BİLGİSAYAR ORTAMINDA SESLİ İFADELERİ TANIMA

Abdulkadir KARACI

YÜKSEK LİSANS TEZİ
BİLGİSAYAR EĞİTİMİ

GAZİ ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

EYLÜL 2006
ANKARA

Abdulkadir KARACI tarafından hazırlanan BİLGİSAYAR ORTAMINDA SESLİ İFADELERİ TANIMA adlı bu tezin Yüksek Lisans tezi olarak uygun olduğunu onaylarım.

.....

Yrd. Doç Dr. Halil İbrahim BÜLBÜL

Bu çalışma, jürimiz tarafından oy birliği ile Bilgisayar Eğitimi Anabilim Dalında Yüksek lisans tezi olarak kabul edilmiştir.

Başkan: : Prof. Dr. Ali Paşa AYDIN

Üye : Yrd. Doç. Dr. Halil İbrahim BÜLBÜL

Üye : Yrd. Doç. Dr. Bülent ÇELİK

Tarih : 20/09/2006

Bu tez, Gazi Üniversitesi Fen Bilimleri Enstitüsü tez yazım kurallarına uygundur.

TEZ BİLDİRİMİ

Tez içindeki bütün bilgilerin etik davranış ve akademik kurallar çerçevesinde elde edilerek sunulduğunu, ayrıca tez yazım kurallarına uygun olarak hazırlanan bu çalışmada orijinal olmayan her türlü kaynağa eksiksiz atıf yapıldığını bildiririm.

Abdulkadir KARACI

BİLGİSAYAR ORTAMINDA SESLİ İFADELERİ TANIMA
(Yüksek Lisans Tezi)

Abdulkadir KARACI

GAZİ ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ
Eylül 2006

ÖZET

Sesli ifade tanıma ile ilgili çalışmalar uzun yıllar önce başlamıştır. Yapılan çalışmalara bakıldığında çoğu çalışma İngilizce dil yapısı üzerine kurulmuştur. Sesli ifade tanıma alanında Türkçe tabanlı çalışmalar çok azdır. Bu çalışmada Windows XP işletim sistemi üzerinde Türkçe olarak verilen komutları kişiye bağımlı olarak tanıyan bir yazılım geliştirilmiştir.

Geliştirilen yazılımda örüntü tanıma yöntemi kullanılmıştır. Ses tanıma yazılımında seslendirilen kelimenin başlangıç ve bitişini belirlemek için sesin enerjisi kullanılmaktadır. Başlangıcı ve bitişi tespit edilerek yakalanan sese sırayla filtreleme, pencereleme, FFT ve özilişki fonksiyonu işlemleri uygulanmaktadır. Böylece referans sinyali elde edilerek veritabanına kaydedilmektedir. Kullanıcı daha sonra aynı kelimeyi seslendirdiğinde seslendirilen bu kelime referans sinyalleriyle karşılaştırılarak tanıma gerçekleştirilmektedir. Yazılımda yapılan denemeler sonucunda sesli ifade tanımada mikrofona olan uzaklık, ortamdaki gürültü, kelimelerin ilk söyleniş tarzına benzer söylenmesi, ses kartının ve mikrofonun özelliği gibi faktörlerin etkili olduğu görülmektedir.

Bilim Kodu : 702.1.014
Anahtar Kelimeler : Ses tanıma, örüntü tanıma, saklı markov model
Sayfa Adedi : 134
Tez Yöneticisi : Yrd. Doç. Dr. Halil İbrahim BÜLBÜL

SPEECH RECOGNITION IN COMPUTER ENVIRONMENT

(M.Sc. Thesis)

Abdulkadir KARACI

GAZİ UNIVERSITY

INSTITUTE OF SCIENCE AND TECHNOLOGY

September 2006

ABSTRACT

It has been for many years since the studies on "Speech Recognition" were first conducted. When these studies are examined, it is seen that most of the studies are based on English language structure. The number of studies on "Speech recognition" which are based on Turkish language structure is very limited.

In this study, a software which recognizes the commands given in Turkish by a speaker has been developed on Windows XP operating system. "Pattern Recognition" method has been used in the software developed. To determine the beginning and the end of a word uttered, the "Speech Recognition software" benefits from the "energy of the sound". Filtering, windowing, FFT, and auto-correlation processes are successively applied to the sound which is caught. In this way, the "reference signal" is received and, then, recorded to the database. When the user utters the same word again, the word is compared with the reference signals, and the recognition process ends. The experiements done on the software during the trail period show that such factors as the distance from the microphone, the noise in the environment, the difference or similarity in the way a word is uttered, and the specifications of the sound-card and of the microphone affects the "Speech Recognition" process.

Science Code : 702.1.014
Key Words : Speech recognition, pattern recognition, hidden markov model
Page Number: 134
Adviser : Assistant Prof. Dr. Halil İbrahim BÜLBÜL

TEŞEKKÜR

Çalışmalarım boyunca değerli yardım ve katkılarıyla beni yönlendiren Hocam Yrd. Doç. Dr. Halil İbrahim BÜLBÜL'e, çalışmam boyunca manevi desteğini hiç esirgemeyen eşime ve kızım Ülkü Berin'e, teşekkürü bir borç bilirim.

İÇİNDEKİLER

Sayfa

ÖZET iv

ABSTRACT v

TEŞEKKÜR vi

İÇİNDEKİLER vii

ÇİZELGELERİN LİSTESİ x

ŞEKİLLERİN LİSTESİ xi

RESİMLERİN LİSTESİ xiii

SİMGELER VE KISALTMALAR xiv

1. GİRİŞ 1

2. SESLİ İFADE TANIMAYA GENEL BAKIŞ 3

2.1. Sesli İfadenin Üretimi 3

2.2. Sesli İfadenin Duyulması ve Algılanması 5

2.3. Ses Tanımada Ön İşlemler 6

2.3.1. Preemhasis filtre 7

2.3.2. Pencereleme 8

2.3.3. Fast fourier transform (FFT) (Hızlı fourier dönüşümü) 8

2.3.4. Sesin enerjisi 9

2.3.5. Sıfırdan geçiş sayısı (Zero crossing rate) 10

2.4. Sesli İfade Tanıma Sistemlerinin Genel Bir Sınıflandırması 10

2.4.1. Ayrışık sözcük tanıma (isolated word recognizion) 10

2.4.2. Sözcük yakalama sistemleri (word spotting systems) 11

2.4.3. Sürekli sesli ifade tanıma sistemleri 11

Sayfa

2.4.4. Konuşmacı bağımlı/bağımsız sistemler (speaker dependent/ independent systems) 12

2.5. Ses Tanımada Kullanılan Yöntemler 13

2.5.1. Örüntü tanıma (pattern recognition) 13

2.5.2. Hidden markov model 15

2.5.3. Dinamik zaman sıkıştırması (dynamic time warping) 17

2.5.4. Sinir ağı 18

2.6. Sesli İfade Tanımayı Etkileyen Faktörler 19

3. SESLİ İFADE TANIMA ALANINDA YAPILAN MEVCUT ÇALIŞMALAR 20

4. SESLİ İFADE TANIMA YAZILIMININ TASARLANMASI 23

4.1. Sesin Bilgisayar Ortamına Alınması 25

4.2. Veriyi Kutuplama 27

4.3. Sesin Enerjisini Hesaplama İşlemi 29

4.4. Kısa Zaman Enerji 31

4.5. Sesi Yakalama İşlemi 33

4.5.1. Sesin başlangıcını tespit etme işlemi 33

4.5.2. Sesin bitişini tespit etme işlemi 35

4.6. Preemhasis Filtre 39

4.7. Pencereleme 42

4.7.1. Kare pencereleme 44

4.7.2. Hamming pencereleme 46

4.7.3. Hanning pencereleme 47

4.7.4. Blackman pencereleme 49

Sayfa

4.7.5. Barlett pencereleme 50

4.8. Fast Fourier Transform (FFT).....52

4.9. Özilişki Fonksiyonu (Autocorrelation Function).....54

4.10. Tanıma.....56

4.10.1 Hamming uzaklığı 57

4.10.2. Öklid uzaklığı.....57

4.10.3. Manhattan uzaklığı 58

4.10.4. Kare uzaklığı 58

4.10.5. Cepstral uzaklık ölçümü 58

5. SONUÇ VE ÖNERİLER 64

KAYNAKLAR.....70

EKLER.....75
EK-1 Sesli ifade tanımayla ilgili olarak geliştirilen programın kaynak kodları 76

ÖZGEÇMİŞ.....134

ÇİZELGELERİN LİSTESİ

Çizelge **Sayfa**

Çizelge 5.1. Tek örnekleme, bayan sesine göre yazılımda yapılan deneme sonuçları....66

Çizelge 5.2. Tek örnekleme, erkek sesine göre yazılımda yapılan deneme sonuçları....67

Çizelge 5.3. İki örnekleme, bayan sesine göre yazılımda yapılan deneme sonuçları....68

Çizelge 5.4. İki örnekleme, erkek sesine göre yazılımda yapılan deneme sonuçları....68

ŞEKİLLERİN LİSTESİ

Şekil **Sayfa**

Şekil 2.1. Sesli ifade tanımaya genel bir bakış.....7

Şekil 2.2. Geleneksel olarak ses bitişinin tespiti 9

Şekil 2.3. Örüntü tanıma işlemi.....13

Şekil 2.4. Örüntü sınıflandırmanın ana kategorileri 14

Şekil 2.5. HMM modeli 16

Şekil 4.1. Ses tanıma yazılımının akış şeması.....24

Şekil 4.2. Sayısal ses sinyali.....25

Şekil 4.3. Sesi bilgisayar ortamına alış akış diyagramı 26

Şekil 4.4. Veriyi kutuplama akış diyagramı.....27

Şekil 4.5. Ses enerji akış diyagramı.....30

Şekil 4.6. Kaydet kelimesinin enerjisi 31

Şekil 4.7. Kısa zaman enerji akış şeması 32

Şekil 4.8. Ses yakalama ayarları penceresi 34

Şekil 4.9. Ses başlangıç akış şeması 35

Şekil 4.10. Ses bitiş akış şeması 36

Şekil 4.11. Preemhasis filtre akış şeması 40

Şekil 4.12. Preemhasis filtre uygulanmış ses sinyali.....41

Şekil 4.13. Sayısallaştırılmış ses sinyali 43

Şekil 4.14. Hanning pencereleme uygulanmış ses sinyali 44

Şekil 4.15. Kare pencereleme akış şeması 45

Şekil 4.16. Hamming pencereleme akış şeması 46

Şekil **Sayfa**

Şekil 4.17. Hanning pencereleme akış şeması 48

Şekil 4.18. Blackman pencereleme akış şeması 49

Şekil 4.19. Barlett Pencereleme akış şeması 51

Şekil 4.20. İşlenmemiş ses sinyali 53

Şekil 4.21. FFT uygulanmış ses sinyali 53

Şekil 4.22. Özilişki fonkisyonu akış şeması.....55

Şekil 4.23. İşlenmemiş ses sinyali 56

Şekil 4.24. Özilişki fonksiyonu uygulanmış ses sinyali 56

Şekil 4.25. Referans sinyalleri okuyan alt programın akış şeması 59

Şekil 4.26. Tanıma işlemini gerçekleştiren alt programın akış şeması.....61

Şekil 4.27. Tanıma işlemi sonuç penceresi 63

Şekil 4.28. Ses tanıma penceresi 63

Şekil 5.1. %97,5 benzerlik oranında tanınmış "İleri" kelimesinin referans sinyali ile karşılaştırılması……65

Şekil 5.2. %100 benzerlik oranında tanınmış "Word" kelimesinin referans sinyali ile karşılaştırılması……...65

Şekil 5.3. "Kaydet" kelimesinin yüksek ve normal ses tonlu karşılaştırma grafikleri……...66

Şekil 5.4. "Kaydet" ve "Kapat" ses sinyallerinin grafiksel karşılaştırılması 69

RESİMLERİN LİSTESİ

Resim **Sayfa**

Resim 2.1. Larynx'in görünümü 4

Resim 2.2. Sesli ifadeyi üreten ses organları 4

Resim 2.3. Kulağın yapısı 6

SİMGELER VE KISALTMALAR

Bu çalışmada kullanılmış bazı simgeler ve kısaltmalar, açıklamaları ile birlikte aşağıda sunulmuştur.

Simgeler	**Açıklama**
$\mathbf{a_n}$	f(t) fourier dönüşüm fonksiyonu katsayısı
$\mathbf{b_n}$	f(t) fourier dönüşüm fonksiyonu katsayısı
E	Sesin enerjisi
f(t)	Fourier dönüşüm foksiyonu
R(k)	Özilişki uzaklığı
X(j)	DFT fonksiyonu
w(n)	Hesaplanan pencere

Kısaltmalar	**Açıklama**
CMU	Carnegie Mellon University
dft	Discrete Fourier Transform (Kesikli Fourier Dönüşümü)
fft	Fast Fourier Transform (Hızlı Fourier Dönüşümü)
hmm	Hidden Markov Model

1. GİRİŞ

Teknolojideki gelişmeler insan makine arasındaki iletişimi önemli düzeyde artırmaktadır. Bu gelişmeleri daha fazla kullanmak için yeni alanlar araştırılmaya başlanmıştır. Geleceği parlak olan araştırma konularından biri konuşma tanımadır. Konuşma doğaldır: insanlar okuma ve yazmadan önce konuşmayı öğrenirler. Aynı zamanda konuşma daha verimli ve etkilidir: birçok insan yazmaya göre daha hızlı konuşur. Ayrıca konuşma esnektir: konuşmaya dokunamayız ve göremeyiz. Konuşma ile makineler kontrol edilebilir ve onlarla iletişim sağlanabilir. Elleri kullanmadan endüstriyel aletler veya motorlar kontrol edilebilir. Güçlü bir konuşma tanıma sistemi felçli hastalar için çok yarar sağlayabilir. Konuşma tanıma tekerlekli sandalye, ev aletleri, telefon içeren sistemler ve acil durum sistemlerini sesli komut vererek işletmek için uygundur [1].

Bilindiği gibi insana ait özelliklerin bilgisayara uygulanmasının ilk evresi insanın bu işlemleri nasıl yürüttüğünü bilmek ve onu taklit etmektir. Sesli ifadeyi bilgisayarın tanıması için ilk adım ses sinyalini sayısal hale dönüştürmek, sinyalin içinden sesin gerçek karakterini çıkarmak, elde edilen bilgilerin tanıma işlemine sokularak tanıma işlemini yapmaktır Sesli ifadelerin tanınması işleminde sesli ifadenin mikrofon aracılığıyla bilgisayara iletilmesi ilk aşamadır. Ses sayısal hale dönüştürülmesiyle pencereleme, filtreleme ve diğer analizler için hazır hale gelmiş olur. Yapılan bu işlemler sayesinde seste bulunan gürültüler ve kullanılacak alana bağlı olarak sesin kişiye bağımlı öğelerden arındırılması gerçekleştirilir.

Ses sinyalinin parametreleri yapılacak filtreleme ve analizler sonucunda elde edilirler. Sesli ifade tanıma seçilen sesli ifade tanıma yöntemlerinden biriyle gerçekleştirilir. Sesli ifade tanımada belirgin olarak aşağıdaki yöntemler kullanılmaktadır [2].

- Örüntü Tanıma,
- Hidden Markov Model,
- Dinamik Zaman Sıkıştırması

- Sinir Ağları

Günümüze kadar yapılmış ve halen yapılmakta olan sesli ifade tanıma ile ilgili çalışmalar, çok büyük oranda İngilizce dil yapınsın özellikleri esas alınarak gerçekleştirilmiştir. Diğer bazı Avrupa dilleri ve Çince ile ilgili çalışmalar da az da olsa vardır. Buna rağmen Türkçe diline özgü, daha doğrusu, Türkçenin dil özellikleri dikkate alınarak yapılmış çalışmalara henüz pek sık rastlanamamaktadır [3]. Microsoft firmasının sesli ifade tanımayla ilgili geliştirmiş olduğu "SPEECH SDK" yazılım geliştirme ortamı bile başta İngilizce olmak üzere birkaç dili desteklemektedir ve Türkçe dil desteği yoktur.

Bu araştırmada amaç kişisel bir bilgisayarda Türkçe olarak seslendirilen sınırlı sayıdaki kelimeleyi örüntü tanıma yöntemini kullanıp, konuşmacıya bağımlı olarak tanıyan bir yazılım geliştirmek ve bunun ilkelerini ortaya koymaktır. Bu amaçla bir yazılım geliştirilmiş ve yazılım Windows XP işletim sistemi ortamında söylenen kelimeleri veritabanına kaydetmektedir. Bu kelimeler istenildiğinde bir programla ilişkilendirilebilecektir. Daha sonra kullanıcı veritabanına kayıtlı bir kelimeyi seslendirdiğinde o kelime tanınanacak ve o kelimeyle ilişkili bir program varsa o program çalıştırılacaktır.

