A DISSERTATION

ON

Performance Analysis of ECG Data compression Technique

Submitted In Partial Fulfilment of the Requirement for the Award of the Degree in

Master of Technology

In

Electronic Circuits and Systems

Submitted By

Samra Zubair

Roll No.:2101311005

Under the Guidance of

Ms. Archana yadav	**Dr. Shailendra Kumar**
(Supervisor)	**(Co-Supervisor)**

Department of Electronics & Communication Engineering.

INTEGRAL UNIVERSITY, LUCKNOW-226026

June-2023

DECLARATION

This is to certify that I, Samra Zubair,(Roll No.: 2101311005) have completed the M.Tech Dissertation work on topic "**Performance Analysis of ECG Data compression Technique**" under the supervision of Ms. Archana Yadav and Dr. Shailendra Kumar for the partial fulfillment of the requirement for the Master of Technology in Electronic Circuits and Systems from Department of Electronics and Communication Engineering, Integral University, Lucknow. This is an original piece of work & I have not submitted it earlier elsewhere.

Date: Signature

Place Samra Zubair

Roll No.: 2101311005

CERTIFICATE

This is to certify that the dissertation titled "**Performance Analysis of ECG Data compression Technique**" has been carried out by Samra Zubair(Roll No.: 2101311005) under my supervision and guidance in partial fulfillment of the requirements for the award of degree of "**Master of Technology**" in Electronic Circuits and Systems at department of Electronics and Communication Engineering, Integral University, Lucknow, India.

Ms. Archana Yadav	**Dr. Shailendra Kumar**
Supervisor	Co-Supervisor
Asst. Professor	Associate Professor
Dept. of ECE	Dept. of ECE
Integral University,	Integral University,
Lucknow	Lucknow

ACKNOWLEDGEMENT

I wish to express my sincere thanks to my supervisor Ms. Archana Yadav, Assistant Professor, Department of Electronics & Communication Engineering, and Co-Supervisor Dr. Shailendra Kumar, Associate Professor, Department of Electronics & Communication Engineering, whose sincerity and encouragement I will never forget. This work would not have been possible without the support and valuable guidance of my supervisor and co-supervisor.

I sincerely wish to thank, Prof. Syed Hassan Saeed, Professor (HOD, Department of Electronics & Communication Engineering) for his valuable feedbacks during my comprehensive examination. I also thankful to all the faculty members and my colleagues who helped me in any way in completion of this dissertation.

Last but not the least, I would like to thanks and my family and friends for supporting me throughout writing this dissertation.

Samra Zubair

Roll No.:2101311005

ABSTRACT

ECG is a device that detects the heart's electrical behavior. The heart is a muscular organ which pumps blood through the body rhythmically. Large signal data must be stored and transmitted. The ECG signal data must then be compressed effectively. We compared the performance of different forms of ECG compression techniques in this dissertation. These strategies are essential in reducing the size of the transmitted data without losing clinical information. These schemes are based on transformation methods such as Discrete Cosine Transform (DCT), Discrete Sine Transform (DST), Fast Fourier Transform (FFT), the improved method Discrete Cosine Transform- II (DCT-II) and Blaschke unwinding AFD .Records selected from MIT-BIH arrhythmia database are tested. For performance evaluation Percent Root Mean Square differences (PRD) and Compression Ratio (CR) are used.

LIST OF ABBREVIATIONS

TP	turning point
AZTEC	amplitude zone time epoch coding
CORTES	coordinate reduction time encoding system
DCT	discrete cosine transform
PRD	Percentage Root Mean Square Difference
DPCM	differential pulse code modulation
SAPA	Scan-Along Polygonal Approximation
CR	compression ratio
FFT	Fast Fourier Transform
LTP	Long term prediction
STP	Short term prediction
JPEG	Joint picture expert group
ECG	Electrocardiogram
DST	Discrete Sine Transform
DCT-II	Discrete Cosine Transform- II
DFT	discrete Fourier transform

LIST OF FIGURES

Figure	Caption	Page No.
Figure 1.1	Time intervals of ECG	3
Figure 1.2	Block diagram of transform based compression method	4
Figure 1.3	Algorithms for basic ECG signal processing	5
Figure 1.4	Block diagram of compression and decompression procedures	6
Figure 1.5	Flowchart of difference offset calculation procedures for making	6
Figure 3.1	Sorting the characters a1c1c2b1a2c3c4a3c5b2a4 based on context: (a) each character in its context, (b) end context moved to front, and (c) characters sorted by their context using reverse lexicographic ordering.	13
Figure 3.2	Burrows-Wheeler Decoding Example mm-wave	16
Figure 3.3	Examples of (a) uniform and (b) non-uniform scalar quantization	19
Figure 3.4	Example of vector-quantization for a height-weight chart	19
Figure 3.5	Transforms	21
Figure 3.6	Steps in JPEG compression	23
Figure 3.7	Zig-zag scanning of JPEG blocks	24
Figure 3.8	MPEG B-frames postponed in data stream.	25
Figure 3.9	P-frame encoding	27
Figure 3.10	A small Haar wavelet family of size seven	30
Figure 3.11	A sampling of popular wavelets	31
Figure 3.12	Self-similarity in the Daubechies wavelet	31
Figure 3.13	Identifying self-similarity	33
Figure 5.1	FFT compression of MIT-BIH record 100	46
Figure 5.2	DCT compression of MIT-BIH record 100	47
Figure 5.3	DST compression of MIT-BIH record 100	48
Figure 5.4	DCT-2 compression of MIT-BIH record 100	49

Figure 5.5 Waveforms of original, reconstructed and error signals with N = 8 taken from record 100 50

Figure 5.6 Waveforms of original, reconstructed and error signals with N = 10 taken from record 100 50

LIST OF TABLES

Table No.	**Caption**	**Page no.**
Table 3.1	JPEG default quantization table, luminance plane	24
Table 5.1	Comparision of resultant compression techniques	51

LIST OF CONTENTS

TITLE	PAGE NO.
Declaration	i
Certificate	ii
Abstract	iii
Acknowledgement	iv
List of abbreviations	v
List of figures	vi-vii
List of tables	viii
CHAPTER 1	**1-7**
1.1 Introduction	1
1.2 ECG Signal Processing	4
1.3. CHALLENGES	7
CHAPTER 2: LITERATURE SURVEY	**8-12**
2.1 DIRECT TIME-DOMAIN TECHNIQUES	8
2.1.1 The Amplitude Zone Time Epoch Coding (AZTEC) Technique	8
2.1.2 The Turning Point Technique	9
2.1.3 THE COORDINATE REDUCTION TIME ENCODING SYSTEM (CORTES) SCHEME	9
2.1.4 Fan and SAPA technique	10
2.1.5 Improved Modified AZTEC	10
2.1.6 DELTA CODING	10
2.2 TRANSFORM-DOMAIN TECHNIQUES	11
2.2.1 FT domain	11
2.2.2 DCT domain	11
2.2.3 Wavelet domain	11
2.2.4 JPEG 2000	11

2.3 PARAMETER EXTRACTION TECHNIQUES	12
2.3.1 Peak picking approach	12
2.3.2 Long term prediction approach	12
CHAPTER3: DATA COMPRESSION TECHNIQUES	**13-34**
3.1 Lossless Compression Techniques	13
3.1.1 Burrows Wheeler	13
3.2 Lossy Compression Techniques	17
3.2.1 Scalar Quantization	17
3.2.2 Vector Quantization	18
3.2.3 Transform Coding	20
3.3 A Case Study: JPEG and MPEG	22
3.3.1 JPEG	22
3.3.2 MPEG	25
3.3.3 MPEG in the Real World	29
3.4 Other Lossy Transform Codes	29
3.4.1 Wavelet Compression	29
3.4.2 Fractal Compression	31
3.4.3 Model-Based Compression	34
CHAPTER 4:METHODOLOGY	**35-44**
4.1 Discrete Cosine Transform(DCT)	35
4.2 Discrete Sine Transform	36
4.3 Fast Fourier Transform (FFT)	37
4.4. Discrete Cosine Transform–II (DCT – II)	38
4.5 Splines and B-splines	39
4.6 Continuous Hermite functions	41
4.7 Blaschke unwinding AFD	42
CHAPTER 5: RESULT ANALYSIS	45-51
5.1 Performance Evaluation	45
5.1.1. Compression Ratio (CR)	45

5.2 Compression Algorithm 46
5.2.1 FFT 46
5.2.2. Discrete Cosine Transform (DCT) 47
5.2.3. Discrete sine Transform (DST) 48
5.2.4. Discrete Cosine Transform-2 (DCT-2) 49
5.2.5. Blaschke unwinding AFD 50
CONCLUSION 52
REFERENCES 53-57
The MIT-BIH Arrhythmia Database 58

CHAPTER 1

1.1 Introduction:

Electrocardiographic signals may be recorded on a long timescale (i.e., several days) for the purpose of identifying intermittently occurring disturbances in the heart rhythm. As a result, the produced ECG recording amounts to huge data sizes that quickly fill up available storage space. Transmission of signals across public telephone networks is another application in which large amounts of data are involved. For both situations, data compression is an essential operation and, consequently, represents yet another objective of ECG signal processing. Signal processing has contributed significantly to a new understanding of the ECG and its dynamic properties as expressed by changes in rhythm and beat morphology. For example, techniques have been developed that characterize oscillations related to the cardiovascular system and reflected by subtle variations in heart rate. ECG Data Compression is required to reduce the disk space required to store the data, as ECG is a continuous data taken for a very long interval of time. Also by compressing redundant data from the signal can be removed which actually takes considerably large area in memory. The need of signal transmission over telephone lines or antenna for remote analysis makes the compression and data reconstruction of the signal an important issue in signal processing. ECG is a graphic display of the electrical activity of the heart. Due to low cost and noninvasion, ECG signal has been extended for heart disease diagnosis and ambulatory monitoring. For storage and transmission of large signal data, it is necessary to compress the ECG signal data. Data compression has its application in many fields and so as in the field of medical science. ECG is an important parameter that measures patient's health and reports abnormalities if any. This thesis has done a survey of various kinds of ECG data compression techniques. Recently, numerous research and techniques have been developed for compression of the signal. These techniques are essential to a variety of application ranging from diagnostic to ambulatory ECG's. Thus, the need for effective ECG compression techniques is of great importance. The non-invasive extraction of physiological and clinical information hidden in biomedical signals is an important and fascinating field of research. Non-invasive assessment of the physiological parameters of a patient enables to study the physiology and patho-physiology of the investigated system, with minimal interference and inconvenience. Endogenous biomedical signals from physiological systems are acquired for a number of reasons including diagnosis, post surgical intensive care monitoring, neonatal monitoring and guide therapy and for research. The electrocardiogram (ECG) is a non- stationary signal containing information about the physiological condition of the heart. The electrical activity of the heart depicts the morphology and durations of the P-QRS- T intervals (Figure 1). The P, QRS complex and T features of ECG reveal the rhythmic depolarization and re polarization of the myocardium contractions of heart’s atria and ventricles [1]. The time intervals between various peaks contain clinical information about the nature of possible disease afflicting a heart [2].

Due to low cost and non-invasion, ECG signal has been extended for heart disease diagnosis and ambulatory monitoring resulting in enormous volume of the data. In course of a 24-h ECG observation or multichannel biological signal acquisition, real-time data compression methods are required for the effective use of communications channels such as wired channel, wireless environment and cloud computing. The ECG data compression is also required for the transmission of ECG signals across intensive care units, emergency tele-medical services, telemedicine, home care, space programs, sports, military, public telephone networks, cellular networks and wireless communication systems [4-5]. ECG is having possibility of redundant information reduction through inter and intra beat correlation, which is the basic cause of its compression [6]. The fundamental goal of data compression is efficient transmission or storage while preserving the significant diagnostic features.

In general, ECG compression can be classified into lossy and lossless techniques [7]. The lossless compression guarantee the integrity of reconstructed data while compromised compression ratio (CR), with nearly 0% reconstruction error, on the other hand lossy compression is having high CR with varying level of reconstruction error [6].

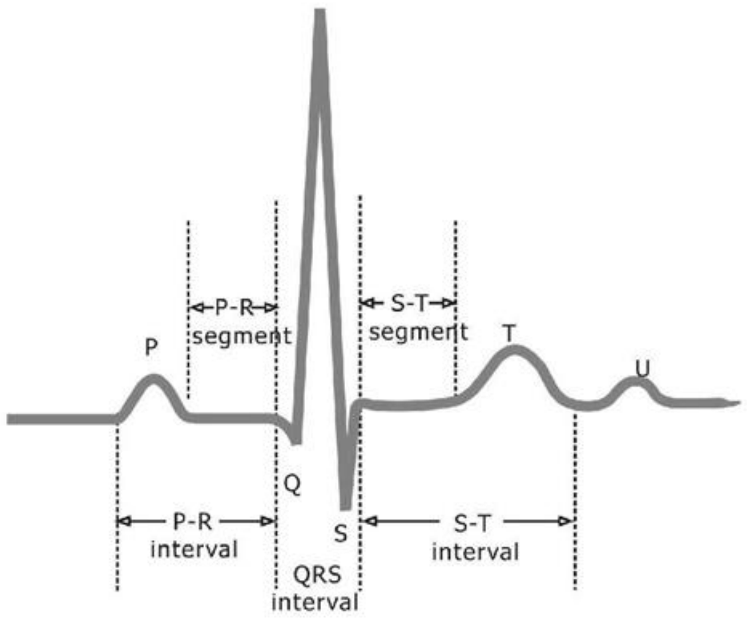


Fig 1.1: Time intervals of ECG

ECG signal compression techniques widely fall into three categories of direct method, transformation method and parameter extraction method [7, 8]. The direct data compression method openly analyzes and reduces data points in the time domain and the example includes turning point (TP) [25], amplitude zone time epoch coding (AZTEC) [3], Improved modified AZTEC technique [9], coordinate reduction time encoding system (CORTES) [48], SLOPE [10], the delta algorithm and the Fan algorithm [11]. The transformed method analyzes energy distribution by converting the time domain to some other domain and example includes Fourier transform, Fourier descriptor [12], the discrete cosine transform (DCT) [13], DCT with modified stages [14, 15] and wavelet transform [16], and the compressed sensing [17]. The parameter extraction method is based upon dominant feature extraction from raw signal; examples include neural based or syntactic methods [18],peak picking and linear prediction method [19]. The other methods for compression includes ASCII based encoding for incorporation of ECG data as ASCII character in existing technology [20-23].

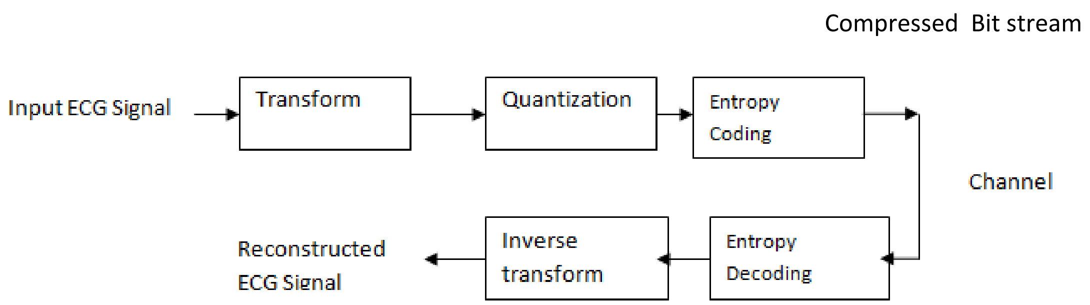


Fig. 1.2 Block diagram of transform based compression method

1.2 ECG Signal Processing:

The block diagram in Figure 1.3 presents this set of signal processing algorithms. Although these algorithms are frequently implemented to operate in sequential order, information on the occurrence time of a heartbeat, as produced by the QRS detector, is sometimes incorporated into the other algorithms to improve performance. The complexity of each algorithm varies from application to application so that, for example, noise filtering performed in ambulatory monitoring is much more sophisticated than that required in resting ECG analysis. Once the information produced by the basic set of algorithms is available, a wide range of ECG applications exist where it is of interest to use signal processing for quantifying heart rhythm and beat morphology properties. The signal processing associated with two such applications—high resolution ECG and T wave alternates are briefly described at the end of this article. The timing information produced by the QRS detector may be fed to the blocks for noise filtering and data compression (indicated by gray arrows) to improve their respective performance. The output of the upper branch is the conditioned ECG signal and related temporal information, including the occurrence time of each heartbeat and the onset and end of each wave.

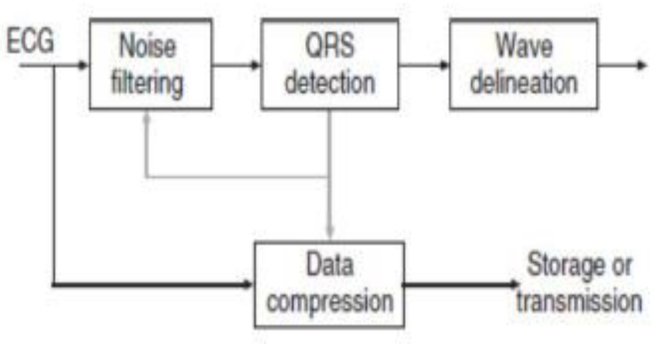


Figure 1.3 Algorithms for basic ECG signal processing.

The algorithm for real-time ECG signal compression and reconstruction is summarized in Figure1.4. As shown in this figure, it is composed of five compressing procedures and four reconstruction procedures. For compression, the first procedure is to obtain backward differences after 1/2 down-sampling of the ECG signal. The second procedure is to detect the peak of the differenced signal and classify it from the current peak to the previous peak and store the result. The third procedure is to obtain the DCT of the stored data. The fourth procedure is to filter the transformed data obtained in the previous procedure using a window filter, and the final procedure is to apply the Huffman coding algorithm. The data transmitted to a server or a base station from e-health devices are the data block coming out of the last compression procedure. The channel number can be added to the protocol header if e-health devices need to transmit multiple bio-signals. Figure 1.4 also shows the reconstruction procedure, which is the reverse order of the compression procedure. The first reconstruction procedure applies the inverse Huffman coding algorithm to the compressed and transmitted data. The second procedure obtains the inverse discrete cosine transform. The third interpolates the recovered time signal during the previous procedure using Spline interpolation, and the final procedure is to reconstruct the original signal after calculating the inverse difference[53].

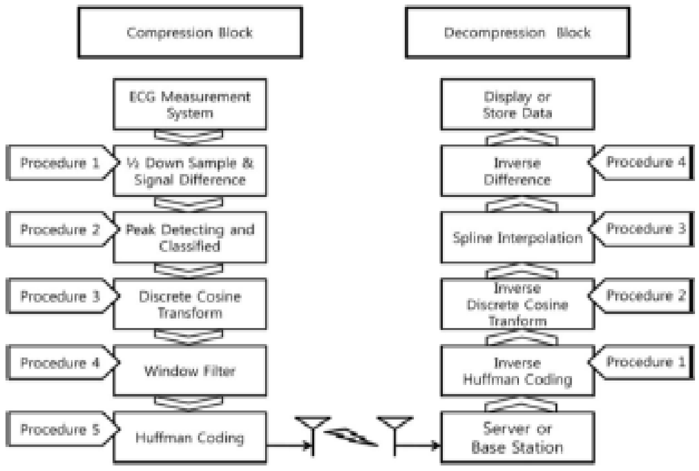


Figure 1.4 Block diagram of compression and decompression procedures[53]

ECG, which is an analog signal, is usually sampled at 200 Hz to 1 kHz depending on the purpose of applications.Usually, the sampled data are represented as a 2-byte data. In the proposed data compression algorithm, the acquired ECG signal is first downsampled by 1/2 and represented as 1-byte data after calculating the backward difference, decreasing its data size by 75%. The signed 1-byte data can be represented from –128 to +127 in decimal.

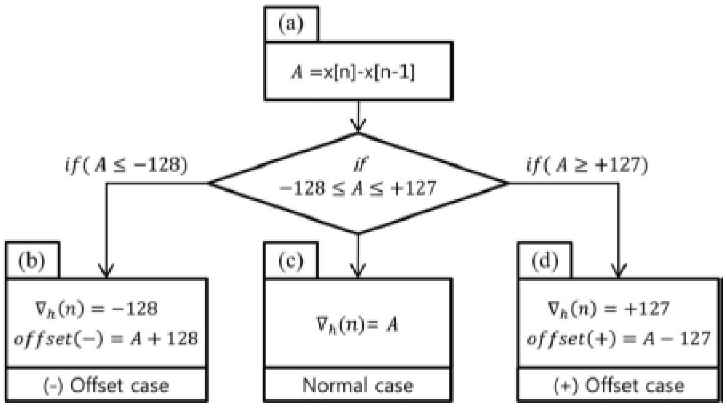


Figure 1.5 Flowchart of difference offset calculation procedures for making

ECG Data Compression is required to reduce the disk space required to store the data, as ECG is a continuous data taken for a very long interval of time. Also by compressing redundant data from the signal can be removed which actually takes considerably large area in memory.

The need of signal transmission over telephone lines or antenna for remote analysis makes the compression and data reconstruction of the signal an important issue in signal processing. ECG is a graphic display of the electrical activity of the heart. Due to low cost and noninvasion, ECG signal has been extended for heart disease diagnosis and ambulatory monitoring. For storage and transmission of large signal data, it is necessary to compress the ECG signal data.

