İSTANBUL TEKNİK ÜNİVERSİTESİ ★ FEN BİLİMLERİ ENSTİTÜSÜ

ENTROPİ KODLAMA İLE EKG VERİ SIKIŞTIRMASI

YÜKSEK LİSANS TEZİ

Müh. Abdurrahman Aktener

Tezin Enstitüye Verildiği Tarih : 12 Haziran 1995

Tezin Savunulduğu Tarih : 3 Temmuz 1995

Tez Danışmanı : Doç. Dr. Mehmet Korürek

Diğer Jüri Üyeleri : Prof. Dr. Ertuğrul Yazgan

Doç. Dr. Ümit Aygölü

TEMMUZ 1995

ÖNSÖZ

Bu çalışmamda bana yol gösteren danışmanım Doç.Dr. Mehmet Korürek'e teşekkür ederim. Ayrıca eğitim ve öğrenim yaşamım boyunca bana rehber olan öğretmenlerime, benim yaşama nedenim, azmim ve isteğim zayıfladığında yanımda olarak bana azim ve istek aşılayan aileme; son olarak da manevi desteğini veren teyzem Sevinç Süer'e en içten teşekkürlerimi sunarım.

Abdurrahman Aktener

İÇİNDEKİLER

ŞEKİL LİSTESİ

TABLO LİSTESİ

ÖZET

Veri sıkıştırma, hem veri saklama, hem de iletişim alanında büyük kolaylıklar sağlamaktadır. Bu yolla, aynı ortamda daha fazla bilgi saklanabilmekte veya aynı bant genişliğinde daha fazla bilgi daha kısa sürede gönderilebilmektedir.

Elektrokardiyogram (EKG) kalpdeki elektriksel aktiviteyi temsil eden grafiksel bir gösterimdir. Bu çalışmada, EKG verisi için uygulanan sıkıştırma teknikleri tanımlanmış ve EKG verisini sıkıştırmak için Huffman entropi kodlama metodu uygulanmıştır. Huffman kodlamasında kodkelimelerin kaynak mesajlara eşlenmesi, kaynak mesajların verideki olasılıklarına göre yapılır. Veride sık geçen mesajlar (yüksek olasılıklı mesajlar) kısa kodkelimelere, nadir geçenler (düşük olasılıklı) uzun kodkelimelere atanır. Bu olasılıklar, sıkıştırma başlamadan önece veri bir kez okunduktan sonra belirlenir. Dinamik yöntemlerde ise, mesaj kümesinden kodkelimesine olan eşleme zamanla değişmektedir. Örneğin dinamik Huffman kodlamasında, kodkelimelerin mesajlara eşlenmesi heerhangi bir zamandaki olasılık değerlerine göre değişir. Sıkıştırma oranını arttırmak amacıyla verideki sıfırların fazlalığı göz önünde tutularak verinin işlenmesinde tekrar uzunluğu kodlama yöntemi de uygulanmıştır. Bu yöntemde, sıkıştırma anında arka arkaya gelen semboller, sadece sembol ve tekrar sayısı ile değiştirilir. Genişletirken ise tersi işlem olan sayı kadar sembol eklenmektedir. Bu çalışmada kullanılan sıkıştırma algoritmasında ilk olarak örneklerin ayrık kosinüs dönüşümü katsayıları hesaplanmakta ve katsayılara normalizasyon ve eşik değeri işlemleri uygulanmaktadır. Kodlama algoritmasında ise Huffman ve tekrar uzunluğu kodlamaları kullanılmıştır.

Tezin içeriğinde ilk bölümde elektrokardiogram verisi için daha önce tanımlanmış sıkıştırma teknikleri sunuldu. İkinci bölümde, üzerinde çalışılan veri olan elektrokardiyogram verisinin analizi amacıyla temel olarak kalbin elektrofizyolojisi anlatıldı. Tezin amacı veri sıkıştırma olduğundan üçüncü ve dördüncü bölümlerde veri sıkıştırmada genel kavramlar ve veri sıkıştırma yöntemleri anlatıldı. Son olarak beşinci bölümde, kullanılan algoritma ve sonuçları verilmiştir.

SUMMARY

ECG DATA COMPRESSION BY ENTROPY CODING

In this study, a variety of data compression techniques for ECG signals are presented. The study especially concentrates on Huffman DCT -based entopy coding algorithm.

The electrocardiogram (ECG) is a graphic representation of the electrical activity of the heart. ECG is recorded from the body surface and it assists an experienced interpreter in giving a prognosis. Compression of digital ECG signals has been a research due to the developments in the techniques of digital processing of analog signals. By the usage of digital signal processing to interpret, store or transmit ECG signals, a need to have memory chips with large capacities has arisen. This makes the compression of digital ECG signals a necessity.

The discrete cosine transform (DCT) was first applied to image compression in Ahmed, Natarajan, and Rao's pioneering work, in which they showed that this particular transform was very close to the KLH (Karhulenen-Loeve-Hotelling) transform, a transform that produces uncorrelated coefficients. Decorrelation of the coefficients is very important for compression, because each coefficient can then be treated independently without loss of compression efficiency. The DCT is an orthogonal transform having some of the features of a transformation to the frequency domain. It is equivalent to a discrete Fouirer transform (DFT) of a symmetricized extension of the input set of samples and fast algorithms are available for efficiently computing the DCT. The DCT has an advantage over the DFT in that it is a purely real transform if the input vector is real. The one dimensional DCT is defined as;

$$C_x(0) = \frac{\sqrt{2}}{N} \sum_{n=0}^{N-1} x(n)$$

$$C_x(k) = \frac{2}{N} \sum_{n=0}^{N-1} x(n) \cos \frac{\Pi(2n+1)k}{2N} \qquad k=1,2,.....,N-1$$

Where $C_x(k)$ is the k^{th} DCT coefficient.

Data compression is the process of transforming a series of bytes into a new string that contains the same information but whose length is as small as possible. Data compression has important applications in the areas of information storage and data communication. Most computer applications nedd a large amount of data storage. Also, computer communication networks transfer large volumes of data over communication lines. Compressing data to be stored or transfered reduces the cost of storage and transmission. When files are stored in compressed form, the capacity of storage medium is approximately doubled. Smilarly, when the amount of data to be transmitted is reduced, the capacity of communication channel is increased. Thus, data compression provides an echonomical and indispensible option to store and transmit the information.

Data compression may be achieved in either software or hardware. When it is hardware, a compressor a hardware device that reads a series of of bytes of data and converts it into a shorter string; and decompressor is a device that converts the compressed data back to its original form. When data compression is done in software, a compression algorithm does the former; and an expansion or decompression algoritm does the latter.

The logic that enables data compression is that the same information retained in the file after some bytes are replaced with shorter representations. In other words, data compression is achieved by removing the redundancy in the file. There are several types of redundancy found in the files. These categories are not independent, but overlap to some extend. The major redundancy types are data distribution, data repetition, high usage patterns and positional redundancy.

In typical files, some bytes occur more frequently than others. Moreover, some bytes may never be used. Thus, a few bits on the average might be saved in the representation of bytes, using less number of bits for most frequently used bytes and more number of bits for least frequently used bytes. In some files, long repetitions of a single value occurs. These files can be encoded more compactly than by just repeating, the value symbol. Examples, strings of zeros.

In general, data compression techniques can be divided into two categories, the nonreversible and reversible. In nonreversible techniques, the size of the physical representation of data is reduced while a subset of the information is preserved. For example, in some data sets "leading zeros" may be considered as irrevelant information. Note that these techniques are, by definition, on the semantics of the data.

In reversible techniques, all of the information is considered to be relevant, and compression/decompression process recovers the original data. The reversible procedures can further divided into two groups, semantic independent and semantic dependent techniques. Semantic dependent techniques can be used on any data with varying degrees of effectiveness. They do not use any information regarding the information content of the data. On the other hand, semantic dependent techniques depend on the context and semantics of the data to provide for redundancy reduction.

A code is a mapping of source messages into codewords. Codes can be categorized as block-block, block-variable, variable-block or variable block, where block block indicates that the source messages and codeword are of fixed length and variable-variable codes map variable-length source messages into variable-length codewords.

Data compression schemes are also classified as either static or dynamic. A static method is one which the mapping from the set of messages to be set of codewords is fixed before transmission begins, so that a given message is represented by the same codeword every time it appears in the message ensemble. In these methods, the mapping is based on the probabilities with which the source messages appear in the source ensemble. Messages that appear frequently are represented by short codewords; messages with smaller probabilities map to longer codewords. These probabilities are determined before transmission begins.

A code is dynamic if the maping from the set of messages to the set of codewords changes over time. These methods involve computing an approximation to the probabilities of occurrence, as the ensemble is being transmitted. The assignment of codewords to messages is based on the values of the relativi frequencies of occurrence at each point in time. A message may be represented by a short codeword early in transmission because it occurs frequently at the begining of the ensemble, even though its probability of occurence over the total ensemble is low. Later, when the more probable messages begin to occur with higher frequency, the short codeword will be mapped to one of the higher probability messages, and it will be mapped to a longer codeword. Dynamic codes are also referred to in the literature as adaptive, in that they adapt to changes in ensemble characteristics over time.

All of the adaptive methods are one-pass methods; only one scan of the ensemble is required. Static methods require two passes; one pass to compute probabilities and determine the mapping, and a second pass for transmission. Thus, as long as the encoding and decoding times of an

adaptivehod are not substantially greater than those of a static method, the fact that an initial scan is not needed implies a speed improvement in the adaptive case. In addition, the mapping determined in the first pass of a static coding method must be transmitted along with compressed ensemble by the encoder to the decoder.

There are two dimensions along which the data compression schemes may be measured: algorithm complexity and amount of compression. When data compression is used in a data transmission application, the goal is speed. Speed of transmission depends on the number of bits sent, the time required for the encoder to generate the coded message, and the time required for the decoder to recover the original ensemble. In a data storage application, altough the degree of compression is the primary concern, it is nonetheless necessary that the algorithm be efficient in for the schemes to be practical. For a static scheme, there are three algoritms to analyse: the map construction algorithm, the encoding algorithm, and the decoding algorithm. For a dynamic method, there are just two algorithms: the encoding algorithm and the decoding algorithm.

Several common measures of compression have been suggested: average message length and compression ratio. Average message length is clearly the average length of a coded message. Compression ratio is defined in several ways. The ratio of the length of the coded message to the length of the original ensemble gives the most common meaning.

Semantic dependent data compression techniques are designed to respond to specific types of applications, one of which is image representation and processing. One of these methods run-length encoding where succesive occurrences of a symbol is replaced by a single symbol along with its frequency. The other is difference mapping, in which the image is represented as an array of differences in brightness (or color) between adjacent pixels rather than the brihtness value themselves. Semantic dependent data compression is also of interest in business data processing. The runs of zeros can compressed using runlength encoding.

Semantic independent data compression techniques are based on the probabilities of messages in the ensemble. Some are static codings that are Shannon - Fano coding, static Huffman coding, universal coding and arithmetic coding. The other is dynamic one that is dynamic Huffman coding.

Huffman coding takes the probabilities of source letters and constructs a full binary tree. Initially, there is a set of singleton trees, each having the

probability of a source letter. At each step, the method merges the two smallest probabilities into a tree whose probability is the total of two probabilities and whose root has two children that are two probabilities. The processed is continued until one tree whose probability is 1 is left. The codewords are assign to each message following the path from root to leaves, such as right child meaning a 0 and left child meaning a 1.

Adaptive Huffman coding determines the mapping from source messages to codewords on the basis of a running estimate of the source message probabilities. The code is adaptive and requires only one pass over the data. The encoder learns the characteristics of the source. The decoder must learn along with the encoder by contiunally updating the Huffman tree in order to stay in syncronization with the encoder.

BÖLÜM 1

GİRİŞ

Veri sıkıştırma yöntemleri gereksiz veri miktarını azaltarak verinin saklanması veya iletimin verimini arttıran yöntemlerdir. Veri sıkıştırmanın bir özelliği, belirli koddaki sembol dizisini yine aynı bilgiyi taşıyan fakat mümkün olduğu kadar kısa yeni bir diziye çevirmektir. Veri sıkıştırma, veri iletimi ve veri saklama alanlarında önemli uygulamalara sahiptir. Bir çok veri işleme uygulamaları büyük oranlarda verinin saklanmasını gerektirir ve bilgisayarın kullanımı yeni alanlara doğru yayıldıkça bu uygulamaların sayısı devamlı olarak artmaktadır. Aynı zamanda bilgisayar iletişim alanlarının yaygınlaşması, iletişim hatlarında yüksek miktarda veri transferine neden olur. Saklanacak ya da iletilecek verinin sıkıştırılması saklama veya iletişim maliyetlerini azaltır. İletilecek verinin miktarı azaltıldığı zaman, sonuçta iletim kanallarının kapasitesi arttırılır. Benzer şekilde bir dosyanın orijinal boyutundan yarısına sıkıştırılması, saklama ortamı kapasitesini iki katına çıkarmaktadır. Bu bölümde veri sıkıştırma kavramına değinilirken EKG verisi için daha önce tanımlanmış bazı sıkıştırma teknikleri kısaca anlatılacaktır.

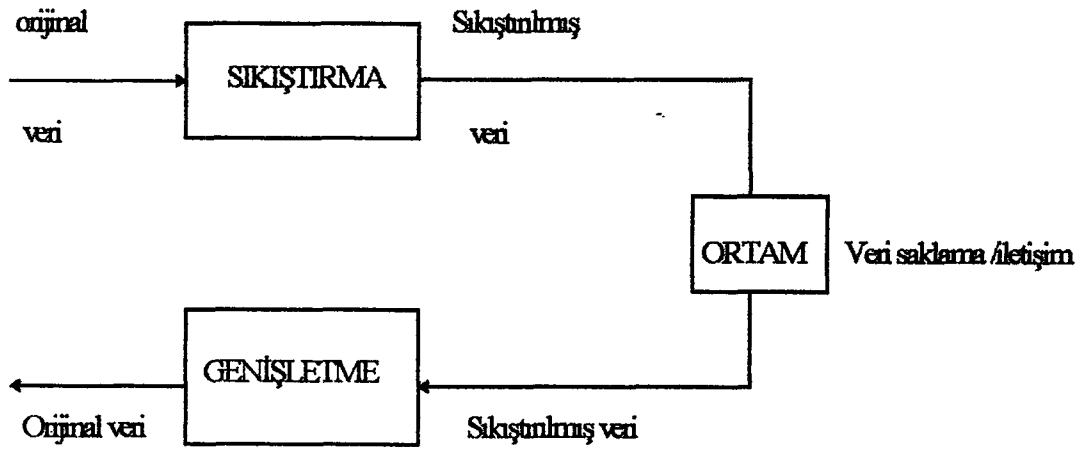
1.1 Veri Sıkıştırma Modeli

Sıkıştırıcı, bir dizi sembolü kabul eden ve daha kısa sembollere çeviren yazılım parçası veya donanım aygıtıdır. Sıkıştırma algoritması, sıkıştırıcının izlediği yöntemdir ve bir semboller dizisini girdi olarak alır ve karşılık gelen sıkıştırılmış veriyi üretir. Genişletici ise sıkıştırılmış veriyi alarak orjinal hale getiren bir yazılım parçası veya donanım aygıtıdır.Bunu yaparken de genişletme algoritmasını kullanır (Şekil 1.1).

ise sıkıştırılmış veriyi alarak orjinal hale getiren bir yazılım parçası veya donanım aygıtıdır.Bunu yaparken de genişletme algoritmasını kullanır (Şekil 1.1).

İkincil bellekte saklanan verilerin sıkıştırılmış olarak tutularak daha az yer kaplaması veya iletişim alanında verinin sıkıştırılmış olarak iletilerek oldukça değerli olan bant genişliğinin tasarruflu olarak kullanılması amaçlarıyla veri sıkıştırma teknikleri uygulanır. Veri sıkıştırma tekniklerini kullanmanın dezavantajları ise, veri sıkıştırma ve genişletmede harcanan zaman ile sıkıştırılmış veride olan herhangi bir hata durumunda veri kurtarmanın zorlaşması ve hatta bazı durumlarda olanaksız hale gelmesidir. Sıkıştırma ve genişletmede harcanan zaman, günümüzde donanım hızı ile gözardı edilebilir, ama parazitli hatlar yüzünden sıkıştırılan veride oluşan hatalar oldukça önemlidir.

Veri saklamada ise veri sıkıştırma, genellikle büyük boyutlardaki verinin uzun vadeli saklanmasında kullanılır. Son yıllarda kalıcı bellekteki dosyaları sıkıştırılmış olarak saklayan işletim sistemleride piyasaya çıkarılmıştır. Sıkıştırıcı, veriyi sıkıştırarak ikincil bellekte daha az yer kaplamasını sağlar. Veri gerektiği zaman, genişletici veriyi genişleterek kullanıma sunar.



Şekil 1.1. Veri sıkıştırma modeli

1.2 Elektrokardiyogram Veri Sıkıştırma Teknikleri

Bu bölümde, ADAPTİF, AZTEC, SAPA-1, SAPA-2 ve NOKTA DEĞİŞİMİ (TP) veri sıkıştırma yöntemleri kısaca tanıtılıcaktır.

Direkt veri sıkıştırma yöntemleri

Bu alt bölümde, özellikle EKG veri sıkıştırması için geliştirilmiş olan direkt veri sıkıştırma tekniklerine değinilmiştir. Temeli, tolerans karşılaştırmalı sıkıştırma tekniğine dayanan ADAPTİF, AZTEC, FAN/SAPA, TP, CORTES gibi sıkıştırma teknikleri bu bölümde sunulmuştur.

