```

% ----- moving average filtering -----
N=18;
A=ones(1,N)./N;
B=1;
mov_avrg_sqr=filter(A,B,ydev_abs);

thresh_mov_avgr_sqr=mov_avrg_sqr;
thresh_mov_avgr_sqr(find(thresh_mov_avgr_sqr<7))=0;

thresh_Yfilt=Yfilt;
thresh_Yfilt(find(thresh_Yfilt<40))=0;

A=[zeros(1,10),thresh_Yfilt]&[thresh_mov_avgr_sqr,zeros(1,10)];

possible_peak_positions=[A,zeros(1,150)];
Y1=[zeros(1,10+150),Y];

[a7]= findpeaks(possible_peak_positions,Y1);
c6=a7-10-150;
BB=c6;
if c6(1)<0,
    c6=c6(2:length(BB));
end
%-----
%           %%% c6 : time of peaks
C61=c6(1);           %%% C61 :c6(1)
y2=diff(c6) ;       %%% y2 : durations
y22=[C61,y2];
c10=650 ;           %%% c10: max duration
c11=length(c6) ;    %%% c11: number of peaks
avgl=round(sum(y2)/length(y2)) ; %%% avgl: avg length

b=zeros(c11-1,Lb) ;
s3=y2-avgl;         %%% s3 : variation of durations
s2=zeros(c11-1,c10) ;
for x=1:c11-1 ,
    s2(x,1:y2(x))=dct(Y(c6(x):c6(x+1)-1) ) ;
end

s1(1,:)=s2(:,1)';
s1=round(s1);       %%% s1 : dc coefficients

```

```

s4=diff(s1);           %%% s4 : change of dc coefficients
S11=s1(1);           %%% S11 : first DC coeff
s44=[S11,s4];
s2(:,1)=0 ;
% s3(1:avgl)=s2(1:avgl);
b(:,:)=s2(:,1:Lb);

Nb=c11-1;

b=idct(b)';
plot(b,'b') ;

```

ideal_lp.m

```
function hd = ideal_lp(wc,M)
```

```

alpha=(M-1)/2;
n=0:(M-1);
m= n - alpha + eps;
hd=sin(wc*m)./(pi*m);

```

findpeaks.m

```
function[a8]= findpeaks(s,ss);
```

```

% s : peaks (possible location of peaks)
% ss : Y (original ECG signal)

```

```

a1=[diff(s)] ;
a2=find(a1==1)+1 ;
a3=find(a1==-1) ;

a4=length(a2);
for x=1:a4,
    a5(x)=max(ss(a2(x):a3(x)));
    a6(x)=max(find(ss(a2(x):a3(x))==a5(x)));
    a7(x)=a2(x)+a6(x)-1;
end
y=1;
a8(y)=a7(1);

```

```

for x=2:length(a7),
    if a7(x)-a8(y)<120,
        if ss(a7(x))>ss(a8(y)),
            a8(y)=a7(x);
        end
    else
        y=y+1;
        a8(y)=a7(x);
    end
end
end

```

THRESHOLDING_and_QUANTIZATION.m

```
function [Bh]=THRESHOLDING_and_QUANTIZATION(B,q0,t0,Lb,Nb)
```

```

for k=1:Lb
    Bh(:,k)=B(:,k)/q0(k);
end

```

```
% -----
```

```

for i=1:Nb
    for k=1:Lb
        if abs(B(i,k))<t0(k)
            Bh(i,k)=0;
        else
            Bh(i,k)=round(Bh(i,k));
        end        %%if
    end            %%k
end                %%i

```

REMOVING_TAIL_ZEROS.m

```
function [y] = REMOVING_TAIL_ZEROS(Lb,Nb,B)
```

```
A=reshape(B',Lb*Nb,1)';
```

```
n=1;
```

```
for x=1:Nb
```

```
    i=find(B(x,:));
```

```
    I=max({1,i});
```

```
    y(1,n:n+I)=[B(x,1:I),23.5];
```

```
    n=n+I+1;
```

```
end
```

JOINING_TAIL_ZEROS.m

```
function [b] = JOINING_TAIL_ZEROS(Lb,Nb,y)
```

```
p=[0,find(y==23.5)];
a=diff(p);
b=zeros(Nb,Lb);

for x=1:Nb,
    b(x,1:a(x)-1)=y(1,p(x)+1:p(x+1)-1);
end
```