Data compression has its application in many fields and so as in the field of medical science. ECG is an important parameter that measures patient's health and reports abnormalities if any. This thesis has done a survey of various kinds of ECG data compression techniques. Recently, numerous research and techniques have been developed for compression of the signal. These techniques are essential to a variety of application ranging from diagnostic to ambulatory ECG's. Thus, the need for effective ECG compression techniques is of great importance. Many existing compression algorithms have shown some success in electrocardiogram compression; however, algorithms that produce better compression ratios and less loss of data in the reconstructed signal are needed. This thesis discusses various techniques proposed earlier in literature for compression of an ECG signal and provide comparative study of these techniques.

1.3. CHALLENGES:

1. To overcome the Problem of Percentage Root Mean Square Difference(PRD).
2. Reduced the storage requirements to develop a more efficient telecardiology system for cardiac analysis and diagnosis.

CHAPTER 2

LITERATURE SURVEY

ECG compression methods are classified as: lossless and lossy. In lossless method, compressed signal is reconstructed in exact form of original signal and in lossy method; compressed signal is reconstructed with some error. Another classification is based on techniques applied for compression and can be categorized as:

2.1 DIRECT TIME-DOMAIN TECHNIQUES:

Direct methods are based on the extraction of a subset of significant samples. Direct time-domain ECG compression techniques have efficient performance in terms of processing speed and CR. These techniques explore the redundancies present directly in the ECG samples. Direct compression techniques can be based on three approaches: tolerance comparison Compression, differential pulse code modulation (DPCM), and entropy coding [3]. Next, the significant work that has been focussed towards direct ECG compression techniques is discussed.

2.1.1 The Amplitude Zone Time Epoch Coding (AZTEC) Technique:

Cox *et al.* developed the AZTEC algorithm for preprocessing of real-time ECG's for rhythm analysis [3]. It has become a popular data reduction algorithm for ECG monitors. The AZTEC algorithm converts raw ECG sample points into plateaus. The amplitude value and length of each plateau are stored for reconstruction. Although the AZTEC technique is capable to compress with CR of 10 but the reconstruction error is not clinically acceptable. The step-like reconstructed signal may misinterpret the ECG features especially in the slow varying slopes of P and T peaks of the ECG [3, 24]. 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). 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 value 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 [50], in which the threshold is not a constant but 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 [51], vector quantization was used along with the m-AZTEC to produce a multi-lead ECG data compressor. This approach yieldes a compression ratio of 8:6:1.

2.1.2 The Turning Point Technique:

The TP is direct data compression technique to reduce the ECG sampling frequency without diminishing the elevation of large amplitude QRS complexes [25]. The TP algorithm achieved fixed CR of 2, with almost zero reconstruction error *i.e.* reconstructed signal resemble the original ECG signal. The drawback of this technique is its unsuitability for equally spaced time intervals.

2.1.3 THE COORDINATE REDUCTION TIME ENCODING SYSTEM (CORTES) SCHEME:

CORTES algorithm is a hybrid approach of AZTEC and TP to achieve high CR of the AZTEC and the low reconstruction error of the TP technique. TP was applied to the high frequency QRS region and AZTEC to the isoelectric regions of the ECG [48]. The performance analysis of the AZTEC, TP, and CORTES based compression for ECG’s at 200 Hz sampling frequency produced CR of 5, 2 & 4.8 and PRD’s of 28, 5 & 7 respectively.

2.1.4 Fan and SAPA technique:

Gardenhire proposed and evaluated the Fan method for ECG compression [11]. Both Fan and Scan-Along Polygonal Approximation (SAPA) techniques are the first-order interpolation with two degrees of freedom (FOI-2DF) algorithms for ECG compression [24]. The Fan method implements the FOI-2DF without storing all the actual data points between the last transmitted and the present point. 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. The location and the amplitude of this sample are stored, and this process recurs. The reconstructed signal looks like a broken line, and its fidelity depends on the error threshold. The greater the threshold is, the better the compression ratio, and the poorer the fidelity. The Scan-Along Polygonal Approximation (SAPA) techniques [52] 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.For signals sampled at 250 Hz with 12 bit resolution, the compression ratio is 3:1 with a PRD of 4%.

2.1.5 Improved Modified AZTEC:

In modified AZTEC algorithm, adaptive statistical parameters of the signal to be compressed are calculated. The adaptive algorithm optimizes the tradeoff between CR and PRD. [9]. The CR ranges between 2.76 and 9.91 for the threshold variation from 0.010 to 0.035 and PRD from 4.54 to 7.99 was achieved.

2.1.6 DELTA CODING:

A technique called delta coding with threshold was proposed in [26] for three-lead ECG compression. ECG samples of three-lead ECG signals with successive differences more than a threshold value are stored. Otherwise samples are considered redundant and removed. In other words, actual ECG samples are replaced by the first-difference signal (amplitude between successive samples) [27]. The DPCM based ECG compression system consists a quantizer in the compression stage and an estimator in both the compression and the reconstruction stages.

2.2 TRANSFORM-DOMAIN TECHNIQUES:

Transform based ECG compression techniques are performed by the application of linear orthogonal transformation to ECG samples. Thus original samples of ECG are subjected to a transformation and the compression is performed in the entirely new domain like Fourier transform (FT), DCT and wavelet etc [8, 12, 16, 28, 31]. These techniques pose higher CR than direct techniques and are insensitive to noise present in ECG signals.

2.2.1 FT domain:

With FT the frequency-amplitude representation of the signal is obtained [8, 12]. To reconstruct the signal inverse FFT is applied. Limitation of FT is that it fails to provide the information regarding the exact location of frequency component in time.

2.2.2 DCT domain:

DCT represents a signal as a sum of varying magnitude and frequency. It implies different boundary condition & often used in signal and image processing for lossy data compression. DCT has strong “energy compaction property” & provide high de-correlation. In DCT compression signal information can restore in a restrict number of DCT coefficients [15, 29, 30]. DCT-II provide very impressive CR but at the cost of high distortion.

2.2.3 Wavelet domain:

Wavelet transform have the beauty to analyze the signal both in time as well as frequency domains simultaneously. Many researchers concentrate on wavelet based ECG compression techniques [32-34]. Recently, one-dimensional (1D) and two- dimensional (2D) Wavelet transform based ECG compression techniques with impressive CR, low PRD and smooth signal quality are presented in literature [35-40]. The 2D compression approaches achieve high CR but involve complex steps; accurate QRS detection, period normalization, amplitude normalization and mean removal etc.

2.2.4 JPEG 2000:

JPEG2000 [41] is the latest image compression standard applied to the compress ECG signals [42]. Some compression techniques use JPEG2000 as the encoder for the 2-D ECG image with high CR [43].

2.3 PARAMETER EXTRACTION TECHNIQUES:

These are irreversible processes which retain the particular characteristics or parameters of the ECG signals. The parameter extraction method is based upon dominant feature extraction from raw ECG signal; examples include neural based or syntactic methods [18, 47], peak picking and linear prediction method [26, 44-48].

2.3.1 Peak picking approach:

The ECG signal is sampled at peaks and significant fiducial points are extracted. Imai et al. [49] presented an ECG peak- picking compression system by reconstructing the signal using spline functions.

2.3.2 Long term prediction approach:

Long-Term Prediction (LTP) explores the "periodicity" property of ECG signal to reduce redundancy and to achieve high CR. Reconstruction error; PRD for LTP approach is lower than the conventional linear short term prediction (STP) [44].

CHAPTER 3

DATA COMPRESSION TECHNIQUES

3.1 Lossless Compression Techniques:

3.1.1 Burrows Wheeler:

The Burrows Wheeler algorithm is a relatively recent algorithm. An implementation of the algorithm called bzip, is currently one of the best overall compression algorithms for text. It gets compression ratios that are within 10% of the best algorithms such as PPM, but runs significantly faster. Rather than describing the algorithm immediately, let’s try to go through a thought process that leads to the algorithm. Recall that the basic idea of PPM was to try to find as long a context as-

	a	**ccbaccacba**	**ccbaccacba$_4$**	**a$_1$**	**cbaccacbaa$_1$**	**c$_1$**
a	**c**	**cbaccacba**	**cbaccacbaa$_1$**	**c$_1$**	**ccbaccacba$_4$**	**a$_1$**
ac	**c**	**baccacba**	**baccacbaac$_1$**	**c$_2$**	**cacbaaccba$_2$**	**c$_3$**
acc	**b**	**accacba**	**accacbaacc$_2$**	**b$_1$**	**baaccbacca$_3$**	**c$_5$**
accb	**a**	**ccacba**	**ccacbaaccb$_1$**	**a$_2$**	**accbaccacb$_2$**	**a$_4$**
accba	**c**	**cacba**	**cacbaaccba$_2$**	**c$_3$**	**ccacbaaccb$_1$**	**a$_2$**
accbac	**c**	**acba**	**acbaaccbac$_3$**	**c$_4$**	**baccacbaac$_1$**	**c$_2$**
accbacc	**a**	**cba**	**cbaaccbacc$_4$**	**a$_3$**	**acbaaccbac$_3$**	**c$_4$**
accbacca	**c**	**ba**	**baaccbacca$_3$**	**c$_5$**	**aaccbaccac$_5$**	**b$_2$**
accbaccac	**b**	**a**	**aaccbaccac$_5$**	**b$_2$**	**accacbaacc$_2$**	**b$_1$**
accbaccacb	**a**		**accbaccacb$_2$**	**a$_4$**	**cbaaccbacc$_4$**	**a$_3$**
	(a)		**(b)**		**(c)**	

Figure 3.1: Sorting the characters a1c1c2b1a2c3c4a3c5b2a4 based on context: (a) each character in its context, (b) end context moved to front, and (c) characters sorted by their context using reverse lexicographic ordering.

We use subscripts to distinguish different occurences of the same character. possible that matched the current context and use that to effectively predict the next character. A problem with PPM is in selecting k. If we set k too large we will usually not find matches and end up sending too many escape characters. On the other hand if we set it too low, we would not be taking advantage of enough context. We could have the system automatically select k based on which does the best encoding, but this is expensive. Also within a single text there might be some very long contexts that could help predict, while most helpful contexts are short. Using a fixed k we would probably end up ignoring the long contexts. Lets see if we can come up with a way to take advantage of the context that somehow automatically adapts. Ideally we would like the method also to be a bit faster. Consider taking the string we want to compress and looking at the full context for each character i.e., all previous characters from the start of the string up to the character. In fact, to make the contexts the same length, which will be convenient later, we add to the head of each context the part of the string following the character making each context n − 1 characters. Examples of the context for each character of the string accbaccacba are given in Figure 3.1. Now lets sort these contexts based on reverse lexical order, such that the last character of the context is the most significant (see Figure 3.1c). Note that now characters with the similar contexts (preceeding characters) are near each other. In fact, the longer the match (the more preceeding characters that match identically) the closer they will be to each other. This is similar to PPM in that it prefers longer matches when "grouping", but will group things with shorter matches when the longer match does not exist. The difference is that there is no fixed limit k on the length of a match—a match of length 100 has priority over a match of 99.

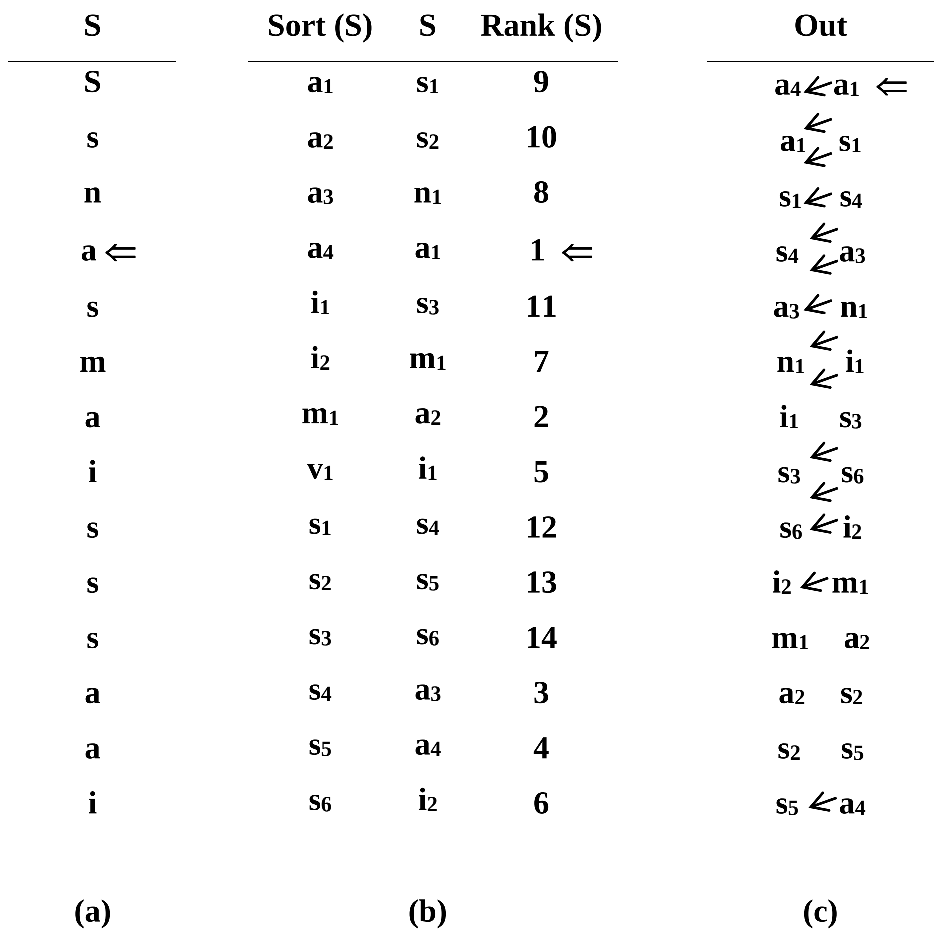
In practice the sorting based on the context is executed in blocks, rather than for the full message sequence. This is because the full message sequence and additional data structures required for sorting it, might not fit in memory. The process of sorting the characters by their context is often referred to as a block-sorting transform. In the discussion below we will refer to the sequence of characters generated by a block-sorting transform as the context-sorted sequence (e.g., c1a1c3c5a4a2c2c4b2b1a3 in Figure 3.1). Given the correlation between neary by characters in a context-sorted sequence, we should be able to code them quite efficiently by using, for example, a move-to-front coder. For long strings with somewhat larger character sets this technique should compress the string significantly since the same character is likely to appear in similar contexts. Experimentally, in fact, the technique compresses about as well as PPM even though it has no magic number k or magic way to select the escape probabilities.

The problem remains, however, of how to reconstruct the original sequence from context sorted sequence. The way to do this is the ingenious contribution made by Burrows and Wheeler. You might try to recreate it before reading on. The order of the most-significant characters in the sorted contexts plays an important role in decoding. In the example of Figure 3.1, these are a1a4a2a3b2b1c1c3c5c2c4. The characters are sorted, but equal valued characters do not necessarily appear in the same order as in the input sequence. The following lemma is critical in the algorithm for efficiently reconstruct the sequence.

Lemma 3.1.1. For the Block-Sorting transform, as long as there are at least two distinct characters in the input, equal valued characters appear in the same order in the most-significant characters of the sorted contexts as in the output (the context sorted sequence).
Proof. Since the contexts are sorted in reverse lexicographic order, sets of contexts whose most significant character are equal will be ordered by the remaining context i.e., the string of all previous characters. Now consider the contexts of the context-sorted sequence. If we drop the least-significant character of these contexts, then they are exactly the same as the remaining context above, and therefore will be sorted into the same ordering. The only time that dropping the least significant character can make a difference is when all other characters are equal. This can only happen when all characters in the input are equal. Based on Lemma 3.1.1, it is not hard to reconstruct the sequence from the context-sorted sequence as long as we are also given the index of the first character to output (the first character in the original input sequence). The algorithm is given by the following code

```
function BW Decode (In,FirstIndex,n)
S = Move To FrontDecode(In,n)
R = Rank(S)
j = First Index
for i = 1 to n − 1
Out[i] = S[j]
j = R[j]
```

For an ordered sequence S, the Rank(S) function returns a sequence of integers specifying for each character c 2 S how many characters are either less than c or equal to c and appear before c in S. Another way of saying this is that it specifies the position of the character if it where sorted using a stable sort.



S	Sort (S)	S	Rank (S)	Out
S	a_1	s_1	9	a_4 ← a_1 ⇐
s	a_2	s_2	10	a_1 s_1
n	a_3	n_1	8	s_1 ← s_4
a ⇐	a_4	a_1	1 ⇐	s_4 a_3
s	i_1	s_3	11	a_3 ← n_1
m	i_2	m_1	7	n_1 i_1
a	m_1	a_2	2	i_1 s_3
i	v_1	i_1	5	s_3 s_6
s	s_1	s_4	12	s_6 ← i_2
s	s_2	s_5	13	i_2 ← m_1
s	s_3	s_6	14	m_1 a_2
a	s_4	a_3	3	a_2 s_2
a	s_5	a_4	4	s_2 s_5
i	s_6	i_2	6	s_5 ← a_4

(a) (b) (c)

Figure 3.2: Burrows-Wheeler Decoding Example.

The decoded message sequence is assanissimassa. To show how this algorithms works, we consider an example in which the MoveToFront decoder returns S = ssnasmaisssaai, and in which FirstIndex = 4 (the first a). The example is shown in Figure 3.2(a). We can generate the most significant characters of the contexts simply by sorting S. The result of the sort is shown in Figure 3.2(b) along with the rank R. Because of Lemma 3.1.1, we know that equal valued characters will have the same order in this sorted sequence and in S. This is indicated by the subscripts in the figure. Now each row of Figure 3.2(b) tells us for each character what the next character is. We can therefore simply rebuild the initial sequence by starting at the first character and adding characters one by one, as done by BW Decode and as illustrated in Figure 3.2(c).

3.2 Lossy Compression Techniques:

Lossy compression is compression in which some of the information from the original message sequence is lost. This means the original sequences cannot be regenerated from the compressed sequence. Just because information is lost doesn't mean the quality of the output is reduced. For example, random noise has very high information content, but when present in an image or a sound file, we would typically be perfectly happy to drop it. Also certain losses in images or sound might be completely imperceptible to a human viewer (e.g. the loss of very high frequencies). For this reason, lossy compression algorithms on images can often get a factor of 2 better compression than lossless algorithms with an imperceptible loss in quality. However, when quality does start degrading in a noticeable way, it is important to make sure it degrades in a way that is least objectionable to the viewer (e.g., dropping random pixels is probably more objectionable than dropping some color information). For these reasons, the way most lossy compression techniques are used are highly dependent on the media that is being compressed. Lossy compression for sound, for example, is very different than lossy compression for images. In this section we go over some general techniques that can be applied in various contexts, and in the next two sections we go over more specific examples and techniques.

3.2.1 Scalar Quantization:

A simple way to implement lossy compression is to take the set of possible messages S and reduce it to a smaller set S0 by mapping each element of S to an element in S0. For example we could take 8-bit integers and divide by 4 (i.e., drop the lower two bits), or take a character set in which upper and lowercase characters are distinguished and replace all the uppercase ones with lowercase ones. This general technique is called quantization. Since the mapping used in quantization is many-toone, it is irreversible and therefore lossy. In the case that the set S comes from a total order and the total order is broken up into regions that map onto the elements of S_0, the mapping is called scalar quantization. The example of dropping the lower two bits given in the previous paragraph is an example of scalar quantization. Applications of scalar quantization include reducing the number of color bits or gray-scale levels in images (used to save memory on many computer monitors), and classifying the intensity of frequency components in images or sound into groups (used in JPEG compression). In fact we mentioned an example of quantization when talking about JPEG-LS. There quantization is used to reduce the number of contexts instead of the number of message values. In particular we categorized each of 3 gradients into one of 9 levels so that the context table needs only 93 entries (actually only (93 + 1)/2 due to symmetry). The term uniform scalar quantization is typically used when the mapping is linear. Again, the example of dividing 8-bit integers by 4 is a linear mapping. In practice it is often better to use a non-uniform scalar quantization. For example, it turns out that the eye is more sensitive to low values of red than to high values. Therefore we can get better quality compressed images by making the regions in the low values smaller than the regions in the high values. Another choice is to base the nonlinear mapping on the probability of different input values. In fact, this idea can be formalized—for a given error metric and a given probability distribution over the input values, we want a mapping that will minimize the expected error. For certain error-metrics, finding this mapping might be hard. For the root-mean-squared error metric there is an iterative algorithm known as the Lloyd-Max algorithm that will find the optimal mapping. An interesting point is that finding this optimal mapping will have the effect of decreasing the effectiveness of any probability coder that is used on the output. This is because the mapping will tend to more evenly spread the probabilities in S_0.

3.2.2 Vector Quantization:

Scalar quantization allows one to separately map each color of a color image into a smaller set of output values. In practice, however, it can be much more effective to map regions of 3-D color space into output values.