2. SESLİ İFADE TANIMAYA GENEL BAKIŞ

İnsan-makine arasında insan için en doğal olan sesli iletişimin kullanılabilmesi, sesli ifadenin makine tarafından tanınabilmesini gerekli kılar. Son 15-20 yılda çeşitli sesli ifade tanıma stratejileri, farklı dillerde ve bu dillere özgü biçimlerde gerçekleştirilmeye çalışılmıştır. Sesli ifade tanıma; sinyal işleme, örüntü tanıma, yapay anlayış, bilgi ve olasılık kuramları, istatistik, bilgisayar bilimleri, psikoloji, dilbilim, biyoloji, ses bilim gibi değişik bilim dallarını ilgilendirmektedir [4, 5].

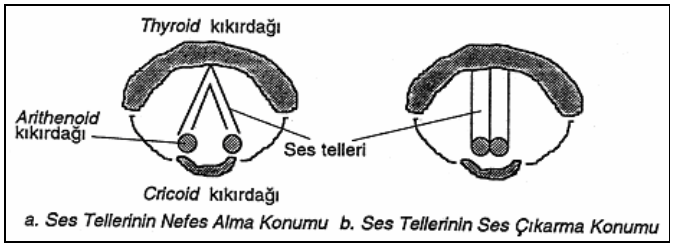
2.1. Sesli İfadenin Üretimi

Sesli ifade tanımanın anlaşılabilmesi için insan sesinin nasıl üretildiğini anlamak önemlidir. İnsan sesini üreten kısımlar şunlardır:

- Gırtlaktan dudaklara kadar uzanan ses yolu (vocal tract)
- Ses telleri (vocal cord)
- Geniz boşluğu (nasal cavity)
- Dudaklar (lips)

Ses yolu (vocal tract) boru şeklinde bir kısımdır ve bitiş noktası dudaktır. Aslında akciğerlerden çıkan hava ses yolu boyunca taşınır [6]. Ses telleri(vocal cord) Larynx denilen kısım üzerinde bulunur. Ses tellerinin larynx içindeki konumu Resim 2.1'de gösterilmiştir. Bu yapıda aritenoid kıkırdakları açılıp kapanarak ses tellerinin nefes borusunu açıp kapaması sağlanmaktadır.

Ses tellerinin tam açık olduğu durum nefes alıp verme durumudur. Ses tellerinin nefes borusunu tam kapadığı durum ise yutma durumudur. Bu iki durumda ses tellerinin titreşmesi söz konusu olmaz. Havanın ses telleri arasında belli bir süreklilik ve şiddette dışarı itilmesi bu tellerin titreşmesine olanak verir. Bu titreşim sesli ifadede ünlülere kaynaklık eden titreşimdir.

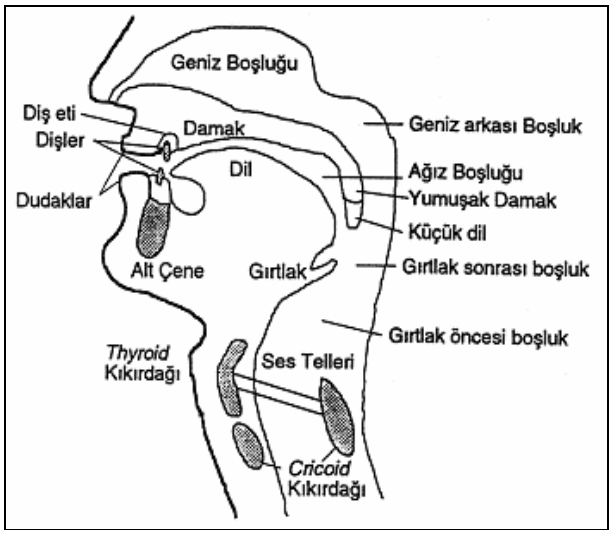


Resim 2.1. Larynx'in görünümü

Sesli ifade üretim sürecine katkı verebilen diğer birimler:

- Gırtlak,
- Alt çene,
- Dil,
- Yumuşak Damak ve Küçük dil,
- (Sert) Damak,
- Dişler
- Diş etleri ve
- Dudaklarıdır.

Sesli ifadeyi üreten ses organları Resim 2.2'de gösterilmektedir.



Resim 2.2. Sesli ifadeyi üreten ses organları

Sesli ifade üretim süreci, titreşimlerin oluşturulması ve bunların modülasyonunu içerir. Titreşimlerin oluşturulması sesin tahriki olarak da bilinir. Sesli ifade üretim sürecinde, sesin tahriki çeşitli biçimlerde gerçekleşir. Bu bağlamda:

- Ses tellerinin titreştirilmesi (phonation),
- Ses telleri arasından havanın fısıltı (whispering) yaratacak biçimde geçirilmesi,
- Ses tellerinden dudaklara değin herhangi bir konumda, havanın türbülans oluşturacak biçimde sürtüşmesinin (frication) sağlanması,
- Ağız içi boşluğun hava ile doldurulup basınç oluşturulması, sonra ağzın birden açılarak patlama (explosion) sağlanması,

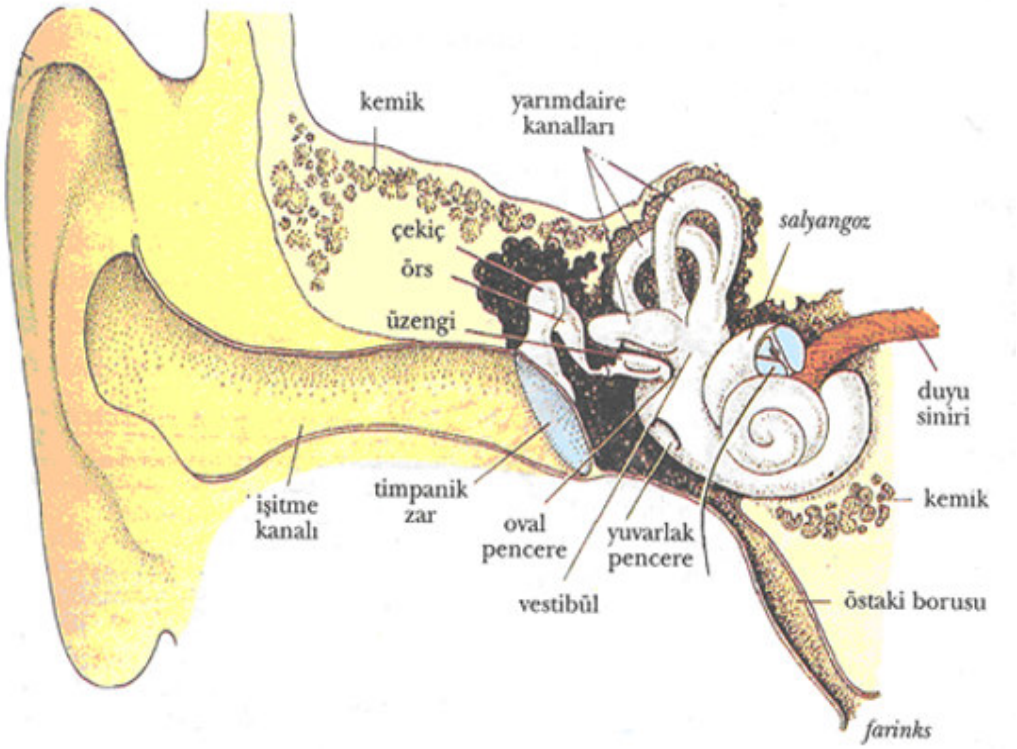
yollarıyla sesin tahriki sağlanır [4].

2.2. Sesli İfadenin Duyulması ve Algılanması

Sesli ifadeler, duyma organı olan kulak ile duyulmakta ve beyin tarafından algılanmaktadır [4]. Kulak, dış kulak, orta kulak ve iç kulak olmak üzere üç bölümden oluşur. Dış kulak sesin toplanması, yükseltilmesi ve orta kulağa iletilmesinde rol oynar. Kulak zarı; dış kulak yolunun sonunda, dış kulakla orta kulak arasında yer alır. İnce, yarı saydam bir zardır. İnsanda yaklaşık 1 cm çapında ve kollajen ipliklerden yapılmıştır.

Orta kulak, dış kulaktan sonra gelen işitme kemikçikleri ve bunlara bağlanan kas bağlarından oluşur. Ayrıca burada östaki kanalı bulunur.

İç kulak, şakak kemiği içinde bulunan ve vücudun en iyi korunan organıdır. Dış ve orta kulak, sadece işitme ile ilgili oldukları halde; iç kulak hem "işitme" hem de "denge" duygusunun algılandığı yapıları taşır [7]. Kulağın bölümleri Resim 2.3'de ayrıntılı olarak gösterilmektedir [8].



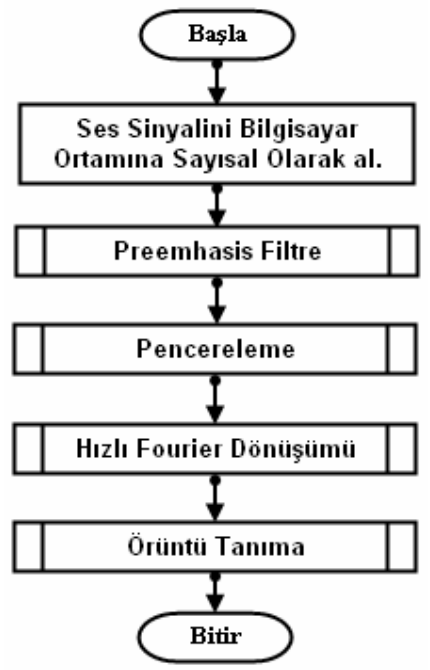
Resim 2.3. Kulağın yapısı

İnsan kulağına etki yapan ve işitme duyusunu uyaran ses dalgaları hava ile iletilir. Su ve birçok katı maddeler ses dalgalarını iletseler de, işitmede önemli olan hava ile iletimdir. Fiziksel yönden sesler; şiddet, frekans ve dalga şeklinde birbirinden ayırt edilir. Şiddet, titremenin şiddetidir, bunu biz kulağımızla sesin şiddeti şeklinde duyarız. Frekans ise saniyedeki titreşim sayısıdır. Bu da sesin tonunu tayin eder. Dalga şekli ise sesin kalitesini ayırt etmemizi sağlar. Örneğin bir keman ve bir piyano telleri aynı şiddet ve aynı frekansta titreşseler de dalga şekilleri farklı olduğundan biz bu aletleri birbirinden ayırabiliriz [7].

2.3. Ses Tanımada Ön İşlemler

Ses tanıma işlevinin yapılabilmesi için ses sinyalinin alınmasından sesi tanımanın gerçekleşmesine kadar bir çok ön süreç vardır. Tanımanın kullanım amacının belirlenmesi ve bu alanda karşılaşılacak sorunları aşmak için kullanılacak ön işlemlerin belirlenmesi ise tanımanın ilk evresidir. Ses tanımada kullanılan ön

işlemlerle tanıma olayına temel bir bakış açısıyla yaklaşılırsa, genel olarak elde edilecek yöntem sırası Şekil 2.1'de gösterilmektedir [9].



Şekil 2.1. Sesli ifade tanımaya genel bir bakış

2.3.1. Preemhasis filtre

Filtereleme işlemi zaman ve sıklık evreninde ayrı, ayrı ele alınabilir. Ön filtreleme işlemi, çoğu kez zaman evreninde uygulanmaktadır. Filtreleme, filtrelenecek sinyal ile filtre fonksiyonu arasında convolution işleminin uygulanmasıdır. Kullanılacak filtre fonksiyonu, sıklık-genlik evreninde anlamlı bir biçimde kolayca belirlenen filtre fonksiyonunun zaman-genlik evrenindeki dönüşümü (transformation) olarak seçilmektedir [4].

Sıradaki ses sinyali örneğinin, sabit bir katsayı ile çarpılmış bir önceki örnekten farkının alınmasıyla elde edilir [10]. Preemhasis filtrenin fonksiyonu Eş. 2.1 deki gibidir [9].

$$H(z)=1-az^{-1} \quad (2.1)$$

Burada z^{-1} standart gecikmeyi ifade eder. Bir önceki eleman olarak düşünülebilir.

2.3.2. Pencereleme

Sayısallaştırılan sinyaller bilgisayarda sözcükler olarak depolanır. Sinyal işleme aşamasında, genellikle sesli ifadeleri temsil eden sözcüklerin tümünü bir seferde ele alma olanağı bulunmaz. Bu nedenle anlamlı uzunluklara bölünmesi gerekir. Bir seferde işleme alınacak sözcük kümesi, sinyal içinde bir pencere olarak tanımlanır [10].

Pencereleme teorisi digital sinyal işlemede eskiden beri araştırılan çok etkin bir konudur. Pencerelemenin bir çok tipi vardır.

- Rectangular
- Hamming
- Hanning
- Blackman
- Barlett
- Kaiser

Hamming window çok kullanılır. Hanning window Hamming window'un özel bir şeklidir [11].

2.3.3. Fast fourier transform (FFT) (Hızlı fourier dönüşümü)

Bir konuşma sinyalini analiz etmede kullanılan en yaygın ve en çok bilgi verici yol, Fourier dönüşümü yöntemini kullanarak kısa zamanlı güç spekturumunun kestirimini yapmaktadır [10]. Fourirer dönüşüm fonksiyonu Eş. 2.2 deki gibidir [12].

$$f(t) = \frac{a_0}{2} + \sum_{n=1}^{\infty} [a_n \cos(w_n t) + b_n \sin(w_n t)] \qquad (2.2)$$

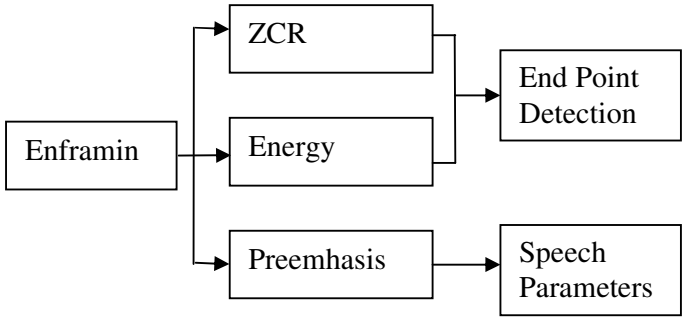
$$a_n = \frac{2}{T}\int_{-T/2}^{T/2} f(t)\cos(w_n t)dt \tag{2.3}$$

$$b_n = \frac{2}{T}\int_{-T/2}^{T/2} f(t)\sin(w_n t)dt \tag{2.4}$$

Eş 2.3 ve Eş. 2.4 deki katsayılar fourier katsayılarıdır.

2.3.4. Sesin enerjisi

Sesin başlangıç ve bitişinin tespit edilebilmesi için şiddetinin yani enerjisinin belirlenmesi gerekir. Gürültülü ortamlarda sesli ifadenin belirlenmesi ses tanıma sistemlerinin ilk adımıdır. Bu özellikle kelime tanıma sistemlerinde ayırma için çok önemlidir. Ses sinyalinin enerjisini hesaplama diğer işlemlerle karşılaştırıldığında basit bir işlemdir. Bitiş noktasını belirleme önemsiz bir işlem değildir. Ses sinyalini işlemede sıfırdan geçiş sayısı (Zero Crossing Rate-ZCR) ve kısa zaman enerji (short time energy) iki temel parametredir. Genellikle bu iki parametre diğerlerinden daha önce hesaplanır. Yanlış tanıma sonuçlarının yarıya yakını bitiş noktasının yanlış tespitindendir. Geleneksel olarak bitiş noktasının belirlenmesi Şekil 2.2'de gösterilmektedir [13].



Şekil 2.2. Geleneksel olarak ses bitişinin tespiti

Ses tanıma sistemleri geliştirilmesi sırasında karşılaşılan en büyük problem, sesin başlangıç ve bitişini tespit etmektir. Kullanıcıdan alınan ses sinyalinin, belirlenen bir

eşik enerjisini aşması ses sinyalinin başladığını ve belirli bir eşik enerjisinin altına düşüp belli bir zaman bu düşük enerjide devam etmesi ses sinyalinin bittiğini gösterir [10].

2.3.5. Sıfırdan geçiş sayısı (Zero crossing rate)

Ses sinyalinin sıfırdan geçiş sayısı zero crossing rate olarak bilinir. Sesli ifade kayıtlarında ses sinyalinin bulunmadığı kesimlerde bu sayı, gürültünün yüksek sıklıkta bir sinyal olmasından dolayı artmaktadır. Sesli ifadelerin yer aldığı kesimlerde ise, zero crossing rate olarak bilinen bu değer düşük olmaktadır. Bu özellik, sesli ifadelerin başlangıç ve bitiş noktalarını belirlemede kullanılmaktadır [4].

2.4. Sesli İfade Tanıma Sistemlerinin Genel Bir Sınıflandırması

Sesli ifade tanıma sistemleri ya da sesli ifade tanıyıcılar, artan zorluk sırasına göre aşağıdaki gibi sıralanabilmektedir:

- Ayrışık sözcük tanıma sistemleri (isolated word recognition systems)
- Sözcük yakalama sistemleri (word spotting systems)
- Sürekli sözcük /sesli ifade tanıma sistemleri (connected word / speech recognition systems) [4].