Değişken Eşikli Adaptif Algoritma Tekniği:

Sözkonusu tekniğin akış mantığı aşağıdaki adımlarla anlatılabilir.

a) Bu algoritma işlenmemiş EKG verilerini alıp AZTEC algoritmasında olduğu gibi kısa çizgiler üretir [Cox 1968]. Orjinal işaret örnek dizisi x_i(i=1,2...) ile ve örneklerinmaksimum ve minimum değerlerini x_{max} ve x_{min} ile belirtilsin. Başlangıç x_{max} ve x_{min} değerleri ilk işaret örneği olan x_1'e eşit kılınır. x_2, x_3 gibi sonraki örnekler devamlı olarak x_{max} ve x_{min} değerleri ile karşılaştırılırlar. Eğer yeni örnek x_{max}'tan büyük ise bu örnek x_{max} 'a atanır, yok eğer bu örnek x_{min}'den küçükse x_{min}'e atanır. Bunun dışındaki durumda x_{max} ve x_{min} değerleri değiştirilmez. Bu işlem, x_{max}-x_{min} farkının T gibi bir eşik değerinden büyük olmasına kadar tekrar edilir [Furht, Perez 1988].

b) Sözkonusu x_{max}-x_{min} > T koşulu sağlandığında çizgi, ($\bar{x}$, t) veri ikilisi kullanılarak kaydedilir. Burada $\bar{x}$, x_i örneğinin hesaplanmış ortalama genlik değerini ve t ise çizginin uzunluğunu, yani o bölgedeki örnek sayısını göstermektedir.

c) T eşiği, işaretin tabiatına bağlı olarak değiştirilir. Tabanhattı (baseline) gibi yavaş değişim gösteren düşük bilgi bölgelerindeki eşik değerleri, P, T, ST, QRS gibi hızlı değişim gösteren bölgelerdeki eşik değerlerinden daha yüksek olacaktır. Böylece algoritma, bir çeşit kendinden adaptasyon içermektedir.

d) Böyle bir bilgi-bölgesinden diğerine atlayabilmek için algoritma, gerçek zamanda bir takım istatistiki bilgileri hesaplar. Bunları şöyle sıralayabiliriz:

1) Ortalama değer;

$$\bar{x} = \frac{1}{n}\sum_{i=1}^{n} xi \tag{1.3}$$

Burada x_i sıkıştırılacak orjinal işaret örnekleridir.

2) İşaretlerin standart sapması s;

$$\sigma = \left[\frac{1}{n}\sum_{i=1}^{n}(x_i - \bar{x})^2\right]^{1/2} \tag{1.4}$$

3) İşaretin üçüncü momenti M;

$$M = \left[\frac{1}{n}\sum_{i=1}^{n}(x_i - \bar{x})^3\right]^{1/3} \tag{1.5}$$

Standart sapma ve 3. moment hesapları, yavaş değişim bölgelerinden hızlı değişim bölgelerine geçişi daha hassas olarak fark etmek amacıyla hesaplanmaktadırlar. Ancak gerçek zaman çalışmalarında bu parametrelerin dinamik olarak rekürsif biçimde hesaplanmaları gerekir. Bunun için de aşağıdaki formüller çıkarılmıştır:

1) Ortalama değer:

$$\bar{x}_k = \frac{(m-1)\bar{x}_{k-1} + x_k}{k} \tag{1.6}$$

2) Standart Sapma:

$$\sigma_k = [\frac{(k-1)\sigma_{k-1}^2 + (x_k - \bar{x}_k)^2}{k}]^{1/2} \tag{1.7}$$

3) Üçüncü moment:

$$M_k = [\frac{(k-1)M_{k-1}^3 + (x_k - \bar{x}_k)^3}{k}]^{1/3} \tag{1.8}$$

e) Her örnekten sonra bir kriter fonksiyonu, KF_k, aşağıdaki şekilde hesaplanır:

$$KF_k = C_1(\sigma_k + M_k) \tag{1.9}$$

Daha sonra bu kriter fonksiyonuna dayanarak, devamlı yenilenecek şekilde eşik değeri hesap edilir:

$$T_k = T_{k-1} - C_2(kF_k - KF_{k-1})T_{k-1} \tag{1.10}$$

C_1 ve C_2 sabitleri deneylerden edinilen tecrübeye göre 1 ve 0,08 civarında seçilebilirler. Eşik değeride (T_{min}, T_{max}) gibi sınırlar arasında tutulabilir.

Böylece standart sapma veya 3. momentteki herhangibir değişme eşik değerine yansımış, yavaş değişim bölgeleri büyük eşik değerlerine sahip olmuş ve sıkıştırma oranı yükselmiş olacaktır.

Sıkıştırma algoritması, belli bir başlangıç eşik değeri ile başlar ve sonra (1.9) ve (1.10) eşitlikleri, her örnekten sonra Tyi yenilemek amacıyla $(x_{max}-x_{min}) > T_k$ oluncaya kadar hesaplanır. Belirtilen eşitsizlik koşulu sağlandığında (x,k) değerleri yardımıyla çizgi saklanır ve işlem devam eder.

Yeniden Yapılama (Reconstruction) Algoritması:

Sıkıştırılmış veri, ($\bar{x}$,k) gibi veri çiftleri dizisinden oluşmuştur. Her çift, çizginin genlik ve uzunluğunu belirtir. Örneğin: (42, 5) = 5 adet 42 birim genlikli örnekten oluşan çizgiyi temsil etmektedir. Veri, bu şekilde açılır ve gözlenirse, bu kabul edilebilir bir görüntü oluşturmaz. Onun için bir eğri yumuşatma algoritması bu diziye uygulanır. Standart en az karesel polinom eğrisi uydurma tekniği burada yumuşatma filtresi olarak uygulanır. Bu teknik, en iyi uygunluğu işaretin her biri 7. örnekten oluşan setlerine tatbik edilmesiyle en iyi sonucu verir.

Burada 2 adım mevcuttur:

1- Elde edilmiş doğru çizgi verilerini alıp ayrım veri noktaları şeklinde açılır;
Örneğin: (42,5) = (42, 42, 42, 42, 42) gibi.

2- Her yedi noktaya yumuşatma filtresi uygulanır;

$$Y_k = \frac{1}{21}(-X_{k-3} + X_{k-2} + 6X_{k-1} + 7X_k + 6X_{k+1} + 3X_{k+2} - 2X_{k+3}) \quad (1.11)$$

Burada Y_k yeni veri noktasını ve x_{k+n} orijinal veri noktalarını temsil etmektedirler

AZTEC Tekniği (Amplitude Zone Time Epoch Coding):

Esasında AZTEC, yukarıda anlatılan değişken eşikli adaptif algoritmanın bir alt algoritması olarak kabul edilebilir. İkisi arasındaki tek fark, AZTEC'de belirlenen eşiğin, işlem boyunca değişmeden sabit kalması ve dolayısı ile standart sapma ve üçüncü moment gibi hesapların yapılmasına gerek kalmamasıdır.

AZTEC algoritması, orjinal EKG örnek noktalarını yatay ve eğimli çizgilere dönüştürür [Cox 1968]. AZTEC platosu (yatay çizgiler), sıfırıncı mertebe interpolatörü (ZOI) kullanılarak üretilirler. Her platonun bellekte saklanan değerleri çizginin genlik değerleri ve uzunluğudur. Bir AZTEC eğiminin başlaması için (plato oluşturabilmek için) eşik kapsamındaki örnek sayısının 3'den az olması lazımdır. Eğimin saklanan değerleri, süresi ve son örnek noktasının genliğidir. İşaretin yeniden yapılanması, AZTEC plato ve eğimlerine ait veri noktalarının ayrık dizilere açılmasıyla elde edilir.

Yeniden yapılanan EKG'deki süreksizliklerden ötürü AZTEC, yüksek oranda veri azaltması sağlasa da, kardiyoglar tarafından kabul edilecek düzeyde değildir. Bu süreksizliklerin büyük oranda azaltılması, yumuşatma yapan bir parabolik filtrenin kullanılmasıyla olur. Bu işlemin kusuru ise EKG dalga biçimine genlik distorsiyonunun katılmasıdır.

Bu durumda, AZTEC algoritmasının ZOI kısmı için kullanılan hata eşiğinin değişen EKG işaretlerine adapte edilmesiyle elde edilen yeni modifiye AZTEC algoritması önerilmiştir [Furth, Perez 1988]. Bu adaptiflik, işaretin ilk üç saniyesinin ardışık (recursive) hesaplanması temeline dayandırılmıştır. Bu teknik, ortalama karesel fark kökü yüzdesine bakıldığında, aynı sıkıştırma oranı için, AZTEC algoritmasına göre daha üstündür. Süreksizliği azaltmak amacıyla AZTEC'e dayalı diğer bir teknik geliştirilmiştir. Platonları üretmek için ZOI kullanmak yerine, eğim çizgileri oluşturmak FAN tekniği kullanılmıştır. Bu teknik, ZOI algoritmasının tabiatında işaret süreksizliği olduğu için kullanılmıştır. Daha sonra EKG işareti, bu eğim çizgileri ve AZTEC eğimleri bağlanarak tekrar yapılanır. Bu teknikle ilgili hesaplamalar, AZTEC algoritması ile karşılaştırıldığında göstermiştir ki, sıkıştırma oranında bir artış ve işaret performans indeksinde (PRD) %50'lik bir iyileşme olmuştur.

Dönüm Noktası Tekniği (TP=Turning Point):

Bu teknik, büyük genlikli QRS'lerin yüksekliğini azaltmadan 200 HZ'lik EKG işareti örnekleme frekansını 100 HZ'ye indirmeyi amaçlamıştır [Mueller 1978].

Algoritma aynı anda 3 veri noktasını işler; X_o referans veri noktası ve bunu takip eden X_1-X_2 veri noktaları. Ya X_1 yada X_2 saklı tutulacaktır. Bu ise hangi noktanın orjinal 3 noktanın eğimini koruduğuna bağlıdır. Bunu, matematiksel bir dille şöyle ifade edebiliriz.

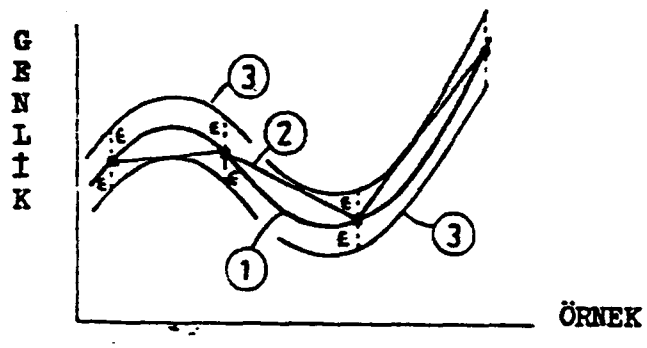
$(x_2 - x_1) \ddot{Y} (x_1-x_o) < 0$ ise $x_o = x_1$,

$(x_2 - x_1) \ddot{Y} (x_1-x_o) > 0$ ise $x_o = x_2$ olarak tutulacaktır.

TP algoritması, 2:1'lik sabit bir sıkıştırma oranı sağlar ki burada rekonstrükte edilen işaret orjinal işarete bazı gürültüleri içereliği halde benzer. TP yönteminin bir kusuru, eşit aralıklarla dizilmiş zaman aralıklarını kayıt edilmiş noktaların temsil etmemesidir

Sapa Teknikleri (Scan Along Polygonal Approximation):

Üç adet SAPA (Poligonal Yaklaşımı İzleme) algoritması vardır. SAPA-2 bunlardan en iyi sonuç verenidir. Bu algoritmanın teorik temeli, doğru çizgiler (approximated) ile orjinal işaret arasındaki sapmanın, hata toleransından hiçbir zaman daha büyük olmamasıdır. SAPA-2 ve FAN algoritmaları arasındaki tek fark, SAPA-2'nin orijinal örnek noktası ve gerçek sonraki örnek noktası arasında 3. bir eğim (merkez eğim, center slope) hesaplamasıdır. Merkez eğim, iki nokta arasında değişen işaretin belli hata sınırları dışına taştığı zaman, sondan bir önceki örnek noktası kalıcı (sürekli) örnek noktası olarak dikkate alınır. Başka bir deyişle, SAPA-2 algoritması, örneğin kalıcı mı yoksa geçicimi olduğunu gerçeklemek için merkez-eğim kriterini kullanır (Şekil 1.2). Oysa ki FAN'da gerçek örnek değeri kriteri kullanılmaktadır [Mueller 1978].



Şekil 1.2. Zaman Poligonal Yaklaşımın Ana Fikri.
1- Orjinal İşaret
2- Orjinal İşarettin. Poligonal Yaklaşımı
3- Yaklaşım Hata Sınırı

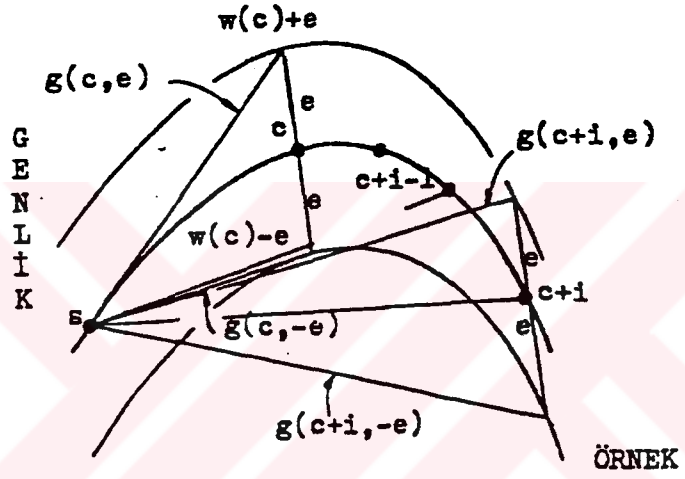
SAPA-1:

Orjinal örneklenmiş veri, w(k) gibi ayrık dizi şeklinde gösterilsin. Burada k, örnek numarası olsun. ε, kullanıcı tarafından belirlenecek olan hata aralığını temsil etsin. Tekrar yapılanmış işaret w(k) ayrık dizisi ile gösterilsin, öyle ki bu dizi $|w(k) - w(k)| < \varepsilon$, k=1,2,3,... şartını sağlasın. Orjinal veri (k, w(k), k=1,2,3,...) ile ve azaltılmış veri ise (k, w(k), $k=s_1, s_2,...$) şeklinde temsil edilsin. (Örnek numarası, genlik) çifti şeklinde gösterilen bu azaltılmış veriye orjinal verinin verteksleri denir. İki verteks arası düz bir çizgi çekmek suretiyle yaklaşıklık yapılır. Algoritmaya başlarken ilk veri noktası verteks olarak seçilir. k=c'deki daha sonra gelen örneğin alınmasından sonra normalize edilmiş iki doğrunun eğimleri hesaplanır. Bunlar,

$$g(c,\varepsilon) = \frac{w(c) + \varepsilon - w(s)}{c - s}$$

$$g(c,-\varepsilon) = \frac{w(c) - \varepsilon - w(s)}{c - s} \qquad (1.12)$$

şeklindedir. Şekil 1.3.'de bu eğimler gösterilmiştir. Her yeni veri noktası işlendiğinde, o anki en küçük g(c,ε) değeri m_1 ve en büyük g(c,-ε) değeri m_2 olarak saklanır. Böylece m_1 ve m_2, s'den gelen ve ±ε aralığı içinde bulunan doğruların maksimumu ve minimum eğimlerini göstermiş olur. Ne zaman $m_2 > m_1$ koşulu sağlanırsa (Şekil 1.3'de k=c+i noktasında olduğu gibi) hemen bir önceki veri noktası yeni verteks olarak seçilir [Ishijima 1983].



Şekil 1.3. SAPA-1 algoritmasını açıklayıcı şekil.

Sapa-2:

Bu algoritma SAPA-1'in biraz daha iyileştirilmişi olup ekstra bir kontrol mekanizması eklenmesiyle elde edilmiştir. Burada g(c,ε) ve g(c,-ε) eğimlerine ek olarak,

$$g(c,0) = \frac{w(c) - w(s)}{c - s} \tag{1.13}$$

hesaplaması katılmıştır.

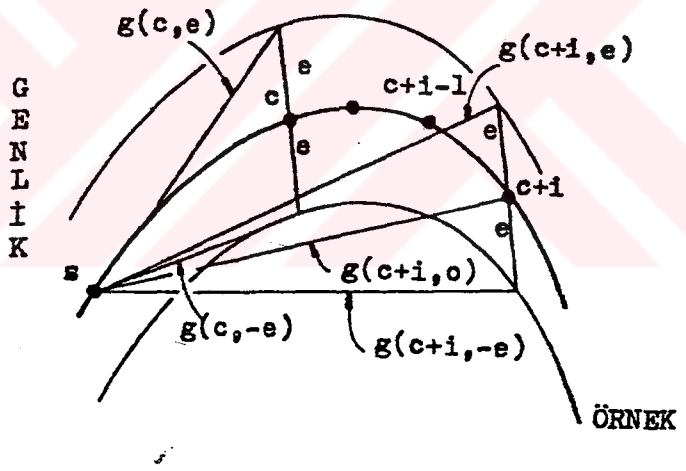
k=s ve k=c+i noktaları arasındaki noktaların uygun şekilde yaklaşıklıklarının yapılıp yapılmadığını anlamak için de,

$$g(c+i,0) \leq m_1 \tag{1.14}$$

ve

$$g(c+i,0) \geq m_2 \tag{1.15}$$

test hesaplamaları yapılmıştır (Şekil 1.4). Bu iki test koşullarından biri bozulduğunda hemen bir önceki veri, verteks olarak tutulur.



Şekil 1.4. SAPA-2 algoritmasını açıklayıcı şekil.