RUN_LENGTH_ENCODING.m

```
function [y6,A11]= RUN_LENGTH_ENCODING(y6,cz);
```

```
%% here cz = 10
```

```
Fseq=y6;
L=length(Fseq);
A2=find(Fseq);
A1=zeros(1,L);

A1(A2)=1;
A3=[A1,1];
A4=[1,A1];
A5=A4-A3;
A6=find(A5==1);
A7=find(A5==-1);
A8=A7-A6;
A9=A8-cz;
A10=find(A9>=0);
A11=A8(A10);    %% Number of consecutive 0s >= cz 0s

A12=A6(A10)-1;
A13=A7(A10);
s=[1,A13];
e=[A12,L];

Ls=length(s);
LPpr=length(Fseq)-sum(A11)+length(A11);
Ppr=31*ones(1,LPpr);
```

```

en=-1;
for x=1:Ls,
    st=en+2;
    en=st+e(x)-s(x);
    Ppr(st:en)=Fseq(s(x):e(x));
end

clear y6;
y6=[Ppr]; % *** y6 : sequence after RLE

```

RUN_LENGTH_DECODING.m

```

function [y6_1]=RUN_LENGTH_DECODING(y6,A11,cz);

L=length(y6)+sum(A11)-length(A11);
Fseq=zeros(1,L);

y6=[1,y6,1];
Ly6=length(y6);

P1=zeros(1,L+2);

Z=find(y6==31);
s=[1,Z+1];
e=[Z-1,Ly6];
Ls=length(s);

st(1)=s(1);
en(1)=e(1);
P1(st(1):en(1))=y6(s(1):e(1));
for x=2:Ls,
    st(x)=en(x-1)+A11(x-1)+1;
    en(x)=st(x)+e(x)-s(x);
    P1(st(x):en(x))=y6(s(x):e(x));
end
Fseq(1,1:L)=P1(1,2:L+1);
y6_1=Fseq;

```

SEG.m

```

function [E]=SEG( A,max)
                                % here max = 31
B=fix(A/max);                   % number of max
C=rem(A,max) ;                  % remainder
F=find(B);
G=B(F);
H=C(F);

E=[0];
m=1;
for x=1:length(A),
    if A(x)>max-1,
        E=[ E , max*ones(1,G(m)) , H(m) ];
        m=m+1;
    else
        E=[ E , A(x) ];
    end
end
E=E( 2:length(E) );

```

invSEG.m

```

function [B]=invSEG(E,max)
B=find([0 0 0]==1);
if length(E)>=1,
    n=2; m=1; B=E(1);
    L=length(E);
    while n<=L,
        c=max;
        while (E(n-1)==max)&(n<=L)
            B(m)=c+E(n) ;
            c=B(m) ;
            n=n+1 ;
        end
        while (E(n-1)~=max)&(n<=L)
            m=m+1 ;
            B(m)=E(n) ;
            n=n+1 ;
        end
    end
end

```

```

    end
end

```

ARITHMATIC_ENCODING.m

```

function [comp,counts_y6,L_y6]=ARITHMATIC_ENCODING(y6)
y6=y6+31+1;
L_y6=length(y6);
el_y6=1:63;
L_el_y6=length(el_y6);
for j=1:L_el_y6
    counts_y6(1,j)=length(find(y6(1,:)==el_y6(1,j))) ;
end
counts_y6(find(counts_y6==0))=1;
comp=arithenco(y6,counts_y6);

```

PERCENT_ROOT_MEAN_SQUARE_DIST.m

```

function [PRD,PRD2]=PERCENT_ROOT_MEAN_SQUARE_DIST(Yp_1,C61,LENGTH,Y1)

Yp1_1 = Yp_1(C61:C61+LENGTH-2);
Y1_1 = Y1(C61:C61+LENGTH-2);
Y1_1avg = sum(Y1_1)/length(Y1_1);

PRD2 = 100*sqrt(( sum((Y1_1-Yp1_1).^2 ))/( sum((Y1_1-Y1_1avg).^2 ) ));
PRD = 100*sqrt(( sum((Y1_1-Yp1_1).^2 ))/( sum((Y1_1).^2 ) ));