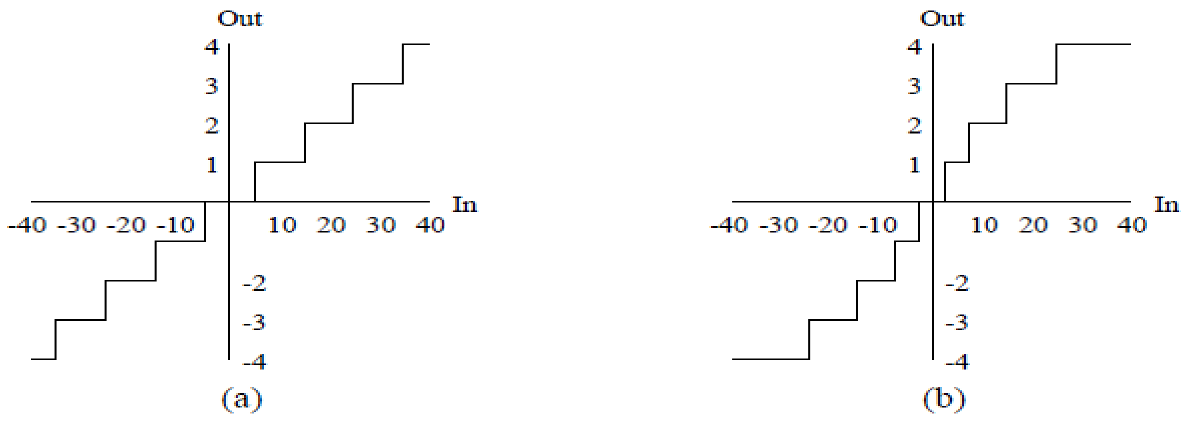


Figure 3.3: Examples of (a) uniform and (b) non-uniform scalar quantization.

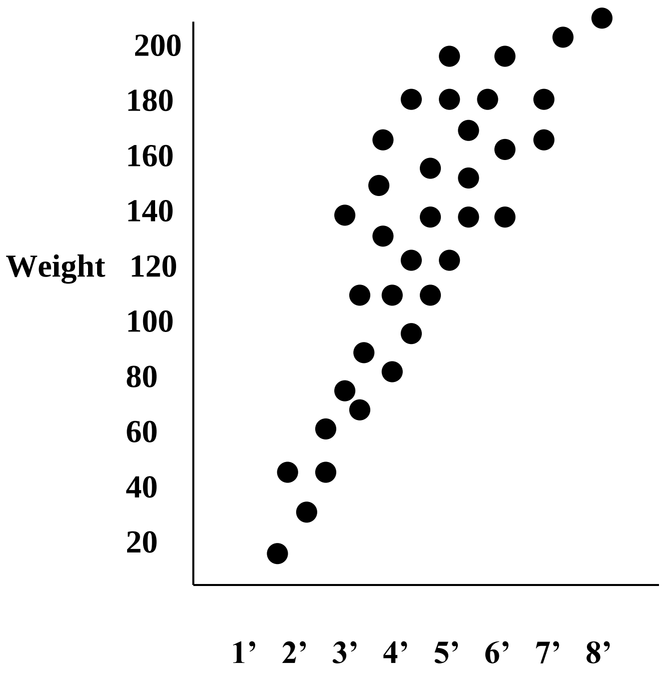


Figure 3.4: Example of vector-quantization for a height-weight chart

By more effective we mean that a better compression ratio can be achieved based on an equivalent loss of quality. The general idea of mapping a multidimensional space into a smaller set of messages S_0 is called vector quantization. Vector quantization is typically implemented by selecting a set of representatives from the input space, and then mapping all other points in the space to the closest representative. The representatives could be fixed for all time and part of the compression protocol, or they could be determined for each file (message sequence) and sent as part of the sequence. The most interesting aspect of vector quantization is how one selects the representatives. Typically it is implemented using a clustering algorithm that finds some number of clusters of points in the data. A representative is then chosen for each cluster by either selecting one of the points in the cluster or using some form of centroid for the cluster. Finding good clusters is a whole interesting topic on its own. Vector quantization is most effective when the variables along the dimensions of the space are correlated. Figure 3.4 gives an example of possible representatives for a height-weight chart. There is clearly a strong correlation between people’s height and weight and therefore the representatives can be concentrated in areas of the space that make physical sense, with higher densities in more common regions. Using such representatives is very much more effective than separately using scalar quantization on the height and weight. We should note that vector quantization, as well as scalar quantization, can be used as part of a lossless compression technique. In particular if in addition to sending the closest representative, the coder sends the distance from the point to the representative, then the original point can be reconstructed. The distance is often referred to as the residual. In general this would not lead to any compression, but if the points are tightly clustered around the representatives, then the technique can be very effective for lossless compression since the residuals will be small and probability coding will work well in reducing the number of bits.

3.2.3 Transform Coding:

The idea of transform coding is to transform the input into a different form which can then either be compressed better, or for which we can more easily drop certain terms without as much qualitative loss in the output. One form of transform is to select a linear set of basis functions that span the space to be transformed. Some common sets include sin, cos, polynomials, spherical harmonics, Bessel functions, and wavelets. Figure 3.5 shows some examples of the first three basis functions for discrete cosine, polynomial, and wavelet transformations. For a set of n values, transforms can be expressed as an $n \times n$ matrix T. Multiplying the input by this matrix T gives, the transformed coefficients. Multiplying the coefficients by $T-1$ will convert the data back to the original form.

For example, the coefficients for the discrete cosine transform (DCT) are

T_{ij} = (p 1/n cos (2j+1)i_ p 2n i = 0,0 _ j < n 2/n cos (2j+1)i_ 2n 0<i < n, 0 _ j < n (3.1)

The DCT is one of the most commonly used transforms in practice for image compression, more so than the discrete Fourier transform (DFT). This is because the DFT assumes periodicity, which is not necessarily true in images. In particular to represent a linear function over a region requires many large amplitude high-frequency components in a DFT. This is because the periodicity assumption will view the function as a sawtooth, which is highly discontinuous at the teeth requiring the high-frequency components. The DCT does not assume periodicity and will only require much lower amplitude high-frequency components. The DCT also does not require a phase, which is typically represented using complex numbers in the DFT. For the purpose of compression, the properties we would like of a transform are (1) to decorrelate the data, (2) have many of the transformed coefficients be small, and (3) have it so that from the point of view of perception, some of the terms are more important than others.

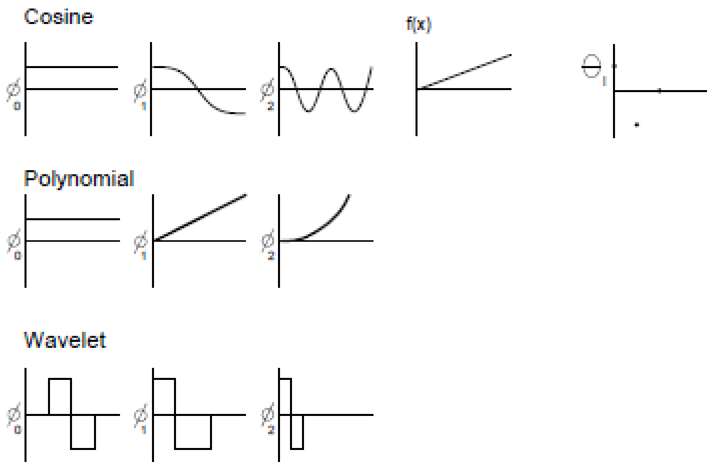


Figure 3.5: Transforms

3.3 A Case Study: JPEG and MPEG:

The JPEG and the related MPEG format make good real-world examples of compression because (a) they are used very widely in practice, and (b) they use many of the compression techniques we have been talking about, including Huffman codes, arithmetic codes, residual coding, runlength coding, scalar quantization, and transform coding. JPEG is used for still images and is the standard used on the web for photographic images (the GIF format is often used for textual images). MPEG is used for video and after many years of debated MPEG-2 has become the standard for the transmission of high-definition television (HDTV). This means in a few years we will all be receiving MPEG at home. As we will see, MPEG is based on a variant of JPEG (i.e. each frame is coded using a JPEG variant). Both JPEG and MPEG are lossy formats.

3.3.1 JPEG:

JPEG is a lossy compression scheme for color and gray-scale images. It works on full 24-bit color, and was designed to be used with photographic material and naturalistic artwork. It is not the ideal format for line-drawings, textual images, or other images with large areas of solid color or a very limited number of distinct colors. The lossless techniques, such as JBIG, work better for such images. JPEG is designed so that the loss factor can be tuned by the user to tradeoff image size and image quality, and is designed so that the loss has the least effect on human perception. It however does have some anomalies when the compression ratio gets high, such as odd effects across the boundaries of 8x8 blocks. For high compression ratios, other techniques such as wavelet compression appear to give more satisfactory results. An overview of the JPEG compression process is given in Figure 3.6. We will cover each of the steps in this process. The input to JPEG are three color planes of 8-bits per-pixel each representing Red, Blue and Green (RGB). These are the colors used by hardware to generate images. The first step of JPEG compression, which is optional, is to convert these into YIQ color planes. The YIQ color planes are

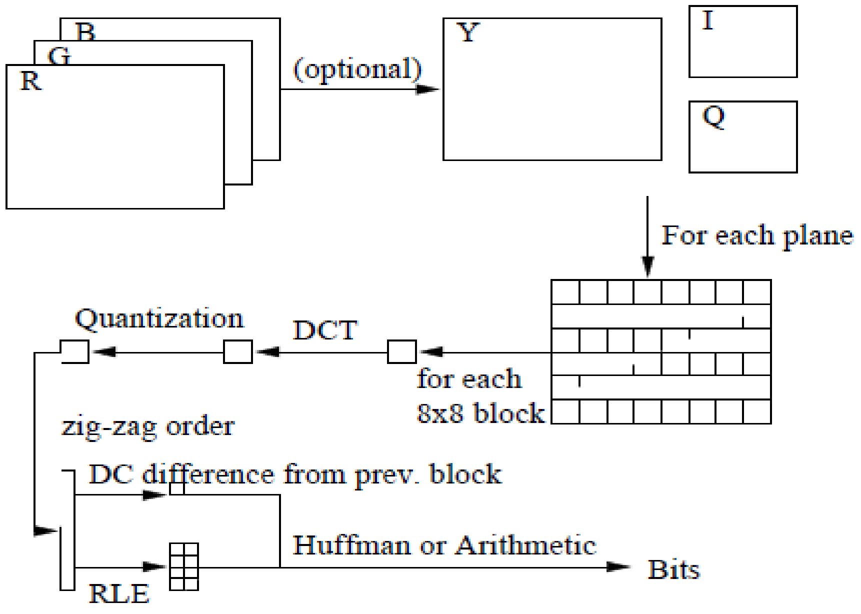


Figure 3.6: Steps in JPEG compression.

designed to better represent human perception and are what are used on analog TVs in the US (the NTSC standard). The Y plane is designed to represent the brightness (luminance) of the image. It is a weighted average of red, blue and green (0.59 Green + 0.30 Red + 0.11 Blue). The weights are not balanced since the human eye is more responsive to green than to red, and more to red than to blue. The I (interphase) and Q (quadrature) components represent the color hue (chrominance). If you have an old black-and-white television, it uses only the Y signal and drops the I and Q components, which are carried on a sub-carrier signal. The reason for converting to YIQ is that it is more important in terms of perception to get the intensity right than the hue. Therefore JPEG keeps all pixels for the intensity, but typically down samples the two color planes by a factor of 2 in each dimension (a total factor of 4). This is the first lossy component of JPEG and gives a factor of 2 compression: (1 + 2 _ .25)/3 = .5. The next step of the JPEG algorithm is to partition each of the color planes into 8x8 blocks. Each of these blocks is then coded separately. The first step in coding a block is to apply a cosine transform across both dimensions. This returns an 8x8 block of 8-bit frequency terms. So far this does not introduce any loss, or compression. The block-size is motivated by wanting it to be large enough to capture some frequency components but not so large that it causes “frequency spilling”. In particular if we cosine-transformed the whole image, a sharp boundary anywhere in a line would cause high values across all frequency components in that line. After the cosine transform, the next step applied to the blocks is to use uniform scalar quantization on each of the frequency terms. This quantization is controllable based on user parameters and is the main source of information loss in JPEG compression. Since the human eye is more perceptive to certain frequency components than to others, JPEG allows the quantization scaling factor to be different for each frequency component. The scaling factors are specified using an 8x8 table that simply is used to element-wise divide the 8x8 table of frequency components. JPEG

16	11	10	16	24	40	51	61
12	12	14	19	26	58	60	55
14	13	16	24	40	57	69	56
14	17	22	29	51	87	80	62
18	22	37	56	68	109	103	77
24	35	55	64	81	104	113	92
49	64	78	87	103	121	120	101

Table 3.1 : JPEG default quantization table, luminance plane.

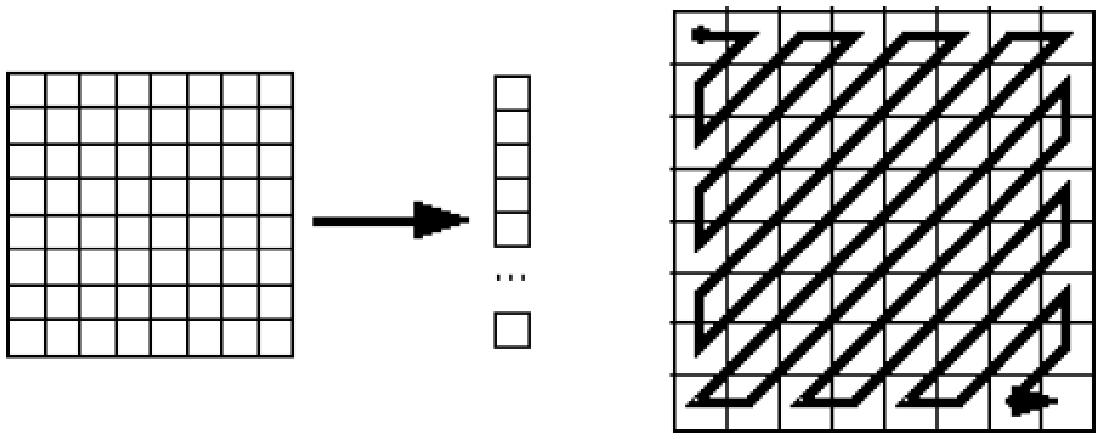


Figure 3.7: Zig-zag scanning of JPEG blocks.

defines standard quantization tables for both the Y and I-Q components. The table for Y is shown in Table 3.1. In this table the largest components are in the lower-right corner. This is because these are the highest frequency components which humans are less sensitive to than the lower-frequency components in the upper-left corner. The selection of the particular numbers in the table seems magic, for example the table is not even symmetric, but it is based on studies of human perception. If desired, the coder can use a different quantization table and send the table in the head of the message. To further compress the image, the whole resulting table can be divided by a constant, which is a scalar "quality control" given to the user. The result of the quantization will often drop most of the terms in the lower left to zero.

JPEG compression then compresses the DC component (upper-leftmost) separately from the other components. In particular it uses a difference coding by subtracting the value given by the DC component of the previous block from the DC component of this block. It then Huffman or arithmetic codes this difference. The motivation for this method is that the DC component is often similar from block-to-block so that difference coding it will give better compression. The other components (the AC components) are now compressed. They are first converted into a linear order by traversing the frequency table in a zig-zag order (see Figure 3.7). The motivation for this order is that it keeps frequencies of approximately equal length close to each other

Playback order: 0 1 2 3 4 5 6 7 8 9

Frame type: I B B P B B P B B I

Data stream order: 0 2 3 1 5 6 4 8 9 7

Figure 3.8: MPEG B-frames postponed in data stream.

in the linear-order. In particular most of the zeros will appear as one large contiguous block at the end of the order. A form of run-length coding is used to compress the linear-order. It is coded as a sequence of (skip,value) pairs, where skip is the number of zeros before a value, and value is the value. The special pair (0,0) specifies the end of block. For example, the sequence [4,3,0,0,1,0,0,0,1,0,0,0,...] is represented as [(0,4),(0,3),(2,1),(3,1),(0,0)]. This sequence is then compressed using either arithmetic or Huffman coding. Which of the two coding schemes used is specified on a per-image basis in the header.

3.3.2 MPEG:

Correlation improves compression. This is a recurring theme in all of the approaches we have seen; the more effectively a technique is able to exploit correlations in the data, the more effectively it will be able to compress that data. This principle is most evident in MPEG encoding. MPEG compresses video streams. In theory, a video stream is a sequence of discrete images. In practice, successive images are highly interrelated. Barring cut shots or scene changes, any given video frame is likely to bear a close resemblance to neighboring frames. MPEG exploits this strong correlation to achieve far better compression rates than would be possible with isolated images. Each frame in an MPEG image stream is encoded using one of three schemes:

I-frame , or intra-frame, are coded as isolated images.

P-frame , or predictive coded frame, are based on the previous I- or P-frame.

B-frame , or bidirectionally predictive coded frame, are based on either or both the previous and next I- or P-frame.

Figure 3.8 shows an MPEG stream containing all three types of frames. I-frames and P-frames appear in an MPEG stream in simple, chronological order. However, B-frames are moved so that they appear after their neighboring I- and P-frames. This guarantees that each frame appears after any frame upon which it may depend. An MPEG encoder can decode any frame by buffering the two most recent I- or P-frames encountered in the data stream. Figure 3.8 shows how B-frames are postponed in the data stream so as to simplify decoder buffering. MPEG encoders are free to mix the frame types in any order. When the scene is relatively static, P- and B-frames could be used, while major scene changes could be encoded using I-frames. In practice, most encoders use some fixed pattern.

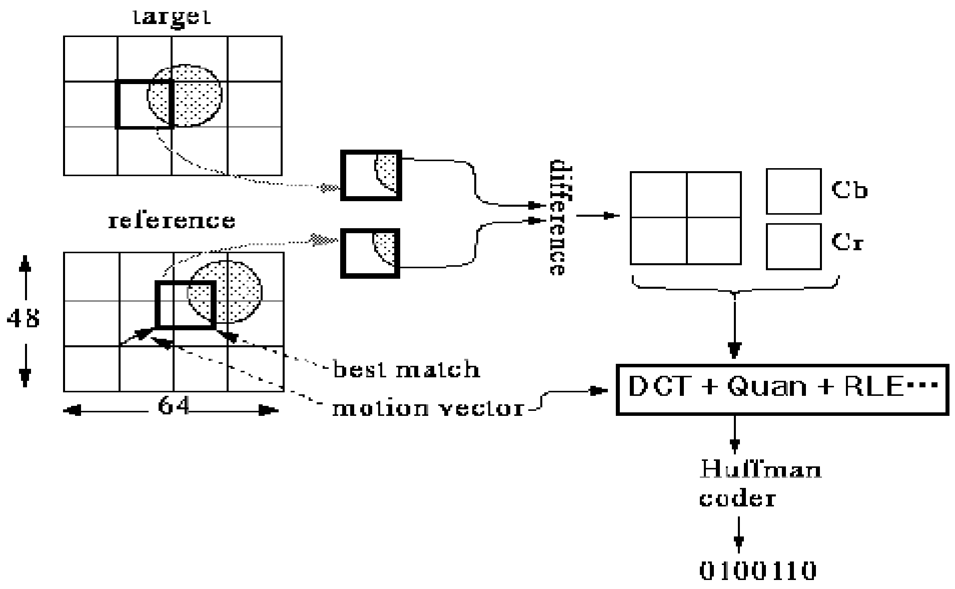


Figure 3.9: P-frame encoding.