BÖLÜM 2

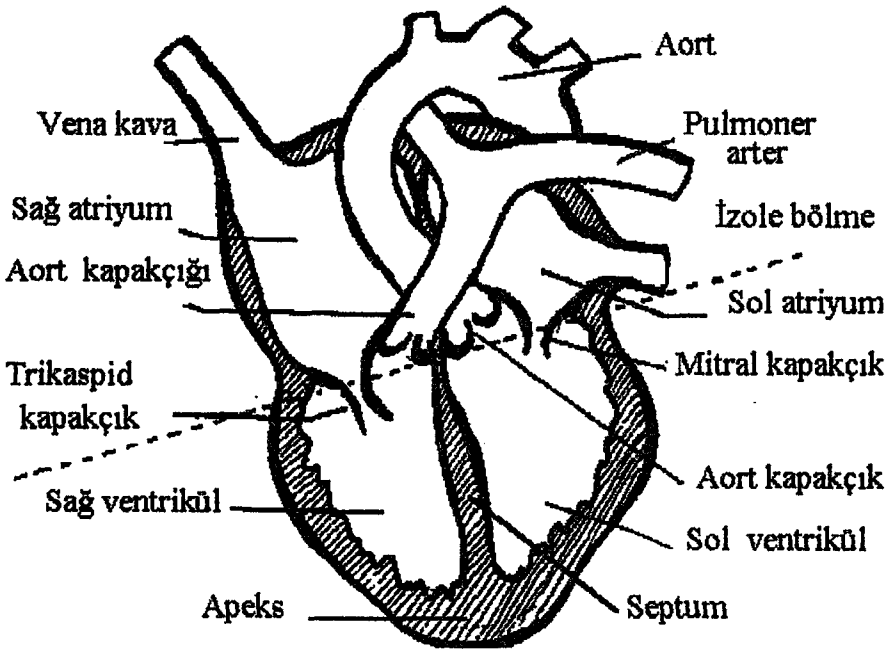
KALBİN ELEKTROFİZYOLOJİK YAPISI

İnsan vücudundaki tüm kaslar biyoelektrik uyarılarla kasılmaktadır. Uyarılan kasın yakınına konan bir çift elektrot aracılığıyla, kasın kasılmasını sağlayan elektriksel işaretlerin izlenmesi mümkündür. İnsan kalbi de, belirli bir ritmde, kendi içindeki sinüs düğümü (SA düğümü) tarafından üretilen elektriksel işaretlerle uyarılmakta, bu uyarı sonucunda da periyodik olarak kasılıp gevşeyerek vücudun ihtiyaç duyduğu kanı pompalamaktadır. Kalpteki elektriksel değişikliklerin kaydedilmesi işlemine Elektrokardiyografi (EKG), kaydedilen (ve/veya görüntülenen) değişime Elektrokardiyogram ve kaydedici cihazlara da Elektrokardiyograf adı verilir. Elektrokardiyografi kardiyolojide kalpteki rahatsızlıkları tanı amacıyla kullanılmaktadır.

Elektrokardiyografi işleminin nasıl gerçekleştirildiği incelenmeden önce aşağıda kalbin elektriksel iletim sistemi açıklanmıştır.

2.1. Kalbin Elektriksel İletim Sistemi

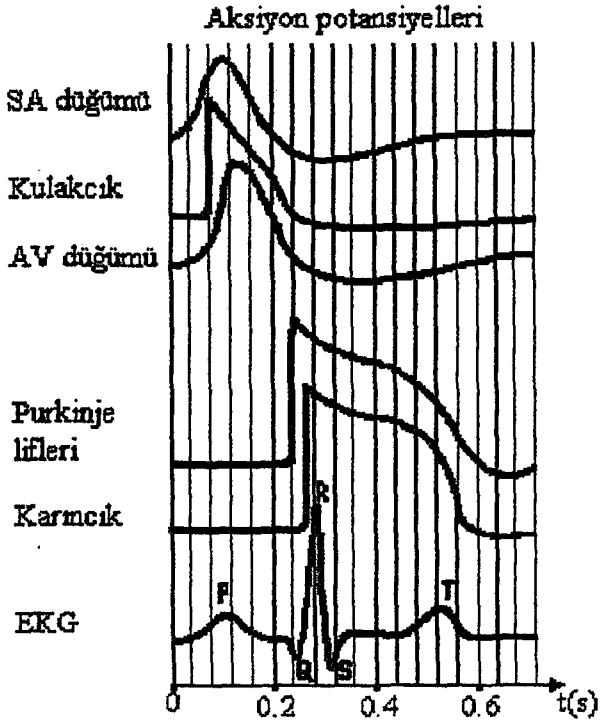
Şekil-2.1'de kalbin elektriksel iletim sistemi görülmektedir. İletim sistemi sinoatriyal düğüm (" sinoatrial node ", SA), his demeti ("bundle of his"), atriyoventriküler düğüm (" atrioventricular node ", AV), demet kolları (" bundle branches ") ve purkinje fiberlerinden oluşur [Yazgan 1989]. Kalbin kasılıp ve gevşemesi bu sistemlerin fonksiyonel bir şekilde çalışmasıyla olmaktadır.



Şekil-2.1 Kalbin elektriksel iletim sistemi

Superior vena-kava'nın sağ kulakçığa açıldığı yerde, kulakçığın iç tarafında (sağ atriyum'un arka duvarında) yer alan, kendi kendine aksiyon potansiyeli üreten (3x10 mm boyutunda) özelleşmiş kalp hücrelerinden oluşmuş olan bir sinoatrial düğüm (sinüs düğümü, SA düğümü) mevcuttur. S-A düğümü, kalbin peysmeykırlarından (kendi kendine uyarım yapan hücrelerinden, "pacemaker") biri olarak çalışır. "Pacemaker", hareketi başlatan, hareketin hızını belirleyen anlamına gelmektedir.

S-A düğümünde kendi kendine oluşan aksiyon potansiyeli, depolarizasyon dalgası halinde tüm kalbe yayılır. S-A düğümünün oluşturduğu aksiyon potansiyellerinin frekansı, değişen koşulların gereksinimini sağlamak üzere merkezi sinir sistemi tarafindanda kontrol edilmektedir. Şekil-2.2'de kalbin çeşitli bölgelerindeki aksiyon potansiyellerinin zamana göre değişim şekilleri görülmektedir. S-A düğümü, bu potansiyellerin kaynağı olduğundan bu düğüme ait aksiyon potansiyeli zaman ekseninde en solda görülmektedir.



Şekil-2.2 Kalpdeki aksiyon potansiyelleri

S-A düğümünde meydana gelen depolarizasyon dalgası, atriyumlar üzerindeki iletim yolları üzerinden hızla yayılarak her iki atriyum'un kasılmasını sağlar ve buradaki kan ventriküllere basılır. Atriyumlardaki aksiyon potansiyeli hızı 30 cm/s kadardır. Şekil-2.2'de görülen, tüm aksiyon potansiyellerinin bileşkesi olan ve kalbin elektriksel aktivitesini temsil eden, elektrotlarla vücut yüzeyinden algılanan EKG (Elektrokardiyogram) işaretinde, kalbin istirahat durumundaki izoelektrik seviyesi üzerinde P dalgası oluşur. 0,1 saniye kadar süren P dalgasının genliği, atriyum kasının aktivitesi hakkında bilgi verir.

Sağ kulakçıkta ("right atrium") (kulakçıklar arası bölmeye ve sağ karıncığa yakın bir yerde) kendi kendine uyarım yapabilen, S-A düğümü gibi ince lifli (Şekil-2.1) ikinci bir düğüm daha vardır ki buna A-V düğümü, atriyo-ventriküler düğüm adı verilir. Bunun kendi kendini uyarma hızı S-A düğümünün uyarı hızından daha düşüktür.

"Pacemaker" hücrelerinin (S-A düğümü, A-V düğümü gibi) ve kalp kası hücrelerinin kendiliklerinden aktivite göstermelerine kateşolaminlerin (epinefrin, norepinefrin vb) neden olduğu sanılmaktadır. Kateşolaminler, hücre içine Ca++ girişini kolaylaştırmaktadırlar. Hücre içine Ca++ girişi ise, kateşolaminlerin hücreden çıkmasına neden olur (depolarizasyon). Bütün kateşolaminlerin hücre dışına çıkması hücre içine Ca++ girişini durdurur ve böylece depolarizasyon sona erer. Aktif pompa ile kalsiyum hücreden dışarı pompalanır ve hücre yeniden polarize (repolarize) duruma geçer. Yavaş yavaş kateşolaminler şekillenerek yeter düzeye erişince depolarizasyon yeniden başlar.

Sino-atriyal ve atriyo-ventriküler düğümler arasında özelleşmiş bir ileti sistemi bulunmaz ve uyarılar, A-V düğüm lifleri arasına kadar uzanan atriyum kas lifleri tarafından iletilir. A-V düğümünde iletim hızı 45 cm/s kadardır. S-A düğümünde oluşan aksiyon potansiyeli, 30-50 milisaniye sonra A-V düğümüne ulaşır. Bu süre atriyumların içindeki tüm kanın ventriküllere boşaltılabilmesi için yeterli değildir. A-V düğümü, içindeki ileti hızının (5 cm/s) düşük olması nedeniyle, bir gecikme elemanı gibi görev yapar ve aksiyon potansiyelinin, ventriküllere 110 ms sonra ulaşmasına neden olur. Atriyumlarla ventriküller (kulakçıklarla karıncıklar) arasında bulunan, elektriksel uyarıları geçirmeyen yalıtkan ve yağlı bir tabakanın varlığı da, uyarının sadece iletim sistemi (AV düğümü) üzerinden gecikerek geçmesini sağlar. Bu şekilde geciktirilen aksiyon potansiyeli sayesinde karıncıklar, kulakçıklardan yeterince bir süre sonra kasılırlar. Aksi halde kulakçık ve karıncıklar aynı anda kasılacak ve bunun sonucunda karıncıklar, tam dolmadan kasılmaya çalışacaktı.

İleti sisteminin A-V düğümünden sonraki elemanı olan His demeti, uyarıyı "apex" bölgesine (kalp ucu bölgesine) kadar çok hızlı bir şekilde (3 m/s) iletir. His demetinin sağ ve sol kolları apeks bölgesinden sonra, miyokardiyum (kalp kası) içinde kalbin yukarı taraflarına doğru uzanır ve ince dallara ayrılarak purkinje sisemini (ağını) oluştururlar.

Ventriküllerin uyarılması Purkinje sistemi ile olur. Kasılan miyokardiyum, ventriküllerdeki kanı arterlere (atar damarlara) pompalar.

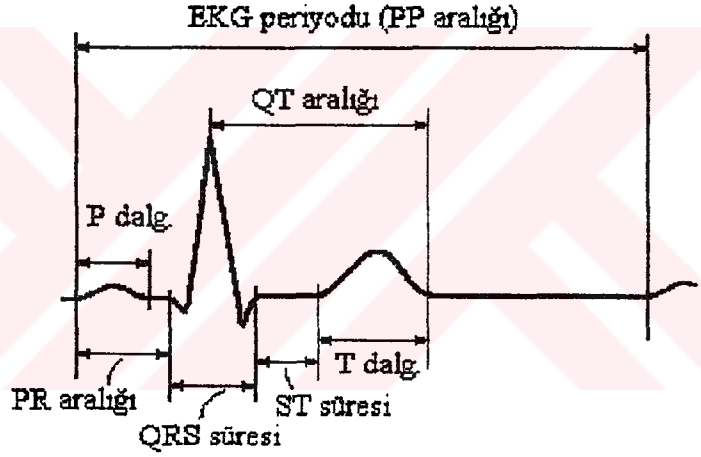
2.2 Kalp Dalgaları ve Elektrokardiyogram

Kalp kaslarının aynı anda kasılması sonucu büyükçe bir elektriksel işaret oluşur. Hastanın vücudu üzerinden algılanabilen bu işaretin zamana göre değişimine de Elektrokardiyogram adı verilir.

EKG eğrisi üzerinde değişik özellikler gösteren kısımlar harflerle karakterize edilir. P dalgası olarak isimlendirilen kısım, atriyumların kasılması sonucu oluşur. Uyarı A-V düğümüne geldiğinde kulakçık kasları kasılma durumundadır. P dalgasının süresi, Aksiyon potansiyelinin S-A düğümünden A-V düğümüne geçişini belirtir. P dalgasından Q dalgasına kadar EKG işareti izoelektrik seviyede kalır [Korürek 1989]. PR aralığı, his demeti iletim zamanını gösterir. P dalgasının başlangıcından Q'ya kadar geçen en çok 0.2 sn'lik süre içinde uyarı dalgası, sinüs düğümünden itibaren kulakçıklardan, A-V düğümünden ve his demetinden geçerek kalp ucuna gelir. Bu ileti zamanının (Şekil-2.3) veya PR aralığının 0.2 sn'den uzun olması, his demetinde bir ileti bozukluğunu belirtir.

QRST dalgaları, Ventriküler Kompleks olarak isimlendirilir. A-V düğümünden his demetine geldiğinde kalp ucuna henüz uyarı gelmemiştir ve kalp ucu tabana (yalıtkan plakaya) göre daha negatiftir (Q dalgası). Uyarı kalp ucuna geldiğinde ise o bölgedeki kas hücreleri depolarize olur ve apeks tabana göre oldukça fazla pozitif olur (R dalgası). Uyarı bütün karıncığa yayılırken potansiyel farkı azalır, hatta bilinmeyen nedenlerle, S çıkıntısına uyan süre içinde uç, tabana göre daha negatif olur. Buna göre QRS kompleksi ventriküllerin depolarize olmasını temsil eder ve 0.1 sn sürer. QRS kompleksinin 0.1 sn'den daha uzun sürmesi karıncıkların hem zaman olarak uyarılmadıklarını gösterir. Karıncık kaslarının, kulakçık kaslarından daha fazla ve güçlü olması nedeniyle R dalgası P dalgasından genlik olarak oldukça büyüktür. Bunun yanında R dalgasının genliğinin çok

yüksek olması hipertansiyonda kalp büyümesinin bir göstergesi olabilir. S dalgasından sonra karıncık kasları aynı potansiyeldedir ve bu nedenle T dalgasına kadar EKG, izoelektrik seviyededir (ST segmenti). Karıncık kaslarının hızlı repolarizasyonu sonucu T dalgası oluşur. T dalgasının pozitif olmasının nedeni repolarizasyona önce, en son depolarize olan, tabana yakın kas hücrelerinin başlamasıdır. Q dalgasından, T dalgasının sonuna kadar karıncık sistolü (kasılması) devam eder. T dalgasından sonra karıncıklar dinlenmeye geçer ve EKG izoelektrik seviyede kalır. Şekil-2.2'de kalbin çeşitli bölümlerindeki aksiyon potansiyellerinin değişimleri gösterilmiştir. Görüldüğü gibi EKG işaretinde dalgalar, aksiyon potansiyellerinin hızlı değişimlerinde ortaya çıkar. Buna göre EKG işareti, aksiyon potansiyellerinin türevleri ile ilgilidir.



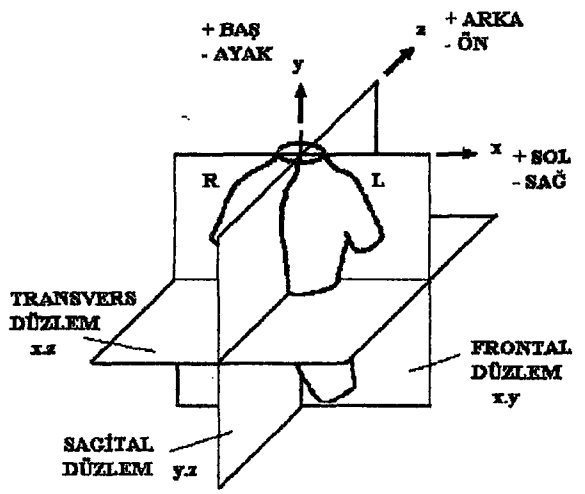
Şekil-2.3 EKG işareti

Kalp kasının diğer bir özelliği de, iskelet kasından daha uzun refrakter (cevap vermeme, tesirsiz kalma) süreye sahip olmasıdır. Kalp, sistol süresince uyarılara cevap vermez ki bu süreye mutlak refrakter devir adı verilir. Kalp kası diyastol (gevşeme) süresinde (fazında) uyarılabilir, ancak diyastol başlangıcında (relatif refrakter fazda) uyarılabilmesi için kuvvetli uyaranlar gereklidir. Diyastol başlangıcında tatbik edilen etkili uyaranla kalpte vaktinden evvel meydana gelen sistole, ekstrasistol denir. Ekstrasistolden sonra gelen dinlenme devri (diyastol süresi), önceki diyastol süresini kompanze edecek

şekilde daha uzundur. Çünkü kalbin normal ritmik faaliyetini sağlayan uyarı, ekstrasistolün mutlak refrakter fazına rastladığı için etkisiz kalır ve bundan sonra yeni doğacak uyarıya kadar, kalpte uzun bir dinlenme devri görülür.

Kalbin ritmik faaliyeti durdurulduktan sonra eşik değerden daha zayıf elektriksel bir uyaran uygulanırsa cevap alınamaz, uyaran eşik değere ulaşırsa, kalp maksimal bir kasılma ile cevap verir. Eşik değerden daha şiddetli uyaranlarla hep aynı maksimal kasılma elde edilir ki buna "Ya Hep Ya Hiç" yasası adı verilir. Bu sayede, kalpte bir kas lifinin uyarılmasıyla bütün lifler faaliyete geçirilerek kısa zamanda toplu kas kasılması elde edilebilir.