2.4.1. Ayrışık sözcük tanıma (isolated word recognizion)

Ayrışık sözcük tanıma sistemleri tek kelime veya bir konuşmacı tarafından seslendirilen ayrışık ifadeleri tanır. Bu tanıma işleminde konuşmanın ses içeren kısmının, gürültüyü içeren ses olmayan kısımdan ayrılması zorunludur. Kelime sınırının doğru ve güvenilir bir şekilde belirlenmesi kelime tanıma sisteminin performansı için kritik önem taşımaktadır [14]. Ayrışık sözcük tanıma sisteminde söylenen her sözcükten sonra kısa bir süre durmak gerekir [15]. Ayrışık sözcük

tanımada kelimelerin farklı söyleniş şekilleri sözlüğe girilmelidir. Bu yüzden yeni bir sözlük oluşturma işlemi uzun bir işlemdir [16]. Küçük ve orta boyuttaki sözlüklerdeki kelimelerin tanınmasını içeren işler için pratik uygulamalar sağlar [17]. Ayrışık sözcük tanımada hız sorunu vardır. Bu sorun kelimenin sınırlarını belirleme işleminden kaynaklanmaktadır. Sesin sınırları enerjisine bakılarak bulunur. Çünkü ses bittiğinde belirli bir süre düşük bir enerji seviyesi elde edilir [18].

2.4.2. Sözcük yakalama sistemleri (word spotting systems)

Sözcük yakalama sistemi gürültülü ortamlar veya diğer sesler içindeki cümlecikleri ve kelimeleri tanımlamak için kullanılan bir sistemdir. Bu kelimelere anahtar kelimeler (keywords), cümleciklere ise anahtar cümlecik (key phrases) denir. Sözcük yakalama sistemi sesli ifadenin tümünü tanıyan bir sistem değildir. Sadece anahtar kelimelerle ilgilenen bir sistemdir [19]. Araştırmacılar anahtar kelimeleri tanımak için bilgisayarlara insanların doğal olarak ne yaptıklarını öğretirler. Word spotting olarak adlandırılan bu teknik bilgisayara cümle içinde kelime tanımayı mümkün kılar. Örnek: “Ben İstanbul’a gidiyorum” cümlesinde olan “İstanbul’a” gibi [10]. Konuşma içinde aranan sözcüklerin yakalanmasını sağlayan bu tür tanıyıcılar sürekli sesli ifadeler üzerinde çalışmaktadır. Time warping ile dinamik programlama tekniklerinin sıkça kullanıldığı bu tür sistemlerde her sözcük bir şablon (template) ile ifade edilmektedir. Tanıma işlemi, aranan şablonun sesli ifade içinde çakıştığı bir örüntü arama biçiminde gerçekleşmektedir [4].

2.4.3. Sürekli sesli ifade tanıma sistemleri

Sürekli sesli ifade tanıma, genelde iki biçimde ele alınmaktadır. Eğer sürekli sesli ifade içinde sözcük sınırları bulunarak sorun sözcük tanımaya indirgenebilirse sürekli sözcük tanımadan; fonem gibi, sözcüğe göre daha alt düzey birimler tabanında bir tanıma gerçekleştiriliyor ise sürekli ifade tanımadan söz edilmektedir [10].

Sürekli sesli ifade tanıma işleminde kelimenin başını ve sonunu belirlemek zordur [20]. Sürekli sesli ifade tanımadaki zorluk üç değişik nedenden kaynaklanmaktadır.

Bunlardan ilki, sürekli sesli ifade içindeki sözcük sınırlarının belirgin olmamasıdır. Ayrışık sözcük tanımada, sözcük sınırları bilindiğinden başarı yüksek olmaktadır. Sürekli sesli ifadede, aranan sözcük sınırlarının bulunması zor, hatta kimi zaman olanaksızdır. Sözcük başına gelen /s/ sesinin çoğu kez kaybolmasının oluşturduğu sorun buna örnek olarak gösterilebilir. İkinci sorun, tanınacak ses birimlerinin, ön ve arkasına gelen diğer ses birimleri tarafından etkilenmeleri, birlikte seslendirilenlerden kaynaklanan sorundur. Üçüncü sorun vurgulama, duraklamadan kaynaklanmaktadır. Duraklama ve vurgulama sözcüklerin söyleyişte kaybedilmesine neden olmaktadır. İsim, sıfat ve kısa süreli sözcükler çoğu kez düşük enerjili olmakta ya da yutulmaktadır. Bu da sistemin, sesli ifadeleri doğru biçimde yakalamasını olumsuz yönde etkilemektedir [4].

2.4.4. Konuşmacı bağımlı/bağımsız sistemler (speaker dependent/independent systems)

Konuşmacı bağımlı sistemler için belirli bir kullanıcı tarafından daha önce tanımlanmış bir kelime ya da cümle ele alınır. Bu çeşit sistemler olanakların ve zamanın, sistemin bir konuşmacıya bağımlı olarak eğitilmesi için yeterli olduğu masaüstü uygulamaları için kullanılır.

Konuşmacıdan bağımsız sistemler konuşmacılardan alınan çok miktarda ses örnekleriyle bir ön öğrenmeden geçirilir. Kullanıcı böyle bir sistemi hemen kullanmaya başlayabilir. Böylece işlemler çok daha kolaylaşır. Bu sistemin dezavantajı, bir dil için bütün konuşmacı varyasyonlarını modellemenin imkansız olmasıdır. Konuşmacı bağımsız sistemler özel konuşmacı eğitimine gerek duymazlar. Bu bir avantaj olarak görülse de daha düşük kalitede performansa sahiptirler.

Konuşmacı bağımlı sistemlerin performansı konuşmacı bağımsız sistemlere göre çok daha yüksektir. Ancak bir sistemin kullanım alanını artırmak için amaç konuşmacıdan bağımsız bir sistem olmasıdır. Fakat tahmin edilebileceği gibi bunu başarmak konuşmacıya bağımlı bir sistem geliştirmekten daha zordur [21].

2.5. Ses Tanımada Kullanılan Yöntemler

2.5.1. Örüntü tanıma (pattern recognition)

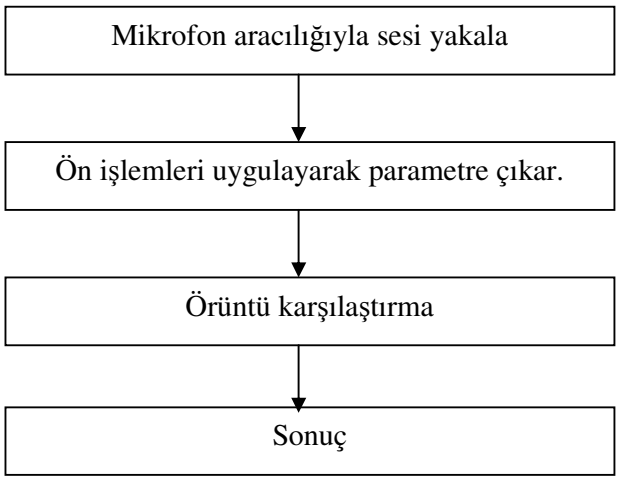
Örüntü tanımanın amacı nesneleri belirli sayıda sınıf veya kategoriye ayırmaktır. Bir örüntü nesne olarak tanımlanır.

Aslında örüntü tanıma işlemi üç aşamada gerçekleşir:

1. Ses verisini elde etme.
2. Ses verisi üzerinde ön işlemleri uygulama.
3. Sınıflandırma.

Ses verisini elde etme aşamasında dış ortamdaki sesler mikrofon aracılığıyla sayısal olarak bilgisayar ortamına alınır. Bilgisayar ortamın alınan ses verisi ikinci aşamada (ön işlemler) kullanılır ve özellik vektörleri çıkarılır. Üçüncü aşamada ise özellik vektörleri sınıflandırıcılar tarafından bazı istatistiksel ölçütler kullanılarak sınıflandırılır [22].

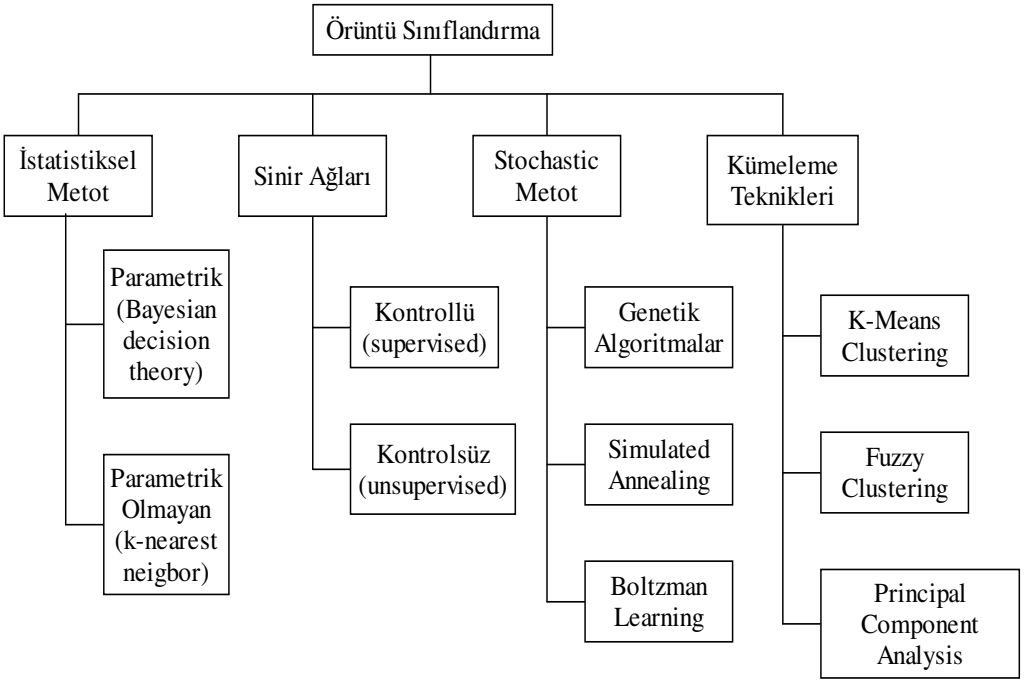
Yukarıdaki aşamalar Şekil 2.3 ile gösterilebilir [23].



Şekil 2.3. Örüntü tanıma işlemi

Örüntü tanıma ile ilgili yapılan araştırmalarının amacı insanın örüntü tanıma kabiliyetini bilgisayara uygulamaktır. Örüntü tanımanın görüntü analiz, karakter tanıma, ses tanıma, kişi ve nesne tanımlama, endüstriyel kontrol, mali veri analizi, çevresel analiz, galaksileri sınıflandırma gibi önemli uygulama alanı vardır [24].

Örüntü tanımada birincil amaç örüntüleri sınıflandırmadır. Örüntü sınıflandırma sistemleri genel olarak ikiye ayrılır: kontrollü (supervised) ve kontrolsüz (unsupervised) teknikler. Kontrollü sınıflandırmada sınıflar ve onların sınırları, sınıflandırıcı kullanmadan önce sınıflandırma sistemini tasarlayan tasarımcı tarafından belirlenir. Diğer taraftan kontrolsüz öğrenme sistemi örüntü gruplarının benzerliklerine bakarak sınıfların sınırlarını öğrenir. Şekil 2.4‘ de örüntü sınıflandırmanın ana kategorileri verilmektedir [25].



Şekil 2.4. Örüntü sınıflandırmanın ana kategorileri

Örüntüye dayalı ses tanımlama yaklaşımında ilgi direkt olarak spektral vektörlerin zaman sıralamasına odaklıdır. Örneğin bir T denek örneği tanımlayalım.

$T=\{t_1, t_2, t_3, \ldots\ldots, t_i\}$ (2.5)

Burada her bir "t" her seferinde girilen konuşmanın spektral "i" ise konuşma çerçevesinin toplam sayısını simgeler. Buna benzer bir şekilde biz referans örneklerinin oluşturduğu sesi $\{r_1^j, r_2^j, r_3^j, ...r_j^j\}$ R olarak simgeleriz. Her bir R_1 bir frekans örüntüsünü hem de spektral çerçevelerin oluş sırasını simgeler. Örüntü karşılaştırma sisteminin amacı, T'nin her bir R^j $1 \le j \le V$ uzaklığını ya da zıtlığını belirlemektir. Bununda amacı ise minimum zıtlığa sahip olan referans örüntüsünü belirlemek ve bu örüntünün konuşma girdisiyle ilişkilendirilmesini sağlamaktır.
T ile R'nin global benzerliğini incelemek için aşağıdaki sorunları çözmelidir:

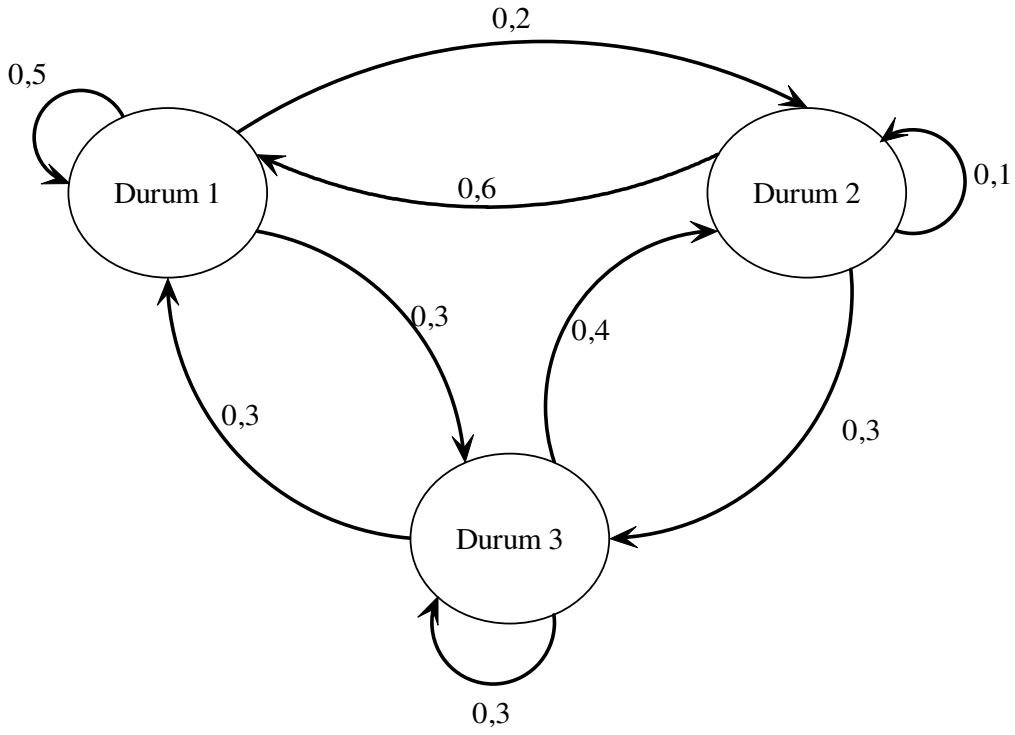
- T ve R'nin uzunlukları birbirine eşit değildir. Bunun sebebi konuşmacıların kullandıkları değişik şiveler ve farklı ses perdeleridir,
- T ve R iyi tanımlanmış bir sıralamaya girmezler. Çünkü farklı sesler aynı ölçüdeki bir süreç içerisinde çeşitlenmezler. Böylece sesliler kolaylıkla uzatılıp ve kısaltılabilir ama çoğu sessizler zaman içerisinde büyük değişikliğe uğramazlar,
- Yapmamız gereken spektral vektör çeşitlerini yani yerel farklılaşmayı ya da uzaklık ölçümünü global benzerlik seviyesine getirerek T ve R arasında zamansal bir sıralama oluşturmayı kolaylaştırmaktır [10].

2.5.2. Hidden markov model

Hidden Markov Model (HMA) olasılık teorisi modelidir. 1960'lardan sonra birçok alanda çok fazla olarak uygulanmıştır. HMM'in deniz altı hedeflerinin sınıflandırılması (sonar target classification), optik karakterler ve el yazısı tanıma, görüntü sınıflandırma, eletroencephalographic örüntü tanıma (elektriksel beyin aktivitelerinin tanınması) ve sesli ifadeleri tanıma, sinyal işleme gibi birçok uygulama alanı vardır [26, 27]. HMM fonem tanıma, ayrışık sözcük tanıma, sürekli sözcük tanıma gibi birçok sesli ifade tanıma yönteminde kullanılmıştır. HMM'in sesli ifade tanımada yaygın olarak kullanılmasının iki ana sebebi vardır. Birincisi sesli ifade tanıma için istatistiksel bir çerçevede güçlü bir algoritma sağlar. İkincisi HMM'in yapısı bizim ses sistemimize uygun bir model olarak görülmektedir [28].