Since I-frames are independent images, they can be encoded as if they were still images. The particular technique used by MPEG is a variant of the JPEG technique (the color transformation and quantization steps are slightly different). I-frames are very important for use as anchor points so that the frames in the video can be accessed randomly without requiring one to decode all previous frames. To decode any frame we need only find its closest previous I-frame and go from there. This is important for allowing reverse playback, skip-ahead, or error-recovery. The intuition behind encoding P-frames is to find matches, i.e., groups of pixels with similar patterns, in the previous reference frame and then coding the difference between the P-frame and its match. To find these “matches” the MPEG algorithm partitions the P-frame into 16x16 blocks. The process by which each of these blocks is encoded is illustrated in Figure 3.9. For each target block in the P-frame the encoder finds a reference block in the previous P- or I-frame that most closely matches it. The reference block need not be aligned on a 16-pixel boundary and can potentially be anywhere in the image. In practice, however, the x-y offset is typically small. The offset is called the motion vector. Once the match is found, the pixels of the reference block are subtracted from the corresponding pixels in the target block. This gives a residual which ideally is close to zero everywhere. This residual is coded using a scheme similar to JPEG encoding, but will ideally get a much better compression ratio because of the low intensities. In addition to sending the coded residual, the coder also needs to send the motion vector. This vector is Huffman coded. The motivation for searching other locations in the reference image for a match is to allow for the efficient encoding of motion. In particular if there is a moving object in the sequence of images (e.g., a car or a ball), or if the whole video is panning, then the best match will not be in the same location in the image. It should be noted that if no good match is found, then the block is coded as if it were from an I-frame. In practice, the search for good matches for each target block is the most computationally expensive part of MPEG encoding. With current technology, real-time MPEG encoding is only possible with the help of custom hardware. Note, however, that while the search for a match is expensive, regenerating the image as part of the decoder is cheap since the decoder is given the motion vector and only needs to look up the block from the previous image. B-frames were not present in MPEG's predecessor, H.261. They were added in an effort to address the following situation: portions of an intermediate P-frame may be completely absent from all previous frames, but may be present in future frames. For example, consider a car entering a shot from the side. Suppose an I-frame encodes the shot before the car has started to appear, and another I-frame appears when the car is completely visible. We would like to use P-frames for the intermediate scenes. However, since no portion of the car is visible in the first I-frame, the P-frames will not be able to "reuse" that information. The fact that the car is visible in a later I-frame does not help us, as P-frames can only look back in time, not forward. B-frames look for reusable data in both directions. The overall technique is very similar to that used in P-frames, but instead of just searching in the previous I- or P-frame for a match, it also searches in the next I- or P-frame. Assuming a good match is found in each, the two reference frames are averaged and subtracted from the target frame. If only one good match is found, then it is used as the reference. The coder needs to send some information on which reference(s) is (are) used, and potentially needs to send two motion vectors. How effective is MPEG compression? We can examine typical compression ratios for each frame type, and form an average weighted by the ratios in which the frames are typically interleaved. Starting with a 356×260 pixel, 24-bit color image, typical compression ratios forMPEG-I are:

Type	Size	Ratio
I	18 Kb	7:1

P	6 Kb	20:1
B	2.5 Kb	50:1
Avg	4.8 Kb	27:1

If one 356 × 260 frame requires 4.8 Kb, how much bandwidth does MPEG require in order to provide a reasonable video feed at thirty frames per second? 30frames/sec,4.8Kb/frame · 8b/bit = 1.2Mbits/sec Thus far, we have been concentrating on the visual component of MPEG. Adding a stereo audio stream will require roughly another 0.25 Mbits/sec, for a grand total bandwidth of 1.45 Mbits/sec. This fits nicely within the 1.5 Mbit/sec capacity of a T1 line. In fact, this specific limit was a design goal in the formation ofMPEG. Real-lifeMPEG encoders track bit rate as they encode, and will dynamically adjust compression qualities to keep the bit rate within some user-selected bound. This bit-rate control can also be important in other contexts. For example, video on a multimedia CD-ROM must fit within the relatively poor bandwidth of a typical CD-ROM drive.

3.3.3 MPEG in the Real World:

MPEG has found a number of applications in the real world, including:

1. Direct Broadcast Satellite. MPEG video streams are received by a dish/decoder, which unpacks the data and synthesizes a standard NTSC television signal.
2. Cable Television. Trial systems are sending MPEG-II programming over cable television lines.
3. Media Vaults. Silicon Graphics, Storage Tech, and other vendors are producing on-demand video systems, with twenty file thousand MPEG-encoded films on a single installation.
4. Real-Time Encoding. This is still the exclusive province of professionals. Incorporating special-purpose parallel hardware, real-time encoders can cost twenty to fifty thousand dollars.

3.4 Other Lossy Transform Codes:

3.4.1 Wavelet Compression:

JPEG and MPEG decompose images into sets of cosine waveforms. Unfortunately, cosine is a periodic function; this can create problems when an image contains strong aperiodic features. Such local high-frequency spikes would require an infinite number of cosine waves to encode properly. JPEG and MPEG solve this problem by breaking up images into fixed-size blocks and transforming each block in isolation. This effectively clips the infinitely-repeating cosine function, making it possible to encode local features. An alternative approach would be to choose a set of basis functions that exhibit good locality without artificial clipping. Such basis functions, called “wavelets”, could be applied to the entire image, without requiring blocking and without degenerating when presented with high-frequency local features. How do we derive a suitable set of basis functions? We start with a single function, called a “mother function”. Whereas cosine repeats indefinitely, we want the wavelet mother function, _, to be contained within some local region, and approach zero as we stray further away:

$$\lim_{x\to\pm\infty} \Phi(x)=0 \qquad (3.2)$$

The family of basis functions are scaled and translated versions of this mother function. For some scaling factor s and translation factor l, _sl(x) = _(2sx − l) A well know family of wavelets are the Haar wavelets, which are derived from the following mother function:

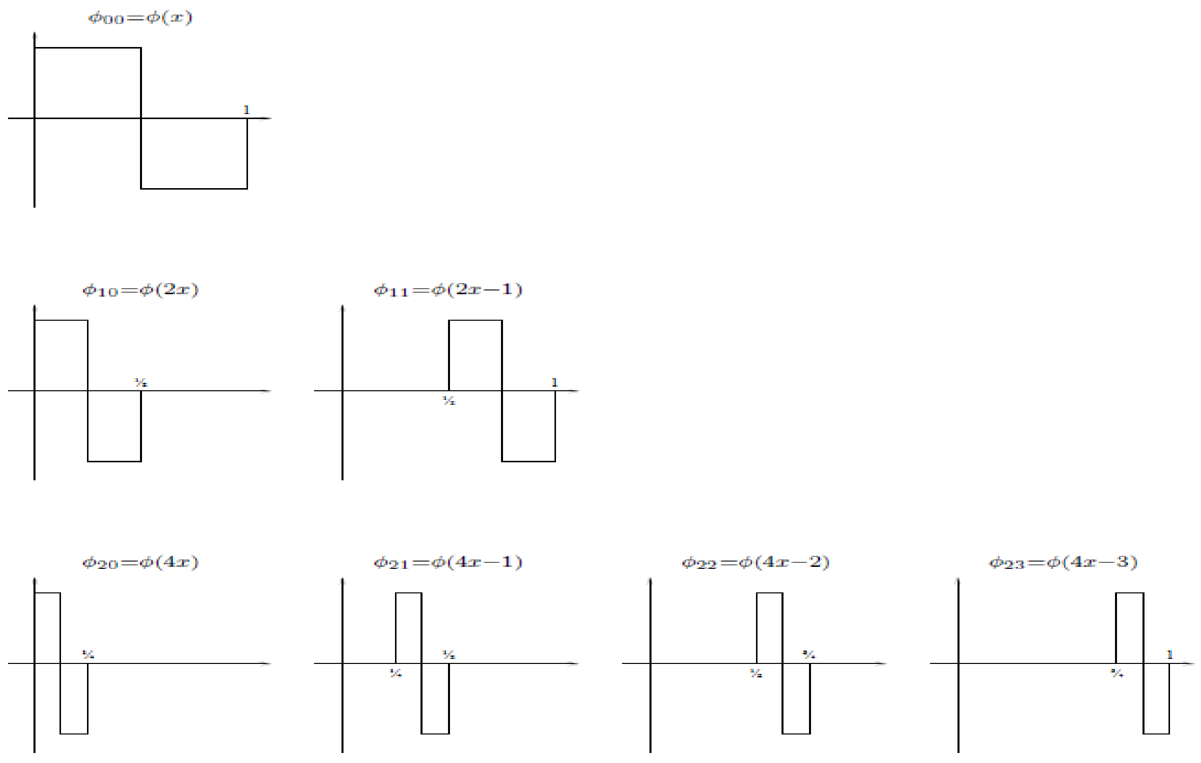
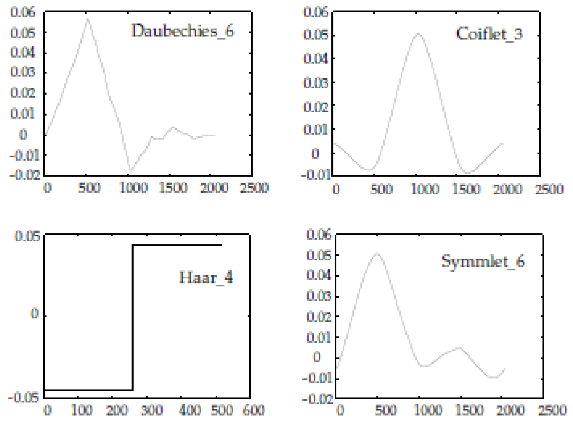
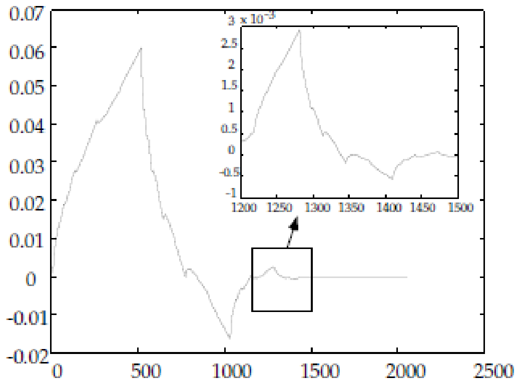


Figure 3.10: A small Haar wavelet family of size seven.

$$\Phi(x) = \begin{cases} 1: & 0 < x \leq 1/2 \\ -1: & 1/2 < x \leq 1 \\ 0: & x \leq 0 \text{ or } x > 1 \end{cases}$$

Figure 3.10 shows a family of seven Haar basis functions. Of the many potential wavelets, Haar wavelets are probably the most described but the least used. Their regular form makes the underlying mathematics simple and easy to illustrate, but tends to create bad blocking artifacts if actually used for compression. Many other wavelet mother functions have also been proposed. The Morret wavelet convolves a Gaussian with a cosine, resulting in a periodic but smoothly decaying function. This function is equivalent to a wave packet from quantum physics, and the mathematics of Morret functions have been studied extensively. Figure 4.11 shows a sampling of other popular wavelets. Figure 4.12 shows that the Daubechies wavelet is actually a self-similarity.





3.11: A sampling of popular wavelets. Figure 3.12: Self-similarity in the Daubechies wavelet.

3.4.2 Fractal Compression:

A function f(x) is said to have a fixed point x_f if $x_f = f(x_f)$. For example:

$$f(x) = ax + b \qquad (3.3)$$

$$= b/(1-a) \qquad (3.4)$$

may be a black box, whose formal definition is not known. In that case, we might try an iterative approach. Keep feeding numbers back through the function in hopes that we will converge on a solution:

x0 = guess

xi = f(xi−1)

For example, suppose that we have f(x) as a black box. We might guess zero as x0 and iterate from there:

x0 = 0

x1 = f(x0) = 1

x2 = f(x1) = 1.5

x3 = f(x2) = 1.75

x4 = f(x3) = 1.875

x5 = f(x4) = 1.9375

x6 = f(x5) = 1.96875

x7 = f(x6) = 1.984375

x8 = f(x7) = 1.9921875

In this example, f(x) was actually defined as 1 2x+1. The exact fixed point is 2, and the iterative solution was converging upon this value. Iteration is by no means guaranteed to find a fixed point. Not all functions have a single fixed point. Functions may have no fixed point, many fixed points, or an infinite number of fixed points. Even if a function has a fixed point, iteration may not necessarily converge upon it. In the above example, we were able to associate a fixed point value with a function. If we were given only the function, we would be able to recompute the fixed point value. Put differently, if we wish to transmit a value, we could instead transmit a function that iteratively converges on that value.

This is the idea behind fractal compression. However, we are not interested in transmitting simple numbers, like "2". Rather, we wish to transmit entire images. Our fixed points will be images. Our functions, then, will be mappings from images to images. Our encoder will operate roughly as follows:

1. Given an image, i, from the set of all possible images, Image.

2. Compute a function f : Image ! Image such that f(i) _ i.

3. Transmit the coefficients that uniquely identify f.

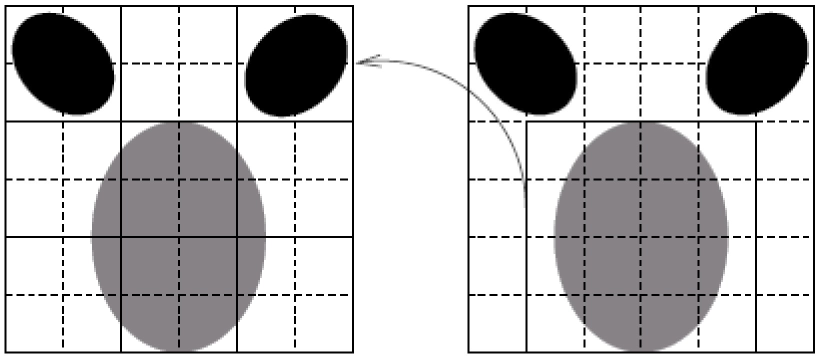


Figure 3.13: Identifying self-similarity.

Range blocks appear on the left; one domain block appears on the left. The arrow identifies one of several collage function that would be composited into a complete image.

Our decoder will use the coefficients to reassemble f and reconstruct its fixed point, the image:

1. Receive coefficients that uniquely identify some function f : Image ! Image.
2. Iterate f repeatedly until its value converges on a fixed image, i.
3. Present the decompressed image, i.

Clearly we will not be using entirely arbitrary functions here. We want to choose functions from some family that the encoder and decoder have agreed upon in advance. The members of this family should be identifiable simply by specifying the values for a small number of coefficients. The functions should have fixed points that may be found via iteration, and must not take unduly long to converge. The function family we choose is a set of “collage functions”, which map regions of an image to similar regions elsewhere in the image, modified by scaling, rotation, translation, and other simple transforms. This is vaguely similar to the search for similar macroblocks in MPEG P-frame and B-frame encoding, but with a much more flexible definition of similarity. Also, whereas MPEG searches for temporal self-similarity across multiple images, fractal compression searches for spatial self similarity within a single image. Figure 3.13 shows a simplified example of decomposing an image info collages of itself. Note that the encoder starts with the subdivided image on the right. For each “range” block, the encoder searchers for a similar “domain” block elsewhere in the image. We generally want domain blocks to be larger than range blocks to ensure good convergence at decoding time.

3.4.3 Model-Based Compression:

We briefly present one last transform coding scheme, model-based compression. The idea here is to characterize the source data in terms of some strong underlying model. The popular example here is faces. We might devise a general model of human faces, describing them in terms of anatomical parameters like nose shape, eye separation, skin color, cheekbone angle, and so on. Instead of transmitting the image of a face, we could transmit the parameters that define that face within our general model. Assuming that we have a suitable model for the data at hand, we may be able to describe the entire system using only a few bytes of parameter data. Both sender and receiver share a large body of a priori knowledge contained in the model itself (e.g., the fact that faces have two eyes and one nose). The more information is shared in the model, the less need be transmitted with any given data set. Like wavelet compression, model-based compression works by characterizing data in terms of a deeper underlying generator. Model-based encoding has found applicability in such areas as computerized recognition of four-legged animals or facial expressions

CHAPTER 4

METHODOLOGY

In this thesis we use FFT, DCT, DCT-II,DST and BLASCHKE UNWINDING AFD transformations for ECG data compression.

4.1 Discrete Cosine Transform(DCT):

The Discrete Cosine Transform (DCT) was developed to approximate Karhunen-Loeve Transform (KLT) when there is high correlation among the input samples, which is the case in many digital waveforms including speech, music, and biomedical signals. The DCT D = [d_0 d_1 d_2 d_3………..d_{N1-1}]^T Of the vector x is defined as follows

$$d_0 = \frac{1}{\sqrt{N}} \sum_{n=0}^{N-1} x_{n_1} \qquad (4.1)$$

$$d_k = \sqrt{\frac{2}{N}} \sum_{n1=0}^{N-1} x_{n_1} \cos \frac{(2n_1+1)K\pi}{2N}, \qquad k = 1,2,\ldots\ldots\ldots\ldots\ldots.. N\text{-}1 \qquad (4.2)$$

Where d_k is the k_{th} DCT coefficient. The inverse discrete cosine transform (IDCT) of d is given by

$$x_{n_1} = \frac{1}{\sqrt{N}} d_0 + \sqrt{\frac{2}{N}} \sum_{k=0}^{N-1} d_k \cos \frac{(2n_1+1)K\pi}{2N} \quad n_1=0,1,2\ldots\ldots\ldots\ldots\ldots\ldots..N\text{-}1 \qquad (4.3)$$

There exist fast algorithms, Order (*N*log*N*), to compute the DCT .Thus, DCT can be implemented in a computationally efficient manner. Two recent image and video coding standards, JPEG and MPEG, use DCT as the main building block. A discrete cosine transform (DCT) expresses a sequence of finitely many data points in terms of a sum of cosine functions oscillating at different frequencies. DCTs are important to numerous applications in science and engineering, from lossy compression of audio (e.g. MP3) and images (e.g. JPEG) (where small high frequency components can be discarded), to spectral methods for the numerical solution of partial differential equations. The use of cosine rather than sine functions is critical in these applications. For compression, it turns out that cosine functions are much more efficient whereas for differential equations the cosines express a particular choice of boundary conditions. In particular, a DCT is a Fourier-related transform similar to the discrete Fourier transform (DFT), but using only real numbers. DCTs are equivalent to DFTs of roughly twice the length, operating on real data with even symmetry (since the Fourier transform of a real and even function is real and even), where in some variants the input and/or output data are shifted by half a sample. Discrete Cosine Transform is a basis for many signal and image compression algorithms due to its high decorrelation and energy compaction property. A discrete Cosine Transform of *N* sample is defined as

$$F(u) = \sqrt{\frac{2}{N}}\, C(u) \sum_{x1=0}^{N-1} f(x_1) \cos\frac{(2x_1+1)u\pi}{2N} \qquad u = 0,1,2\ldots\ldots\ldots\ldots\ldots..N\text{-}1 \qquad (4.4)$$

Where

$$C(u) = \begin{cases} \frac{1}{\sqrt{2}}, & for\ u = 0 \\ 1, & otherwise \end{cases}$$

The function *f(x)* represents the value of x_{th} samples of input signals. *F(u)* represents DCT coefficients. The inverse DCT is defined in similar fashion as

$$f(x_1) = \sqrt{\frac{2}{N}}\, C(u) \sum_{u=0}^{N-1} C(u)\, F(u) \cos\frac{(2x_1+1)u\pi}{2N} \qquad x_1 = 0,1,2\ldots\ldots\ldots\ldots N\text{-}1 \qquad (4.5)$$

4.2 Discrete Sine Transform:

Discrete sine transform (DST) is a Fourier-related transform similar to the discrete Fourier transform (DFT), but using a purely real matrix. It is equivalent to the imaginary parts of a DFT of roughly twice the length, operating on real data with odd symmetry (since the Fourier transform of a real and odd function is imaginary and odd), where in some variants the input and/or output data are shifted by half a sample. Like any Fourier-related transform, discrete sine transforms (DSTs) express a function or a signal in terms of a sum of sinusoids with different frequencies and amplitudes. Like the discrete Fourier transforms (DFT), a DST operates on a function at a finite number of discrete data points. The obvious distinction between a DST and a DFT is that the former uses only sine functions, while the latter uses both cosines and sines (in the form of complex exponentials). However, this visible difference is merely a consequence of a deeper distinction: a DST implies different boundary conditions than the DFT or other related transforms.

Formally, the discrete sine transform is a linear, invertible function F: R^N -> R^N (where R denotes the set of real numbers), or equivalently an N × N square matrix. There are several variants of the DST with slightly modified definitions. The N real numbers $x_0,\dots x_{N-1}$ are transformed into the N real numbers $X_0,\dots X_{N-1}$ according to

$$X_k = \sum_{n=0}^{N-1} x_n \sin\frac{\pi}{N+1}(n+1)(k+1) \qquad k = 0,1,\dots\dots N\text{-}1 \qquad (4.6)$$

4.3 Fast Fourier Transform (FFT) :

A fast Fourier transform (FFT) is an efficient algorithm to compute the discrete Fourier transform (DFT) and it's inverse. There are many distinct FFT algorithms involving a wide range of mathematics, from simple complex-number arithmetic to group theory and number theory. A DFT decomposes a sequence of values into components of different frequencies but computing it directly from the definition is often too slow to be practical. An FFT is a way to compute the same result more quickly. Computing a DFT of *N* points in the naive way, using the definition, takes O(*N*2) arithmetical operations , while an FFT can compute the same result in only O(*N* log *N*) operations.

Fast Fourier Transform is a fundamental transform in digital signal processing with applications in frequency analysis, signal processing etc. The periodicity and symmetry properties of DFT are useful for compression. The uth FFT coefficient of length N sequence {f(x)} is defined as

$$F(u) = \sum_{x=0}^{N-1} f(x)\, e^{\frac{-j2\pi ux}{N}} \qquad u = 0,1,2\dots\dots\dots\dots N\text{-}1 \qquad (4.7)$$

And its inverse transform is calculated from

$$f(x) = \frac{1}{N}\sum_{u=0}^{N-1} F(u)\, e^{\frac{j2\pi ux}{N}} \qquad x = 0,1,2\dots\dots\dots N\text{-}1 \qquad (4.8)$$

4.4. Discrete Cosine Transform–II (DCT – II):

The most common variant of discrete cosine transform is the type-II DCT [54]. The DCT-II is typically defined as a real, orthogonal (unitary), linear transformation by the formula

$$C_k^{II}=\sqrt{\frac{2-\delta_{k,0}}{N}}\sum_{n=0}^{N-1} x_n \cos\left[\frac{\pi}{N}\left(n+\frac{1}{2}\right)k\right] \qquad (4.9)$$

for N inputs x_n and N outputs C_k^{II}, where $\delta_{k,0}$ is the Kronecker delta (= 1 for k = 0 and = 0 otherwise). DCT-II can be viewed as special case of the discrete Fourier transform (DFT) with real inputs of certain symmetry. This viewpoint is fruitful because it means that any FFT algorithm for the DFT leads immediately to a corresponding fast algorithm for the DCT-II simply by discarding the redundant operations. The discrete Fourier transform of size N is defined by

$$X_K=\sum_{n=0}^{N-1} x_n \qquad (4.10)$$

where $\omega_N=e^{-2\pi i N}$ is an Nth primitive root of unity. In order to relate this to the DCT-II, it is convenient to choose a different normalization for the latter transform as

$$C_K=2\sum_{n=0}^{N-1} x_n \cos\left[\frac{\pi}{n}\left(n+\frac{1}{2}\right)k\right] \qquad (4.11)$$

$$2\cos\left(\frac{\pi l}{N}\right)=\omega_{4N}^{2l}+\omega_{4N}^{4N-2l} \qquad (4.12)$$

$$C_K=2\sum_{n=0}^{N-1} x_n \cos\left[\frac{\pi}{N}(n+1)k\right] \qquad (4.13)$$

$$C_K=\sum_{n=0}^{N-1} x_n\,\omega_{4N}^{(2n+1)k}+\sum_{n=1}^{N-1} x_n\,\omega_{4N}^{(4N-2n-1)k} \qquad (4.14)$$

Thus, the DCT-II of size N is precisely a DFT of size 4N, of real-even inputs, where the even-indexed inputs are zero.