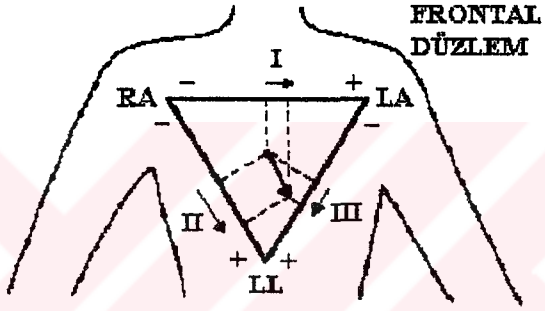
Kalpteki elektriksel değişikliklerin kaydedilmesi işlemine ve bu işlemle ilgili sisteme elektrokardiyografi (EKG) denildiği belirtilmişti. Bir elektrik üreteci gibi çalışan kalbin gövde içerisine tamamen gömülü olması nedeniyle üreteç çıkışı ancak bir ameliyatla doğrudan ölçülebilir. Bir hacimsel iletken olan gövdenin yüzeyindeki çeşitli noktalar arasında yapılan potansiyel ölçümleri aracılığı ile kalbin durumu belirlenebilir. Böylece kardiyak vektörü istenilen referans düzlemlerindeki eksenler üzerine izdüşürülebilir. Uygulamada referans düzlem olarak, "Frontal", "Transverse" ve "Sagittal" düzlemler alınır (Şekil 2.4).



Şekil-2.4 Referans düzlemler

Einthoven Üçgeni

Elektrokardiyografin elektrotları sol kol, sağ kol ve sol bacağa bağlanarak (sağ bacak referans noktası alınmak üzere) Şekil-2.5'de gösterilen Einthoven üçgeni oluşturulur. Bu üçgenin merkezinde kalp bulunur. Burada kalp tabanı ile ucu (apex) arasında bağlı olan ve şekilde bir vektörle gösterilen bir biyoelektrik gerilim kaynağı ile temsil edilmektedir. Bu vektörün, Einthoven üçgeninin kanarları üzerindeki izdüşümleri, standart bipolar derivasyonlar olarak bilinir :



Şekil-2.5 Einthoven üçgeni

Derivasyon I : sağ kol, sol kol,

Derivasyon II : sağ kol, sol bacak

Derivasyon III : sol kol, sol bacak

Üç elektroddan ikisi derivasyon seçici (komutatör) ile elektrokardiyografa bağlanabilmektedir. Ayrıca unipolar göğüs derivasyonları (V1'den V6'ya kadar), unipolar kol ve ayak derivasyonları (aVR, aVL, aVF) da mevcuttur.

BÖLÜM 3

GENEL KAVRAMLAR

Bu bölümde Ayrık Kosinüs Dönüşümü tanımlanarak tek boyutlu Ayrık Kosinüs Dönüşümünün nasıl elde edildiği gösterilecektir. Ayrıca enformasyon teorisine kısa bir giriş yapılıp veri sıkıştırma yöntemlerinin değerlendirilmesi için gerekli tanımlar verilecektir.

3.1. Ayrık Kosinüs Dönüşümü– DCT –

Altmışların sonu ve yetmişlerin başında dijital ortagonal transformlar ve onların kullanım alanları olan görüntü sıkıştırmasında çok büyük bir araştırma hareketi vardı. Öyle ki bir çok yeni transform diğerlerinden daha iyi performansa sahip oldukları iddiası ile ortaya çıkmıştı. Aynı zamanlarda bir takım araştırmacılar da bu transformların birbirleriyle karşılaştırılması üzerinde çalışmaktadır. Sonunda Karhunen-Loeve transformu (KLT), karşılaştırma amacı ile optimal transform olarak türetildi. Bu sıralarda Nasır Ahmet, Kansas State Üniversitesi'nde bir grup araştırmacıyla AKD üzerine çalışmaktaydı. Grubun başındaki Nasır Ahmet ve arkadaşlarının 1973 yılının yazında yaptıkları çalışmada AKD'den aldıkları sonuç beklediklerinden çok daha iyi oldu. Öyle ki performansı neredeyse KLT kadar iyi idi. İlk olarak 1974 yılında IEEE Computer Transactions'da yayınlanan AKD şimdi bir standart haline gelmiştir[Ahmet 1974]. Konumuz veri sıkıştırma olduğundan tek boyutlu ayrık kosinüs dönüşümü tanımlanacaktır.

3.2 Tek Boyutlu AKD

x(n), N-noktada tanımlı bir dizi olsun öyle ki; $0 \le n \le (N-1)$'in dışında iken 0 olsun. En sık kullanılan uygulama olan çift simetrik AKD'yi oluşturacağız. Bunun için önce yeni bir 2N noktada tanımlı olan y(n)'i oluşturursak, önce y(n)'in 2N noktada tanımlı AFD Y(k) alırız, buradan Y(k) ile Cx(k) arasında ilişki kurarız ki Cx(k), N noktalı AKDdir.

$$\underset{x(n)}{\overset{N-noktalı}{}} \leftrightarrow \underset{y(n)}{\overset{N-nokta}{}} \xleftrightarrow{AFD} \underset{Y(k)}{\overset{N-nokta}{}} \leftrightarrow \underset{C_x(k)}{\overset{N-nokta}{}} \qquad (3.1)$$

x(n) ile y(n)'in arasındaki bağıntı:

$$y(n) = x(n) + x(2N-1-n) \qquad (3.2)$$

$$y(n) = \begin{cases} x(n), \ldots\ldots\ldots\ldots\ldots\ldots 0 \le n \le N-1 \\ x(2N-1-n), \ldots\ldots\ldots 0 \le n \le N-1 \end{cases} \qquad (3.2b)$$

2N-noktada tanımlı y(n)'in AFD'nü alırsak (Ayrık Fourier Dönüşümü), Y(k) elde edilir.

$$Y(k) = \Sigma y(n) W_{2N}^{kn}, \ldots\ldots\ldots\ldots 0 \le k \le 2N-1 \qquad (3.3)$$

$$W_{2N} = \exp(-j2\pi / 2N) \qquad (3.4)$$

(3.3) ve (3.4) nolu denklemlerden,

$$Y(k) = \sum_{n=0}^{N-1} x(n) W_{2N}^{kn} + \sum_{n=N}^{2N-1} x(2N-1-n) W_{2N}^{kn}, \ldots . 0 \le k \le 2N-1 \qquad (3.5)$$

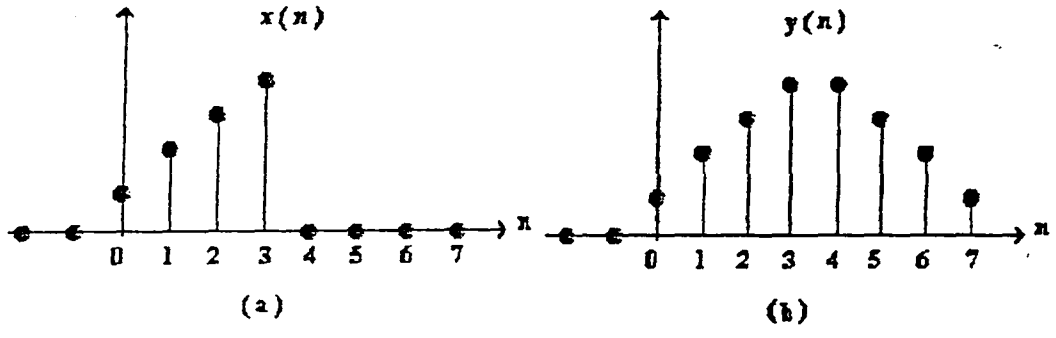
(3. 5) denklemini birkaç değişkenini değiştirip biraz da cebrik işlemle alttaki şekle sokabiliriz.

$$Y(k) = W_{2N}^{-k/2} \sum_{n=0}^{N-1} 2x(n) \cos \frac{\pi}{2N} k(2N+1), \ldots\ldots\ldots 0 \le k \le 2N-1 \qquad (3.6)$$

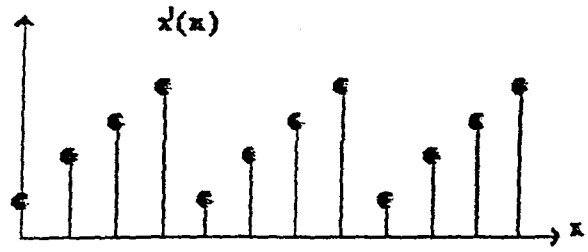
ve N noktada tanımlı AKD (Ayrık Kosinüs Dönüşümü) Cx(k)'yı aşağıdaki şekilde elde ederiz.

$$C_x(k) = \begin{cases} W_{2N}^{kn} Y(k), \ldots\ldots\ldots 0 \le k \le N-1 \\ 0, \ldots\ldots\ldots\ldots\ldots\ldots\ldots dışında \end{cases} \qquad (3.7)$$

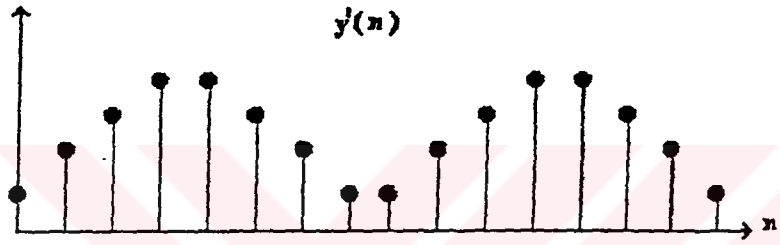
x(n) ve y(n) bağıntılarına N=4 için bir örnek Şekil-3.1'de görülmektedir. Şekilden de görebileceğimiz gibi y(n) bağıntısı, örnekleme noktasının yarısı olan n=(N-1)/2 noktasına göre simetriktir. Eğer periyodik bir x'(n) bağıntısı oluşturursak, x'(n)'nin süreksizlik noktaları olduğunu görürüz, çünkü x(n)'in başlangıç ve bitiş noktaları ard arda birleştirilmiştir. Fakat y'(n)'i oluşturduğumuzda görüyoruz ki bu süreksizlikler yoktur.



Şekil-3.1 (a) x(n) ve (b) y(n) bağıntılarına örnek. y(n) = x(n) + x(2N - 1 - n) dir ve ayrık kosinüs transformunda ara adımdır.



(a)



(b)

Şekil 3.2 Periyodik x'(n) bağıntısı ve ondan elde edilen y'(n) bağıntısı

(3.6) ve (3.7) denklemlerinden,

$$C_x(k)=\begin{cases}\sum_{n=0}^{N-1} 2x(n)\cos\frac{\Pi}{2N}k(2N+1), \ldots\ldots\ldots 0\le k\le N-1\\ 0, \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \textit{dışında}\end{cases} \qquad (3.8)$$

(3.8) Denklemi AKD'nün tanım denklemidir. Denklemden de anlayabileceğimiz gibi Cx(k), N-noktalı bir bağıntıdır. Dolayısıyla x(n)'in N noktası Cx(k)'nın N noktası ile temsil edilir. Eğer x(n) reel ise Cx(k) da reeldir ve tersi eğer x(n) kompleks ise Cx(k) da komplekstir.

Ters ayrık kosinüs dönüşümünü almak istersek (IDCT), ilk önce Cx(k)'yı Y(k) bağıntısına dönüştürmeliyiz. Daha sonra da Ters Fourier Dönüşümü alarak y(n)'i elde ederiz. Y(k) ve Cx(k) arasındaki ilişkinin aşağıdaki şekilde olduğunu gösterebiliriz.

$$Y(k)=\begin{cases} W_{2N}^{-k}Y(2N-k),.....0\le k\le 2N-1 \\ 0,.....k=N \end{cases} \tag{3.9}$$

(3.6) ve (3.9) denklemlerinden,

$$Y(k)=\begin{cases} W_{2N}^{-k/2}C_x(k),.....0\le k\le N-1 \\ 0,.....k=N \\ -W_{2N}^{-k/2}C_x(2N-k),.....N+1\le k\le 2N-1 \end{cases} \tag{3.10}$$

elde edilir. Y(k) bağıntısına IDFT ("Ters Ayrık Fouirer Dönüşümü") uygulanırsa elde edilen denklem şöyle olur,

$$y(n)=\frac{1}{2N}\sum_{k=0}^{2N-1}Y(k)W_{2N}^{-kn},.....0\le k\le 2N-1 \tag{3.11}$$

ve sonunda x(n) ifadesi y(n)'den şu ilişki ile elde edilir.

$$x(n)=\begin{cases} y(n),.....0\le n\le N-1 \\ 0,.....dışında \end{cases} \tag{3.12}$$

(3.10), (3.11) ve (3.12) denklemlerini birleştirirsek, elde edeceğimiz yeni bağıntı şöyle olur...

$$x(n)=\begin{cases} \frac{1}{N}\left[\frac{C_x(0)}{2}+\sum_{k=1}^{N-1}C_x(k)\cos\frac{\Pi}{2N}k(2n+1)\right],.....0\le n\le N-1 \\ 0,.....dışında \end{cases} \tag{3.13}$$

(3.13) Denklemi şu şekilde de ifade edilebilir...

$$x(n)=\begin{cases}\frac{1}{N}\left[\frac{C_x(0)}{2}+\sum_{k=1}^{N-1}C_x(k)\cos\frac{\Pi}{2N}k(2n+1)\right],.....0\le n\le N-1\\ 0,.....dışında\end{cases} \quad (3.14a)$$

burada $$w(k)=\begin{cases}\frac{1}{2},.....k=0\\ 1,.....1\le k\le N-1\end{cases} \quad (3.14b)$$

(3.8) ve (3.14) denklemlerinden tek boyutlu DCT çiftini görebiliriz.

Tek Boyutlu Ayrık Kosinüs Transformu

ID-DCT

$$C_x(k)=\begin{cases}\sum_{n=0}^{N-1}2x(n)\cos\frac{\Pi}{2N}k(2N+1),.....0\le k\le N-1\\ 0,.....,dışında\end{cases} \quad (3.8)$$

$$x(n)=\begin{cases}\frac{1}{N}\left[\frac{C_x(0)}{2}+\sum_{k=1}^{N-1}C_x(k)\cos\frac{\Pi}{2N}k(2n+1)\right],.....0\le n\le N-1\\ 0,.....dışında\end{cases} \quad (3.13)$$

3.3 Veri Sıkıştırma

Veri sıkıştırma genellikle kodlama olarak anlaşılsada, kodlama gerekli verinin özel temsili için kullanılan genel bir terimdir. Enformasyon teorisi, verimli kodlama ile sonucundaki iletişim hızı ve hata olasılığı çalışmaları olarak

tanımlanabilir. Veri sıkıştırma enformasyon teorisinin bir kolu olarak görülebilir. Amacı, iletilen veri miktarını en aza indirgemektir.

Kod, kaynak mesajların (A kaynak alfabesinden verilerin), kodkelimelere (B kod alfabesindeki kelimelere) eşlenmesidir. Kaynak mesajlar, verinin parçalandığı temel birimlerdir. Bu temel birimler, kaynak alfebesinden tek bir sembol veya sembol dizisi olabilirler.

Kodlar, blok-blok, blok-değişken, değişken-blok veya değişken-değişken olarak sınıflandırılabilir. Blok-blok kodlar, kaynak mesajların ve kod verilerin sabit uzunlukta olduklarını belirtirken değişken-değişken kodlar, değişken uzunluktaki kaynak mesajları yine değişken uzunlukta kodkelimelere eşler. Blok-blok kodlar, sıkıştırma sağlamadığından bu çalışmanın konusu değildir. Çalışmanın konusu blok-değişken kodlar olacaktır.

Her kodkelimesi birbirinden ayrılabilen, yani kaynak mesajlarla kodkelimeler arasında birebir eşleme olan kodlara belirgin kod adı verilir. Her kodkelimesi, bir dizi kod kelimesi içinden ayrı olarak tanınabiliyorsa, belirgin kod tek olarak çözünebilir denir. Tek olarak çözülebilir kod, önek özelliğini taşıyorsa, yani hiç bir kodkelimesi bir diğerinin öneki değilse önek kodu adını alır. Kodkelimeleri (1,10000,00) kümesinden oluşan kod, tek olarak çözülebilen bir kod olduğu halde önek kodu değildir. Önek kodları anında çözülebilirler, yani kodlanmış bir mesaj kodkelimelerine ileriye bakmadan ayrılabilir. (1,10000,00) kodkelimeleriyle kodlanmış bir mesajı açmak için ileriye bakmak gerekir. Örneğin, "1000000001" kodlanmış mesajının ilk kod kelimesi "1" dir. Fakat bu, mesajın son kelimesi okunmadan anlaşılmaz. Bu çalışmadaki kodlar, tek olarak çözülebilme ve önek kod özelliklerini taşımaktadırlar.

3.4 Yöntemlerin Sınıflandırılması

Veri sıkıştırma yöntemleri mesaj ve kodkelimeleri uzunluklarından başka, statik veya dinamik olarak da iki gurupta incelenebilir.

Statik yöntemler, mesaj kümesinden kodkelime kümesine eşlemenin iletişim (ya da sıkıştırma) başlamadan önce belirlendiği yöntemlerdir. Bunlarda, herhangi bir mesaj hep aynı kod kelimesiyle temsil edilir. Veri sıkıştırma klasiği olan Huffman [Huffman 1952] kodlama statik bir yöntemdir. Huffman kodlamasında kodkelimelerin kaynak mesajlara eşlenmesi, kaynak mesajların verideki olasılıklarına göre yapılır. Veride sık geçen mesajlar (yüksek olasılıklı mesajlar) kısa kodkelimelere, nadir geçenler (düşük olasılıklı) uzun kodkelimelere atanır. Bu olasılıklar, sıkıştırma başlamadan önce veri bir kez okunduktan sonra belirlenir. Tablo 3.1, örnek verinin Huffman kodlamasını göstermektedir.