Bir markov işleminde değer, herhangi bir anda sadece en son değere bağlı ve işlemdeki diğer değerlerden bağımsızdır. Şekil 2.5' deki okların üzerindeki sayılar bir sonraki adımda (şu anda işlemin hangi durumda olduğuna bağlı olarak) hangi duruma geçebileceğinin olasılığını verir. Temelde olan şudur, eğer durum 1'de ise o zaman:

- %50 durum 1'de kalma şansı,
- %20 durum 2'ye girme şansı,
- %30 durum 3'e gitme şansı vardır.



Şekil 2.5. HMM modeli

HMM de sesli ifade üretiminin sonlu durumlu bir sistem tarafından üretildiği varsayılır. Bu sistemde her durumun, sonlu sayıda sesli ifade çıktısı üretilebileceği düşünülür. Sesli ifade üretilirken, bu sonlu durumlu sistemin bir durumdan diğerine geçtiği ve her durumda belirli bir sesin üretildiği varsayılır. Bir durumdan diğerine geçiş olasılığı geçiş ağırlığı olarak bilinir [10].

Hidden Markov modeli çerçevesinde üç soruna yanıt arayarak bu modelin pratikte kullanılması sağlanır. Bu sorunlar değerlendirme sorunu, kod çözümü sorunu ve öğrenme sorunudur. Değerlendirme sorunu çerçevesinde bir dizi gözlemin verilen bir modelce oluşturulma olasılığı araştırılır. Bu amaçla forward algorithm adlı bir algoritma kullanılır. Kod çözme sorunu eldeki bir dizi gözlemin hangi durum dizisi ile üretildiği sorusuna yanıt arar. Bu sorunun yanıtı Viterbi algoritması ile bulunmaya çalışılır. Viterbi algoritması sesli ifadelerin kesimlenmesi ve sürekli sesli ifade tanımada uygulama bulmaktadır. Öğrenme aşamasında model parametreleri forward-backward algoritması kullanılarak optimize edilir [4].

2.5.3. Dinamik zaman sıkıştırması (dynamic time warping)

Dinamik zaman sıkıştırması dinamik programlamadan türemiş şablon (template) karşılaştırma algoritmasıdır. Bu yöntem sesli ifade tanımada çok etkili bir yöntemdir. Konuşma tanımada referans sözlükteki şablonlarla konuşulan kelime karşılaştırılmak zorundadır. Kullanıcıya bağımlı olarak değişen konuşma hızı bu işlemdeki ilk zorluktur [29, 30]. İşte dinamik zaman sıkıştırması esas olarak, iki konuşma örüntüsünün (pattern) değişik konuşma hızlarını düzenleyen bir genel zaman ayarlama prosedürü olarak kullanılır.

Dinamik zaman sıkıştırması modelinde aşağıdakiler kabullenilir;

- Yerel değişimler ufaktır ve uzaklık hatası yerel devamlılık sınırlaması olarak bilinen uzaklık hatası yöntemi ile bulunur,
- Aynı kelimenin değişik durumlarda genel söyleniş değişimlerinin doğrusal zaman normalizesi ile ele alınması,
- Test konuşmasının her birim çerçevesi (frame) tanımaya eşit katkıda bulunur,
- Tüm çerçeveler boyunca sabit olarak uygulanan tek bir mesafe ölçüsü yeterlidir [10].

Bu yöntem sesli ifade tanımada oldukça sık kullanılır. Bu yöntem daha çok diğer yöntemlerle birlikte kullanılan ve tanıma işlemlerinin verimliliğini artırmak amacıyla

kullanılan bir yöntemdir. Bu yöntemde, sesli ifadelerin seslendirme süreleri sıkıştırılarak ya da genişletilerek referanslarla karşılaştırılmaları ilkesi kullanılmaktadır [4].

2.5.4. Sinir ağı

Nöron ağı sesli ifade tanımada geniş olarak kullanılmaya başlanan bir yöntemdir. Nöron ağı diğer yöntemlerden daha yavaştır.

Nöron ağı bir örüntü tanıma sistemidir. Bilgisayar belli bir örüntüyü verilen bir sonuçla ilişkilendirmeyi öğrenir. Örneğin, bir nöral ağa 1, 2 ve 3 rakamlarının karakteristikleri öğretildiğinde artık aynı rakamları farklı bir font ya da başka birinin el yazısıyla bile sunulduğunda tanıyabilmektedir.

Nöron ağı birkaç nöron tabakasından oluşur; bir giriş tabakası, bir ya da daha çok gizli tabaka ve bir çıkış tabakası. Her bir nöron tabakası girdisini bir önceki tabakadan ya da ağ yapı girdisinden alır. Her nöronun çıktısı diğer seviyeyi ya da ağ yapısının çıktısını besler. Nöron ağı da birkaç giriş tabakasına ayarlanabilir ağırlıklı bağlantılarına, tümüyle bağlı olan bir gizli tabaka ve çıkış tabakasının gizli tabakayla tamamen bağıntılı olduğu, bir ya da daha fazla çıkış birimi içerebilen çıkış tabakası mevcuttur.

Nöron ağı kabaca insan beynine dayanmaktadır. İnsan beyni birbirine bağlı milyonlarca nörondan oluşmaktadır. Bunlar diğer nöronlardan oluşan ağa bağlıdırlar. Meydana gelen tüm işlemler bu birbirine bağlı nöronların veri alış verişlerinden kaynaklanır. Bağlantıların bazıları kuvvetli bazıları da zayıf olabilirler. Böylece insan beyni öğrenme ağıyla, nöronların tepkilerine göre birçok görevi yerine getirebilir. Bu öğrenme mühendislere bilgisayarda Nöron Ağı modellemesini yapma ilhamı vermiştir. Böylece bir bilgisayara programlandığı konu dışında milyonlarca başka görevi de yerine getirebilir [10].

2.6. Sesli İfade Tanımayı Etkileyen Faktörler

Tanıma işleminin yapay olarak bilgisayarda uygulanması için ilk adım olarak ses tanımada karşılaşılan problemlerin tanımlanması gerekir. Tutarlı ve mükemmel güvenilir ses tanıma birçok sebepten dolayı zordur. Mesela sorunlardan birisi sesin insandan insana değişmesidir. Çoğu kişi seslerinde gürültü ve rahatsız edici (öksürme, hapşırma, duraklama) durumlar bulundurur ve herkes aynı kelimeyi her zaman aynı nitelikte telaffuz etmez [10]. Sözcüklerin seslendiriliş biçimi kişiden kişiye değişebileceği gibi bir kişinin konuşması da konuşmanın içeriğine, kişinin içinde bulunduğu ortamın gürültü düzeyine göre değişiklikler göstermektedir.

Sesli ifadede dilbilgisi kurallarına tam olarak uyulmaması bir diğer sorundur. İnsanlar arası karşılıklı konuşmada dilbilgisi kurallarına göre bir sonraki sözcüğü tahmin etmek mümkün olabilmektedir [31].

Diğer bir sorun ise birçoğumuzun konuşma tarzıdır. Konuştuğumuz zaman genelde yanlış telaffuz edilen veya gereksiz kullanılan (um, ee, hımm) kelimelerdir. Kelimeler aynı zamanda arka arkaya söylendiğinde yutulabilirler. Buna ilaveten, bir çok kelime birbirine benzer. Sorunlar maddeler halinde aşağıdaki gibi sıralanabilir:

- Akıcı bir konuşmada kelimeler arasında boşluklar olmayabilir. Bu konuşmayı tanımaya çalışan yazılım için büyük sorunlar çıkarır,
- Sesler kişiden kişiye çok değişik olabilir. Aynı kişi bile aynı kelimeyi değişik bir şekilde telaffuz edebilir,
- Kadın, erkek ve çocuk sesleri arasında büyük farklılıklar vardır,
- Konuşmanın yapıldığı ortamda tanımayı zorlaştıran başka seslerde bulunabilir. Bu seslerin ana sesten ayrıştırılması işlemi zordur.
- Konuşmacının sesi tam olarak anlaşılmasa bile insan gelen kelimeyi konuyu bilmesinden, kelimenin gelişinden dolayı anlayabilir. Böyle bir tanımayı bilgisayara uygulamak zordur [10].

3. SESLİ İFADE TANIMA ALANINDA YAPILAN MEVCUT ÇALIŞMALAR

Çalışmanın bu bölümünde bu çalışmanın konusu ile ilgili YÖK Dökümentasyon Merkezi, Milli Kütüphane, Gazi Üniversitesi Merkez Kütüphanesi ve internet kaynakları taranmak suretiyle yerli ve yabancı literatürdeki çalışmalar elde edilmiş ve konuyla ilgili yönleri ile burada özetlenmiştir.

Bir makine ile kendi sesini kullanarak iletişim kurma bilim adamları ve mühendislerin uzun zamandan beri ilgisini çeken bir konudur. Sesli ifade tanıma işleminde söylenen kelime bir mikrofon aracılığıyla elektriksel sinyale dönüştürülür. Daha sonra sinyal özellik çıkarma işlemine tabi tutulur. Bu özellikler sözlükteki örüntülerle karşılaştırılır. Eğer örüntülerden biriyle eşleştirilirse kelime bulunmuş demektir [32].

Ses tanımayla ilgili olarak ilk ciddi girişim yaklaşık 55 yıl önce başladı. Dreyfus-Graf 1950'de The Acoustical Society Of America dergisinde yayınlanan "Sohographand Sound Mechanics" isimli çalışmasında kendi geliştirdiği "stenosonograph" ile ses sinyalinin enerjisine göre sesleri CRT ekran üzerinde ayırmıştır [33]. Sesli ifade tanımayla ilgili diğer bir çalışma 1956 yılında RCA' dan Olsen ve Belar tarafından yapılan "Phonetic Typewriter" isimli çalışmadır. Bu çalışmada tek heceli 10 sözcükten oluşan kelime haznesi bulunan ve %98 oranında tanıma yüzdesine sahip, söylenen kelimeleri yazan bir makine geliştirilmiştir [34]. Olsen ve Belar 1961 yılında "Phonetic TypewriterIII" isimli çalışmalarında bu makineyi biraz daha geliştirerek sözcük sayısını 100'e çıkarmıştır [35]. Shearme ve Leach 1968 yılında "Some Experimentswith a Simple Word Recognition System" isimli makale çalışmalarında bilgisayarda örüntü tanımayı uygulamışlardır. Bu çalışmalarında 10 konuşmacı tarafından 32 kelimelik bir sözlük oluşturmuşlar ve tanıma yüzdesi %90 çıkmıştır [36]. Sesli ifade tanıma alanında yapılan diğer bir çalışma ise 1968 yılında Purton tarafından yapılmıştır. Purton "Speech recognition using autocorrelation analysis" isimli çalışmasında özilişki tekniği ile bir sistem üretmiştir. Her sesli ifadeye özilişki fonksiyonunu uygulayarak 36x30'luk bir matris üretmiştir. Bu örüntüyü en iyi eşleştirme yöntemi kullanılarak şablonla

karşılaştırmıştır. En iyi tanıma oranı %98 ile %99 arasında değişmiştir [37]. Bu tez kapsamında yapılan çalışmada da örüntü tanıma yöntemi ve özilişki fonksiyonu kullanılmaktadır. Özilişki fonksiyonu sesdeki ana sıklıklıları ortaya çıkarmakta ve gürltünün etkisini azaltmaktadır. Ayrıca yazılımın tanıma oranı da erkek sesinde en fazla % 100'dür.1970 yılının başında Carnegie Mellon Üniversitesinden mezun olan James Baker Profesör Raj Reddy'nin gözetiminde tamamen istatistiksel bir yaklaşımla Hidden Markov Model (HMM) denen anlaşılması güç olan bir matematiksel model geliştirmiştir [38]. Bu model ses tanıma alanında hala sıklıkla kullanılmaktadır. Son yıllarda akustik modeli temel alan bir çok başarılı ses tanıma sistemleri gerçekleştirilmiştir. CMU'nun SPHINX sistemi, BBN'nin BYBLOS'u ve MIT'in VOYAGER sistemi. SPHINX sistemi bunlar arasında en başarılı bilinen sistemdir. PHINX sistemi Hidden Markov Model (HMM) ve dil (niagram linguistic) modelini kullanır [10]. Sesli ifade tanıma alanında yapılan diğer bir çalışma ise Ohga ve arkadaşlarının 1982 'de "A walsh-hadamard transform LSI for speech recognition" isimli makale çalışmalarında geliştirdiği kullanıcıya bağımlı ses tanıma birimidir. Geliştirilen bu ses tanıma biriminde örüntü tanıma yöntemi ve dinamik programlama kullanılmıştır. Tanıma yüzdesi %98,8'dir [32].

Ülkemizde ise 1991 yılından bugüne kadar, konuşma kodlama, konuşmacıdan bağımsız / bağımlı ayrık ve birleştirilmiş sözcük tanıma ve metinden bağımsız konuşmacı tanıma konularında araştırma-geliştirme çalışmaları genel olarak TÜBİTAK tarafından yapılmaktadır. TÜBİTAK mobil telefon uygulamaları için sözcük tanıma birimi geliştirmiştir. Bu sistem kazalara neden olduğu gerekçesiyle kullanımına kısıtlama getirilen araç telefonlarının tuş takımı yerine mikrofon ile kullanılmasını sağlar. Yapılacak her işlem mikrofona söylenir ve gerekli işlem yapılır.

TÜBİTAK tarafından konuşmacı tanıma alanında üzerinde çalışılan konular vektör niceleme, sinir ağları, ikinci dereceden istatistikler ve karar verme yöntemleridir. Konuşmacı tanımada asıl amaç verilen sesin grup içindeki herhangi bir kişiye ait olup olmadığının bulunmasıdır. Örneğin kapı güvenlik sistemlerinde girilen ses, giriş izni olanlarla karşılaştırılır ve kapı açma ya da açmama konusunda karar verilir.

Öte yandan ses tanıma konusunda ülkemizdeki üniversitelerde çeşitli araştırmalarda bulunulmuştur. Bu araştırmalar genelde ayrık ses tanıma veya sesli harf tanıma sistemleri üzerine yoğunlaşmıştır [39]. Türkiye'de sesli ifade tanıma alanında yapılan diğer bir çalışma 1999 yılında Doğan, S. tarafından Marmara üniversitesinde yüksek lisans tezi olarak yapılmıştır. Bu tezde örüntü tanıma yöntemi ayrıntılı bir şekilde incelenmiş ve kullanıcıya bağımlı ses tanıma yazılımı üzerinde çalışılmıştır [10]. Diğer bir çalışma ise 2005 yılında Trakya üniversitesinde Aydın, Ö. tarafından hazırlanan ve yapay sinir ağlarıyla sesli ifade tanımayı içereren yüksek lisans tezidir. Bu tez kapsamında yapay sinir ağları kullanılarak sesli ifade tanıma yazılımı hazrılanmaya çalışılmış ancak çok başarılı sonuçlar elde edilememiştir [21]. Artuner, H. 1994 yılında hazırladığı "Bir Türkçe Fonem Kümeleme Sistemi tasarımı ve Gerçekleştirilmesi" isimli doktora çalışmasında ses tanımayla ilgili olan ön işlemleri ayrıntılı bir şekilde belirtmiştir. Ayrıca bu çalışmasında Türkiye Türkçesi'nin fonetik özelliklerini ayrıntılı olarak ortaya koymuştur [4].

4. SESLİ İFADE TANIMA YAZILIMININ TASARLANMASI

Sesli ifade tanımayla ilgili yazılım geliştirilirken Windows XP işletim sistemi ortamında Delphi programlama dili kullanılmıştır.