4.5 Splines and B-splines:

A spline function consists of polynomial pieces on subintervals joined together by continuity conditions.The segmented nature allows splines to adjust very efficiently to the local characteristics of the data and represent it better (i.e. with smaller deviations) than other classes of functions.

Definition 4.5.1. A function f(x), defined on a finite interval [a, b], is called a spline function of order k > 0,having as knots the sequence t = {t_0,t_1,. . . ,t_{n+l}}(t_0 = a, & t_{n+l} = b) such that ti < t_{i+k} (the knots coordinates ti may not be distinct), if the following two conditions are satisfied:

1. In each knot interval [t<, t_{i+l}], f(x) is given by a polynomial of degree k - 1 at most.

2. At any knot t_i such that $t_{i-1} < t_i =\ldots = t_{i+l} < t_{i+l+1}$, the function f(x) has continuous k - l - 2 derivatives (and is discontinuous at t_i if l = k - 1).

The vector space of functions satisfying Definition 2.1 will be denoted by $S_{k,t}$. The dimension of the Vector space $S_{k,t}$ is

$$\dim(S_{k,t}) = n + k \qquad (4.15)$$

To perform computations with splines, one must choose a suitable representation in which any member of $S_{k,t}$ can be written as a unique linear combination of n + k basis functions such that Definition 4.2.1 is automatically satisfied. A common choice is to use B-splines. Computations with B-splines are particularly convenient, due to their local-support property,i.e. they are nonzero only over a finite interval, Moreover, B-splines are a unique minimum-support basis the only set of basis functions in which each covers the minimum number of knots. Using B-splines, curve-fitting problems are easy to pose and lead to well-conditioned, banded positive-definite systems. They also provide an easy to manipulate representation for splines having different degrees of smoothness at each knot.

Definition 4.2.2. A B-spline $B_{i,k,t}(x)$ of order k > 0,with knots ti, . . , t_{i+k}, can be defined using the following recurrence relation:

$$B_{i,1,t}(x) = \begin{cases} 1, & if\ t_i \le x < t_{i+1} \\ 0, & otherwise \end{cases} \qquad (4.16)$$

and

$$\mathrm{B}_{\mathrm{i,k,t}}(x)=\frac{t_{i+r}-x}{t_{i+r}-t_{i+1}}B_{i+1,r-1,t}(x)+\frac{x-t_i}{t_{i+r-1}-t_i}B_{i,r-1,t}(x) \qquad (4.17)$$

for r = 2,3,. . . , k, where we interpret the terms

$$\frac{t_{i+r}-x}{t_{i+r}-t_{i+1}}\ and\ \frac{x-t_i}{t_{i+r-1}-t_i}$$

as zero whenever $t_{i+r}-t_{i+1}$= 0 and $t_{i+r-1}-t_i$ = 0, respectively.

From this definition one can observe that B-splines have local support

$$B_{i,k,t}(x)=0\ if\ x\in[t_i,t_{i+k}] \qquad (4.18)$$

$$t_{-k+1}\le t_{-k+2}\le\cdots t_{-1}\le t_0=a,$$

$$\mathrm{b}=t_{n+1}\le\ t_{n+2}\ \le\ldots\ldots\ldots\ldots t_{n+k-1}\le\ t_{n+k}$$

but which are otherwise arbitrary. Every function f(x) satisfying Definition 4.2.1 then has a unique representation (the Curry-Schoenberg Theorem)

$$\mathrm{f(x)}=\textstyle\sum_{i=-k+1}^{n}a_i\,c, \qquad (4.19)$$

in which a_i is called the ith B-spline coefficient of f(x). The upper bound for the absolute value of the spline function f(x) is given by [55]

$$\max_x|f(x)|\le\ \max_i|a_i| \qquad (4.20)$$

The most common choice for boundary knots are coincident knots

$$t_{-k+1}=t_{-k+2}=\cdots t_{-1}=t_0=a,$$

$$\mathrm{b}=t_{n+1}=\ t_{n+2}\ =\ldots\ldots\ldots\ldots t_{n+k-1}=t_{n+k}$$

This choice implies that all B-splines vanish outside the interval [a, b] and allows very easily to impose the boundary conditions

$$f(a)=a_{-k+1}\qquad f(b)=\ a_n \qquad (4.21)$$

The knots t_1,..., t_n are called interior knots, and in the data reduction scheme, these are the only knots which can be eliminated by the knot removal algorithm. One of the most powerful tools in studying B-splines also used extensively in this algorithm is an explicit formula which allows us to write basis functions $B_{i,k,\rho}(x)$ defined for the knot sequence ρ = t\{t_p}, t = {t_{-k+1},. . .,t_{n+k}}, in terms of basis functions $B_{i,k,t}(x)$ defined for t [56]:

$$f(x) = \begin{cases} B_{i,k,t}(x), & if\ i \le p-k-1 \\ \frac{t_{i+r}-x}{t_{i+r}-t_{i+1}} B_{i+1,r-1,t}(x) + \frac{x-t_i}{t_{i+r-1}-t_i} B_{i,r-1,t}(x) & if\ p-k \le i \le p-1 \\ B_{i+1,k,t}(x), & if\ p \le i \le n-1 \end{cases} \qquad (4.22)$$

4.6: Continuous Hermite functions:

Consider the family of polynomials $H_l(x)$, $l \geq 0$, that satisfy the recursion

$$H_l(t) = \ 2tH_{l-1}(t) - 2(l-1)H_{l-2}(t) \qquad (4.23)$$

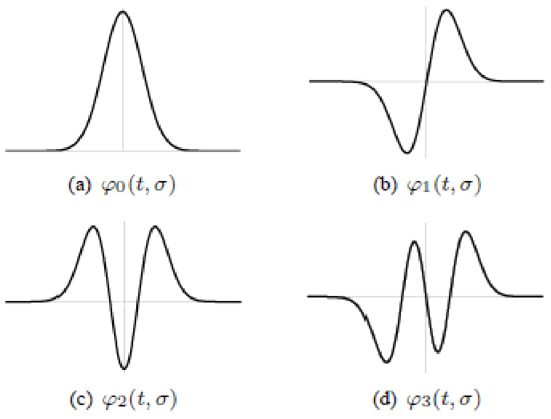


Fig. 4.1. First four Hermite functions (plotted for the same scale σ).

for $l \geq 2$, with H$_0$(t) = 1 and $H_l(t)$ = 2t. They are known as Hermite polynomials. These polynomials are orthogonal on the real line ℝ with respect to the weight function e^{t^2}:

$$\int_{-\infty}^{\infty} H_l(t)\, H_m(t)\, e^{t^2} dt = 2^l\, l!\, \sqrt{\pi}\delta_{l-m} \qquad (4.24)$$

$$\varphi_l(t,\sigma) = \frac{1}{\sqrt{\sigma 2^l l! \sqrt{\pi}}} e^{-t^2/2\sigma^2} H_l(t/\sigma) \qquad (4.25)$$

are orthonormal on ℝ with respect to the standard inner product

$$\langle \varphi_l(t,\sigma), \varphi_m(t,\sigma) \rangle = \int \varphi_l(t,\sigma), \varphi_m(t,\sigma) dt = \delta_{l-m} \qquad (4.26)$$

The set of functions {ϕℓ(t, σ)}ℓ≥0, called continuous Hermite functions, is an orthonormal basis in the Hilbert space of continuous functions defined on ℝ [57]–[58]. Any such function s(t) can be represented as a linear combination of the basis functions

$$s(t) = \sum_{l \geq 0} c_l\, \varphi_l(t,\sigma) \qquad (4.27)$$

where

$$c_l = \langle s(t), \varphi_l(t,\sigma)\rangle = \int s(t), \varphi_l(t,\sigma)dt \qquad (4.28)$$

The first four continuous Hermite functions are shown in Fig. 4.1. Notice that each $\phi\ell(t, \sigma)$ quickly approaches zero as the value of |t| increases: since $H\ell(t/\sigma)$ is a polynomial of degree ℓ,

$$\lim_{|t|\to\infty} e^{-t^2/2\sigma^2} H_l(t/\sigma) = 0. \qquad (4.29)$$

As a consequence, for practical purposes we can assume that each continuous Hermite function has a compact support. Since in this thesis we often work only with the first L continuous Hermite functions, we assume that $\varphi_0(t,\sigma), \varphi_1(t,\sigma). . . , \varphi_{L-1}(t,\sigma)$have the same compact support $[-T_\sigma, T_\sigma]$, where T_σ is a suitably chosen constant that depends on σ and L. In other words, we assume

$$\varphi_l(t,\sigma) = 0 \; for \; t \; \in [-T_\sigma, T_\sigma], \qquad (4.30)$$

where $0 \le \ell < L$. If a signal s(t) also has a compact support of $[-T_\sigma, T_\sigma]$, then we can compute the coefficients c_ℓ using a finite integral:

$$c_l = \int s(t), \varphi_l(t,\sigma)dt = \int_{-T_\sigma}^{T_\sigma} s(t), \varphi_l(t,\sigma)dt \qquad (4.31)$$

4.7 Blaschke unwinding AFD:

Algorithm 1 illustrates how the Blaschke unwinding AFD is applied to compress a real-valued signal. First, the input real-valued signal F is projected to H^2 space and we get F^+. In practice, we could safely assume that

$$2\Re F^+ = F + c_0 \qquad (4.32)$$

holds, where $\Re$ means taking the real part and c_0 is the zeroth Fourier coefficient of F. c_0 is the first data point we save for the signal compression, and F^+ is initialized as the first remainder F1.

Algorithm I : Blaschke Unwinding AFD based Compression

Input: Real-valued input signal F, sets of parameters a∈ D

and the decomposition level N.

```
Output: ${c_n}_{n=1}^{N}$, ${a_n}_{n=1}^{N}$ and a finite number of zeros $\left\{\left\{r_{n_j}\right\}_{j=1}^{M_n}\right\}_{n=1}^{N}$
l: Get the projection signal $F^+$ of F in the Hardy space.
2: Initialize $F_l = F^+$.
3: for n = 1 to N do.
4: Obtain the inner function $I_n$ and outer function $O_n$ of $F_n$ so that $F_n = I_n O_n$;
5: Get zeros $\left\{r_{n_j}\right\}_{j=1}^{M_n}$, of $I_n$ by Algorithm 2;
6: Get $a_n = \arg\max\{(1-|a|^2)|O_n(a)|^2 : a \in D\}$;
7: Get $c_n = \langle O_n e_{a_n} \rangle$;
8: Get $F_{n+1} = \frac{F_n - c_n I_n e_{a_n}}{I_n} \frac{1-\overline{a_n}z}{z-a_n}$;
9: return ${c_n}_{n=1}^{N}$, ${a_n}_{n=1}^{N}$, $\left\{\left\{r_{n_j}\right\}_{j=1}^{M_n}\right\}_{n=1}^{N}$.
```

Second, extract the inner function by calculating zeros of F_1 by the method introduced in [59], where we assume that F_1 has finite roots on $\overline{D} := \{z \in C| \ \|z\| \leq 1\}$ [59]. The detailed steps of numerical calculation for calculating zeros of F_1 are performed in Algorithm 2. Then accordingly, get the outer function O_1 by the Nevanlinna factorization. Third, The set of $\{a_n\}$, n = 1, 2, . . . , in consisting of discrete points in $\mathbb{D}$ is generated by dividing $\mathbb{D}$ into rectangular grid to get the TM system and evaluators $\{e_a\}$. Then, the decomposition of O_1 is based on the TM system. During the decomposition, the maximal selection principle is applied in the selection of a1 with the aid of evaluators. Suppose the decomposition level is N ∈ N. Iterate the above three steps, each on the remainder of the previous step, for N times, and we end up with $\{e_{a_n}\}$, the modified Blaschke products $\{B_n\}$, and M_n zeros, for n = 1, . . . N. As a result, we obtain 2N + 1 parameters, including$\{c_n\}_{n=0}^{N}$, and $\{a_n\}_{n=1}^{N}$, as well as $\sum_{n=1}^{N} M_n$ zeros. c_n and a_n, where n = 1, . . . N, as well as $\sum_{n=1}^{N} M_n$ zeros, are other data points we save for the data compression.

Algorithm II: Procedure for calculating zeros

```
Input: F, $\delta > 0$
Output: zeros of F, $\left\{r_{n_j}\right\}_{j=1}^{M_n}$
l: Determine for $M$, $M = \frac{1}{2\pi i}\int_{|z|=1}\frac{F'(z)}{F(z)}dz$
2: Initialize $G_1$ : $F$
3: for j = 1 to $M_1$ do.
4: Evaluate arg $\min_{z\in D_{1-\delta}}\left|G_j(z)\right|$;
5: Get $r_j$ satisfying $G_j(r_j) = 0$;
6: Get $G_{j+1} \coloneqq G_j\,\frac{1-r_j z}{z-r_j}$ ;
7: return $\{r_j\}_{j=1}^{M_n}$.
```

CHAPTER 5
RESULT ANALYSIS

We used data in the MIT-BIH database to test the performance of the six coding techniques. The ECG data is sampled at 142Hz and the resolution of each sample is 11bits/samples. The amount of compression is measured by CR and the distortion between the original and reconstructed signal is measured by PRD. A data compression algorithm must represent the data with acceptable fidelity while achieving high CR.

5.1 Performance Evaluation:

The effectiveness of an ECG compression technique is described in terms of: Percentage Mean Square Difference (PRD) and Compression Ratio (CR).

5.1.1 Compression Ratio (CR):

CR is the ratio of the original data to compressed data without taking into account factors such as bandwidth, sampling frequency, precision of the original data, word- length of compression parameters, reconstruction error threshold, database size, lead selection, and noise level. It is given by:

CR = Bit rate of original file / Bit rate of reconstructed file ……………… (5.1)

That is, Higher the CR, smaller the size of the compressed file.

5.1.2. Percentage Mean Square Difference (PRD):

Percentage Mean Square Difference (PRD) is a measure of error loss. This measure evaluates the distortion between the original and the reconstructed signal.

PRD calculation is as follows:

$$\text{PRD} = \sqrt{\frac{\sum (X_i - X_{i2})^2}{\sum X_i^2}} \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \quad (5.2)$$

Where X_i is the original file and X_{i1} is its reconstructed version.

We used data in the CSE database to test the performance of the five coding techniques. The ECG data is sampled at 333 Hz. The amount of compression is measured by CR and the distortion between the original and reconstructed signal is measured by PRD.

5.2 Compression Algorithm:

5.2.1 FFT:

- Separate the ECG components into three components x, y, z.
- Find the frequency and time between two samples.
- Find the FFT of ECG signal and check for FFT coefficients (before compression)

  = 0, increment the counter A if it is between +25 to-25 and assign to Index=0.
- Check for FFT coefficients (after compression) =0, increment the Counter B.
- Calculate inverse FFT and plot decompression, error.
- Calculate the compression ratio CR and PRD.

  Figure 5.1 shows the original ECG signal record 100 which are selected from MIT-BIH arrhythmia database and its reconstructed waveform when compressed by FFT.

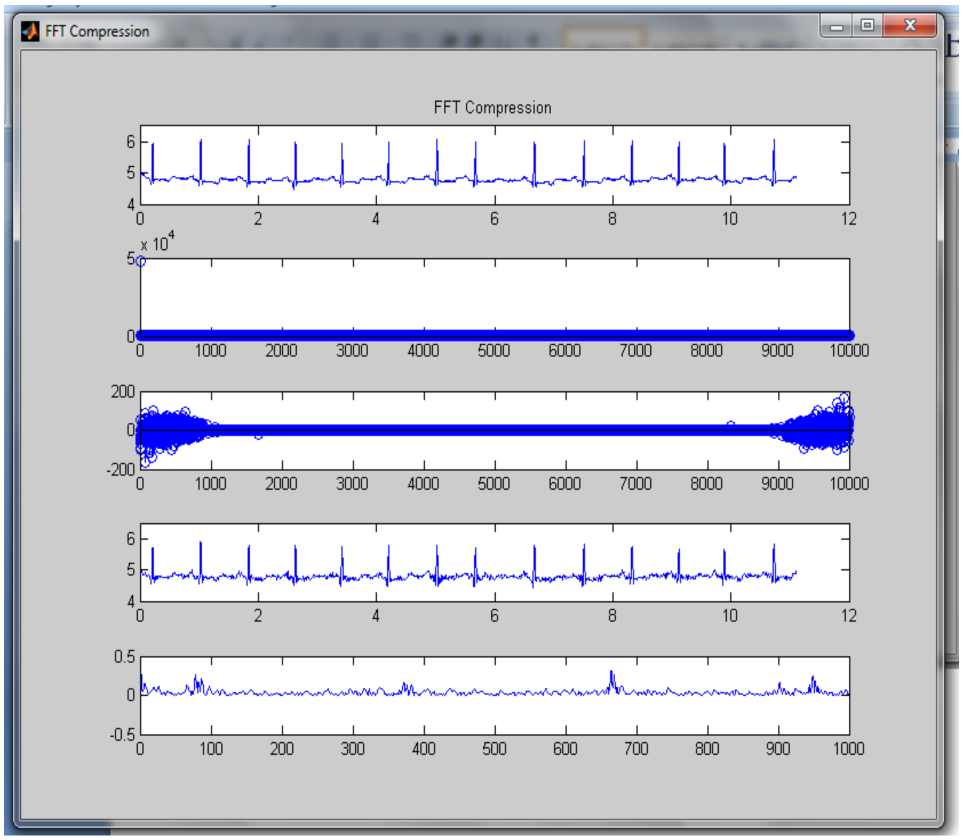


Figure 5.1 FFT compression of MIT-BIH record 100

Limitation of FFT is it fails to provide the information regarding the exact location of frequency component in time

5.2.2. Discrete Cosine Transform (DCT):

- Separate the ECG components into three components x, y, z.
- Find the frequency and time between two samples.
- Find the DCT of ECG signal and check for DCT coefficients (before compression) = 0, increment the counter A if it is between +0.22 to-0.22 and assign to Index=0.
- Check for DCT coefficients (after compression) =0, increment the Counter B.
- Calculate inverse DCT and plot decompression, error.
- Calculate the compression ratio CR and PRD.

Fig.5.2 shows the original ECG signal record 100 which are selected from MITBIH arrhythmia database and its reconstructed waveform when compressed by DCT.

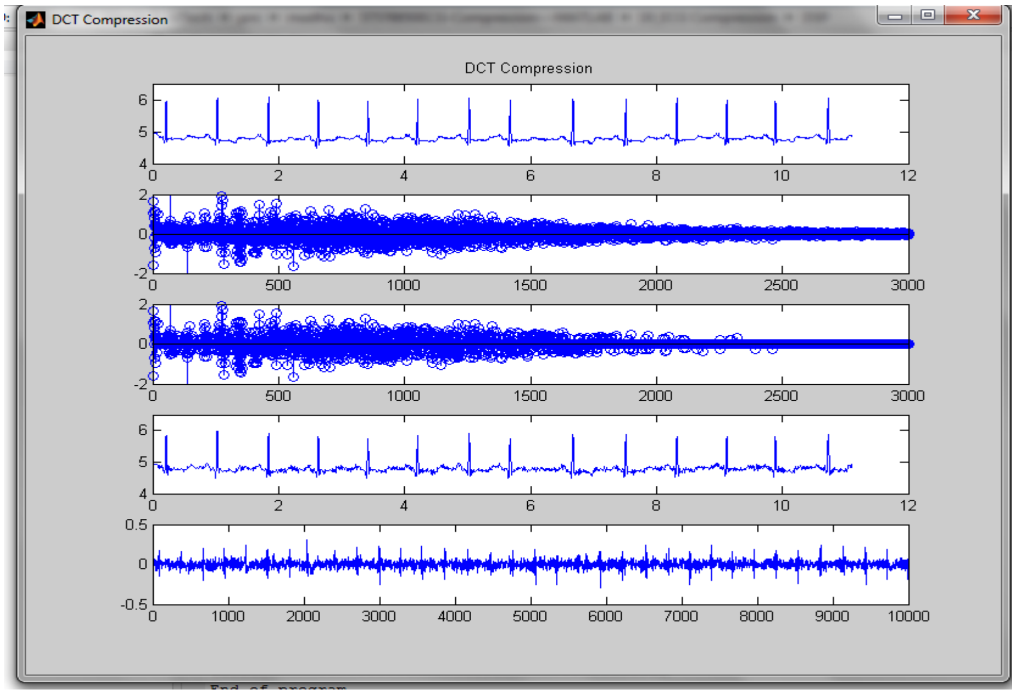


Figure 5.2 DCT compression of MIT-BIH record 100

- **Limitation of DCT:**

  Distortion is more in reconstructed signal.