Dinamik yöntemlerde ise, mesaj kümesinden kodkelime kümesine olan eşleme zamanla değişir. Örneğin dinamik Huffman kodlamasında, mesajların

Tablo 3.1. Örnek veri için Huffman kodlaması

Kaynak mesaj	Olasılık	Kodkelime
1	2/40	1001
-1	3/40	1000
2	4/40	011
-2	5/40	010
3	6/40	111
-3	7/40	110
0	8/40	00
5	5/40	101

olasılıkları o ana kadar olan geçiş sayılarına göre hesaplanır. Kodkelimelerin mesajlara eşlenmesi herhangi bir zamandaki olasılık değerlerine göre devamlı olarak değişir. Bir mesaj iletişimin başında, verinin başlangıcında çok geçtiği için kısa kodkelimesiyle temsil edilebilir, halbuki bu mesaj, bütün veride çok düşük bir geçiş olasılığına sahip olabilir. İletişim ilerledikçe, daha yüksek olasılıklı mesajlar geçtiği zaman, kısa kodkelimeleri yüksek olasılıklı mesajlara eşlenecek ve bu mesaj, verinin kalan kısmında nadir geçtiği için daha uzun kodkelimeleriyle temsil edilmeye başlanacaktır.

Dinamik kodlar, verinin zamanla değişen karakteristiğine adapte oldukları için adaptif olarak da adlandırılırlar. Bazı adaptif yöntemler kaynak verideki değişen kalıplara adapte olurken [Welch 1984], bazıları referans yerelliğine [Bentley 1986], adapte olurlar. Referans yerelliği, bir çok verinin ortak özelliği olarak görülen bazı kalıpların kısa zaman aralıklarında sık geçmesi, sonra uzun süre kullanılmamasıdır.

Bütün adaptif yöntemler, tek-geçiş yöntemleridir, yani verinin sadece bir kez okunması yeterlidir. Statik yöntemler ise iki-geçiş gerektirir; biri mesajların verideki olasılıklarını hesaplamak ve mesaj ile kodkelimelerin eşlemesini yapmak için, diğeri sıkıştırmayı yapmak için... Bu yüzden, adaptif yöntemlerin kodlama ve kodçözme süreleri statik yöntemlerinkinden çok daha büyük olmadıkça, bu yöntemler ilk okuma gerektirmediğinden zaman açısından iyileştirme sağlarlar. Ek olarak, statik kodlamanın ilk geçişinde belirlenen eşleme kodlayıcıdan kodçözücüye gönderilmelidir. Bu eşleme, her dosya iletimi öncesi gönderilebilir veya bir eşleme üzerinde anlaşılıp bir kaç iletimde kullanılabilir. Tek-geçiş yöntemlerinde kodlayıcı, iletim süresince eşlemeyi dinamik olarak tekrar tekrar tanımlar. Kodçözücü de, kodkelimelerini aldıkça eşlemeyi tekrar tekrar tanımlar.

Bir yöntem de, ne tamamen statik ne de tamamen dinamik olmayıp melez de olabilir. Basit bir melez kodlamada, gönderici ve alıcı belli bir sayıda statik kodun bulunduğu birbirinin aynı kod kitapları tutarlar. Her iletişim öncesi kodlayıcı, kod kitabındaki kodlardan birini seçer ve seçtiği kodun adını ya da numarasını göndererek kod çözücüyü seçiminden haberdar eder. Kodçözücüde gelen kodkelimelerini seçilen koda göre çözer.

Bu çalışmada, melez kodlama yöntemi kullanılmıştır. Gönderici ve alıcı belli bir kod kitabını tutmaktadırlar. Bloklar içinde gelen veriler, olasılıklarına göre büyükten küçüğe doğru dizilmekte olasılıklara karşı gelen kodlarla eşleştirilmektedir. Kodçözücüde ise gelen kodkelimeleri seçilen koda göre çözülür.

3.5 Veri Sıkıştırma Modeli

Veri sıkıştırma tekniklerinin birbirleriyle karşılaştırılmasında bazı kıstaslar gerekmektedir. Tekniklerin karşılaştırılmasında kullanılan ölçülebilecek iki boyut vardır: algoritma karmaşıklığı ve sıkıştırma miktarı. Veri sıkıştırma veri iletiminde kullanıldığı zaman, amaç iletişimin en hızlı şekilde tamamlanmasıdır. İletişimin hızı da, gönderilen bit sayısına, kodlayıcının mesajı kodlaması için gerekli zamana ve kodçözücünün orjinal mesajı elde etmesi için geçen zamana bağlıdır. Veri saklama uygulamalarında ise sıkıştırma derecesi ana kaygı olsa da, tekniğin kullanılması için algoritmanın da verimli olması önemlidir. Statik teknikler için üç algoritma vardır: eşleme-bulma, kodlama ve kodçözme algoritmaları. Dinamik teknikler içinse algoritma sayısı ikiye düşer: kodlama ve kodçözme algoritmaları.

Çeşitli sıkıştırma ölçümleri önerilmiştir. Bunlar, *gereksizlik* [Shannon 1949], *ortalama mesaj uzunluğu* [Huffman 1952] ve *sıkıştırma oranıdır* [Davidson 1966; Rubin 1976; Ruth 1972]. Bu ölçülerin herbiri veri hakkında bazı varsayımlarda bulunur. Enformasyon teorisinde, genellikle kaynak mesajın bütün istatistiksel parametrelerinin tam olarak bilindiği varsayılır [Gilbert 1971]. En yaygın model, *ayrık belleksiz kaynaktır*. Bu kaynak, a1, a2, ... , an sembollerinden oluşan sabit bir alfabeden seçilen sembol dizisidir. Bu semboller alfabeden, bazı sabit p(a1),..., p(an) olasılıklarıyla istatistiksel açıdan bağımsız olarak rasgele seçilmiştir [Gallager 1968].

3.5.1 Gereksizlik ve Entropi

Veri sıkıştırıldığı zaman amaç, gereksizliği azaltarak sadece bilgiyi bırakmaktır. Bir kaynak mesaj x'in bilgi ölçüsü, -log p(x)'dir. Buradaki ve bundan sonra kullanılacak bütün logaritma fonksiyonlarında logaritma iki tabanına göredir. Burada p(x) x mesajının geçme olasılığıdır. p(x)=1 olduğu zaman, veride sadece bu mesaj geçmesi gerektiğinden x hiç bir bilgi taşımaz. Benzer olarak p(x) küçük olduğu zaman, x mesajı daha az geçer ve daha çok bilgi taşır. Bütün kaynak alfabesinin ortalama bilgi miktarı, her kaynak harfinin bilgi miktarının geçme olasılığı kadar ağırlığını almakla bulunur. Böylece, $\Sigma[p(x) \log p(x)]$ ifadesi elde edilir. Burada x, A kaynak mesaj alfabesinde bir semboldür. Bu ifade, kodlanmış mesaj için gerekli bit sayısına bir alt sınır koyan *entropi* kavramıdır [Shannon 1949]. Her mesajın kodkelimesinin uzunluğu, o mesajın bilgi miktarını taşımaya yetecek kadar olması gerektiğinden, bir kodlanmış mesajın boyutu entropiden küçük olamaz. Hiçbir sıkıştırma yöntemi, böyle bir modelde bir veriyi bilgi kaybı olmaksızın entropisinden daha az bite indiremez. Kodlanmış mesajın toplam bit sayısı en az, entropi ile verinin mesaj uzunluğunun çarpımı kadar olabilir.

Bir verinin bilgi gereksizliği, $\Sigma[\ p(x)\ l(x)\ ] - \Sigma[\ -p(x)\ \log\ p(x)\ ]$ ifadesi olarak tanımlanır. Burada l(x), x mesajını temsil eden kodkelimesinin uzunluğudur. $\Sigma[p(x)l(x)]$ ifadesi, kodkelime uzunluklarının geçiş olasılıklarıyla ağırlığını yani, ortalama kodkelime uzunluğunu verir. $\Sigma[-p(x)\ \log\ p(x)]$ ise ortalama bilgi miktarıdır. Böylece gereksizliğin, ortalama kodkelime uzunluğu ile ortalama bilgi miktarı farkı olduğu bulunur. Eğer kod, bir ayrık olasılık dağılımı için en az ortalama kodkelime uzunluğunu veriyorsa, bu koda minimum gereksizlik kodu denir.

Veride dört tip gereksizlik bulunabilir [Welch 1984]. Bu gereksizlik tipleri veride, yalnız başına yada bir kaçı birden bir gurup halinde görülebilir. Veri sıkıştırma yöntemleri, verideki bu gereksizlikleri yok etmeye çalışırlar. Bazı yöntemler sadece bir gereksizlik üzerinde konsantre olurlarken, bazıları birkaçına birden adapte olabilirler. En iyi yöntem, bütün gereksizlikleri yokeden yöntemdir. Fakat bu, teoride mümkün olsa da, uygulamada aşırı bellek ve zaman istediğinden imkansızdır.

3.5.2 Ortalama Mesaj Uzunluğu

Bir kodun verimliliğinin ölçüsü olarak Huffman [Huffman 1952], $\Sigma[p(x)l(x)]$ ifadesiyle verilen ortalama mesaj uzunluğunu kullanmıştır. Bu miktar aslında kodlanmış mesajın, yani kodkelimelerin ortalama uzunluğudur. Gereksizlik, ortalama kodkelime uzunluğu ile entropi arasındaki fark olduğundan ve entropi verilen bir olasılık dağılımına göre sabit olduğundan, ortalama kodkelime uzunluğunu en aza indirmek, gereksizliği en aza indirmek anlamına gelir. Eğer belli olasılık dağılımı için entropi sonsuza doğru giderken, ortalama kodkelime uzunluğunun oranı 1'e yaklaşırsa, o koda optimal kod denir. Optimallik, ortalama kodkelime uzunluğunu teorik minimuma yaklaştırır.

3.6 Elektrokardiogram veri sıkıştırması

EKG veri sıkıştırma teknikleri arasında karşılaştırma yapılırken aşağıda sıralanan özellikler dikkate alınmaktadır.

1) İşaret örnekleme frekansı (f_s): EKG işaretlerini sayısala çevirmekte kullanılan analog/sayısal çeviricilerde örnekleme frekansı amaca göre çeşitlilik gösterip yaygın olarak 500Hz'lik örnekleme frekansı kullanılmaktadır.

2) Sayısal örneklerdeki bit sayısı (p): Saklanacak EKG sayısal bilgilerinin çözünürlüğünün (rezolüsyonunun) göstergesi olan bit sayısı 8 veya 12 olabilmektedir.

3) Veri sıkıştırma oranı (CR, compression ratio): Veri sıkıştırma algoritmasının önemli parametrelerinden biri olup bu oranın büyüklüğü algoritmanın üstünlüğünü gösterir.

$$CR = \frac{Sıkıştırılacak \quad örneklerin \quad sayısı}{Sıkıştırılmış \quad örneklerin \quad sayısı} \tag{3.15}$$

4) Performans indeksi (PRD, percent rootmean square difference): Orijinal EKG işareti, x_{org} ile sıkıştırılmışının yeniden yapılanması, x_{rec} (rekonstrükte edilmişi) arasındaki farkın bağıl karesel ortalamasının karekökü olarak tanımlı olup yeniden yapılanmış (oluşturulmuş) olan EKG işaretinin orijinaline ne derece benzer olduğunun ölçüsü olarak kullanılmaktadır. PRD, veri sıkıştırma algoritmalarının önemli karşılaştırma parametrelerinden bir diğeri olup PRD'nin küçüklüğü algoritmanın başarı derecesini gösterir.

$$PRD = \sqrt{\frac{\sum_{i=1}^{n} [xorg\ (i) - xrec\ (i)]^2}{\sum_{i=1}^{n} x^2 org\ (i)}} \tag{3.16}$$

5) Algoritmanın hızı ne kadar yüksekse o oranda hızlı işlem yapılabilir ve gerçek zaman çalışmaları gerçekleştirilebilir.

6) EKG sıkıştırma algoritmalarında kullanılan veri tabanlarının çoğu standart dışıdır. Oysa kullanılacak veri tabanlarına göre de algoritma sonuçları farklı olabilmektedir.

BÖLÜM 4

SIKIŞTIRMA YÖNTEMLERİ

Sıkıştırma yöntemleri genel olarak iki gruba ayrılır. Birincisi, gürültülü diğeri gürültüsüz yöntemlerdir. Gürültülü yöntemler, iletişim sırasında veride meydana gelebilecek hata ya da kayıplara tölerans gösteren yöntemlerdir. Bunlar genellikle parazitli iletişim hatlarında görüntü iletişiminde kullanılabilirler. İletişim sırasında hattın parazitli olmasından dolayı sıkıştırılan görüntüde bazı bozulmalar meydana gelebilir. Bu yöntemler sıkıştırılmış görüntüyü orjinal görüntüye yakın bir görüntüye genişletirler. Gürültüsüz yöntemler ise, veri saklama gibi güvenilir ve kayba tahammülsüz ortamlarda kullanılır. sıkıştırılmış veri elde edilirken orjinal verinin aynısı elde edilir. Bu çalışma sadece gürültüsüz yöntemleri içine almaktadır. Gürültüsüz yöntemler de içerik-bağımlı, statik ve dinamik olmak üzere üç başlık altında anlatılacaktır.

4.1 İçerik Bağımlı Yöntemler

Bu yöntemler, çeşitli uygulamalarda görülen bazı yerel gereksizlik tipleri için kullanılırlar. Bunlar ilginç ve kendi alanlarında değer taşıyan yöntemlerdir. En büyük dezavantajları, sınırlı sayıda kullanılabilmeleridir. Programlandıkları tipte dosyalardaki gereksizliklere çok iyi adaptasyonsağladıklarından, bu tip dosyalarda genel sıkştırma yöntemlerinden çok daha büyük başarı sağlarlar.

Verinin işlenmesinde yaygın olarak kullanılan bir teknik, tekrar uzunluğu kodlamasıdır ("run-length encoding"). Bu yöntemde, sıkıştırma anında arka arkaya gelen semboller sadece sembol ve tekrar sayısı ile değiştirilir. Genişletirken ise tersi işlem olan sayı kadar sembol eklenir.

Örnek olarak "000000" bilgisinin sıkıştırılacağını varsayalım. Tekrar uzunluğu kodlamasında, bilgi "60" olarak sıkıştırılmaktadır. Görüldüğü gibi orjinal bilgideki 6 sembolün yerine aynı bilgi 2 sembolle temsil edilmiş yani sıkıştırılmıştır. "60" bilgisini tekrar orjinal hali olan "000000" bilgisine genişletebilmek için genişletici sıkıştırıcının tam tersi işlemini yaparak orjinal bilgiyi elde eder.

4.2 Statik Yöntemler

Veri sıkıştırmanın klasiği sayılan Huffman kodlaması [Huffman 1952] 40 yıl önce geliştirildi. Huffman kodlaması, minimum gereksizlik kodlarının kurulmasına ilk çözümü getirmiştir. Uzun yıllar Huffman kodlamadan daha iyi sıkıştıran bir kodlama olamayacağı düşünülmüştür. Bundan önce, birbirinden ayrı olarak Shannon [Shannon 1949] ve Fano [Fano 1949] tarafından geliştirilen bir yöntem Huffman'dan daha kötü sonuçlar vermiştir. Huffman yöntemi, daha sonra geliştirilen yöntemlere kıyaslama için bir temel oluşturmuştur ve bu alanda yazılan makalelerin hemen tamamı tarafından referans olarak kullanılmıştır. Huffman ve Shannon-Fano kodlamaları, kaynak mesajların nasıl tanımlandığına bağlı olarak blok-değişken veye değişken-değişken olabilirler.

Aritmetik kodlama ise, veri sıkıştırmaya diğer statik yöntemlerden farklı şekilde yaklaşım göstermiştir. Bu yöntem, kaynak mesajla

kodkelimelere eşlendiği bir kod yaratmaz. Bunun yerine, diğer yöntemlerin yaptığı gibi kodkelimelerin yanyana gelmesiyle oluşan, bir kod dizisiyle kaynağı temsil eder. Aritmetik kodlama, kaynağın entropisine oldukça yakın sonuçlar elde eder.

4.2.1 Shannon-Fano Kodlaması

Bu tekniğin en büyük avantajı basitliğidir. Bu yöntemde kod oluşturma algoritması şöyle gerçekleşir: Kaynak mesajlar ve olasılıkları, olasılıkların artmayan sırasıyla listelenir. Bu liste, alt ve üst tarafta yaklaşık eşit toplamlı olasılıkların bulunduğu iki gruba bölünür. İlk grupta bulunan mesajlara kodkelimelerinin ilk rakamı olarak "0", ikinci gruptakilere de "1" verilir. Daha sonra bu iki grup, yine aynı kritere göre ikiye bölünerek ek kod rakamları eklenir. Bu işlem, bütün gruplarda sadece bir mesaj kalana kadar sürdürülür. Kolayca görülebileceği gibi ,Shannon-Fano kodlaması minimum gereksizlik kodunu elde eder.