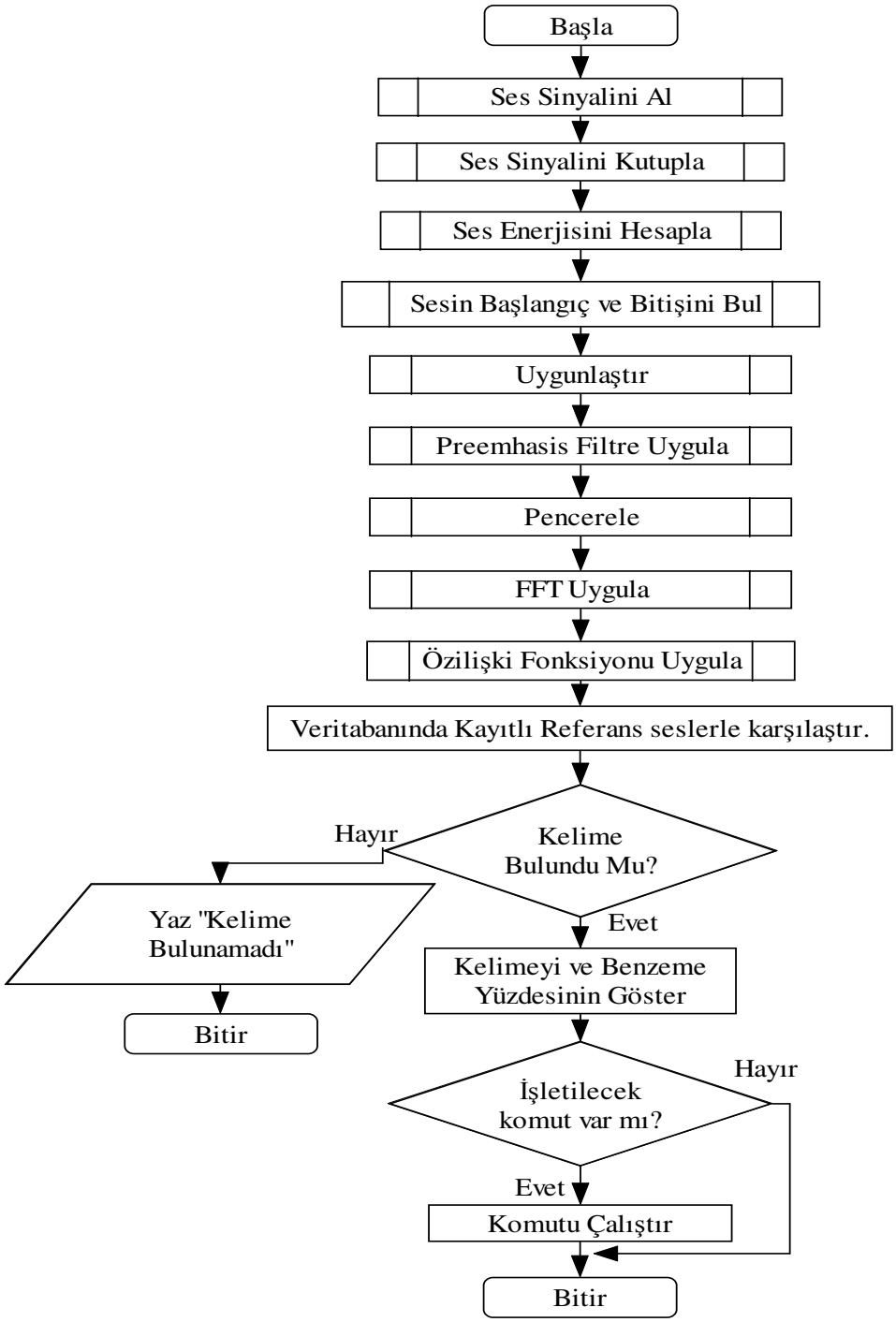
Geliştirilen yazılım aşağıdaki donanım ve yazılımlara ihtiyaç duymaktadır:

- Mikrofon,
- Ses kartı,
- Windows 98/XP işletim sistemi bulunan bir bilgisayar.

Ayrıca geliştirilen yazılımın en iyi performansı vermesi için aşağıdaki hedefler belirlenmiştir:

- Dış ortamdaki gürültü en asgariye indirilmeli ya da olmamalıdır.
- Kullanıcı kelimeleri düzgün bir şekilde telaffuz etmelidir.
- Kelimeleri seslendiren kişi ilk söylediği telaffuza yakın bir telaffuzda kelimeleri söylemelidir.
- Telaffuz sorununun etkisini en aza indirmek için veritabanındaki örnekleme artırılmalıdır.
- Bir kelimeye bağlı olarak bir program çalıştırılmak isteniyorsa Windows 98/XP ortamında çalışan bir program (Word, Excel gibi) yolu(path) doğru verilerek kelimeyle ilişkilendirilmelidir.

Bu çalışmada geliştirilen yazılım örüntü tanıma yöntemine göre sınırlı sayıdaki kelimeyi kişiye bağımlı olarak tanımaktadır. Yazılımda kullanıcı öncelikle tanınmasını istediği sözcüklerle sözlük veritabanını oluşturur. Yazılım kelime tanıma esnasında kullanıcının söylediği kelimeyle sözlük veritabanındaki tüm kelimeleri karşılaştırır ve en uygun kelimeyi bulmaya çalışır. Sözlük veritabanındaki tüm kelimelerin ayrı, ayrı benzeme oranlarını listeler ve oranı en yüksek olan kelimeyi bulunan kelime olarak gösterir. Geliştirilen yazılımın en genel akış şeması Şekil 4.1'de gösterilmektedir.



Şekil 4.1. Ses tanıma yazılımının akış şeması

Sesli ifade tanıma ile ilgili geliştirilen program aşağıdaki unitlerden oluşmaktadır.

Sessayisal : Ses sinyalinin alındığı ve sesin sayısal hale dönüştürülmesi için gerekli işlemlerin başlatıldığı unittir.

Sesyakalaayar : Ses yakalama ile ilgili ayarların yapıldığı unittir.

Sinyalisle : Kutuplama, preemhasis filtre, pencereleme, enerji hesabı, sesin başlangıç ve bitişini tespit eden alt programların bulunduğu unittir.

Sozluk : Veritabanına kaydedilen seslerle ilgili işlemlerin yapıldığı unittir.

Uyakalanan : Yakalanan ses sinyalini gösterir, ses sinyali üzerinde ön işlemlerin yapılmasını ve sesin tanınmasını sağlar.

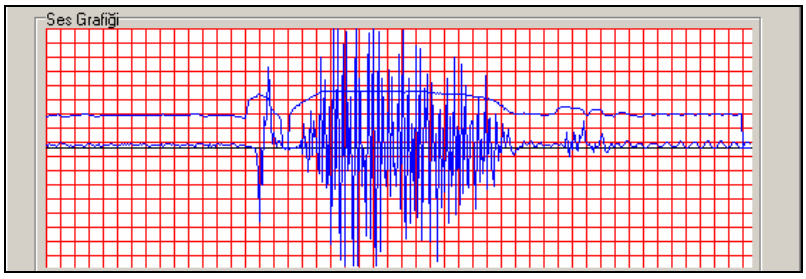
Ffthesapla : FFT hesaplama ile ilgili alt programları içerir.

Goruntu : Programın her aşamasında sesin grafiğini çizer.

Seslib : Programla ilgili değişkenlerin toplu olarak bulunduğu unittir.

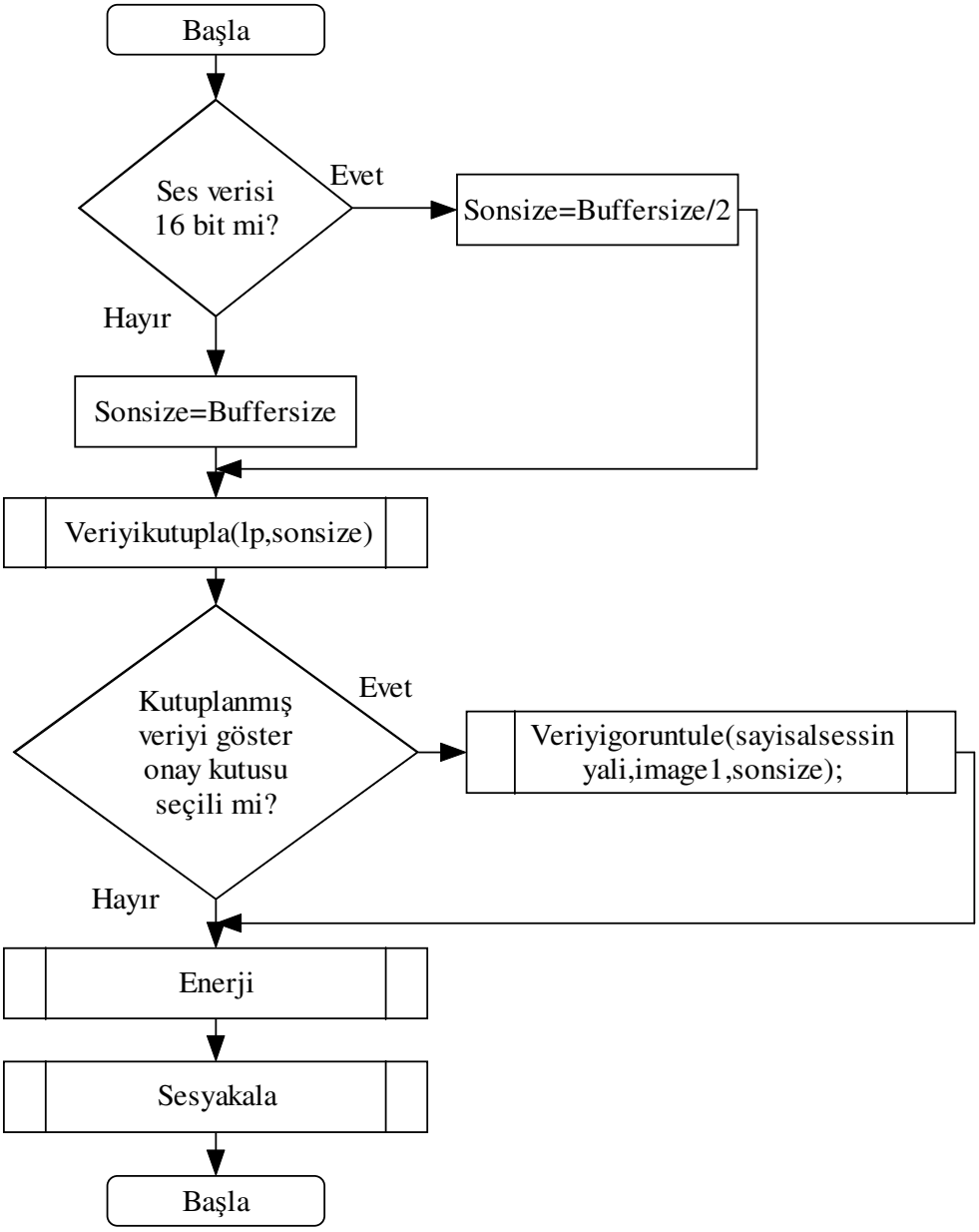
4.1. Sesin Bilgisayar Ortamına Alınması

Ses bilgisayar ortamına mikrofon aracılığıyla alınmaktadır. Sesi bilgisayar ortamına almak için audio1 isimli kompenentin record olayı kullanılmaktadır. Bilgisayar ortamına alınan ses veri tampon alanını doldurduğunda Audio1 komponentinin Record olayı devreye girer. “Audio1Record” alt yordamındaki “LP” değişkeninde ses verisi, “Buffersize” değişkeninde ise ses verisinin boyutu tutulmaktadır. Ses 16 bit ise 2 byte’lık yer kapladığı için veri boyutunu yarıya indirmek için Buffersize değişkeni 2’ye bölünmektedir. Şekil 4.2’de bilgisayar ortamına alınmış sayısal ses sinyali gösterilmektedir.



Şekil 4.2. Sayısal ses sinyali

Şekil 4.3’de sesi bilgisayar ortamına alan alt programın akış şeması gösterilmektedir.



Şekil 4.3. Sesi bilgisayar ortamına alış akış diyagramı

Program kodları aşağıdaki gibidir.

```
procedure TForm1.Audio1Record(Sender: TObject; LP, RP:pointer;BufferSize:
Word);
  begin
```

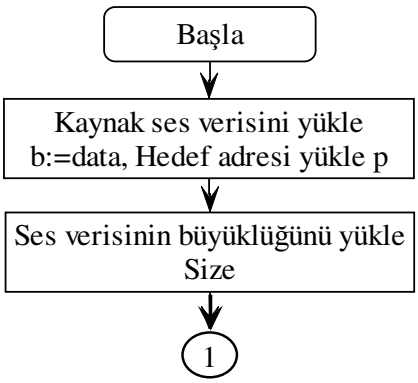
```
 defa:=defa+1;
  if defa=1 then liste1.items.add('Başla') else  liste1.items.add(inttostr(buffersize));
  if bitpersamples=16 then sonsize:=buffersize div 2 else sonsize:=buffersize;
   Veriyikutupla(lp,sonsize);
 if kutupkont.Checked=true then  Veriyigoruntule(sayisalsessinyali,image1,sonsize);
  Enerji;
   sesyakala;
end;
```

4.2. Veriyi Kutuplama

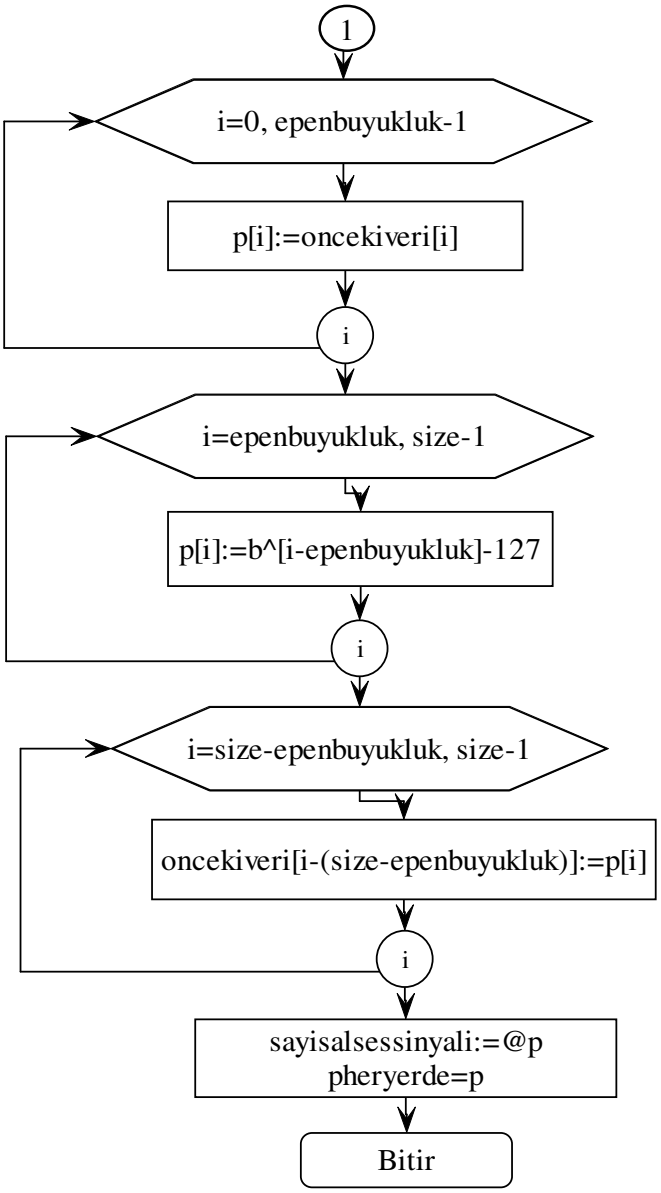
Veriyi kutuplama, ses kartından örneklenen veriyi, 8 bit veya 16 bit örneklemesinden bağımsız olarak işleyebilmek için kullanılır. Alınan veri 32 bit olarak sonraki işlemler için saklanır. Ses sinyali negatif ve pozitif ölçekte salınım yapması nedeniyle maksimum genlik seviyesinin alt yarısı negatif üst yarısı pozitif kabul edilmiştir [10].

Veriyi kutuplama alt yordamımız 8 bitlik veri için ses sinyalinden 127 değerini çıkarır. Bu sesin pozitif ve negatif salınım yapması için gereklidir. Sonuçlar 32 bitlik veriye çevrilerek 8 bit veya 16 bit olmasından bağımsız olarak işlenebilmektedir.

Şekil 4.4'de veriyi kutuplama ile ilgili akış şeması gösterilmektedir.



Şekil 4.4. Veriyi kutuplama akış diyagramı



Şekil 4.4. (Devam) Veriyi kutuplama akış diyagramı

Veriyi kutuplama ile ilgili alt yordam aşağıda verilmektedir.

```
procedure veriyikutupla(data:pointer;size:integer);
var
  p:array[0..50000] of integer;
  b:pbytedizi;  //seslibde
```

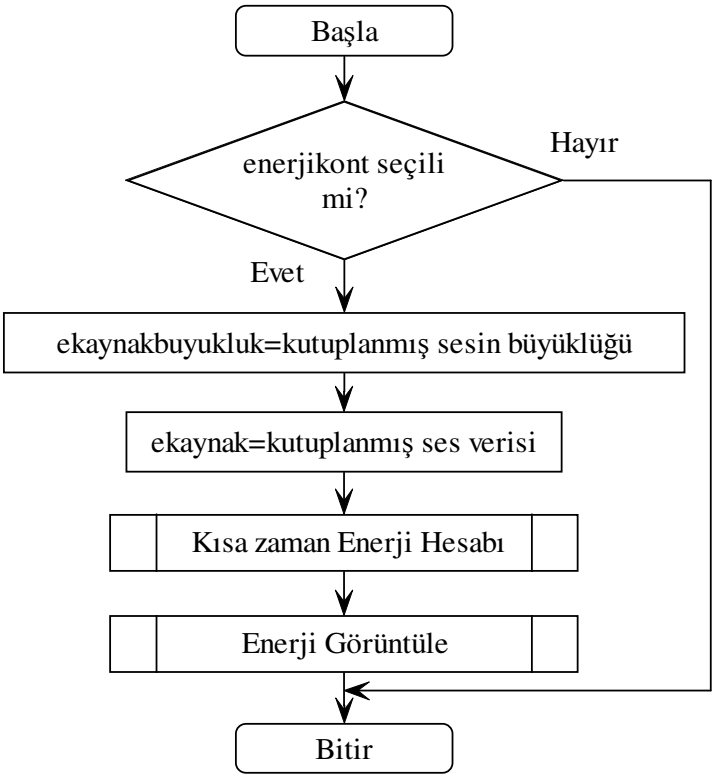
```
  i:integer;   z:integer;
begin
        b:=data;
        for i:=0 to epenbuyukluk-1 do  //Önceki veri bloğunun son
            p[i]:=oncekiveri[i];         //penceresini başa ekle.
        for i:=epenbuyukluk to size-1 do
          p[i]:=b^[i-epenbuyukluk]-127;  //yeni veriyi ekle
        for i:=size-epenbuyukluk to (size-1) do  //yeni verinin son penceresi
          oncekiveri[i-(size-epenbuyukluk)]:=p[i];  // önceki veri olarak saklanıyor.
   sayisalsessinyali:=@p;
    for z:=1 to 50000 do pheryerde[z]:=p[z];
end;
```

4.3. Sesin Enerjisini Hesaplama İşlemi

Ses sinyalinin en basit temsillerinden biri onu enerjisi ile göstermektir [39]. Ses tanıma sistemleri geliştirilmesi sırasında karşılaşılan en büyük problem, sesin başlangıç ve bitişini tespit etmektir. Kullanıcıdan alınan ses sinyali, belirlenen bir eşik enerjisini aşması ses sinyalinin başladığını ve belirli bir eşik enerjisinin altına düşüp belli bir zaman bu düşük enerjide devam etmesi ses sinyalinin bittiğini gösterir [10]. Konuşma tanıma sistemlerinde sıklıkla kullanılan sinyal değerlendirme yöntemi belirli bir zaman penceresi içinde dalganın her noktasında aldığı değerlerin karelerinin toplamları olarak hesaplanan dalga enerjisi veya yoğunluğunu bulmaktadır [3]. Bu enerji kısa zaman enerji formülü ile hesaplanmaktadır.