5.2.3. Discrete sine Transform (DST):

- Separate the ECG components into three components x, y, z.
- Find the frequency and time between two samples.
- Find the DST of ECG signal and check for DST coefficients (before compression)

  = 0, increment the counter A if it is between +15 to - 15 and assign to Index=0.
- Check for DST coefficients (after compression) =0, increment the Counter B.
- Calculate inverse DST and plot decompression, error.
- Calculate the compression ratio CR and PRD.

  Figure 5.3 shows the original ECG signal record 100 which are selected from MIT-BIH arrhythmia database and its reconstructed waveform when compressed by DST.

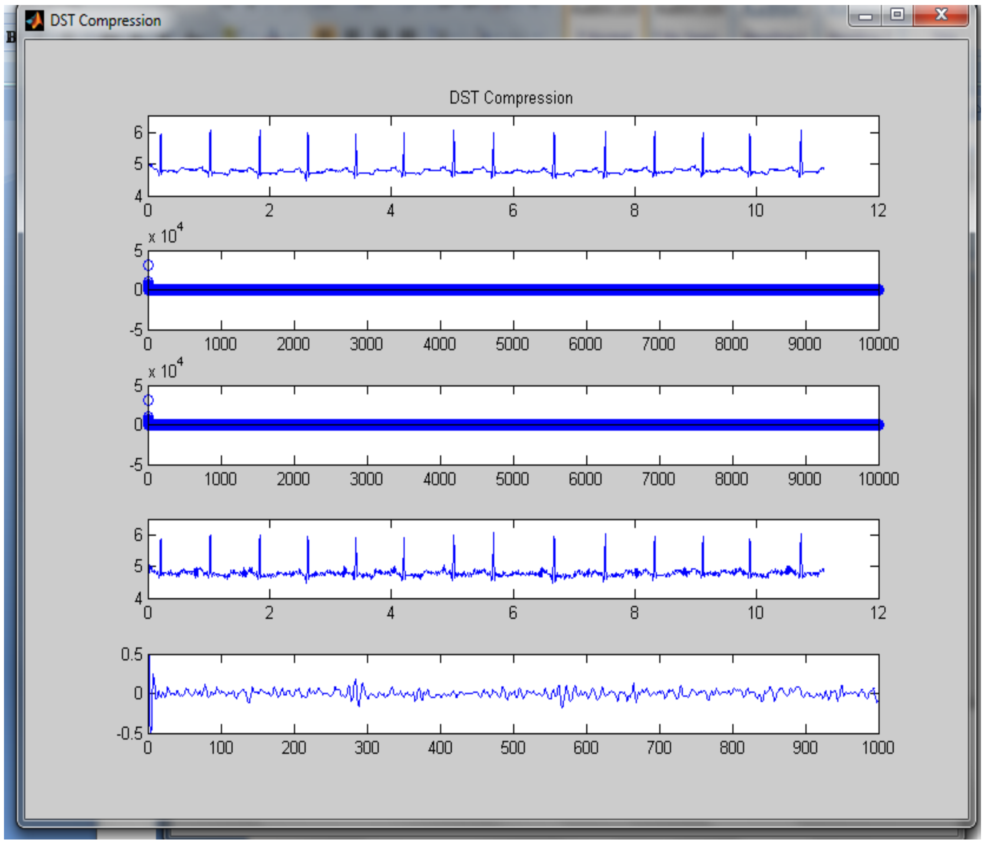


Figure 5.3 DST compression of MIT-BIH record 100

5.2.4. Discrete Cosine Transform-2 (DCT-2):

- Partition of data sequence x in N_b consecutive blocks bi, i =0, 1, ¼.., Nb-1, each one with Lb samples.
- DCT computation for each block.
- Quantization of the DCT coefficients.
- Lossless encoding of the quantized DCT coefficients.

Figure 5.4 shows the original ECG signal record 100 which are selected from MIT-BIH arrhythmia database and its reconstructed waveform when compressed by DCT-2.

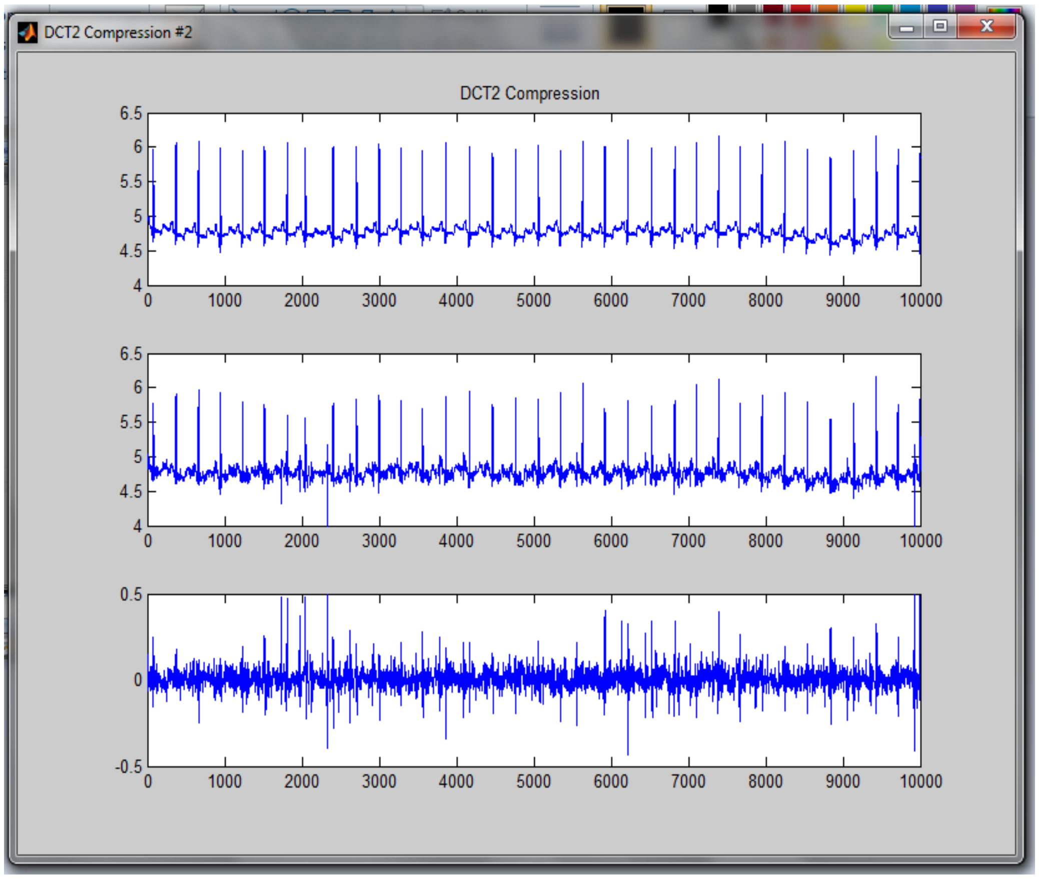


Figure 5.4 DCT-2 compression of MIT-BIH record 100

5.2.5. Blaschke unwinding AFD:

The compression consists of two steps. The first step carries out the Hardy projection and the Blaschke unwinding AFD compression. The second step is the lossless Huffman encoding.

For the decompression, it is the inverse of the compression, including the Huffman decoding and the inverse Blaschke unwinding AFD process.

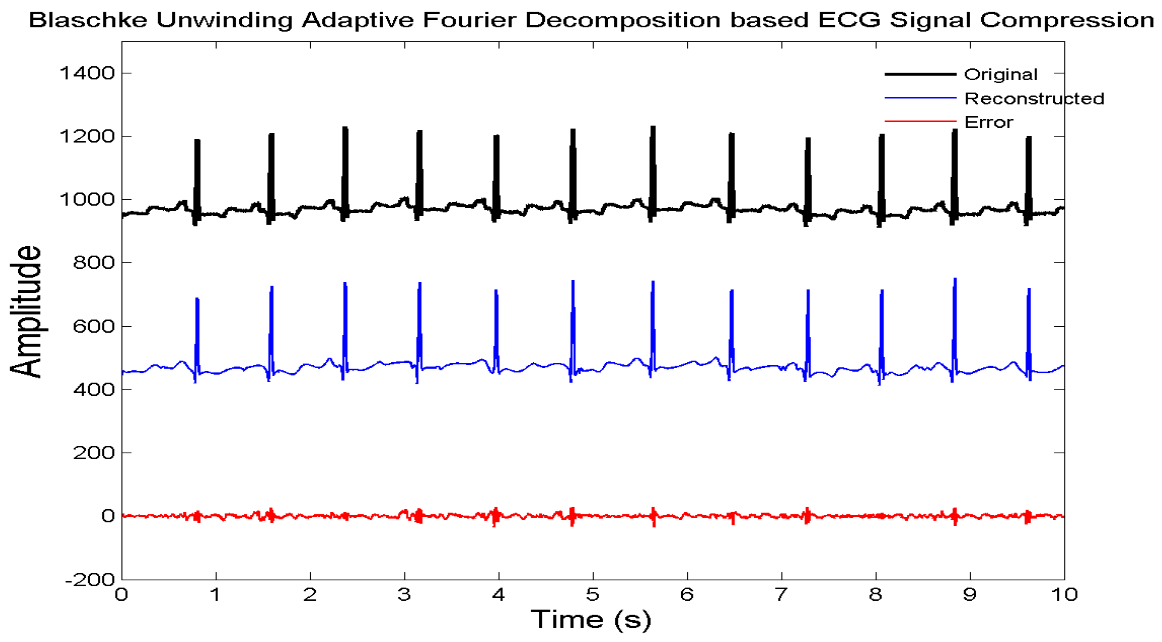


Figure:5.5 Waveforms of original, reconstructed and error signals with N = 8 taken from record 100

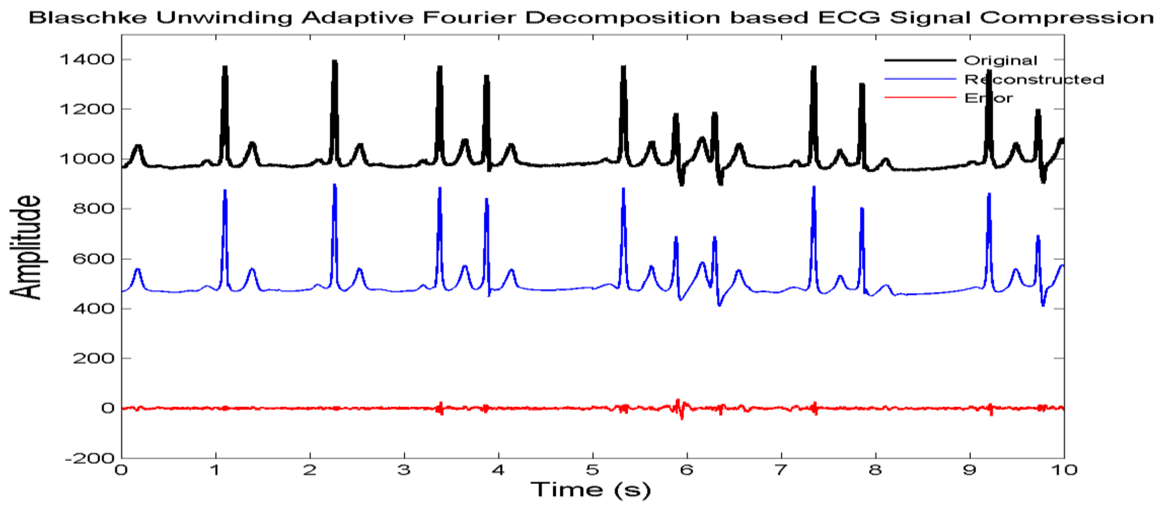


Figure:5.6 Waveforms of original, reconstructed and error signals with N = 10 taken from record 100

The comparison table shown in Table 5.1 details the resultant compression techniques. This gives the choice to select the best suitable compression method. A data compression algorithm must represent the data with acceptable fidelity while achieving high CR. As the PRD indicates reconstruction fidelity; the increase in its value is actually undesirable. Blaschke unwinding AFD which leads to a high compression rate with a high fidelity. Compared with existing algorithms, like FFT, DCT, DST and DCT-2.

Table 5.1 Comparision of resultant compression techniques

Method	**CR**	**PRD**
FFT	16.01	1.10
DCT	16.87	1
DST	11.62	1.19
DCT2	22.21	1.27
Blaschke unwinding AFD(N=8)	39.34	0.71
Blaschke unwinding AFD(N=10)	26.09	0.57

CONCLUSION

Among the five techniques presented, DST provides lowest CR and distortion is also high. DCT improves CR and lowers PRD. Next is FFT which gives CR 16.01 with PRD as 1.10. But DCT-II provides an improvement in terms of CR of 22.21 but PRD increases up to 1.27. Thus an improvement of a discrete cosine transform (DCT)-based method for electrocardiogram (ECG) compression is presented as DCT-II in terms of amount of compression. From table 5.1 we can observe that using Blaschke unwinding AFD based compression we are getting higher compression rate in comparision to other compression technique. Hence this technique is better than other ECG compression technique in both point of consideration.

REFERENCES

[1] Houghton, A. R. and Gray D. 2003. Making sense of the ECG: A Hands-on Guide. Aold Publishing Company.

[2] Singh B., Singh D., Jaryal A.K. and Deepak K.K. 2012. Ectopic beats in approximate entropy and sample entropy-based HRV assessment. International Journal of Systems Science, 43(5), 884-893.

[3] Cox J.R., Nolle F.M., Fozzard H.A., and Oliver G.C. 1968. AZTEC, a preprocessing program for real-time ECG rhythm analysis. IEEE Trans. Biomed. Eng., 15, 128-129.

[4] Mitra M., Bera J.N. and Gupta R. 2012. Electrocardiogram compression technique for global system of mobile-based offline telecardiology application for rural clinics in India. IET Sci. Meas. Technol., 6(6), 412-419.

[5] Kulkarni P.K., Kumar V. and Verma H.K. 1997. Direct data compression techniques for ECG signals: effect of sampling frequency on performance. International Journal of Systems Science, 28(3), 217-228.

[6] Koski A. 1997. Lossless ECG encoding. Comput. Methods and Programs Biomed., 52(1), 23–33.

[7] Kumar V., Saxena S.C. and Giri V.K. 2006. Direct data compression of ECG signal for telemedicine. Int. J. Syst. Sci., 37(1), 45–63.

[8] Shinde A. and Kanjalkar P. 2011. The comparison of different transform based methods for ECG data compression. Proceedings of International Conference on Signal Processing, Communication, Computing and Networking Technologies, 332-335.

[9] Kumar V., Saxena S.C., Giri V.K. and Singh D. 2005. Improved modified AZTEC technique for ECG data compression: Effect of length of parabolic filter on reconstructed signal. Comput. Electr. Eng., 31,334–344.

[10] Tai S.C. 1991. SLOPE- A real-time ECG data compressor. Int. J. Bio-Med. Comput., 29(2), 175-179.

[11] Barr R.C., Blanchard S.M., and Dipersio D.A. 1985. SAPA-2 Is the Fan. IEEE Trans. Biomed. Eng., 32, 337.

[12] Reddy B.R.S. and Murthy I.S.N. 1986. ECG data compression using Fourier descriptors. IEEE Trans. Biomed. Eng., 33(4), 428–434.

[13] Batista L.V., Melcher E.U. and Carvalho L.C. 2001. Compression of ECG signals by optimized quantization of discrete cosine transform coefficients. Med. Eng. Phys., 23(2), 127–34.

[14] Lee S., Kim J. and Lee Jong-Ho. 2011. A real-time ECG data compression and transmission algorithm for an e- health device. IEEE Trans. Biomed. Eng., 58(9), 2448– 2455.

[15] Duarte R.C.M., Matos F.M. and Batista L.V. 2007. Near- lossless compression of ECG signals using perceptual masks in the DCT domain, IFMBE Proceedings, 18, 229–231.

[16] Lu Z., Kim D.Y., and Pearlman W.A. 2000. Wavelet compression of ECG signals by the set partitioning in hierarchical trees algorithm(SPIHT). IEEE Trans. Biomed. Eng. Wavelet, 47(7), 849–856.

[17] Mamaghanian H., Khaled N., Atienza D., and Vandergheynst P. 2011. Compressed sensing for real- time energy-efficient ECG compression on wireless body sensor nodes. IEEE Trans. Biomed. Eng., 58(9), 2456– 2466.

[18] Iwata A., Nagasaka Y., and Suzumura N. 1990. Data compression of the ECG using neural network for digital holter monitor. IEEE Eng. Med. Biol. Mag., 9(3), 53– 57.

[19] Al-Shrouf A., Abo-Zahhad M., and Ahmed S.M. 2003. A novel compression algorithm for electrocardiogram signals based on the linear prediction of the wavelet coefficients. Digit. Signal Process., 13(4), 604–622.

[20] Mukhopadhyay S.K., Mitra S. and Mitra M. 2011. A lossless ECG data compression technique using ASCII character encoding. Comput. Electr. Eng., 37(4), 486– 497.

[21] Mukhopadhyay S.K., Mitra S., and Mitra M. 2012. An ECG signal compression technique using ASCII character encoding. Measurement, 45(6), 1651–1660.

[22] Mukhopadhyay S.K., Mitra S. and Mitra M. 2013. ECG signal compression using ASCII character encoding and transmission via SMS. Biomed. Signal Process. Control, 8(4), 354–363.

[23] Singh B., Sharma D., Singh M. and Singh D. 2014. An improved ASCII character encoding method for lossless ECG compression. Advances in Biomedical Science and Eng., 1(2), 1-11.

[24] Jalaleddine M.S., Hutchens C.G., Strattan R.D., and Coberly W.A. 1990. ECG data compression techniques- A unified approach. IEEE Trans. Biomed. Eng., 37(4), 329–343.

[25] Mueller W.C. 1978. Arrhythmia detection program for an ambulatory ECG monitor. Biomed. Sci. Instrument., 14, 81-85.

[26] Borjesson P., Einarsson G., and Pahlm O. 1980. Comments on compression of the ECG by prediction or interpolation and entropy encoding. IEEE Trans. Biomed. Eng., 27(11), 674-675.

[27] Peric Z., Denic D., Nikolic J., Jocic A., and Jovanovic A. 2013. DPCM quantizer adaptation method for efficient ECG signal compression. Journal of Communications Technology and Electronics, 58(12), 1241–1250.

[28] Kuklinski W.S. 1983. Fast Walsh transform data- compression algorithm ECG applications. Med. & Biol. Eng. Comput., 21, 465-472.

[29] Benzid R., Messaoudi A. and Boussaad A. 2008. Constrained ECG compression algorithm using the block-based discrete cosine transform. Digital Signal Processing, 18(1), 56–64.

[30] Bendifallah A., Benzid R. and Boulemden M. 2011. Improved ECG compression method using discrete cosine transform. Electronics Letters, 47 (2), 87-89.

[31] Fira C.M. and Goras L. 2008. An ECG signals compression method and its validation using NNs. IEEE Transactions on Biomedical Engineering, 55(4), 1319 - 1326.

[32] Byung S., Yoo S. K., and Lee M.H. 2006. Wavelet-based low-delay ECG compression algorithm for continuous ECG transmission. IEEE Transactions on Information Technology in Biomedicine, 10(1), 77-83.

[33] Sabarimalai M. M and Dandapat S. 2006. Wavelet threshold based ECG compression using USZZQ and Huffman coding of DSM. Biomedical Signal Processing and Control, 1(4), 261–270.

[34] Benzid, R., Marir, F., and Bouguechal N.E. 2007. Electrocardiogram compression method based on the adaptive wavelet coefficients quantization combined to a modified two-role encoder. IEEE Signal Processing Letters, 14(6), 373–376.

[35] Blanco-Velasco M., Cruz-Roldan F.,Godino-Llorente J.I., and Barner K.E. 2007. Wavelet packets feasibility study for the design of an ECG compressor. IEEE Transactions on Biomedical Engineering, 54(4), 766– 769.

[36] Aggarwal V. and Patterh M.S. 2013. ECG compression using slantlet and lifting wavelet transform with and without normalisation. International Journal of Electronics, 100(5), 626-636.

[37] Xingyuan W.J. 2008. A 2-D ECG compression algorithm based on wavelet transform and vector quantization. Digit. Signal Process., 18(2), 179–188.

[38] Xingyuan W. and Juan M. 2009. Wavelet-based hybrid ECG compression technique. Analog Integrated Circuits and Signal Processing 59(3), 301–308.