Tablo 4.1 Bir Shannon-Fano kodlaması

Kaynak mesaj	Olasılık	Kodkelime
m(1)	1/2	0
m(2)	1/4	10
m(3)	1/8	110
m(4)	1/16	1110
m(5)	1/32	11110
m(6)	1/64	111110
m(7)	1/64	111111

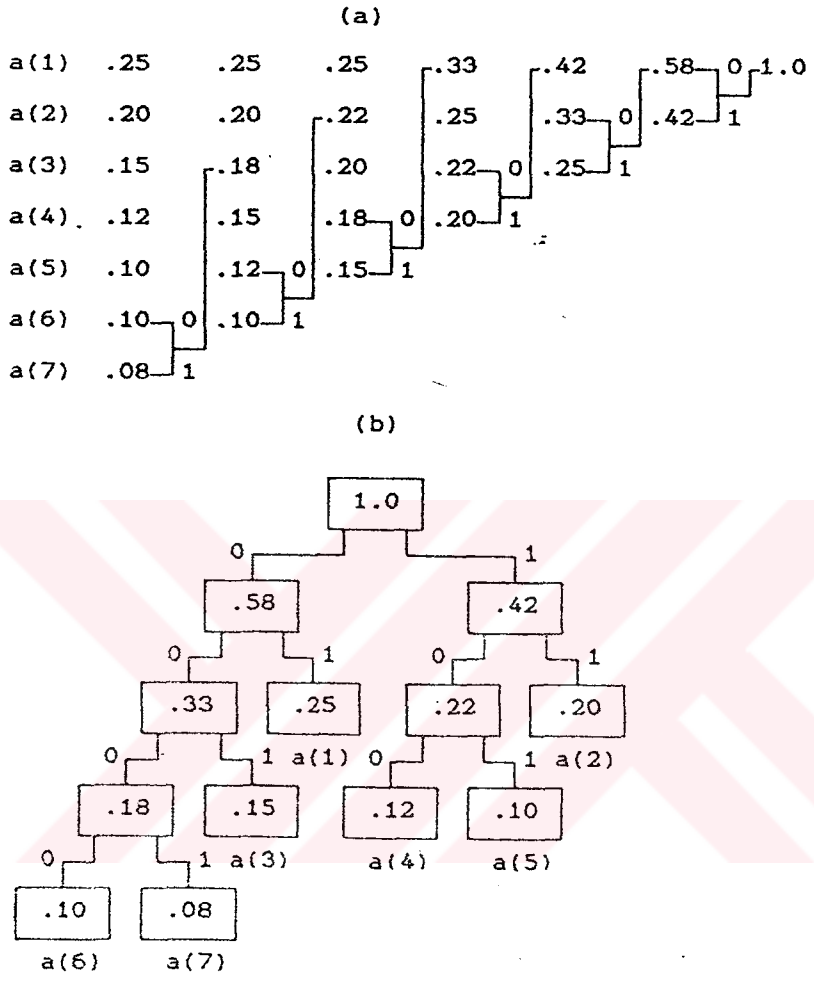
Tablo 4.1, yöntemin basit bir olasılık dağılımı için uygulamasını göstermektedir. Burada herhangi bir x mesajının kodkelimesinin uzunluğu -log (x)'dir. Bu ancak, listenin devamlı iki eşit olasılıklı gruplara bölündüğü zaman gerçekleşebilir. Bu mümkün olmadığı zaman, bazı kodkelimelerin, uzunluğu bunun bir fazlası olabilir. Shannon-Fano kodlaması, alt sınırı entropi, üst sınırı entropinin bir fazlası olan bir ortalama kodkelime uzunluğu elde eder.

4.2.2 Huffman Kodlama

Huffman yöntemi, {a(1),a(2),....,a(n)} pozitif ağırlıklar dizisi alır ve yaprakları ağırlık olan bir tam ikili ağaç kurar. Tam ikili ağaç, her düğümü ya iki çocuğu olan, ya da hiç çocuğu olmayan ağaçtır. Başlangıçta, her ağırlık için tekli ağaçlar kümesi vardır. Kod oluşturma algoritmasının her adımında, ağırlıkları en küçük ağırlıklı iki ağaç birleştirelerek, ağırlığı bu ağaçların ağırlıkları toplamı olan ve kökü bu iki ağaç tarafından temsil edilen altağaçlara sahip yeni bir ağaç oluşturulur. Birleştirilen ağaçlar listeden çıkarılarak, yeni ağaç listeye eklenir. Bu işlem, listede sadece bir ağaç kalana kadar sürdürülür. Eğer herhangi bir anda en küçük ağırlık çiftini seçmede birden fazla yol varsa, bu çiftlerden herhangi biri seçilebilir. Listenin sıralı olması gerekmez, ama sıralı olması kolaylık yaratır.

Bu algoritma, her mesaja karşılık gelen kodkelime uzunluklarını belirler. Kodkelimelerin seçiminde bir çok yol vardır. Bunların hepsi önek özelliğinde kod oluşturur. Normal belirlemede, soldaki ağacın bağına "0" rakamı, sağdakine "1" rakamı verilir. Her mesaj için kodkelimesi, kökten o harfi temsil eden yaprağa giden yoldaki rakamlardan oluşur. Şekil 4.1'deki

olasılık listesi için Huffman kodkelimeleri {01,11,001,100,101,0000,0001} kümesi olur. Bu işlem minimum gereksizlik kodu elde eder.



Şekil 4.1 Liste (a) ve ağaç (b) için Huffman işlemi

Kodkelimelerini kurmada bir çok yol olduğundan dolayı, Huffman kodlama her zaman aynı uzunlukta kodkelimeleri elde etmeyebilir. Örnek olarak {0.4,0.2,0.2,0.1,0.1} mesaj olasılık kümesiyle, kodkelime uzunlukları {1,2,3,4,4} ve {2,2,2,3,3} olan kodlar çıkabilir. Fakat bu kodların her ikisi de, aynı ortalama kodkelime uzunluğunu sağlar.

Schwartz [Schwartz 1964a], yeni bir ağırlığı her zaman listede aynı ağırlıkların üzerine yerleştiren ve listenin en altındaki iki ağırlığı birleştiren bir Huffman varyasyonu önermiştir. Böylelikle bulunan kodlar, maksimum uzunluktaki kodkelimelerin minimumuna ve minimum kodkelime uzunluğu toplamına sabittir. Schwartz ile Kallick [Schwartz 1964b] ve Connel [Connel 1973], Huffman algoritmasının birbirini takip eden ikili kodlar üreten varyasyonunu önermişlerdir.

Daha önce belirtildiği gibi, Shannon-Fano yönteminin gereksizlik üst sınırı 1'dir. Huffman yönteminde bu, minimum kaynak mesaj olasılığı ile 0.086 toplamına eşittir [Gallager 1978]. Minimum kaynak mesaj olasılığının maksimum değeri 0.5'dir ve genellikle çok daha küçük, hatta sıfıra yakın olabilir. Gereksizliğin tanımında, bu yöntemler için kod eşlemenin iletilmesi gözardı edilmiştir. Fakat gerçek uygulamalarda kod eşlemenin de iletilmesi gerekmektedir ve statik yöntemlerde hesaplanan entropi miktarının da, kod eşleme faktörünün de eklenmesi gerektiği akıldan çıkarılmamalıdır.

4.2.3 Melez Yöntemler

Eğer optimal kodlamadan biraz daha kötü sonuçlar veren bir yöntem kullanılmak istenirse, kod eşlemenin iletilmesi maliyeti, gönderici ve alıcı arasında üzerinde önceden anlaşılan bir kod eşlemesi kullanılarak yok edilebilir. Gönderici ve alıcı herbiri değişik gruptaki mesajların istatistiklerine uygun kodlamalar içeren bir kod kitabı kullanabilirler. Bu durumda gönderici, basitçe bu ortak kodlardan hangisini kullandığını çok az bit ile gönderilecek kod eşlemesini göndermekten kurtulabilir.

Ayrıca, melez teknik kullanılarak eşleme hesaplaması zamanı maliyeti de en aza iner. Yani, istatistiksel açıdan örnek bir dosya için eşleme bir kere yapılır ve bu, bütün o gruptaki dosyalar için kullanılabilir. Yalnız bu yöntemde örnek dosya seçiminin temsil ettiği grubu tam olarak karakterize edememesi ve herhangi bir dosyanın o grup dosyalarının olasılık dağılımına uymaması gibi çeşitli riskleri vardır.

4.2.4 Aritmetik Kodlama

Bu yöntem, Abramson [Abramson 1963] tarafından kitabında sunulmuştur. Daha sonra bir çok kişi tarafından geliştirilmiştir [Rissanen 1976, Pasco 1976,Rubin 1979, Witten 1987]. Aritmetik kodlamada kaynak mesaj, sayı doğrultusundaki [0,1) aralığıyla belirlenir. Mesajdaki her sembol, bu aralığı daraltır. Aralık daraldıkça, onu belirten bit sayısı artar. Bu kodlama, kaynağı temsil etmek için kaynak mesajların olasılıklarını kullanarak aralığı daraltır. Yüksek olasılıklı bir mesaj, aralığı düşük olasılıklı bir mesajdan daha az daraltır ve böylece kodlanmış mesaja daha az sayıda bit ekler. Yöntem, kaynak mesajların sıralı olmayan bir listesi ile başlar. Sayı doğrusu, kümülatif olasılıklara göre alt aralıklara bölünür. Aritmetik kodlama bir örnek ile şöyle açıklanabilir. {1,2,3,4,5} kaynak mesajları ve sırasıyla {0.2,0.4,0.1,0.2,0.1} olasılıklarıyla Tablo 4.2, sayı doğrusunun başlangıçtaki bölünüşü göstermektedir.

"1" sembolü [0,1) aralığının 1/5<'i olan [0,0.2), "2" sonraki 2/5'i olan [0.2,0.6), "3" sonraki 1/10'u olan [0.6,0.7), "4" sonraki 1/5'i olan [0.7,0.9) ve "5" kalan 1/10'u olan [0.9,1) aralıkları ile temsil edilir. Kodlama başladığı zaman, bütün veri [0,1) aralığı ile temsil edilir. "11425" verisi

için ilk "1" aralığı [0,0.2)'ye ve ikinci "1" [0,0.04)'e düşürür. Sonraki "4", aralığı [0.028,0.036)'ya , "2", [0.0296,0.0328)'e ve "5", son aralık olan [0.03248,0.0328)'e indirir. Bu son aralık ya da aralığın içindeki herhangi bir sayı "11425" verisini temsil eder.

Tablo 4.2 Aritmetik kodlama

Kaynak mesaj	Olasılık	Aralık
1	0.2	[0,0.2)
2	0.4	[0.2,0.6)
3	0.1	[0.6,0.7)
4	0.2	[0.7,0.9)
5	0.1	[0.9,1)

Bu aralık azaltma işlemi, aralığın sol sınırı ve aralığın yeni boyutu değerlerinin belirlenmesidir. Bunlardan aralığının sol sınırı, önceki aralığın sol sınırına, o anki mesajın sol sınırıyla önceki aralığın boyutu çarpımının eklenmesiyle bulunur. Örnekte başlangıç aralık [0,1) dir. İlk "1" aralığının sol sınırını 0 + 0×1, yani 0 yapar. Aralığın boyutunu ise, önceki aralığın boyutu ile o anki mesajın aralığının boyutunun çarpımı belirler. İlk "1" için bu 1×0.2, yani 0.2'dir. Yani, ilk "1" aralığı [0,0.2) yapar.

Orjinal mesajı elde edebilmek için, kodçözücü kodlayıcının kullandığı kaynak modelini ve kodlayıcının daha önce anlatıldığı gibi bulduğu son aralıkda bulunan tek bir sayı alır. Kod çözmek, iletilen sayının kaynak mesaj aralıklarından hangisinde olduğunun anlaşılması için bir dizi karşılaştırma yapmaktır. Verilen örnekde bu sayı 0.0325 olabilir. Kodçözücü, bu sayıyı kullanarak kodlayıcının hareketlerine benzer hareketler yapar. 0.0325 sayısı, 1 mesajının aralığı olan [0,0.2) aralığında

olduğundan ilk mesaj "1" dir. Bu aralığı [0,0.2)'ye düşürür. Kodçözücü, bundan sonraki mesaj "1" ise aralığın [0,0.04), "2" ise [0.04,0.12), "3" ise [0.12,0.14), "3" ise [0.12,0.14), "4" ise [0.14,0.18) ve "5" ise [0.10.2) olacağını anlar. Sayı 0.0325 olduğundan, ikinci mesajın "1" olduğu anlaşılır. Bu işlem bütün veri elde edilinceye kadar sürdürülür. En son alt aralığın boyutu, o aralıkdaki bir sayıyı belirtmek için gerekli bit sayısını belirtir. [0,1) aralığının s boyutundaki bir alt aralığı, -log(s) sayıda bit ile temsil edilebilir. Son aralığın boyutu kaynak mesajların olasılıklarının çarpımına eşittir. Bu ifade tam olarak entropiye eşit olarak bulunur [Lelewer 1987]. Yani aritmetik kodlama, diğer yöntemlerden daha iyi sonuçlar elde eder.

Aritmetik kodlamanın uygulamasında bazı sorunlarla karşılaşılmaktadır. Bunlardan ilki, kodçözücünün nerde durması gerektiğini bilmemesidir. Gerçekten de 0 sayısı yukarda ki örnekte aynı zamanda "1", "11", "111", yani bir çok sayıda "1""in yanyana geldiği mesajı belirtir. Bu sorunun çözümü, kodlanmış mesajla beraber orjinal mesaj uzunluğunun da gönderilmesidir. Başka bir çözüm de mesaj sembolünün mesaj sonunu belirtmesidir. İkinci sorun ise, yukardaki anlatımdan en son aralık belli olana kadar, kodlayıcının hiç mesaj gönderemediği gibi görünmektedir. Oysa durum böyle değildir. Aralık daraldıkça, aralık sınırlarını belirleyen sayıların solundaki rakamlar sabit kalmaktadır. Üçüncü sorun, üretilen sayının çok basamaklı olması ve bu sayıları işlemedeki zorluktur.

4.3 Dinamik Yöntemler

Bu yöntemler veri üzerinden sadece bir kez geçerek, kaynak mesajın değişken özelliklerine, devamlı olarak eşleme tablosunu değiştirerek

adaptasyon sağlayan yöntemdir. Bu yöntemlerin kaynak noktası yine Huffman kodlama olmuştur.

Adaptif Huffman kodlama, Huffman yöntemine benzer olarak, Huffman ağacı kullanır. Aradaki fark ağacın devamlı olarak değişmesidir. Ağacın değiştirilme algoritması, yöntemin çeşitli varyasyonları olmasına yol açmıştır.

Adaptif Huffman kodlamada, o ana kadar geçen mesaj olasılıklarına göre kaynak mesajlar kodkelimelerine eşlenir. Kodlama, kaynağın yerel özelliklerine cevap vererek, yerel gereksizliği bir anlamda yok etmeye çalışır. Bu bir anlamda kaynak karakteristiğinin öğrenilmesi demektir. Kodçözücü, kodlayıcı ile eş hareketle Huffman ağacını aynı anda değiştirerek öğrenir. Bu kodlamanın üstünlüğü, veri üzerinden sadece bir kez geçmesidir. Ayrıca adaptif olduğundan, eşleme tablosununda iletilmesi gerekmemektedir.

Adaptif Huffman kodlaması, ilk olarak birbirinden ayrı olarak Faller [Faller 1973] ve Gallager [Gallager 1978] tarafından önerilmiş ve algoritma Knuth [Knuth 1985] tarafından geliştirilmiştir. Bu algoritmanın temelini Gallager'in tanımladığı kardeş özelliği oluşturur. Bir ikili kod ağacı, eğer kökdüğümü hariç bütün düğümlerinin kardeşi varsa ve düğümler yanındaki kardeş düğümüyle beraber, artmayan ağırlıklarla sıralanabiliyorsa kardeş özelliğine sahiptir. Gallager, bir ikili önek kodunun sadece ve sadece kod ağacı kardeş özelliğindeyse, Huffman kodu olduğunu kanıtlamıştır. Bu yöntemde, kodlayıcı ve kodçözücü sürekli değişen Huffman kod ağacı tutarlar. Kod ağacının yaprakları kaynak mesajları ve yaprakların

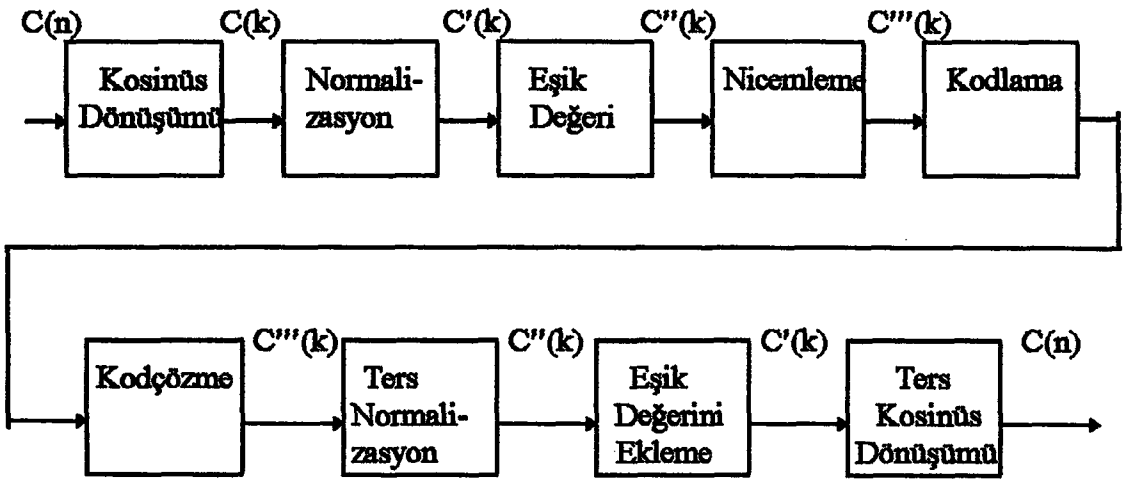
ağırlıklarıda mesajların geçme frekanslarını temsil ederler. Herhangi bir anda n olası mesajdan k tanesi geçer.