Enerji alt programında Audio1 nesnesinin “nosamples” özelliği vasıtasıyla ses sinyalinin büyüklüğü “ekaynakbuyukluk” değişkenine ve kutuplanmış ses sinyali de (sayisalsessinyali değişkenindeki ses verisi) ekaynak değişkenine aktarılmaktadır. Daha sonra kisazamanenerji alt programı çağrılarak kısa zaman enerji hesabı yapılmakta ve sesin enerjisi enerjigoruntule alt yordamı vasıtasıyla ekrana yansıtılmaktadır.

Programdaki Enerji alt programının akış şeması Şekil 4.5'deki gibidir.



Şekil 4.5. Ses enerji akış diyagramı

Programdaki enerji alt yordamı aşağıdaki gibidir.

```
procedure tform1.enerji;
var
temp:integer;
begin
  if enerjikont.checked then begin
    ekaynakbuyukluk:=form1.Audio1.Recorder.NoSamples;
   ekaynak:=sayisalsessinyali;
   kisazamanenerji;
   enerjigoruntule(@ehedef,form1.Image1,ekaynakbuyukluk-epencerebuyukluk);
 end;
end;
```

4.4. Kısa Zaman Enerji

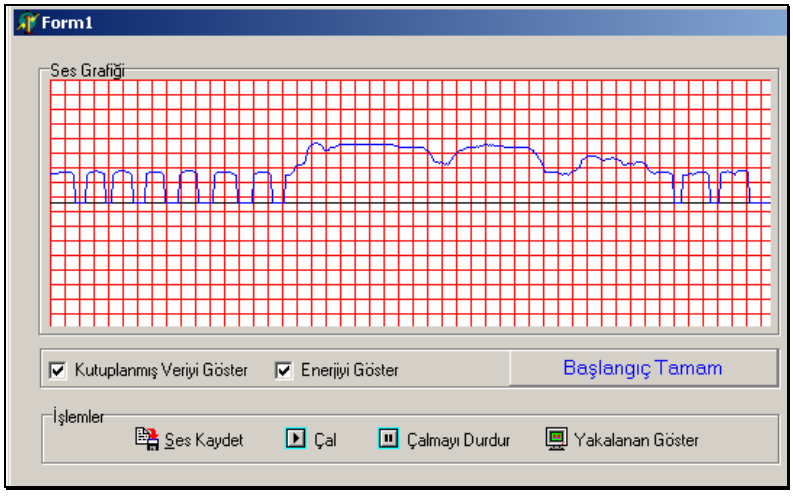
Ses sinyalinin belirli bir zamanda sahip olduğu enerjiyi hesaplamak için kullanılır. Kısa zaman enerji hesabının 3 farklı tanımlaması vardır [13].

Logaritmik Enerji(Logarithm Energy): $E = \sum_{i=1}^{N} \log x(i)^2$ (4.1)

Kare Toplam Enerji(Sum Of Square Energy: $E = \sum_{i=1}^{N} x(i)^2$ (4.2)

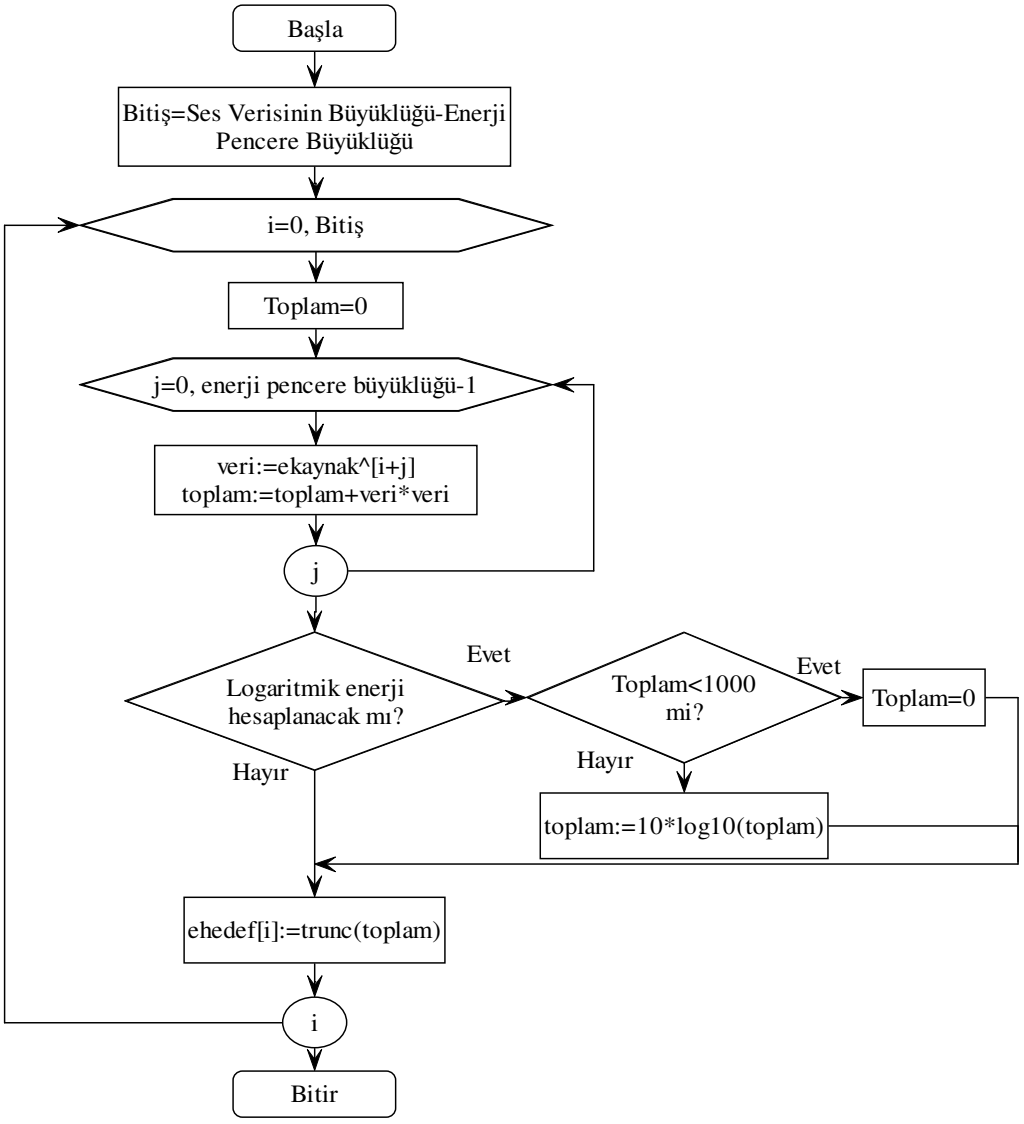
Mutlak Toplam Enerji(Sum Of Absolute Energy): $E = \sum_{i=1}^{N} |x(i)|$ (4.3)

Geliştirdiğimiz yazılımda kare toplam enerji (sum of square energy) fonksiyonu kullanılmıştır. Şekil 4.6'da kaydet kelimesinin program tarafından hesaplanan enerjisi gösterilmektedir.



Şekil 4.6. Kaydet kelimesinin enerjisi

Kısa zaman enerji hesabı ile ilgili alt programın akış şeması Şekil 4.7'de gösterilmektedir.



Şekil 4.7. Kısa zaman enerji akış şeması

Kısa zaman enerji hesabını yapan alt programda iki döngü vardır. Dış döngü değişkeni “i”, iç döngü değişkeni ise “j” dir. Dış döngüdeki elemandan itibaren (i. eleman) , enerji pencere büyüklüğü (epenbuyukluk değişkeni) kadar elemanın kareleri toplamı iç döngüde hesaplanır ve bu enerji değeri i. enerji değeri olarak “ehedef” değişkenine aktarılır. Programda enerji pencere büyüklüğü 256 olarak alınmıştır. Bir örnekle anlatmak gerekirse kısa zaman enerji hesabı yapılırken 0. elemandan itibaren 256 değerin kareleri toplamı bulunur bu 1. enerji değeridir. Daha

sonra 1. elemandan itibaren 256 değerin kareleri toplamı bulunur buda 2. enerji değeridir. Bu işlem bitiş değerine(Ses Verisinin Büyüklüğü-Enerji Pencere Büyüklüğü) kadar devam eder. Böylece tüm ses verisi için tek, tek enerji değeri hesaplanmış olur.

4.5. Sesi Yakalama İşlemi

Gürültülü ortamlarda sesin doğru olarak belirlenmesi çok önemlidir. Ses tanımayla ilgili bir program geliştirmeden önce etkili bir bölümleme yöntemine sahip olmak gerekir. Ses bitişini belirleme birçok amaç için kullanılabilir. Fakat ses bitişinin belirleme önemsiz bir işlem değildir [40]. Ses tanıma sisteminde sesin bitiş noktasının yanlış belirlenmesinin iki negatif etkisi vardır:

1. Ses sınırı yanlış belirlendiğinden tanıma hatası ortaya çıkar.
2. Sesin sınırı yanlış belirlendiğinde ses olmayan bölümlerde devreye gireceği için hesaplama işlemi artar [41].

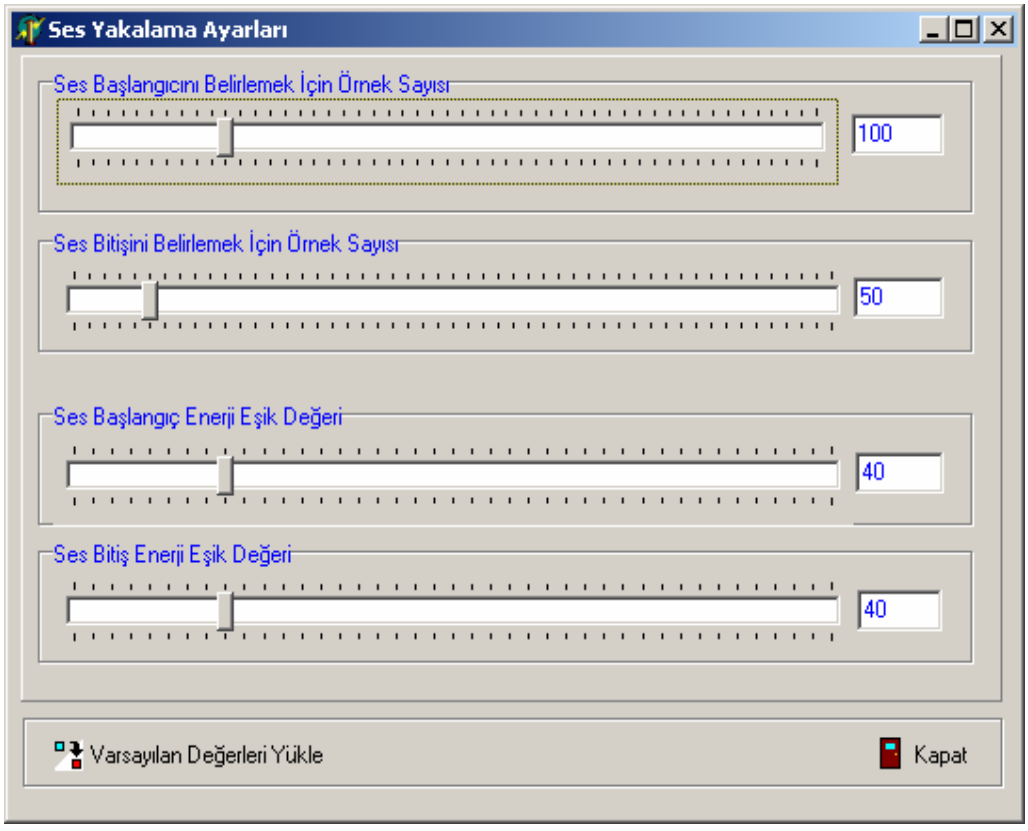
Çok gürültülü ortamlarda ses sinyalinin özelliği gürültüye çok benzerdir ve ses sinyalini gürültüden ayırmak zor bir problemdir. Ses sinyalinin bitişini belirlemek için çeşitli yöntemler vardır. Kelimenin sınırlarını belirlemek için en yaygın, esnek ve doğru yöntem ses sinyalin enerjisini kullanmaktır [40].

4.5.1. Sesin başlangıcını tespit etme işlemi

Sesin başlangıcını tespit etmek için hesaplanan enerji değerine bakmak gerekir. Programda ses başlangıcının kabulü için minimum enerji eşik değeri ve bu eşik değerinin kaç örnek boyunca devam edeceği belirlenmelidir. Bu işlem program içinde "seslib" unit'indeki "sesyakalaorneksayisi" ve "sesyakalaesikdeger" değişkenlerine değer atanarak yapılmaktadır. Programda örnek sayısı 100 eşik değer ise 40 olarak belirlenmiştir. En iyi değerler deneme yanılma yoluyla bulunabilir. Program belirli bir noktadan başlayarak enerji değerlerine bakar ve ard arda 100 enerji değeri 40 seviyesinin üzerinde ise ses başlangıcı bulunmuş demektir. Eğer

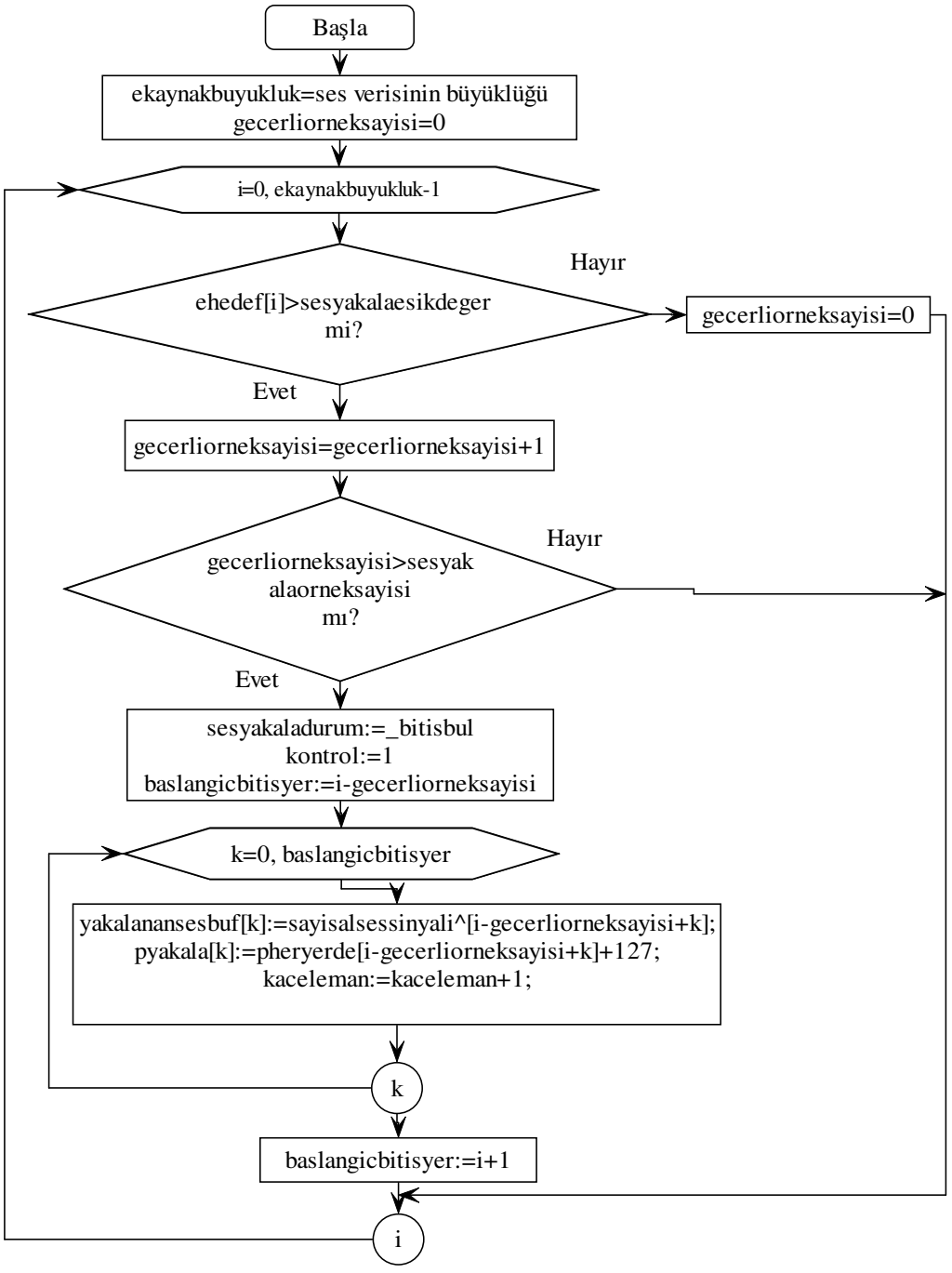
arada bir değer bu 40 seviyesinin altında ise tüm değerler sıfırlanır ve bir sonraki noktadan işlem tekrarlanır. Enerji değerlerimiz kısa zaman enerji alt programında "ehedef" değişkenine aktarıldığı için ses başlangıcı bitişi bulunurken kaynak olarak bu değişken kullanılır. Ses başlangıcı tespit edildikten sonra ses bitişini bulmak için "sesbitara" alt programı çağrılır.