[39] Hilton M.L. 1997. Wavelet and wavelet packet compression of electrocardiograms. IEEE Trans. Biomed. Eng., 44(5), 394–402.

[40] Johan D., Nguyen T.Q., and Tompkins W.J. 1995. ECG compression using discrete symmetric wavelet transform. 17th Int. Conf. IEEE Medicine and Biology, 1, 167-168.

[41] Taubman D.S., Marcellin M.W., and Rabbani M. 2002. JPEG2000: Image compression fundamentals, standards and practice. Journal of Electronic Imaging, 11. 286.

[42] Bilgin A., Marcellin M.W. and Altbach M.I. 2003. compression of electrocardiogram signals using JPEG2000. IEEE Transactions on Consumer Electronics, 49(4), 833-840.

[43] Filho E.B.L. and N. M. M. Rodrigues. 2008. ECG signal compression based on DC equalization and complexity sorting. IEEE Trans. Biomed. Eng., 55(7), 1923–1926.

[44] Nave G. and Cohen A. 1993. ECG compression using long-term prediction. IEEE Trans. Biomed. Eng., 40(9), 877–885.

[45] Ruttimann U.E and Pipberger H.V. 1979. Compression of the ECG by prediction or interpolation and entropy encoding. IEEE Trans. Biomed. Eng., 26(11), 613-623.

[46] Furht B. and Perez A. 1988. An adaptive real-time ECG compression algorithm with variable threshold. IEEE Trans. Biomed. Eng., 35(6), 489-494.

[47] Trahanias P. and Skordalakis E. 1990. Syntactic pattern recognition of ECG. IEEE Trans. Pattern Anal. Machine Intell., 12, 648-657.

[48] Abenstein J.P. and Tompkins W.J. 1982. New data- reduction algorithm for real-time ECG analysis. IEEE Trans. Biomed. Eng., 29, 43-48.

[49] Imai H., Kimura N., and Yoshida Y. 1985. An efficient encoding method for electrocardiography using spline functions. Syst. Comput. Japan, 16(3), 85-94.

[50] B. Furht and A. Perez, “An adaptive real-time ECG compression algorithm with variable threshold,” IEEE Trans. Biomed. Eng., vol. 35, pp. 489–494, June 1988.

[51] C. P. Mammen and B. Ramamurthi, “Vector quantization for compression of multichannel ECG,” IEEE Trans. Biomed. Eng., vol. BME-37, pp. 821–825, Sept. 1990.

[52] M. Ishijima, S. B. Shin, G. H. Hostetter, and J. Sklansky, “Scan along polygon approximation for data compression of electrocardiograms,” IEEE Trans. Biomed. Eng., vol. BME-30, pp. 723–729, 1983.

[53] SangJoon Lee et.al, “A Real-Time ECG Data Compression and Transmission Algorithm for an e-Health Device”, IEEE Transactions on Biomedical Engineering, VOL. 58, NO. 9, September 2011

[54] S. Chan and K. Ho (1990), “Direct Methods for computing discrete sinusoidal transforms”, IEEE Proceedings, 137, 433-442.

[55] C. de Boor, On calculating with B-splines, J. Approx. Theory 6 (1972) 50-62.

[56] W. Boehm, Inserting new knots into B-spline curves, Comput. Aided Des. 12 (4) (1980) 199-201.

[57] G. Szeg¨o, Orthogonal Polynomials, Amer. Math. Soc. Colloq. Publ.,3rd edition, 1967.

[58] R. Askey, Orthogonal Polynomials and Special Functions, SIAM, 1966.

[59] W. Mai, P. Dang, L. Zhang, and T. Qian, “Consecutive minimum phase expansion of physically realizable signals with applications,” Math. Meth. App. Sci. 2015.

Appendix A

The MIT-BIH Arrhythmia Database

The database used in this work is a collection of files from the MIT-BIH Arrhythmia Database CD-ROM (third edition) [Moody, 1997].

The source of the ECGs included in the MIT-BIH Arrhythmia Database is a set of over 4000 long-term Holter recordings that were obtained by the Beth Israel Hospital Arrhythmia Laboratory between 1975 and 1979. Approximately 60% of these recordings were obtained from inpatients. The database contains 23 records (numbered from 100 to 124 inclusive with some numbers missing) chosen at random from this set, and 25 records (numbered from 200 to 234 inclusive, again with some numbers missing) 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 of Holter recordings.

The first group is intended to serve as a representative sample of the variety of waveforms and artifact that an arrhythmia detector might encounter in routine clinical use. A table of random numbers was used to select tapes, and then to select half-hour segments of them. Segments selected in this way were excluded only if neither of the two ECG signals was of adequate quality for analysis by human experts.

Records in the second group were chosen to include complex ventricular, junctional, and supraventricular arrhythmias and conduction abnormalities. Several of these records were selected because features of the rhythm, QRS morphology variation, or signal quality may be expected to present significant difficulty to arrhythmia detectors; these records have gained considerable notoriety among database users.

The subjects were 25 men aged 32 to 89 years, and 22 women aged 23 to 89 years. Records 201 and 202 came from the same male subject. Each record in this directory is slightly over 30 minutes in length. Each signal file contains two channels of ECG signals sampled at 360 Hz. Each sample is represented by 12-bit two’s complement amplitude. 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.

Performance Analysis of ECG Data Compression Techniques

Samra Zubair[1], Archana Yadav[2], Shailendra Kumar[3]
[1, 2, 3] Dept of ECE
[1, 2, 3] Integral University, Lucknow

Abstract- *ECG can identify the electrical activity of the heart. A muscular organ, the heart rhythmically pumps blood throughout the body. It's necessary to send and store large signal data. The ECG signal data must then be effectively compressed. In this article, we compared the effectiveness of various ECG compression techniques.These techniques are crucial for lowering communicated data size without losing clinical information. Discrete Cosine Transform (DCT), Discrete Sine Transform (DST), Fast Fourier Transform (FFT), the enhanced method Discrete Cosine Transform- II (DCT-II), and Blaschke unwinding AFD are the transformation methods on which these schemes are based.We test records that have been chosen from the MIT-BIH arrhythmia database. Percent Root Mean Square Differences (PRD) and Compression Ratio (CR) are used to evaluate performance.*

Keywords- ECG, DCT, DST, FFT, DCT-II, Blaschke unwinding AFD

I. INTRODUCTION

Electrocardiographic signals may be recorded on a long timescale (i.e., several days) for the purpose of identifying intermittently occurring disturbances in the heart rhythm. As a result, the produced ECG recording amounts to huge data sizes that quickly fill up available storage space. Transmission of signals across public telephone networks is another application in which large amounts of data are involved. For both situations, data compression is an essential operation and, consequently, represents yet another objective of ECG signal processing. Signal processing has contributed significantly to a new understanding of the ECG and its dynamic properties as expressed by changes in rhythm and beat morphology. For example, techniques have been developed that characterize oscillations related to the cardiovascular system and reflected by subtle variations in heart rate. ECG Data Compression is required to reduce the disk space required to store the data, as ECG is a continuous data taken for a very long interval of time. Also by compressing redundant data from the signal can be removed which actually takes considerably large area in memory. The need of signal transmission over telephone lines or antenna for remote analysis makes the compression and data reconstruction of the signal an important issue in signal processing. ECG is a graphic display of the electrical activity of the heart. Due to low cost and noninvasion, ECG signal has been extended for heart disease diagnosis and ambulatory monitoring. For storage and transmission of large signal data, it is necessary to compress the ECG signal data. Data compression has its application in many fields and so as in the field of medical science. ECG is an important parameter that measures patient's health and reports abnormalities if any. This thesis has done a survey of various kinds of ECG data compression techniques. Recently, numerous research and techniques have been developed for compression of the signal. These techniques are essential to a variety of application ranging from diagnostic to ambulatory ECG's. Thus, the need for effective ECG compression techniques is of great importance. The non-invasive extraction of physiological and clinical information hidden in biomedical signals is an important and fascinating field of research. Non-invasive assessment of the physiological parameters of a patient enables to study the physiology and patho-physiology of the investigated system, with minimal interference and inconvenience. Endogenous biomedical signals from physiological systems are acquired for a number of reasons including diagnosis, post surgical intensive care monitoring, neonatal monitoring and guide therapy and for research. The electrocardiogram (ECG) is a non- stationary signal containing information about the physiological condition of the heart. The electrical activity of the heart depicts the morphology and durations of the P-QRS- T intervals (Figure 1). The P, QRS complex and T features of ECG reveal the rhythmic depolarization and re polarization of the myocardium contractions of heart's atria and ventricles [1]. The time intervals between various peaks contain clinical information about the nature of possible disease afflicting a heart [2].

Due to low cost and non-invasion, ECG signal has been extended for heart disease diagnosis and ambulatory monitoring resulting in enormous volume of the data. In course of a 24-h ECG observation or multichannel biological signal acquisition, real-time data compression methods are required for the effective use of communications channels such as wired channel, wireless environment and cloud computing. The ECG data compression is also required for the transmission of ECG signals across intensive care units, emergency tele-medical services, telemedicine, home care, space programs, sports, military, public telephone networks, cellular networks and wireless communication systems [4-5]. ECG is having possibility of redundant information reduction through inter and intra beat correlation, which is the basic cause of its compression [6]. The fundamental goal of data compression is efficient transmission or storage while preserving the significant diagnostic features.

In general, ECG compression can be classified into lossy and lossless techniques [7]. The lossless compression guarantee the integrity of reconstructed data while compromised compression ratio (CR), with nearly 0% reconstruction error, on the other hand lossy compression is having high CR with varying level of reconstruction error [6].

ECG signal compression techniques widely fall into three categories of direct method, transformation method and parameter extraction method [7, 8]. The direct data compression method openly analyzes and reduces data points in the time domain and the example includes turning point (TP) [25], amplitude zone time epoch coding (AZTEC) [3], Improved modified AZTEC technique [9], coordinate reduction time encoding system (CORTES) [48], SLOPE [10], the delta algorithm and the Fan algorithm [11]. The transformed method analyzes energy distribution by converting the time domain to some other domain and example includes Fourier transform, Fourier descriptor [12], the discrete cosine transform (DCT) [13], DCT with modified stages [14, 15] and wavelet transform [16], and the compressed sensing [17]. The parameter extraction method is based upon dominant feature extraction from raw signal; examples include neural based or syntactic methods [18],peak picking and linear prediction method [19]. The other methods for compression includes ASCII based encoding for incorporation of ECG data as ASCII character in existing technology [20-23].

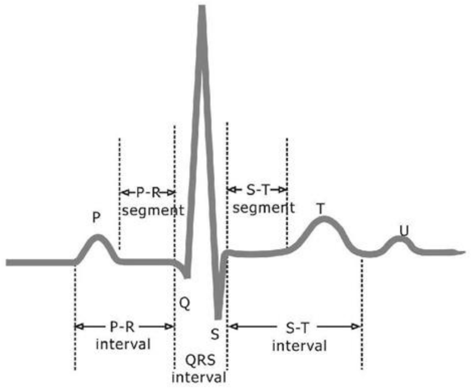


Fig 1 : Time intervals of ECG

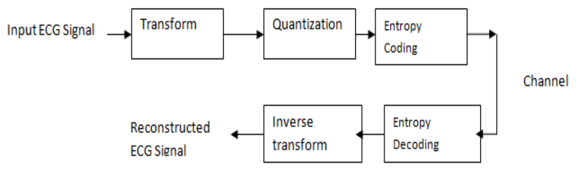


Fig. 2 Block diagram of transform based compression method

ECG Data Compression is required to reduce the disk space required to store the data, as ECG is a continuous data taken for a very long interval of time. Also by compressing redundant data from the signal can be removed which actually takes considerably large area in memory. The need of signal transmission over telephone lines or antenna for remote analysis makes the compression and data reconstruction of the signal an important issue in signal processing. ECG is a graphic display of the electrical activity of the heart. Due to low cost and noninvasion, ECG signal has been extended for heart disease diagnosis and ambulatory monitoring. For storage and transmission of large signal data, it is necessary to compress the ECG signal data.

Data compression has its application in many fields and so as in the field of medical science. ECG is an important parameter that measures patient's health and reports abnormalities if any. This thesis has done a survey of various kinds of ECG data compression techniques. Recently, numerous research and techniques have been developed for compression of the signal. These techniques are essential to a variety of application ranging from diagnostic to ambulatory ECG's. Thus, the need for effective ECG compression techniques is of great importance. Many existing compression algorithms have shown some success in electrocardiogram compression; however, algorithms that produce better compression ratios and less loss of data in the reconstructed signal are needed.

II. DISCRETE COSINE TRANSFORM(DCT)

The Discrete Cosine Transform (DCT) was developed to approximate Karhunen-Loeve Transform (KLT) when there is high correlation among the input samples, which is the case in many digital waveforms including speech, music, and biomedical signals. The DCT D = $[d_0\ d_1\ d_2\ d_3 \ldots\ldots\ldots d_{N1-1}]^T$ Of the vector x is defined as follows

$$d_0 = \frac{1}{\sqrt{N}} \sum_{n=0}^{N-1} x_{n_1} \qquad (1)$$

$$d_k = \sqrt{\frac{2}{N}} \sum_{n1=0}^{N-1} x_{n_1} \cos \frac{(2n_1+1)K\pi}{2N}, \quad k = 1,2,\ldots\ldots\ldots\ldots\ldots N\text{-}1 \qquad (2)$$

Where d_k is the k_{th} DCT coefficient. The inverse discrete cosine transform (IDCT) of d is given by

$$x_{n_1} = \frac{1}{\sqrt{N}} d_0 + \sqrt{\frac{2}{N}} \sum_{k=0}^{N-1} d_k \cos \frac{(2n_1+1)K\pi}{2N}$$

$n_1=0,1,2……………….N-1$ (3) (3.3)

There exist fast algorithms, Order (*N*log*N*), to compute the DCT .Thus, DCT can be implemented in a computationally efficient manner. Two recent image and video coding standards, JPEG and MPEG, use DCT as the main building block. A discrete cosine transform (DCT) expresses a sequence of finitely many data points in terms of a sum of cosine functions oscillating at different frequencies. DCTs are important to numerous applications in science and engineering, from lossy compression of audio (e.g. MP3) and images (e.g. JPEG) (where small high frequency components can be discarded), to spectral methods for the numerical solution of partial differential equations. The use of cosine rather than sine functions is critical in these applications. For compression, it turns out that cosine functions are much more efficient whereas for differential equations the cosines express a particular choice of boundary conditions. In particular, a DCT is a Fourier-related transform similar to the discrete Fourier transform (DFT), but using only real numbers. DCTs are equivalent to DFTs of roughly twice the length, operating on real data with even symmetry (since the Fourier transform of a real and even function is real and even), where in some variants the input and/or output data are shifted by half a sample. Discrete Cosine Transform is a basis for many signal and image compression algorithms due to its high decorrelation and energy compaction property. A discrete Cosine Transform of *N* sample is defined as

$$F(u) = \sqrt{\frac{2}{N}} C(u) \sum_{x1=0}^{N-1} f(x_1) \cos \frac{(2x_1+1)u\pi}{2N}$$

u = 0,1,2……………..*N*-1 (4)

Where

$$C(u) = \begin{cases} \frac{1}{\sqrt{2}}, & for\ u = 0 \\ 1, & otherwise \end{cases}$$

The function *f(x)* represents the value of x_{th} samples of input signals. *F(u)* represents DCT coefficients. The inverse DCT is defined in similar fashion as

$$f(x_1) = \sqrt{\frac{2}{N}} C(u) \sum_{u=0}^{N-1} C(u)\ F(u) \cos \frac{(2x_1+1)u\pi}{2N}$$

$x_1 = 0,1,2…………N-1$ (5)

III. DISCRETE SINE TRANSFORM

Discrete sine transform (DST) is a Fourier-related transform similar to the discrete Fourier transform (DFT), but using a purely real matrix. It is equivalent to the imaginary parts of a DFT of roughly twice the length, operating on real data with odd symmetry (since the Fourier transform of a real and odd function is imaginary and odd), where in some variants the input and/or output data are shifted by half a sample. Like any Fourier-related transform, discrete sine transforms (DSTs) express a function or a signal in terms of a sum of sinusoids with different frequencies and amplitudes. Like the discrete Fourier transforms (DFT), a DST operates on a function at a finite number of discrete data points. The obvious distinction between a DST and a DFT is that the former uses only sine functions, while the latter uses both cosines and sines (in the form of complex exponentials). However, this visible difference is merely a consequence of a deeper distinction: a DST implies different boundary conditions than the DFT or other related transforms.

Formally, the discrete sine transform is a linear, invertible function F: R^N -> R^N (where R denotes the set of real numbers), or equivalently an N × N square matrix. There are several variants of the DST with slightly modified definitions. The N real numbers $x_0,….x_{N-1}$ are transformed into the N real numbers $X_0,…..X_{N-1}$ according to

$$X_k = \sum_{n=0}^{N-1} x_n \sin \frac{\pi}{N+1} (n+1)(k+1)$$

k =0,1,…….. *N*-1 (6)

IV. FAST FOURIER TRANSFORM (FFT)

A fast Fourier transform (FFT) is an efficient algorithm to compute the discrete Fourier transform (DFT) and it's inverse. There are many distinct FFT algorithms involving a wide range of mathematics, from simple complex-number arithmetic to group theory and number theory. A DFT decomposes a sequence of values into components of different frequencies but computing it directly from the definition is often too slow to be practical. An FFT is a way to compute the same result more quickly. Computing a DFT of *N* points in the naive way, using the definition, takes O(*N*2) arithmetical operations , while an FFT can compute the same result in only O(*N* log *N*) operations.

Fast Fourier Transform is a fundamental transform in digital signal processing with applications in frequency analysis, signal processing etc. The periodicity and symmetry properties of DFT are useful for compression. The uth FFT coefficient of length N sequence {f(x)} is defined as

$$F(u) = \sum_{x=0}^{N-1} f(x)\, e^{\frac{-j2\pi ux}{N}}$$
u = 0,1,2……………..*N*-1 (7)

And its inverse transform is calculated from

$$f(x) = \frac{1}{N}\sum_{u=0}^{N-1} F(u)\, e^{\frac{j2\pi ux}{N}}$$
x = 0,1,2………. *N*-1 (8)

V. DISCRETE COSINE TRANSFORM–II (DCT – II)

The most common variant of discrete cosine transform is the type-II DCT [54]. The DCT-II is typically defined as a real, orthogonal (unitary), linear transformation by the formula

$$C_K^{II} = \sqrt{\frac{2-\delta_{k,0}}{N}} \sum_{n=0}^{N-1} x_n \cos\left[\frac{\pi}{N}\left(n+\frac{1}{2}\right)k\right] \quad (9)$$

for N inputs x_n and N outputs C_K^{II}, where $\delta_{k,0}$ is the Kronecker delta (= 1 for k = 0 and = 0 otherwise). DCT-II can be viewed as special case of the discrete Fourier transform (DFT) with real inputs of certain symmetry. This viewpoint is fruitful because it means that any FFT algorithm for the DFT leads immediately to a corresponding fast algorithm for the DCT-II simply by discarding the redundant operations. The discrete Fourier transform of size N is defined by

$$X_K = \sum_{n=0}^{N-1} x_n \quad (10)$$

where $\omega_N = e^{-2\pi i N}$ is an N_{th} primitive root of unity. In order to relate this to the DCT-II, it is convenient to choose a different normalization for the latter transform as

$$C_K = 2\sum_{n=0}^{N-1} x_n \cos\left[\frac{\pi}{n}\left(n+\frac{1}{2}\right)k\right] \quad (11)$$

$$2\cos\left(\frac{\pi l}{N}\right) = \omega_{4N}^{2l} + \omega_{4N}^{4N-2l} \quad (12)$$

$$C_K = 2\sum_{n=0}^{N-1} x_n \cos\left[\frac{\pi}{N}(n+1)k\right] \quad (13)$$

$$C_K = \sum_{n=0}^{N-1} x_n\, \omega_{4N}^{(2n+1)k} + \sum_{n=1}^{N-1} x_n\, \omega_{4N}^{(4N-2n-1)k} \quad (14)$$

Thus, the DCT-II of size N is precisely a DFT of size 4N, of real-even inputs, where the even-indexed inputs are zero.

VI. BLASCHKE UNWINDING AFD

Algorithm 1 illustrates how the Blaschke unwinding AFD is applied to compress a real-valued signal. First, the input real-valued signal F is projected to H^2 space and we get F^+. In practice, we could safely assume that

$$2\Re F^+ = F + c_0 \quad (15)$$

holds, where $\Re$ means taking the real part and c_0 is the zeroth Fourier coefficient of F. c_0 is the first data point we save for the signal compression, and F^+ is initialized as the first remainder F1.

Algorithm I : Blaschke Unwinding AFD based Compression

Input: Real-valued input signal F, sets of parameters a$\in$ D and the decomposition level N.