Başlangıçta, kod ağacı 0-düğümü adı verilen tek yaprak düğümden oluşur. Bu 0-düğümü, o ana kadar kullanılmayan mesajları temsil eden özel bir düğümdür. Bir mesaj iletildiği anda her iki taraf ağırlıkları arttırmalı ve yeni ağırlıklara göre kardeş özelliğinin korunması için kod ağacı tekrar hesaplanmalıdır. t sayıda mesaj iletildiği anda, bunlardan k tanesi birbirinden farklıysa, Huffman kod ağacında k+1 yaprak vardır. Bunlardan k tanesi birbirinden farklı mesajlar için, 1 tanesi ise 0-düğümü yani o ana kadar hiç kullanılmamış mesajlar içindir. Eğer bundan sonra gönderilen mesaj, daha önce kullanılmamış olan k mesajdan biriyse, bu mesajın kodkelimesi gönderilir ve frekansı arttırılarak kod ağacı yeniden hesaplanır. Eğer daha önce kullanılmayan bir mesaj kullanılırsa, 0-düğümü ikiye ayrılarak biri yeni mesaj, diğeri 0-düğümü olan bir çift yaprak yaratılır ve ağaç tekrar hesaplanır. Bu durumda, 0-düğümünün kodkelimesi ve yeni mesaj gönderilir

BÖLÜM 5

KODLAMA ALGORİTMASI VE SONUÇLAR

Şekil 5.1 kodlama algoritmasında kullanılan blok şemayı göstermektedir. Bu çalışmada transform kodlarında etkinliği bilinen AKD (Ayrık Kosinüs Dönüşümü) metodu kullanılmıştır. Temelde, EKG verisi ilk önce 64 örnek boyutlu AKD katsayılarını hesaplayacak bloğa girer. AKD bloğu'nun çıkışında elde edilen 64 katsayı, normalizasyon faktörü ile çarpılır ve bir eşik değerinnden ("threshold") değerinden geçirilir. Eşik değeri bloğu'nun amacı EKG verisindeki gereksiz olan bilgiyi ayıklamaktır. Bu adımdan sonra AKD katsayıları sıfıra dönüşür ve sadece birkaç tane alçak frekans bileşeni ile DC bileşen sıfırdan farklı olarak kalır. Eşik değeri bloğundan sonraki adımda ise veri nicemlenir. Kodlama bloğunda, sıfır değerlere tekrar uzunluğu kodlaması, diğer değerlere de Huffman kodlaması uygulanır. Tekrar uzunluğu kodlaması birçok sıfırın oluştuğu dönüşümlerde çok iyi performans göstermektedir. Kodlama işleminin bitişiyle veri sıkıştırılmış olur ve sıkıştırılan veride saklanır. Alıcı kısmında ise kodçözücüde kodlar çözülür, değerler normalizasyon katsayısına bölünür ve çıkan değerlere eşik değeri eklenir. En sonda da ters ayrık kosinüs dönüşümü alınarak sıkıştırılan işaret genişletilmiş olur. Alt bölümler de kodlama algoritması ayrıntılarıyla açıklanmıştır.



Şekil 5.1 Kodlayıcı/Kodçözücü sistemin blok şeması

5.1 Algoritmada Temel Adımlar

Bir x(n) dizisi, n=0,1,...,N-1, için tek boyutlu ayrık kosinüs dönüşümü şu şekilde tanımlanmıştı.

$$C_x(0) = \frac{\sqrt{2}}{N} \sum_{n=0}^{N-1} x(n)$$

$$C_x(k) = \frac{2}{N} \sum_{n=0}^{N-1} x(n) \cos \frac{(2n+1)k\Pi}{2N} \qquad k=1,2,.....,(N-1) \qquad (3.8)$$

Ters ayrık kosinüs dönüşümü:

$$x(n) = \frac{1}{\sqrt{2}} C_x(0) + \sum_{k=1}^{N-1} C_x(k) \cos \frac{(2n+1)k\Pi}{2N} \qquad n=0,1,2,.....,(N-1) \qquad (3.13)$$

64 örnek boyutlu ayrık kosinüs dönüşümü uygulamasından sonra dönüşüm gösretimi katsayılarını elde ederiz (Şekil 5.1.'de *C(k)*). Eğer kayıtta keskin

değişimler vuku buluyorsa, dönüşüm katsayıları α faktörü ile belirlenen bir değere ölçeklendirilebilinir.

Sonraki işlemde, dönüşüm katsayıları doğru akım (DC) bileşeni *C(0)* hariç tutularak eşik değeri işlemine tabi tutulur. Bu işlemdeki amaç eşik değerinden küçük katsayıları sıfırlamaktır.

$$C''(k) = \begin{cases} C'(k) - \beta & C'(k) > \beta \\ 0 & diğer \end{cases}$$

β' dan büyük *C(k)* değerlerinden β'nın çıkarılması genlik değerlerinin dinamik aralığını sıkıştırmaktadır. Böylece, genlik değerlerinin histogramındaki süreksizliklerde ortadan kalkar. Ölçekleme ve eşik belirleme işlemleri yeniden tanımlama sürecinde sıkıştırma oranı ve hatanın kontrolüne yardımcı olur.

Nicemleme ise temel olarak her bir katsayının kendine en yakın tamsayıya yuvarlatılmasıdır. Karar ve yeniden tanımlama tablolarının kullanılmasına gerek yoktur.

$$C'''(k) = \begin{cases} (tamsayı)(C''(k) + 0.5) & C''(k) \geq 0 \ \ için \\ (tamsayı)(C''(k) - 0.5) & diğer \end{cases}$$

Tamsayı yuvarlatmasından sonra elde edilen dizi kodlanır. Kodlama yapılırken, bütün bloklarda daha önceden belirlenmiş sabit bir kodtablosu kullanılır. Sıkıştırma süresi boyunca kodtablosu değişmez. Bu bağlamda, üç tane kodtablo tanımlanmıştır. Bunlar Huffman kodtablosu, tekrar uzunluğu kodlama için bir kodtablo ve sonuncu olarak da ilk tabloda belirlenen değer aralığından büyük değerler için ayrı bir kodtablosudur.

Huffman kodlamada Tablo 5.1 deki kodlar kullanılmaktadır. Algoritmada 0 ile ±9 değerleri kodlanmaktadır. Tablodaki AMPRE sembolik olarak |9|'dan büyük olan değerleri ifade eder ve ilk adımda Huffman tablosundaki önek koduyla kodlanır. İkinci adımdaki kodları ise kendi kodtablosu belirlemektedir. Aynı durum sıfır değerleri içinde sözkonusudur. Tablo 5.1 sembolik olarak kodeşlemesini göstermek ve hangi değerlerle çalışıldığını belirlemek amacıyla sunulmuştur yani tablodaki olasılıkların ve değerlerin sıralımının gerçek veriyle bir ilgisi yoktur..

Tablo 5.1 Huffman kodtablosu

Genlik değeri	Olasılık	Kod	Bit sayısı
0	0.137000	10	2
AMPRE	0.130600	11	2
-1	0.058400	0010	4
1	0.058400	0011	4
-2	0.046800	0100	4
2	0.046800	0101	4
-3	0.039000	0110	4
3	0.039000	00000	5
-4	0.030200	00010	5
4	0.030200	00011	5
-5	0.023400	01110	5
5	0.023400	01111	5
-6	0.015600	000010	6
6	0.007800	0000110	7
-7	0.003800	00001110	8
7	0.002000	000011110	9
-8	0.001000	0000111110	10
8	0.000100	00001111110	11
-9	0.000010	000011111110	12
9	0.000002	0000111111110	13
DC değer	0.000001	0000111111111	13

Tablo 5.2 de tekrar uzunluğu kodlaması yapılırken Tablo 5.3 ise |9|'dan büyük olan değerler kodlanırken kullanılır. Bu kodlar Tablo 5.1'de belirlenen önek kodlarına eklenirler. Tablo 5.2 ve Tablo 5.3,sembolik birer gösterimdir ve algoritmada Tablo 5.3 deki gibi bir değer sınırlaması yoktur.

Tablo 5.2 Tekrar Uzunluğu Kodtablosu

Sıralı sıfır sayısı	Olasılık	Kod	Bit sayısı
1	0.572873	0	1
2	0.139275	10	2
3	0.033860	110	3
4	0.008232	1110	4
5	0.002001	11110	5
6	0.000487	111110	6
7	0.000118	1111110	7
8	0.000029	11111110	8
9	0.000006	111111110	9
10	0.000002	1111111110	10
11	0.000001	1111111111	10

Tablo 5.3 Huffman Kodtablosu2

Genlik değeri	Olasılık	Kod	Bit sayısı
-10	0.031250	0	1
10	0.031250	10	2
-11	0.015625	110	3
11	0.015625	1110	4
-12	0.001500	11110	5
12	0.001250	111110	6
-13	0.000200	1111110	7
13	0.000180	11111110	8
-14	0.000150	111111110	9
14	0.000020	1111111110	10
-15	0.000017	11111111110	11
15	0.000010	111111111110	12
16	0.000001	111111111111	12

Kodlama yapılırken, herbir bloğun içindeki değerlerin olasılıkları hesaplanır ve olasılıklar büyükten küçüğe doğru sıralanır. Sıralamadan sonra her değer kendisine karşılık gelen bir kodla eşlenir. Algoritmada 16 blok olduğuna göre bu işlem her blokta tekrar tekrar yapılacaktır. Örnek olarak α=2 ve eşikdeğeri=0.2 için birinci blokta elde edilen kodtablolarını inceleyelim. Tablo 5.4 bu değerleri göstermektedir. Hatırlanacağı üzere, Tablo 5.1 de 21 değişik değer için kodeşlemesi vardı. Fakat bu her blokta 21 değişkenin aynı anda varolacağını göstermez. Nitekim, Tablo 5.4'de 12 değer çıkmıştır. Bu değerlerden "-3" değerini örnek olarak inceleyelim. Tablo 5.4 de -3'e karşılık gelen kod "00011" dir ve 5 bit ile temsil edilir. Tablo 5.5 ise α=2 eşikdeğeri=1 için bulunan tablodur. Dikkat edilirse, bu durumda -3'ün olasılığı daha yüksektir ve bu sebeble "0010" koduna karşı düşer ve 4 bit ile temsil edilir. Birinci blok dışında diğer bloklar için bulunan kodtablolarınıda inceleseydik bu duruma yine şahit olurduk. Kodtablolarında -3 verisinin olasılık değerine göre koduda değişmektedir. Bir ihtimalde -3 değerinin blokların bazılarında hiç olmamasıdır.

Tablo 5.4 Örnek veri tablosu1

Genlik	Olasılık	Kod	Bit sayısı
0	0.625000	10	2
1	0.062500	11	2
-1	0.062500	0010	4
2	0.062500	0011	4
-2	0.046875	0100	4
-4	0.031250	0101	4
AMPRE	0.031250	0110	4
DC değer	0.015625	00000	5
3	0.015625	00010	5
-3	0.015625	00011	5
4	0.015625	01110	5
5	0.015625	01111	5

Tablo 5.5 Örnek veri tablosu2

Genlik	Olasılık	Kod	Bit sayısı
0	0.828125	10	2
1	0.031250	11	2
-3	0.031250	0010	4
AMPRE	0.031250	0011	4
DC değer	0.031250	0100	4
2	0.015625	0101	4
-2	0.015625	0110	4
3	0.015625	00000	5
4	0.015625	00010	5

Kodların devamlı değişken olması dikkate alındığında kullanılan yöntem dinamiktir fakat her blokta aynı kodtablosu kullanıldığından yöntem aynı zamanda statiktir. Bu sebeble bu yöntemi melez yöntem diye adlandırıyoruz.

Sıralı sıfırlar için tekrar uzunluğu kodlama tekniği uygulanmaktadır. Tablo 5.5 α=2 ve eşikdeğeri=1 için birinci blokta elde edilen tabloyu göstermektedir. Algoritma sıralı sıfırların bir blokta kaç tane olduğunu hesaplar, olasılıkları bulur, sıralar ve Tablo 5.5 deki kodeşlemesini yapar. Örnek Tablo 5.5 de birinci blok için bir 0 dört kere, arka arkaya iki 0 bir kere, arka arkaya üç 0 bir kere, arka arkaya beş 0 bir kere ve arka arkaya onbir 0 da 1 kere vuku bulmaktadır. Arka arkaya sıfırların sayısı 60'a kadar çıkabilmektedir. Algoritma arka arkaya olan sıfırların sayısı 11'i geçtiğinde bunları toplayıp kodtablosunda ki sıralı sıfır değeri için yine 11 değerini atamaktadır. Algorittmada, tekrar uzunluğu kodlamasına geçilmeden önce Tablo 5.4 kodtablosundaki sıfıra karşılık gelen kodu önek olarak alır sonra tekrar uzunluğu kodlamasıyla eşlenilen kodlara bu öneki ekler. Bu şekilde sıfırlar daha az sayıda bit ile kodlanmış olur.

Tablo 5.6 Örnek veri tablosu3

Sıralı sıfırlar	Olasılık	Kod	Bit sayısı
1	0.062500	0	1
2	0.015625	10	2
3	0.015625	110	3
5	0.015625	1110	4
11	0.015625	11110	5

Örnek vermek gerekirse, birinci bloktaki onbir sıralı sıfırı kodlayalım. Tablo 5.4 deki kodtablosunda "0" iki bite karşılık düşer. Tekrar uzunluğu kodlaması yapmasaydık bu onbir sıfırı yirmi iki bitle temsil edecektik. Fakat tekrar uzunluğu kodlaması yaparsak ilk önce Tablo 5.6 de onbir sıfıra karşı hangi kodun geleceğini tespit ederiz. Bu kod "11110" dır. Sonrada Tablo 5.4 de sıfır için tespit edilen "10" kodunu bulduğumuz "11110" koduna önek olarak getiririz. Sonuçta onbir sıfır "1011110" koduyla kodlanır ve bu da yedi bite karşılık düşer.

|9|'dan büyük değerleri kodlamada α=2 ve eşikdeğeri=1 için birinci blokta Tablo 5.7 deki değerler elde edilmiştir. Bu tablodaki kodlar Tablo 5.4 de AMPRE sembolü ile belirlenen önek kodunun sonuna eklenirler. Dikkat edlecek bir başka özellik de Tablo 5.7 deki genlikler için bir aralık sınırlaması olmamasıdır. Bloğun içindeki veri ne ise tabloda bu veri aynı değeriyle temsil edilir.

Tablo 5.7 Örnek tablo3

Genlik	Olasılık	Kod	Bit sayısı
35	0.015625	0	1
-12	0.015625	10	2

5.2 Sonuçlar

EKG işareti MIT-BIH aritmi veritabanı CD-ROM'undan sağlandı. CD-ROM içindeki EKG dosyaları " .dat" isimli işaret dosyaları ile belirtilmiştir. Bu çalışmada 109.dat isimli dosya kullanıldı. Yalnız bu dosya içindeki veriye doğrudan ulaşmak mümkün olmadığından işaret dosyasının bir listesini döken yardımcı programlar geliştirilmiştir. "rdsamp" bu amaçla kullanılan bir uygulama programıdır.

"rdsamp" = uygulama programı

kullanışı: rdsamp -r KAYIT [Opsiyonlar]
KAYIT EKG kayıdının ismidir. Opsiyonlar kısmıda şunları içermektedir.

-f ZAMAN belirtilen zamanda başlatır.
-t ZAMAN belirtilen bir zamana sonlandırır.
-p PRINT işaretin zaman ve örnek değerlerini fiziksel bir ünitenin içine atar.

Bu program EKG dosyasının bir listesini döker. "-f" ve "-t" opsiyonları ile bir zaman dilimi seçilir ve "-p" opsiyonu zaman ve örnek değerlerini belirttiğimiz dosyanın içine atar. Örnek olarak,

rdsamp -r 109 -f 0 -t 30 > EKG

0 ila 30 sn arasındaki 109.dat dosyasındaki veriler ismini bizim verdiğimiz EKG dosyasının içine yazılır. Sonuçta EKG dosyasında şu değerleri görürüz (Tablo 5.8).

Tablo 5.8 EKG dosyası

time (sec)	MLI (mV)	V1 (mV)
0.000	0.175	0.635
0.003	0.175	0.635
.	.	.
.	.	.
.	.	.
0.0025	0.180	0.635
0.0028	0.185	0.645
.	.	.
.	.	.

Tablo 5.8 de işaretin zaman aralıkları ile birlikte derivasyonlarıda gösterilmiştir. MLII birinci derivasyonu V1 ikinci derivasyonu temsil eder. Değerler milivolt mertebesindedir. Bu çalışmada birinci derivasyon değerleri kullanılmıştır.

İşaretin örnekleme frekansı 360 Hz, çözünürlüğü ise 13 bittir. Bu çalışmada 1024 örnekli bir EKG dosyası oluşturulmuş ve algoritmada bu dosya kullanılmıştır. AKD dönüşümünde blok boyutu 64 örnek olarak alınmıştır. Bu bağlamda, 16 ayrı blok elde edilmiştir.

Tablo 5.9 da eşikdeğeri=0.5 ve değişik alfa değerleri için elde edilen sıkıştırma oranları (CR) ve performans indeksleri (PRD) görülmektedir. Görüldüğü gibi, alfayla sıkıştırma oranı CR arasında ters bir orantı mevcuttur yani, alfa değeri arttıkça CR düşmekte buna mukabil işaretin performans indeksi iyileşmektedir.