Ses başlangıç ve bitişinin tespitiyle ilgili parametreler ses yakalama ayarları penceresinden yapılmaktadır. Bu pencere Şekil 4.8'de gösterilmektedir.



Şekil 4.8. Ses yakalama ayarları penceresi

Ses başlangıcını bulan alt programın akış şeması Şekil 4.9'da gösterilmektedir.



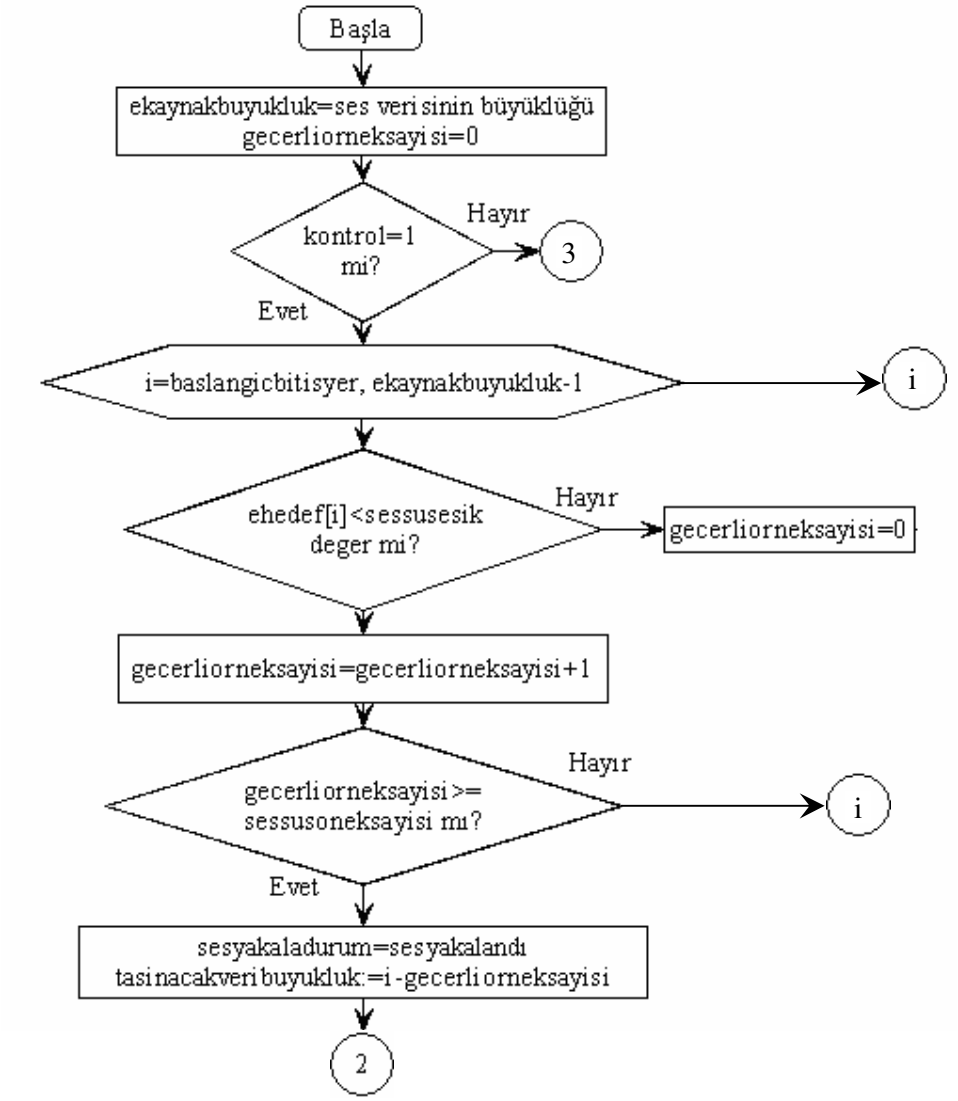
Şekil 4.9. Ses başlangıç akış şeması

4.5.2. Sesin bitişini tespit etme işlemi

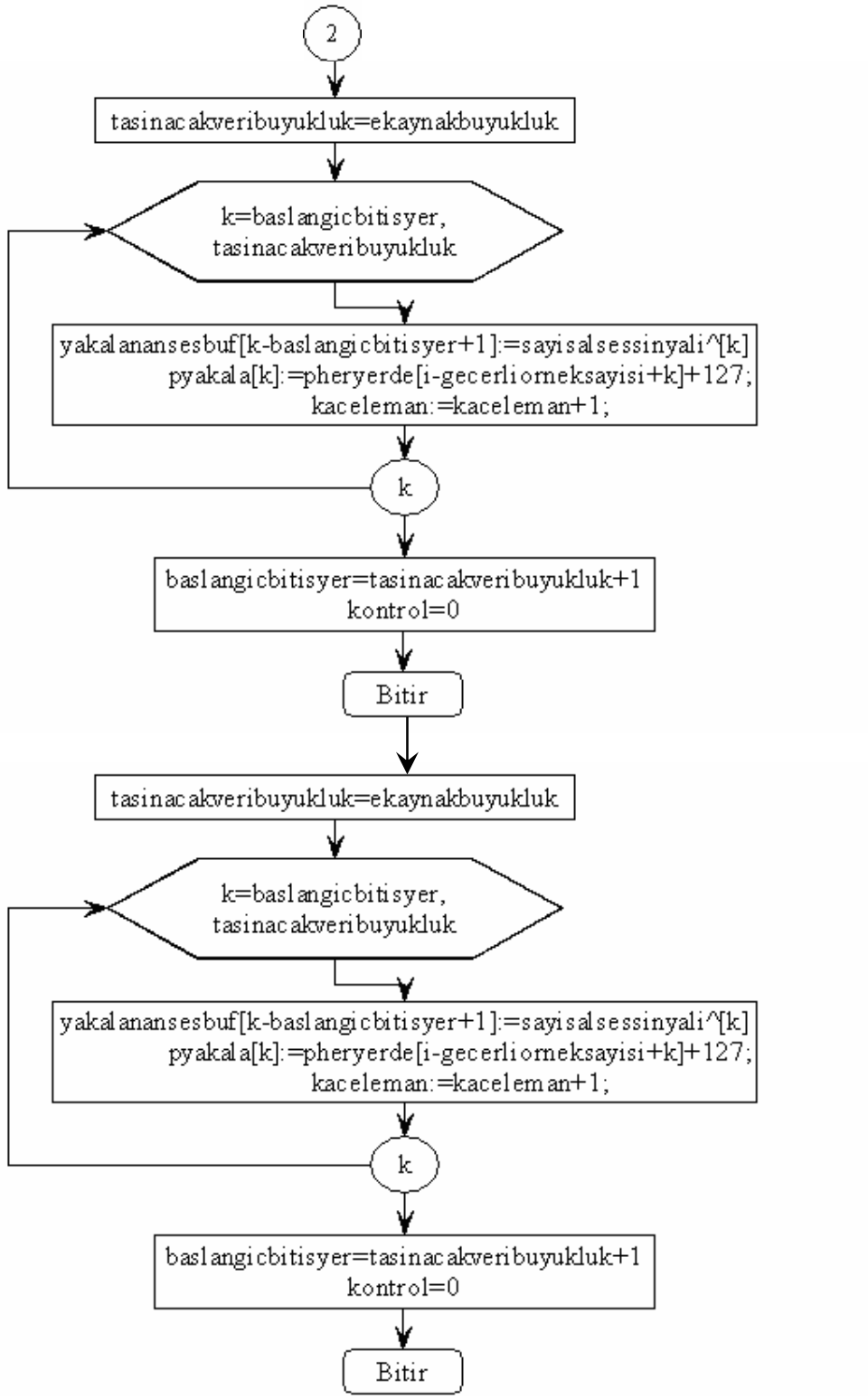
Ses bitişi bulunurken yine enerji değerlerine bakılır. Enerji değerinin program içinde

belirlenen örnek sayısı kadar yine program içinde belirlenen maksimum enerji eşik değerinin altına düşmesi sesin bittiğini belirtir. Bu iki parametre program içinde "sessusorneksayisi", "sessusesikdeger" değişkenlerine 50 ve 40 değerleri aktarılarak tanımlanmaktadır. Yani 50 örnek boyunca enerji 40 seviyesinin altına düşerse ses bitişi tespit edilmiş demektir.

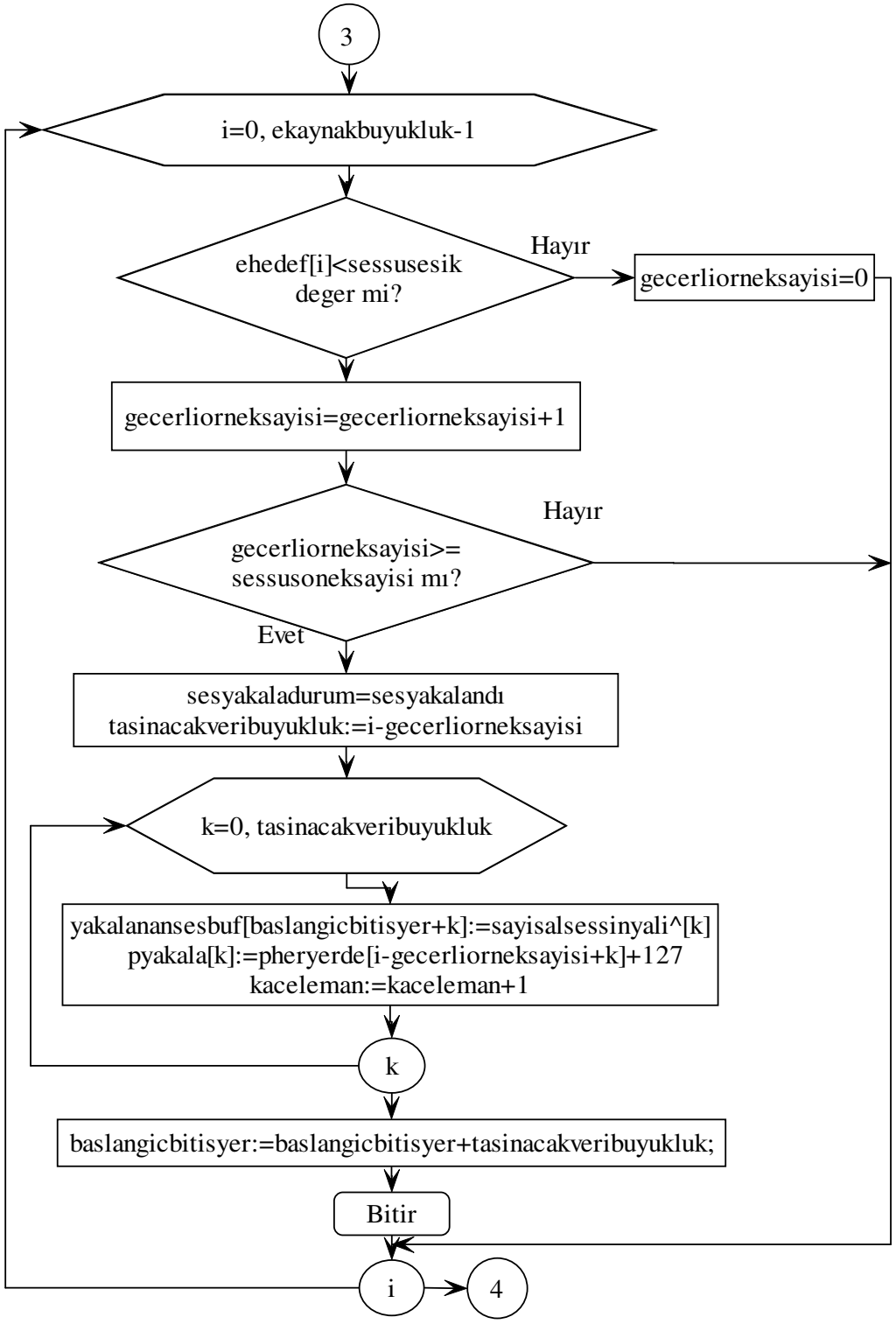
Ses bitişini tespit eden "sesbitara" alt programının akış şeması Şekil 4.10'da gösterilmektedir.