Output: $\{c_n\}_{n=1}^{N}$, $\{a_n\}_{n=1}^{N}$ and a finite number of zeros $\left\{\left\{r_{n_j}\right\}_{j=1}^{M_n}\right\}_{n=1}^{N}$

```
l: Get the projection signal $F^+$of F in the Hardy space.
2: Initialize $F_l = F^+$.
3: for n = 1 to N do.
4: Obtain the inner function $I_n$ and outer function $O_n$ of $F_n$ so that $F_n = I_n O_n$;
5: Get zeros $\left\{r_{n_j}\right\}_{j=1}^{M_n}$, of $I_n$ by Algorithm 2;
6: Get $a_n$ = arg max{$(1-|a|^2)|O_n(a)|^2$:a$\in D$};
7: Get $c_n$ = $\langle O_n e_{a_n}\rangle$;
8: Get $F_{n+1}$ = $\frac{F_n - c_n I_n e_{a_n}}{I_n}\frac{1-\overline{a_n}z}{z-a_n}$;
9: return $\{c_n\}_{n=1}^{N}$, $\{a_n\}_{n=1}^{N}$, $\left\{\left\{r_{n_j}\right\}_{j=1}^{M_n}\right\}_{n=1}^{N}$.
```

Second, extract the inner function by calculating zeros of F_1 by the method introduced in [59], where we assume that F_1 has finite roots on $\overline{D}$:= {z ∈ C| $\|z\|$ ≤ 1} [59]. The detailed steps of numerical calculation for calculating zeros of F_1 are performed in Algorithm 2. Then accordingly, get the outer function O_1 by the Nevanlinna factorization. Third, The set of {a_n}, n = 1, 2, . . . , in consisting of discrete points in $\mathbb{D}$ is generated by dividing $\mathbb{D}$ into rectangular grid to get the TM system and evaluators {e_a} . Then, the decomposition of O_1 is based on the TM system. During the decomposition, the maximal selection principle is applied in the selection of a1 with the aid of evaluators. Suppose the decomposition level is N ∈ N. Iterate the above three steps, each on the remainder of the previous step, for N times, and we end up with {e_{a_n} }, the modified Blaschke products {B_n}, and M_n zeros, for n = 1, . . . N. As a result, we obtain 2N + 1 parameters, including $\{c_n\}_{n=0}^{N}$, and $\{a_n\}_{n=1}^{N}$, as well as $\sum_{n=1}^{N} M_n$ zeros. c_n and a_n, where n = 1, . . . N, as well as $\sum_{n=1}^{N} M_n$ zeros, are other data points we save for the data compression.

Algorithm II: Procedure for calculating zeros

Input: F, $\delta > 0$

Output: zeros of F, $\left\{r_{n_j}\right\}_{j=1}^{M_n}$

```
l: Determine for $M$, $M= \frac{1}{2\pi i}\int_{|z|=1}\frac{F'(z)}{F(z)}dz$
2: Initialize $G_1 : F$
3: for j = 1 to $M_1$ do.
4: Evaluate arg $\min_{z\in D_{1-\delta}}|G_j(z)|$;
5: Get $r_j$ satisfying $G_j(r_j) = 0$;
6: Get $G_{j+1} := G_j \frac{1-r_j z}{z-r_j}$ ;
7: return $\{r_j\}_{j=1}^{M_n}$.
```

VII. RESULT ANALYSIS

We used data in the MIT-BIH database to test the performance of the six coding techniques. The ECG data is sampled at 142Hz and the resolution of each sample is 11bits/samples. The amount of compression is measured by CR and the distortion between the original and reconstructed signal is measured by Percentage Mean Square Difference (PRD). A data compression algorithm must represent the data with acceptable fidelity while achieving high CR.

Figure 3 shows the original ECG signal record 100 which are selected from MIT-BIH arrhythmia database and its reconstructed waveform when compressed by FFT.

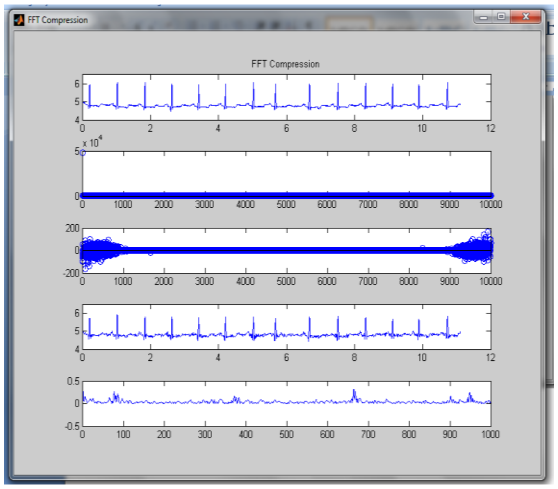


Figure 3 FFT compression of MIT-BIH record 100

Fig.4 shows the original ECG signal record 100 which are selected from MITBIH arrhythmia database and its reconstructed waveform when compressed by DCT.

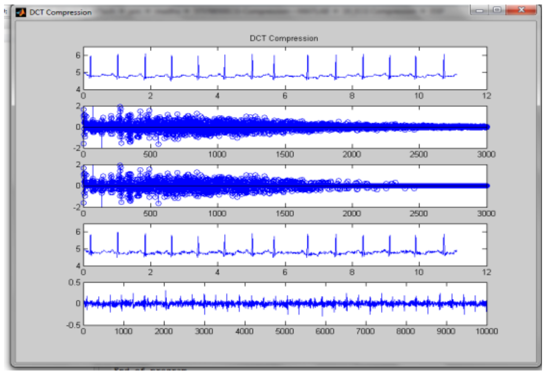


Figure 4 DCT compression of MIT-BIH record 100

Figure 5 shows the original ECG signal record 100 which are selected from MIT-BIH arrhythmia database and its reconstructed waveform when compressed by DST.

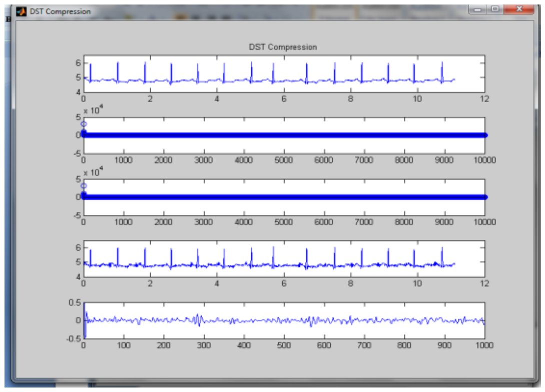


Figure 5. DST compression of MIT-BIH record 100

Figure 6 shows the original ECG signal record 100 which are selected from MIT-BIH arrhythmia database and its reconstructed waveform when compressed by DCT-2.

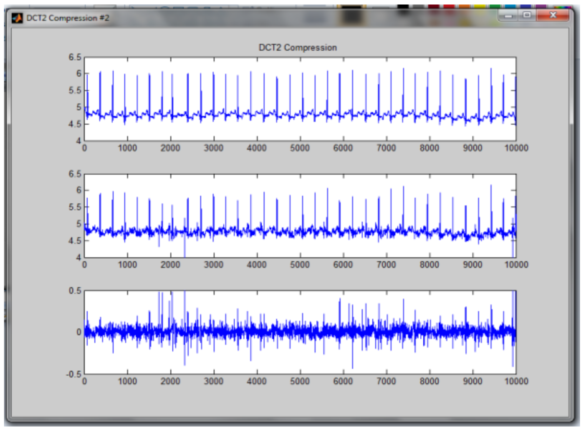


Figure 6 DCT-2 compression of MIT-BIH record 100

In Blaschke unwinding AFD, the compression consists of two steps. The first step carries out the Hardy projection and the Blaschke unwinding AFD compression. The second step is the lossless Huffman encoding. For the decompression, it is the inverse of the compression, including the Huffman decoding and the inverse Blaschke unwinding AFD process.

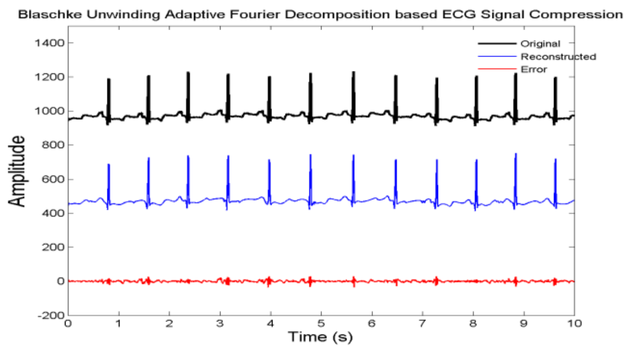


Figure:7 Waveforms of original, reconstructed and error signals with N = 8 taken from record 100

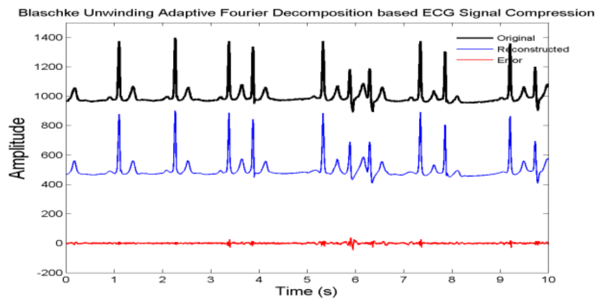


Figure:8 Waveforms of original, reconstructed and error signals with N = 10 taken from record 100

The comparison table shown in Table 1 details the resultant compression techniques. This gives the choice to select the best suitable compression method. A data compression algorithm must represent the data with acceptable fidelity while achieving high CR. As the PRD indicates reconstruction fidelity; the increase in its value is actually undesirable. Blaschke unwinding AFD which leads to a high compression rate with a high fidelity. Compared with existing algorithms, like FFT, DCT, DST and DCT-2.

Table 1 Comparision of resultant compression techniques

Method	**CR**	**PRD**
FFT	16.01	1.10
DCT	16.87	1
DST	11.62	1.19
DCT2	22.21	1.27
Blaschke unwinding AFD(N=8)	39.34	0.71
Blaschke unwinding AFD(N=10)	26.09	0.57

VIII. CONCLUSION

Among the five techniques presented, DST provides lowest CR and distortion is also high. DCT improves CR and lowers PRD. Next is FFT which gives CR 16.01 with PRD as 1.10. But DCT-II provides an improvement in terms of CR of 22.21 but PRD increases up to 1.27. Thus an improvement of a discrete cosine transform (DCT)-based method for electrocardiogram (ECG) compression is presented as DCT-II in terms of amount of compression. From table 1 we can observe that using Blaschke unwinding AFD based compression we are getting higher compression rate in comparison to other compression technique. Hence this technique is better than other ECG compression technique in both point of consideration.

REFERENCES

[1] Houghton, A. R. and Gray D. 2003. Making sense of the ECG: A Hands-on Guide. Aold Publishing Company.

[2] Singh B., Singh D., Jaryal A.K. and Deepak K.K. 2012. Ectopic beats in approximate entropy and sample entropy-based HRV assessment. International Journal of Systems Science, 43(5), 884-893.

[3] Cox J.R., Nolle F.M., Fozzard H.A., and Oliver G.C. 1968. AZTEC, a preprocessing program for real-time ECG rhythm analysis. IEEE Trans. Biomed. Eng., 15, 128-129.

[4] Mitra M., Bera J.N. and Gupta R. 2012. Electrocardiogram compression technique for global system of mobile-based offline telecardiology application for rural clinics in India. IET Sci. Meas. Technol., 6(6), 412-419.

[5] Kulkarni P.K., Kumar V. and Verma H.K. 1997. Direct data compression techniques for ECG signals: effect of sampling frequency on performance. International Journal of Systems Science, 28(3), 217-228.

[6] Koski A. 1997. Lossless ECG encoding. Comput. Methods and Programs Biomed., 52(1), 23–33.

[7] Kumar V., Saxena S.C. and Giri V.K. 2006. Direct data compression of ECG signal for telemedicine. Int. J. Syst. Sci., 37(1), 45–63.

[8] Shinde A. and Kanjalkar P. 2011. The comparison of different transform based methods for ECG data compression. Proceedings of International Conference on

Signal Processing, Communication, Computing and Networking Technologies, 332-335.

[9] Kumar V., Saxena S.C., Giri V.K. and Singh D. 2005. Improved modified AZTEC technique for ECG data compression: Effect of length of parabolic filter on reconstructed signal. Comput. Electr. Eng., 31,334–344.

[10] Tai S.C. 1991. SLOPE- A real-time ECG data compressor. Int. J. Bio-Med. Comput., 29(2), 175-179.

[11] Barr R.C., Blanchard S.M., and Dipersio D.A. 1985. SAPA-2 Is the Fan. IEEE Trans. Biomed. Eng., 32, 337.

[12] Reddy B.R.S. and Murthy I.S.N. 1986. ECG data compression using Fourier descriptors. IEEE Trans. Biomed. Eng., 33(4), 428–434.

[13] Batista L.V., Melcher E.U. and Carvalho L.C. 2001. Compression of ECG signals by optimized quantization of discrete cosine transform coefficients. Med. Eng. Phys., 23(2), 127–34.

[14] Lee S., Kim J. and Lee Jong-Ho. 2011. A real-time ECG data compression and transmission algorithm for an e-health device. IEEE Trans. Biomed. Eng., 58(9), 2448–2455.

[15] Duarte R.C.M., Matos F.M. and Batista L.V. 2007. Near-lossless compression of ECG signals using perceptual masks in the DCT domain, IFMBE Proceedings, 18, 229–231.

[16] Lu Z., Kim D.Y., and Pearlman W.A. 2000. Wavelet compression of ECG signals by the set partitioning in hierarchical trees algorithm(SPIHT). IEEE Trans. Biomed. Eng. Wavelet, 47(7), 849–856.

[17] Mamaghanian H., Khaled N., Atienza D., and Vandergheynst P. 2011. Compressed sensing for real-time energy-efficient ECG compression on wireless body sensor nodes. IEEE Trans. Biomed. Eng., 58(9), 2456–2466.

[18] Iwata A., Nagasaka Y., and Suzumura N. 1990. Data compression of the ECG using neural network for digital holter monitor. IEEE Eng. Med. Biol. Mag., 9(3), 53– 57.

[19] Al-Shrouf A., Abo-Zahhad M., and Ahmed S.M. 2003. A novel compression algorithm for electrocardiogram signals based on the linear prediction of the wavelet coefficients. Digit. Signal Process., 13(4), 604–622.

[20] Mukhopadhyay S.K., Mitra S. and Mitra M. 2011. A lossless ECG data compression technique using ASCII character encoding. Comput. Electr. Eng., 37(4), 486–497.

[21] Mukhopadhyay S.K., Mitra S., and Mitra M. 2012. An ECG signal compression technique using ASCII character encoding. Measurement, 45(6), 1651–1660.

[22] Mukhopadhyay S.K., Mitra S. and Mitra M. 2013. ECG signal compression using ASCII character encoding and transmission via SMS. Biomed. Signal Process. Control, 8(4), 354–363.

[23] Singh B., Sharma D., Singh M. and Singh D. 2014. An improved ASCII character encoding method for lossless ECG compression. Advances in Biomedical Science and Eng., 1(2), 1-11.

[24] Jalaleddine M.S., Hutchens C.G., Strattan R.D., and Coberly W.A. 1990. ECG data compression techniques-A unified approach. IEEE Trans. Biomed. Eng., 37(4), 329–343.

[25] Mueller W.C. 1978. Arrhythmia detection program for an ambulatory ECG monitor. Biomed. Sci. Instrument., 14, 81-85.

[26] Borjesson P., Einarsson G., and Pahlm O. 1980. Comments on compression of the ECG by prediction or interpolation and entropy encoding. IEEE Trans. Biomed. Eng., 27(11), 674-675.

[27] Peric Z., Denic D., Nikolic J., Jocic A., and Jovanovic A. 2013. DPCM quantizer adaptation method for efficient ECG signal compression. Journal of Communications Technology and Electronics, 58(12), 1241–1250.

[28] Kuklinski W.S. 1983. Fast Walsh transform data-compression algorithm ECG applications. Med. & Biol. Eng. Comput., 21, 465-472.

[29] Benzid R., Messaoudi A. and Boussaad A. 2008. Constrained ECG compression algorithm using the block-based discrete cosine transform. Digital Signal Processing, 18(1), 56–64.

[30] Bendifallah A., Benzid R. and Boulemden M. 2011. Improved ECG compression method using discrete cosine transform. Electronics Letters, 47 (2), 87-89.

[31] Fira C.M. and Goras L. 2008. An ECG signals compression method and its validation using NNs. IEEE Transactions on Biomedical Engineering, 55(4), 1319 - 1326.

[32] Byung S., Yoo S. K., and Lee M.H. 2006. Wavelet-based low-delay ECG compression algorithm for continuous ECG transmission. IEEE Transactions on Information Technology in Biomedicine, 10(1), 77-83.

[33] Sabarimalai M. M and Dandapat S. 2006. Wavelet threshold based ECG compression using USZZQ and Huffman coding of DSM. Biomedical Signal Processing and Control, 1(4), 261–270.

[34] Benzid, R., Marir, F., and Bouguechal N.E. 2007. Electrocardiogram compression method based on the adaptive wavelet coefficients quantization combined to a modified two-role encoder. IEEE Signal Processing Letters, 14(6), 373–376.

[35] Blanco-Velasco M., Cruz-Roldan F.,Godino-Llorente J.I., and Barner K.E. 2007. Wavelet packets feasibility study for the design of an ECG compressor. IEEE Transactions on Biomedical Engineering, 54(4), 766– 769.

[36] Aggarwal V. and Patterh M.S. 2013. ECG compression using slantlet and lifting wavelet transform with and

without normalisation. International Journal of Electronics, 100(5), 626-636.
[37] Xingyuan W.J. 2008. A 2-D ECG compression algorithm based on wavelet transform and vector quantization. Digit. Signal Process., 18(2), 179–188.
[38] Xingyuan W. and Juan M. 2009. Wavelet-based hybrid ECG compression technique. Analog Integrated Circuits and Signal Processing 59(3), 301–308.
[39] Hilton M.L. 1997. Wavelet and wavelet packet compression of electrocardiograms. IEEE Trans. Biomed. Eng., 44(5), 394–402.
[40] Johan D., Nguyen T.Q., and Tompkins W.J. 1995. ECG compression using discrete symmetric wavelet transform. 17th Int. Conf. IEEE Medicine and Biology, 1, 167-168.
[41] Taubman D.S., Marcellin M.W., and Rabbani M. 2002. JPEG2000: Image compression fundamentals, standards and practice. Journal of Electronic Imaging, 11. 286.
[42] Bilgin A., Marcellin M.W. and Altbach M.I. 2003. compression of electrocardiogram signals using JPEG2000. IEEE Transactions on Consumer Electronics, 49(4), 833-840.
[43] Filho E.B.L. and N. M. M. Rodrigues. 2008. ECG signal compression based on DC equalization and complexity sorting. IEEE Trans. Biomed. Eng., 55(7), 1923–1926.
[44] Nave G. and Cohen A. 1993. ECG compression using long-term prediction. IEEE Trans. Biomed. Eng., 40(9), 877–885.
[45] Ruttimann U.E and Pipberger H.V. 1979. Compression of the ECG by prediction or interpolation and entropy encoding. IEEE Trans. Biomed. Eng., 26(11), 613-623.
[46] Furht B. and Perez A. 1988. An adaptive real-time ECG compression algorithm with variable threshold. IEEE Trans. Biomed. Eng., 35(6), 489-494.
[47] Trahanias P. and Skordalakis E. 1990. Syntactic pattern recognition of ECG. IEEE Trans. Pattern Anal. Machine Intell., 12, 648-657.
[48] Abenstein J.P. and Tompkins W.J. 1982. New data-reduction algorithm for real-time ECG analysis. IEEE Trans. Biomed. Eng., 29, 43-48.
[49] Imai H., Kimura N., and Yoshida Y. 1985. An efficient encoding method for electrocardiography using spline functions. Syst. Comput. Japan, 16(3), 85-94.
[50] B. Furht and A. Perez, “An adaptive real-time ECG compression algorithm with variable threshold,” IEEE Trans. Biomed. Eng., vol. 35, pp. 489–494, June 1988.
[51] C. P. Mammen and B. Ramamurthi, “Vector quantization for compression of multichannel ECG,” IEEE Trans. Biomed. Eng., vol. BME-37, pp. 821–825, Sept. 1990.
[52] M. Ishijima, S. B. Shin, G. H. Hostetter, and J. Sklansky, “Scan along polygon approximation for data compression of electrocardiograms,” IEEE Trans. Biomed. Eng., vol. BME-30, pp. 723–729, 1983.
[53] SangJoon Lee et.al, “A Real-Time ECG Data Compression and Transmission Algorithm for an e-Health Device”, IEEE Transactions on Biomedical Engineering, VOL. 58, NO. 9, September 2011
[54] S. Chan and K. Ho (1990), “Direct Methods for computing discrete sinusoidal transforms”, IEEE Proceedings, 137, 433-442.
[55] C. de Boor, On calculating with B-splines, J. Approx. Theory 6 (1972) 50-62.
[56] W. Boehm, Inserting new knots into B-spline curves, Comput. Aided Des. 12 (4) (1980) 199-201.
[57] G. Szeg¨o, Orthogonal Polynomials, Amer. Math. Soc. Colloq. Publ.,3rd edition, 1967.
[58] R. Askey, Orthogonal Polynomials and Special Functions, SIAM, 1966.
[59] W. Mai, P. Dang, L. Zhang, and T. Qian, “Consecutive minimum phase expansion of physically realizable signals with applications,” Math. Meth. App. Sci. 2015.