Tablo 5.10 da α=1.5 ve değişik eşikdeğerleri için elde edilen sıkıştırma oranları (CR) ve performans indeksleri (PRD) görülmektedir. Bu tabloda görülen ise eşikdeğeri ile CR arasında doğru bir oranın mevcut olduğudur. Eşikdeğeri arttıkça sıkıştırma oranı CR de artmaktadır. α=1.5 ve eşikdeğeri=4 için yaklaşık 46 değerinde sıkıştırma oranı elde edilmiştir. Şekil 5.2 de bu değerler için çizilmiş EKG şekilleri gösterilmiştir.

Tablo 5.9 Sıkıştırma tablosu1

alfa (α)	CR	PRD (%)
1	17.678	2.675
1.2	15.075	2.279
1.4	13.272	1.980
1.6	12.179	1.837
1.8	11.310	1.741
2	10.947	1.669

Tablo 5.10 Sıkıştırma tablosu2

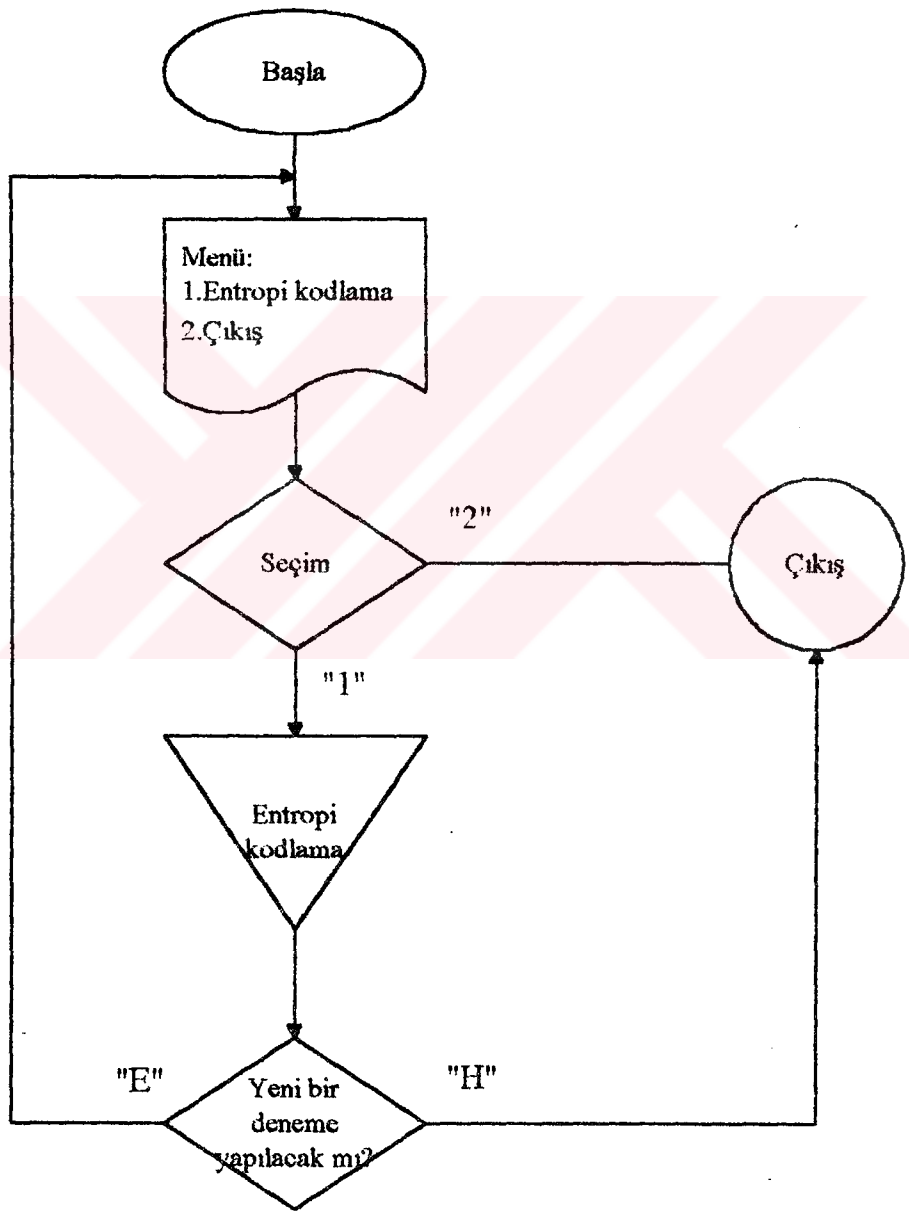
Eşikdeğeri	CR	PRD (%)
0.2	10.965	1.710
0.5	12.653	1.872
1	19.126	2.816
2	32.788	4.512
3	40.834	5.565
4	46.062	6.311

Şekil 5.2 Sırasıyla orjinal işaret, yeniden yapılanmış işaret ve ikisi arasındaki fark işareti.

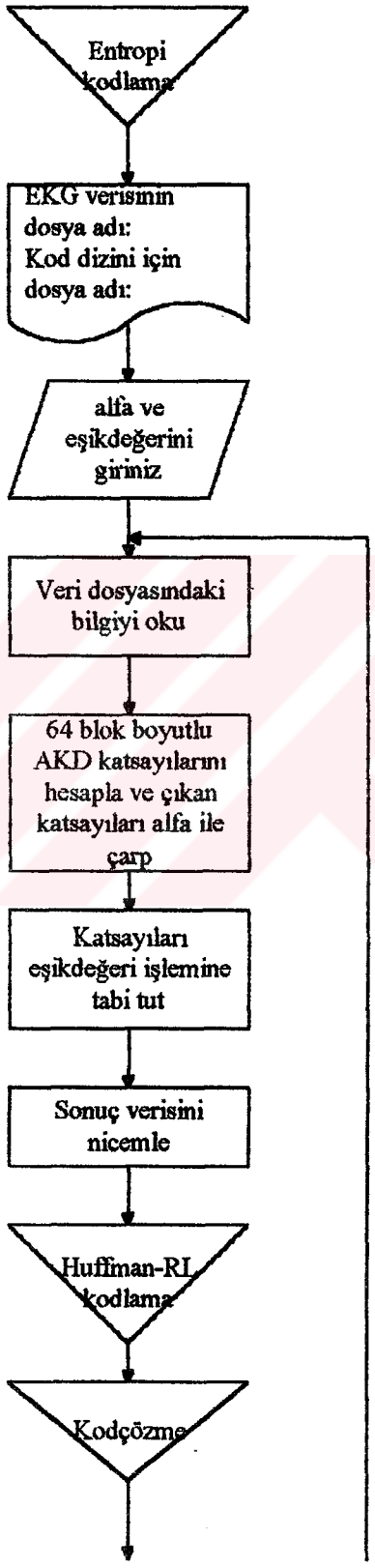
5.3 Programın Akış Diyagramları

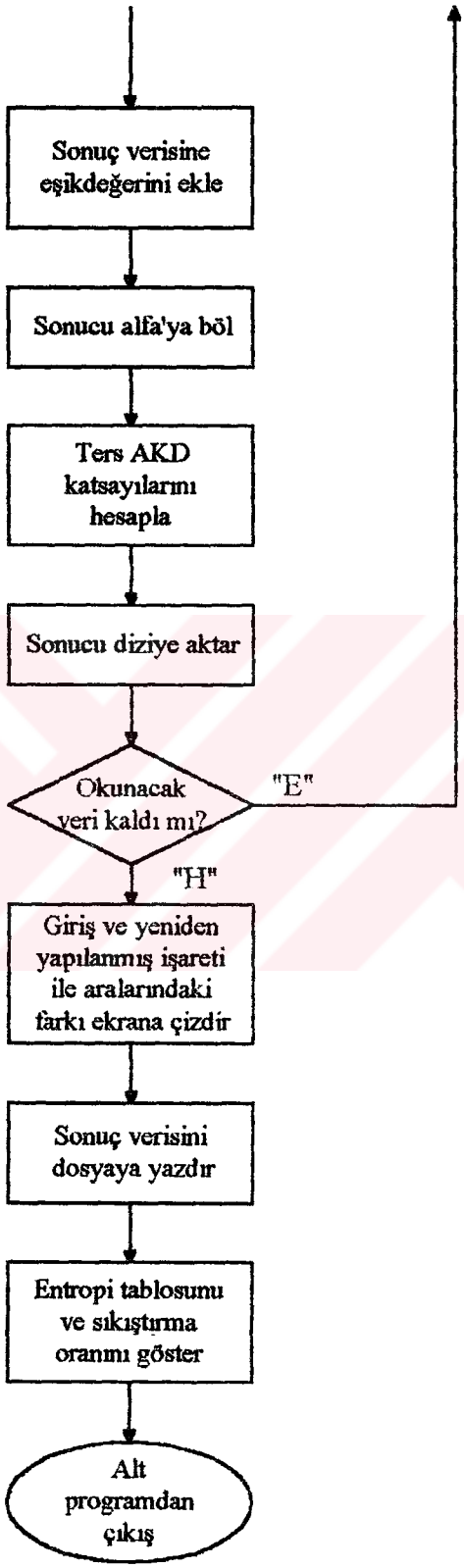
Programlar Turbo C programlama dilinde yazılmıştır.

1. Ana Fonksiyonun Akış Diyagramı

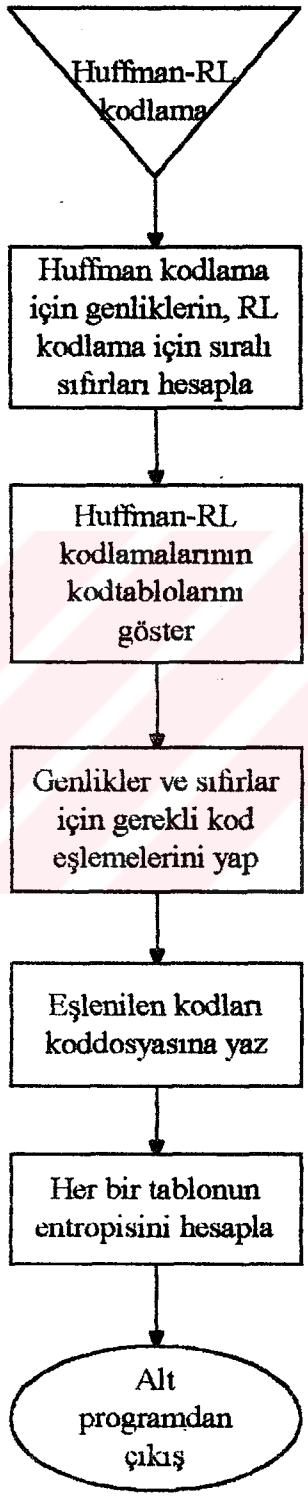


2. Entropi Kodlama alt programının akış diyagramı

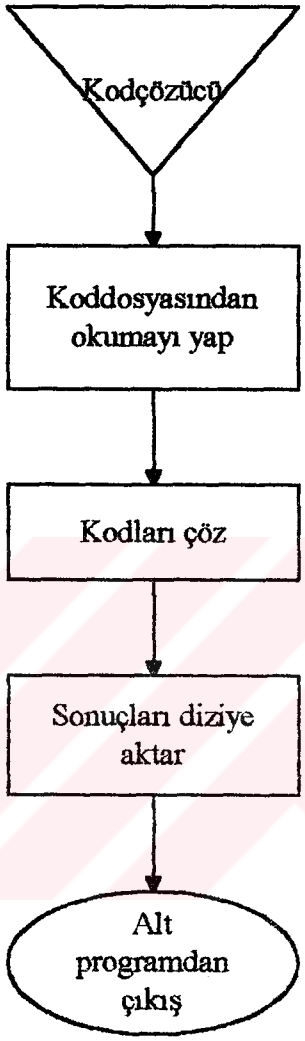



Sonuç verisine eşikdeğerini ekle
Sonucu alfa'ya böl
Ters AKD katsayılarını hesapla
Sonucu diziye aktar
Okunacak veri kaldı mı?
"E"
"H"
Giriş ve yeniden yapılanmış işareti ile aralarındaki farkı ekrana çizdir
Sonuç verisini dosyaya yazdır
Entropi tablosunu ve sıkıştırma oranını göster
Alt programdan çıkış

3. Huffman-RL (Tekrar Uzunluğu) kod altprogramının akış diyagramı



4. Kodçözücü alt programının akış diyagramı



SONUÇLAR VE ÖNERİLER

EKG verisi sıkıştırılırken kırkaltı civarlarında bir sıkıştırma oranı elde edilmiştir. Programı geliştirmek için bazı tavsiyelerde bulunulabilinir. Bunlardan ilki, hızlı kosinüs dönüşümü algoritmasının kullanılmasıdır. Bu sayede ayrık kosinüs dönüşümümde hesaplanacak olan toplam ve çarpımların adedinde önemli miktarda azalmalar olur ve bu da programın hızını arttırıcı önemli bir etkendir. Dezavantajı ise programın karmaşıklık derecesini arttırır. Bir diğer öneri ise algoritmada daha önceden tanımlanmış sabit kodkitapçıklarının yerine her bir blokta kodkitapçıklarının dinamik olarak tanımlanmasıdır. Böylece kodların belirlenmesinde optimuma daha yaklaşılır fakat bu işlemin programa hız kaybettireceğide gözardı edilmemelidir.

KAYNAKLAR

ABRAMSON, N., (1963), Information Theory and Coding, McGraw-Hill, NewYork

AHMED, N., NATARAJAN, T., RAO, K.R, (1974), Discrete Cosine Transform, IEEE Trans. Comput., Vol C-25, pp. 90-93

BENTLEY, J.L., SLEATOR, D.D, TARJAN, R.E, WEI, V.K, A Locally Adaptive Data Compression Scheme. Comm. ACM, Vol.29, No.4, pp. 320-330

CONNEL, J.B., (1973), A Huffman-Shannon-Fano Code, Proc. IEEE, Vol.61, No.7, pp. 397-403

COX, J.R. et. al., (1968), AZTEC: A Preprocessing Program Real-Time ECG Rhythm Analysis, IEEE Trans. Biomed. Eng., Vol. BME-15, pp.128-129

DAVIDSON, L.D., (1966), Theory of Adaptive Data Compression Edited by A.V.Balakrishinan, Academic of Press, New York, pp.173-192

FALLER, N., (1973), An Adaptive System for Data Compression, In record of the 7th Asimolar Conference on Circuits, Systems and Computers, Naval Postgraduate School, Monterey, pp. 593-597

FANO, R.M., (1949), Transmission of Information, M.I.T. Press, Cambridge

FURTH, B, PEREZ, A., (1988), An Adaptive Real-Time ECG Compression Algorithm With Variable Threshold, IEEE Trans. Biomed. Eng., Vol. BME-35

GALLAGER, R.C., (1968), Information Theory and Reliable Communication, Wiley, New York

GALLAGER, R.G., (1978), Variations on a theme by Huffman, IEEE, Trans. Inf. Theory, Vol.24, No.6, pp. 668-674

GILBERT, E.N, (1971), Codes Based on Inaccurate Source Probabilities, IEEE, Trans. Inf. Theory, Vol.17, No.3, pp. 304-314

HUFFMAN, D.A., (1952), A Method for the Reconstruction of Minimum Redundancy Codes, Proc. IRE, Vol.40, No.9, pp. 1098-1101

ISHIJIMA, M., HOSTETTER, G.M., (1983), Scan Along Polygonal Approximation of Electrocardiograms, IEEE Trans. Biomed. Eng. Vol.BME-30

KNUTH, D.E., (1985), Dynamic Huffman Coding, J. Algorithms, Vol.6, No.2, pp. 163-180

KORÜREK, M., (1988), Fizyoloji Ders Notları, İ.T.Ü Elektrik-Elektronik Fakültesi

LELEWER, D.A., DANIEL, S.H, (1987), Data Compression, A.C.M Comp. Surveys, Vol.19, No.3, pp. 261,296

MUELLER, W.C., (1978), Arrhythmia Detection Program for an Ambulatory ECG Monitor, Biomed. Sci. Instrument., Vol. 14, pp. 81-85

PASCO., R., (1976), Source Coding Alghorithms for Fast Data Compression, Phd. dissertation. Dept. of Electrical Eng., Stanford Univ., Stanford

RISSANEN, J.J., (1976), Generalized Kraft Inequality and Arithmetic Coding, IBM J. Res. Dev., Vol.20, pp. 198-203

RUBIN, F., (1979), Arithmetic Stream Coding Using Fixed Precision Registers, IEEE Trans. Inf. Theory, Vol.25, No.6, pp. 672-675

RUTH, S.S., KREUTZER, P.J, (1979), Data Compression for Large Business Files, Datamation, Vol. 18, No. 9, Pp. 62-66

SCHWARTZ, E.S., (1964 a), An Optimum Encoding with Minimum Longest Code and Total Number of Digits, Inf. Control, Vol.7, No. 1, pp. 37-44

SCHWARTZ, E.S., KALLIC, B., (1964 b), Generating a Canonical Prefix Encoding, Comm. ACM., Vol. 7, No. 3, pp. 166-169

SHANNON, C.E., WEAVER, W., (1949), The Mathematical Theory of Communication, Univerisity Of Illionis Press, Urbana

WELCH, T.A., (1984), A Technique for High-Performance Data Compression, Computer, Vol.17, No.6, pp, 8-19

YAZGAN, E., (1989), Tıp Elektroniğine Giriş Ders Notları, İTÜ Elektrik-Elektronik Ders Notları

ÖZGEÇMİŞ

Yazar 1970 senesinde Ankarada doğdu. 1987 senesinde İstanbul Fenerbahçe Lisesini bitirdi. Lisans öğrenimini Yıldız Teknik Üniversitesi, Kocaeli Mühendislik Fakültesi, Elektronik ve Haberleşme Mühendisliği bölümünde tamamlayan yazar, 1991 yılında İstanbul Teknik Üniversitesi Biyomedikal Programında Yüksek Lisans öğrenimine başlamıştır.