```

References

- [1] Jalaliddine SM, Hutchens CG, Strattan RD & Coberly WA. "ECG data compression techniques – a unified approach", *IEEE Trans. on BME*; vol. 37 (4), pp. 329-343, 1990.
- [2] Nave G & Cohen A. "ECG compression using long term prediction", *IEEE Trans. On BME.*; vol. 40 (9), pp. 877-885, 1993.
- [3] Cohen A, Poluta PM & Scott-Millar R. "Compression of ECG signals using vector quantization", *Proc. of the IEEE-90 S. A. Symposium on Communications and Signal Proc., COMSIG-90*; pp. 45-54, 1990.
- [4] Kleijn WB & Paliwal KK. "Speech Coding and Synthesis", Amsterdam: *Elsevier*, 1995.
- [5] Zigel Y, Cohen A. "On the optimal distortion measure for ECG compression", "EMBEC '99, Vienna; pp. 1618-19, November 1999
- [6] Zigel Y, Cohen A, Abu-Ful A, Wagshal A, Katz A. "Analysis by synthesis ECG signal compression", *Computers in Cardiology*; vol. 24, pp. 279-92, 1997.
- [7] Ishijima M. "Fundamentals of the decision of optimum factors in the ECG data compression", *IEICE Trans. Inf. and Sys.*; E76-D (12), pp. 1398-1403, 1993.
- [8] Cox JR, Nolle FM, Fozzard HA & Oliver CG. "AZTEC, a preprocessing program for real time ECG rhythm analysis", *IEEE Trans. on BME*; vol. 15, pp. 128-129, 1968.
- [9] Furht B & Perez A. "An adaptive real-time ECG compression algorithm with variable threshold", *IEEE Trans. on BME*; vol. 35, pp. 489-494, 1988.
- [10] Mammen CP & Ramamurthi B. "Vector quantization for compression of multichannel ECG", *IEEE Trans. on BME*; vol. 37 (9), pp. 821-825, 1990.
- [11] Mueller WC. "Arrhythmia detection program for an ambulatory ECG monitor", *Biomed. Sci. Instrument.*; vol. 14, pp. 81-85, 1978.
- [12] Abenstein JP & Tompkins WJ. "A new data reduction algorithm for real time ECG analysis", *IEEE Trans. on BME*; vol. 29, pp. 43-48, 1982.
- [13] Pollard AE & Barr RC. "Adaptive sampling of intra-cellular and extracellular cardiac potentials with the fan method", *Med. and Biol. Eng. and Comp.*; vol. 25, pp. 261-269, 1987.

- [14] Ishijima M, Shin SB, Hostetter GH & Sklansky J. "Scan along polygon approximation for data compression of electrocardiograms", *IEEE Trans. on BME*; vol. 30, pp. 723-729, 1983.
- [15] Jalaleddine SM, Hutchens CG. "SAIES – A new ECG data compression algorithm", *J. of Clinical Eng.*; vol. 15 (1), pp. 45-51, 1990.
- [16] Tai SC. "SLOPE – a real time ECG data compression", *Med. and Biol. Eng. And Comp.*; vol. 29, pp. 175-179, 1991.
- [17] Tai SC. "ECG data compression by corner detection", *Med. and Biol. Eng. and Comp.*; vol. 30, pp. 584-590, 1992.
- [18] Ruttiman UE & Pipberger HV. "Compression of the ECG by prediction or interpolation and entropy encoding", *IEEE Trans. on BME*; vol. 26, pp. 613-623, 1979.
- [19] Hsia PW. "Electrocardiographic data compression using precoding consecutive QRS information", *IEEE Trans. on BME*; vol. 36, pp. 465-468, 1989.
- [20] Hamilton PS & Tompkins WJ. "Compression of the ambulatory ECG by average beat subtraction and residual differencing", *IEEE Trans. on BME*; vol. 38, pp. 253-259, 1991.
- [21] Huffinan DA. "A method for the construction of minimum redundancy coders", *Proc. IRE*; vol. 40, pp. 1098-1101, 1952.
- [22] Hamilton PS. "Adaptive compression of the ambulatory electrocardiogram", *Biomedical Inst. and Tech.*; pp. 56-63, January 1993.
- [23] Iwata A, Nagasaka Y & Suzumura N. "Data compression of ECG using neuralnetwork for digital holter monitor", *IEEE Eng. in Med. and Biolo. Mag.*; pp. 53-57, September 1990.
- [24] Hamilton DJ, Thomson DC & Sandham WA. "ANN compression of morphologically similar ECG complexes", *Med. and Biol. Eng. and Comp.*; vol. 33, pp. 841-843, 1995.
- [25] Kuklinsky WS. "Fast Walsh transform data compression algorithm for ECG applications", *Med. and Biol. Eng. and Comp.*; vol. 21, pp. 465-473, 1983.
- [26] Ahmed N, Milne P, Harris S. "Electrocardiographic data compression via orthogonal transforms", *IEEE Transactions on Biomedical Engineering*; vol. 22(6), pp. 484-7, 1975.

- [27] Womble ME, Halliday JS, Mitter SK, Lancaster MC & Triebvasser JH. "Data compression for storing and transmitting ECGs/VCGs", *Proc. IEEE*; vol. 65, pp. 702-706, 1977.
- [28] Chen J, Itoh S & Hashimoto T. "ECG data compression by using wavelet transform", *IEICE Trans. Inf. and Sys.*; E76-D (12), pp. 1454-1461, 1993.
- [29] Ramakrishnan AG & Supratim S. "ECG coding by wavelet-based linear prediction", *IEEE Trans. on BME*; vol. 44 (12), 1997.
- [30] Poel J. "Compressão de sinais de eletrocardiograma", Master Thesis, Mestrado em Engenharia Biome'dica, NETEB/UFPB, João Pessoa, May 1999.
- [31] Lee H, Buckley K. "ECG data compression using cut and align beats approach and 2-D transforms", *IEEE Transactions on Biomedical Engineering*; vol. 46(5), pp. 556-64, 1999.
- [32] Rao K, Yip P. "Discrete cosine transform—algorithms, advantages, applications", *San Diego: Academic Press*; 1990.
- [33] Strang G, Nguyen T. "Wavelets and filter banks", *Wellesley: Wellesley-Cambridge Press*; 1996.
- [34] Ahmed N, Milne P, Harris S. "Electrocardiographic data compression via orthogonal transforms", *IEEE Transactions on Biomedical Engineering*; vol. 22(6), pp. 484-7, 1975.
- [35] Zou F, Gallagher R. "ECG data compression with wavelet and discrete cosine transforms", *Biomed Sci Instrum*; vol. 30, pp. 57-62, 1994.
- [36] Ratnakar V. "Quality-controlled lossy image compression", Ph.D. Thesis, *University of Wisconsin, Madison*, 1997.
- [37] Wallace G. "The JPEG still picture compression standard", *Communications of the ACM*; vol. 34(4), pp. 30-44, 1991.
- [38] Batista LV, Melcher EU, Carvalho LC. "Compression of ECG signals by optimized quantization of discrete cosine transform coefficients", *Medical Engineering & Physics, Elsevier*; vol. 23 pp. 127-134, 2001
- [39] Huszar RJ. "Basic Dysrhythmias: interpretation & management", 2nd ed. *St. Louis, Missouri: Mosby Lifeline*; 1994.
- [40] Cardenas-Barreras J, Lorenzo-Ginori J. "Mean-shape vector quantizer for ECG signal compression", *IEEE Transactions on Biomedical Engineering* ;vol.46(1), pp. 62-70, 1999.

- [41] Husoy J, Gjerde T. "Computationally efficient sub-band coding of ECG signals", *Med Eng Phys*; vol. 18(2), pp. 132-42, 1996.
- [42] Lu Z, Kim D, Pearlman W. "Wavelet compression of ECG signals by set partitioning in hierarchical trees algorithm", *IEEE Transactions on Biomedical Engineering*; vol. 47(7), pp. 849-56, 2000.
- [43] Batista LV, Melcher EU, Carvalho LC. "An ECG compression method using peak selection and discrete cosine transform (in Portuguese)", *Brazilian Journal of Biomedical Engineering* ; vol. 16(1), pp. 39-48, 2000.
- [44] Miaou S, Yen H. "Quality driven gold washing adaptive vector quantization and its application to ECG data compression", *IEEE Transactions on Biomedical Engineering*; vol. 47(2), pp. 209-18, 2000.
- [45] Michael L Hilton. "Wavelet and wavelet packet compression of electrocardiograms", *Technical Report TR9505, Dept of computer Science, The University of South Carolina, Columbia*