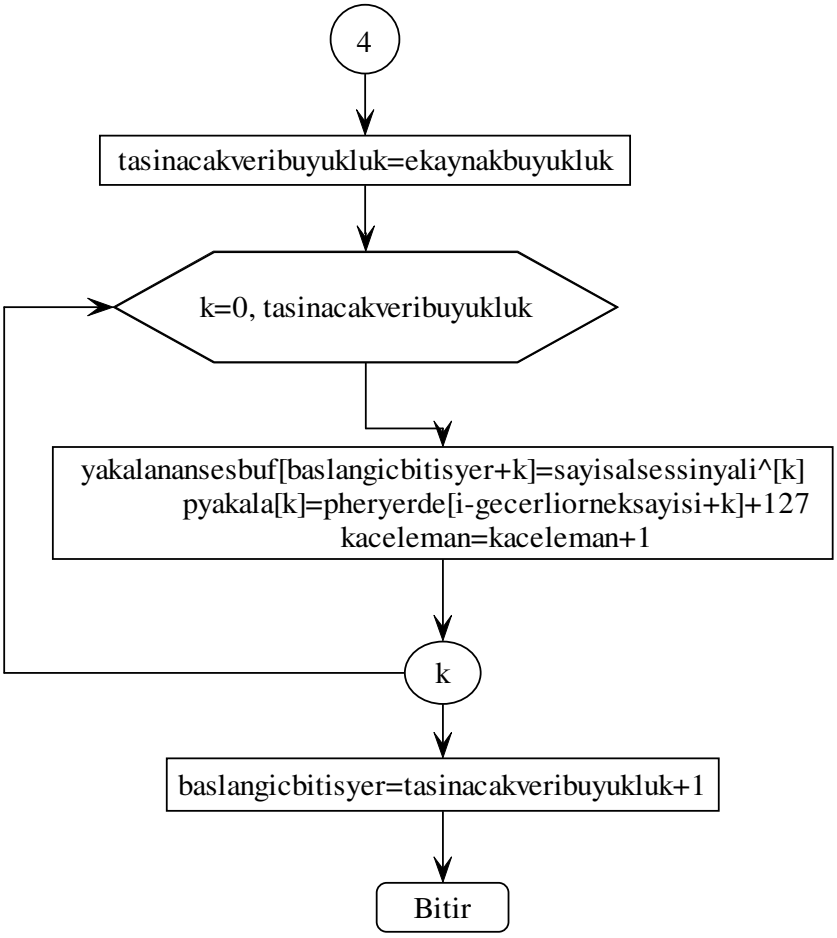
Şekil 4.10. Ses bitiş akış şeması



Şekil 4.10. (Devam) Ses bitiş akış şeması



Şekil 4.10. (Devam) Ses bitiş akış şeması



Şekil 4.10. (Devam) Ses bitiş akış şeması

4.6. Preemhasis Filtre

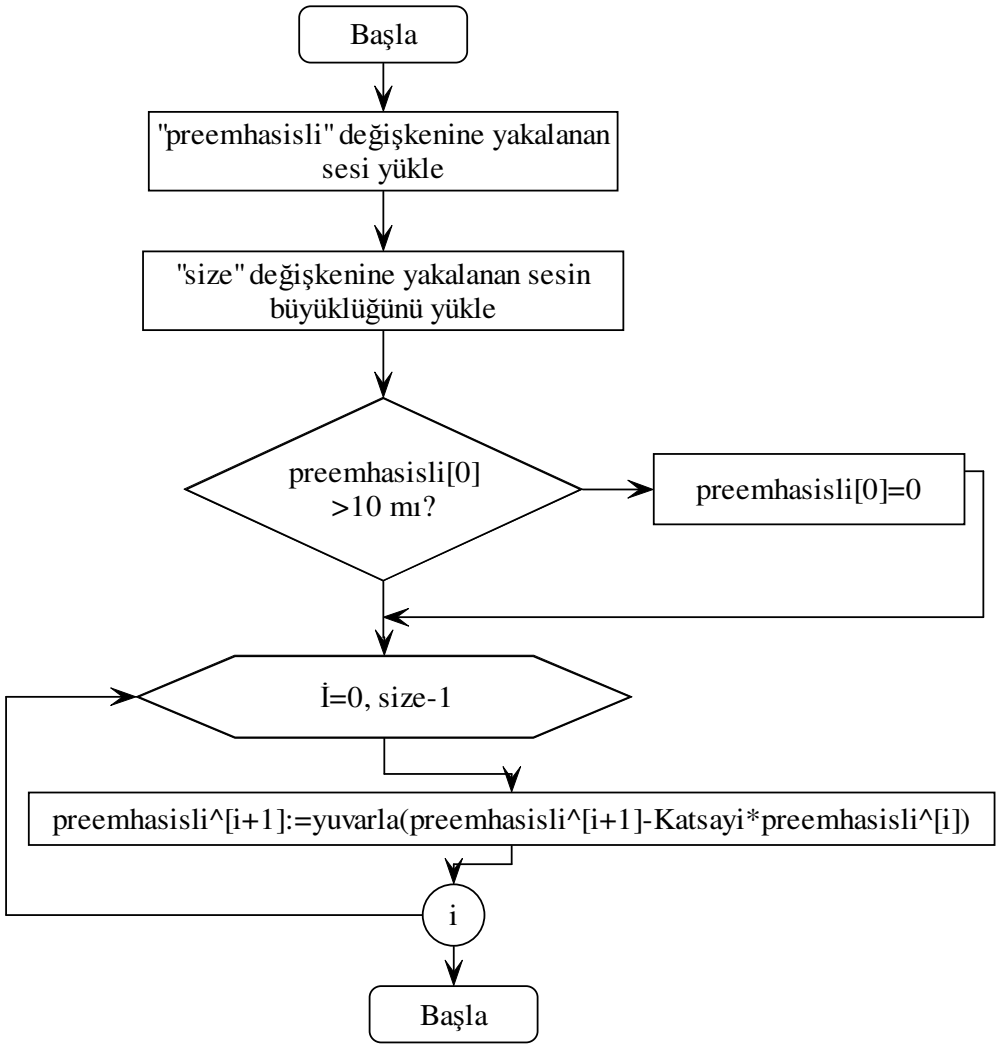
Sesli ifadeler düşük sıklıklar için yüksek enerji değerlerine sahip olmaktadır. Düşük sıklıktaki bileşenlerin yüksek sıklıktakileri maskelememesi için preemhasis adı verilen filtre kullanılmaktadır [4].

Genellikle preemhasis filtre Eş. 4.4 deki bağıntıyla ifade edilir.

$$P(z)=1-\mu z^{-1} \quad (4.4)$$

μ katsayısı 0,9 ile 0,95 arasında değişmektedir [2]. z^{-1} ifadesi standart gecikmeyi temsil eder [10].

Preemhasis filtreyi uygulayan “preemhasisfiltre” alt programının akış şeması Şekil 4.11’de gösterilmektedir.



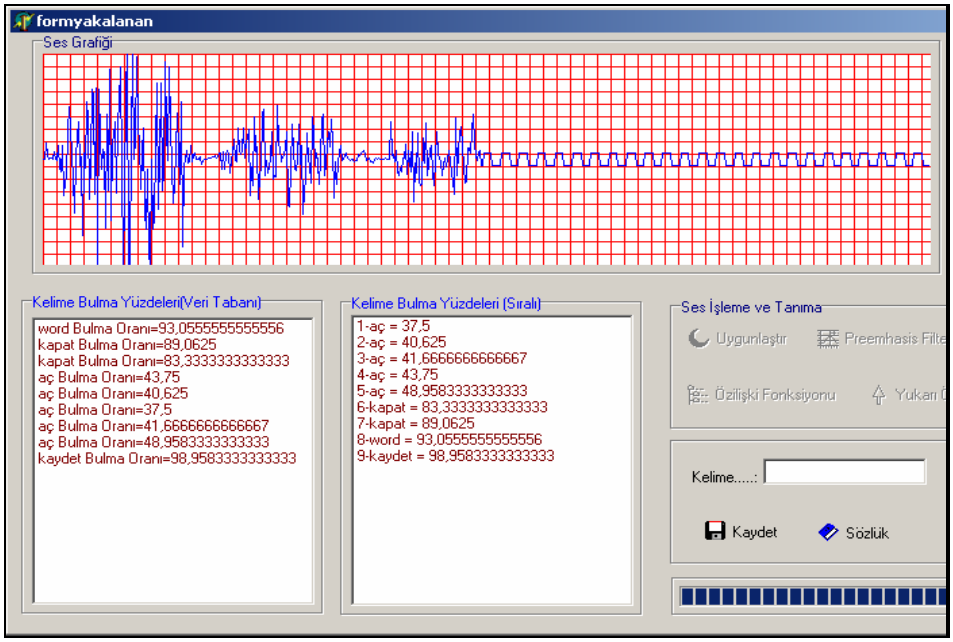
Şekil 4.11. Preemhasis filtre akış şeması

Preemhasis filtreyi uygulayan “preemhasisfiltre” alt programının kaynak kodları aşağıda gösterilmektedir.

```
Procedure preemhasisfiltre(veri:pointer;size:integer;katsayi:real);
var
```

```
preemhasisli:pointerintegerdizi;
i:integer;
begin   preemhasisli:=veri;                                    //sayısallaştırılmış ses verisi
   if preemhasisli^[0]>10 then preemhasisli^[0]:=0; //ilk değer çok büyükse
formyakalanan.progressbar1.Position:=0;  formyakalanan.progressbar1.max:=size;
   for i:=0 to size-1 do begin
      preemhasisli^[i+1]:=round(preemhasisli^[i+1]-Katsayi*preemhasisli^[i]);
      formyakalanan.progressbar1.Position:=i;   end; end;
```

Preemhasisli pointer dizi değişkeni Preemhasis filtre uygulanmış ses verisini tutar. Preemhasis filtre uygulanırken öncelikle sayısal ses verisinin ilk elemanı büyükse preemhasisli^[0]:=0; satırı ile sıfırlanır. Daha sonra bir For döngüsü ile sayısal ses verisinin o anki örnek verisinden (preemhasisli^[i]), bir önceki örnek verisi (preemhasisli^[i-1]) katsayı ile çarpılarak çıkarılır ve o anki örnek verinin preemhasis filtre uygulanmış hali elde edilir. Bu işlem örnek sayısının 1eksiğine kadar devam eder. Böylece sayısal ses örneğinin tümüne filtre uygulanmış olur. Kaydet kelimesine Preemhasis filtreleme uygulanmış ses sinyali Şekil 4.12'de görülmektedir.



Şekil 4.12. Preemhasis filtre uygulanmış ses sinyali

4.7. Pencereleme

Yakalanan ses verisinin Fourier analizi ile işlenmesi için pencerelenmesi yani belli bir kısmının alınması gerekir. Alınan bu kısım sesin en küçük anlam ifade eden kısmı ve Fourier analizinin kısa zamanda hesaplanacağı kadar olmalıdır. Genelde bu süre 30 ms dir [10].

Sayısallaştırılan sinyaller bilgisayar ortamında ikili sözcükler olarak tutulurlar. Sinyal işleme aşamasında, genellikle sesli ifadeleri temsil eden sözcüklerin tümünü bir seferde ele alma olanağı bulunmaz. Bu nedenle, bütünün anlamlı uzunlukta parçalara bölünmesi gerekir. Zira FFT (Fast Fourier Transformation) gibi işlemlere ilişkin algoritmalar belirli uzunlukta (belirli sayıda sözcükten oluşan) veri kümeleri üzerinde çalışabilmektedir. Bir seferde işleme alınacak sözcük kümesi, sinyal içinde bir pencere (window) olarak tanımlanır. Sözcükler örnekleme sonucu elde edilen değerler olduklarından sözcük sayısının örnekleme periyodu ile çarpılması ile bir zaman birimi elde edilir. Bu nedenle pencere, bir zaman dilimi olarak da düşünür. Sesli ifadeleri belirli uzunluktaki sözcüklere ayırma işlemi ise pencereleme (windowing) olarak bilinir [4].

Preemhasis filtrelemeden sonra ses sinyali bir seri pencereye bölünür. Pencereleme fikri insanın ses sisteminden çıkan sesin farklı özelliklere sahip olmasından dolayı ortaya çıkmıştır. Ses sinyalinin istatistiksel olarak 10-15 ms aralıklarla durağan olduğu varsayılmaktadır. Pencereleme sesi düzenli aralıklara böler [9].

Pencere aralığının seçiminde genellikle üç faktör rol oynar.

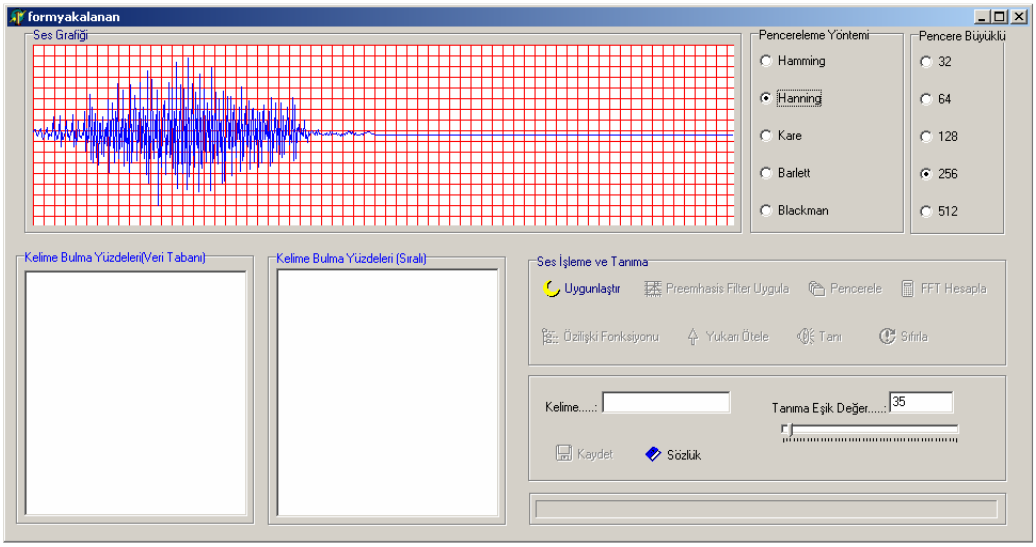
- Pencere aralığı öyle seçilmelidir ki, sesin özellikleri bu aralıkta çok önemli değişikliklere uğramamalıdır.
- Pencerenin boyu ses örneklerinden istenilen parametrenin elde edilmesine yetecek kadar uzun olmalıdır.
- Birbirini takip eden pencereler, sesin bazı bölümlerini atlayacak kadar kısa olmamalıdır. Çünkü analiz periyodik olarak tüm sinyal boyunca yapılır. Atlanılan

bölümler ses parametrelerinin yanlış olarak temsil edilmesine neden olur.

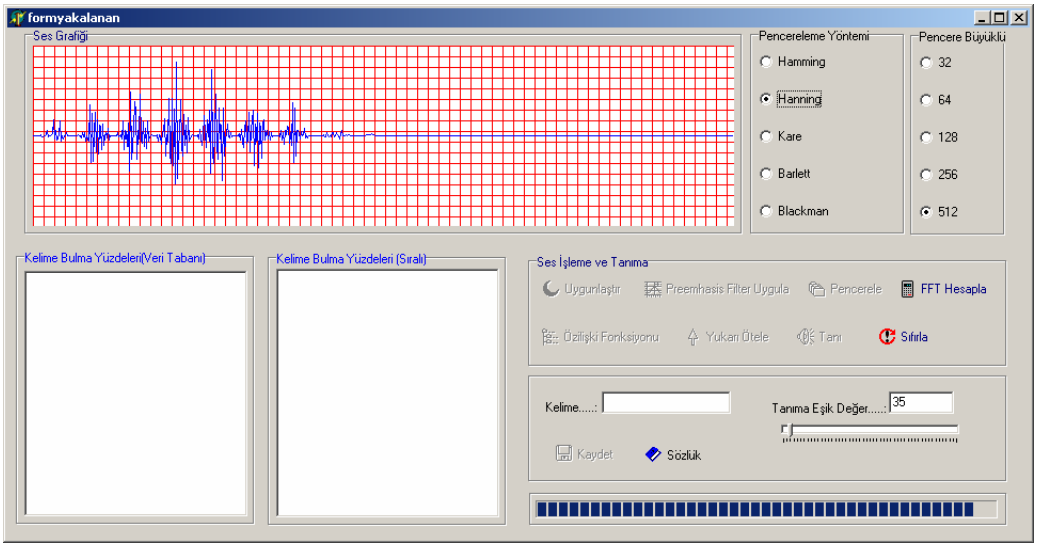
Pencerelemenin anlamı, y(n) ses sinyalinin sonlu süreli bir W(n) ile çarpılmasıdır. Bu w(n) penceresi, pencerenin şekli ile ağırlıklandırılmış bir dizi ses örneği elde eder. Fakat pencerelerin uzunluğunun sonlu olması hesapları basitleştirir [39].

Pencereleme işlemi, bir pencere bir öncekinin bitme noktasında başlatılarak örtüşmesiz yapılabildiği gibi daha geride başlatılarak pencereler arası örtüşme sağlanabilir. Bu sayede işaretteki kesin sınırlar yumuşatılabilir [42].

Geliştirilen programda pencereleme fonksiyonlarından biri uygulanmadan önce veri boyutu pencere büyüklüğüne tam bölünerek pencere sayısı hesaplanmaktadır. Ancak veri boyutu pencere büyüklüğüne tam bölünmediği zaman artan veriler ortaya çıkmaktadır. Bunların da hesaba girmesi için programda bu artan verilerden yeni bir pencere yapılmaktadır ve pencerenin boş kalan verilerine 0 değeri yerleştirilmektedir. Bu işlemden sonra kullanıcının seçimine göre Kare, Blackman, Hanning, Hamming, Barlett pencerelerinden biri uygulanır.



Şekil 4.13. Sayısallaştırılmış ses sinyali



Şekil 4.14. Hanning pencereleme uygulanmış ses sinyali

4.7.1. Kare pencereleme

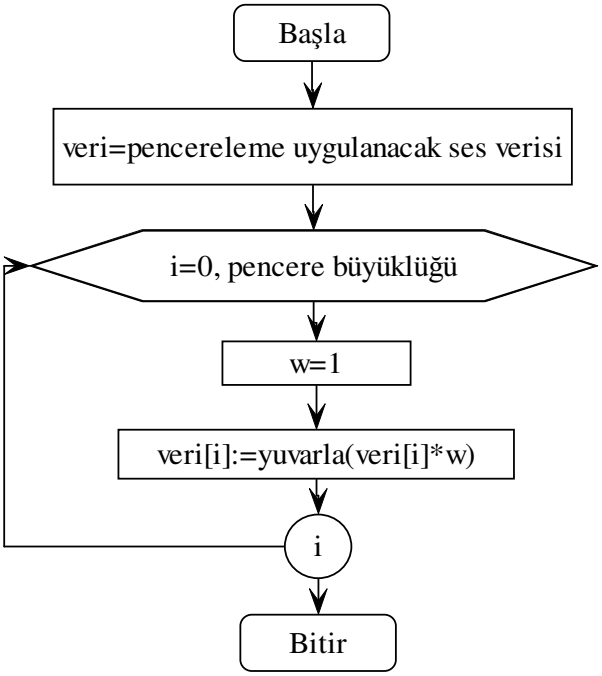
Kare pencereleme fonksiyonunun formülü [43]

$$w(n) = \begin{cases} 1 \quad ; \quad 0 \le n \le N-1 & (4.5) \\ 0, \quad ; \quad \text{ortherwise} & (4.6) \end{cases}$$

Kare pencereleme alt programında w=1 alınır ve ilgili penceredeki tüm sayısal ses verileri w ile çarpılır.

Kare pencerelemede aslında w=1 olduğu için çok fazla değişiklik olmamaktadır. Çünkü 1 etkisiz elemandır.

Kare pencereleme işlemini gerçekleştiren alt programın akış şeması Şekil 4.15'de gösterilmektedir.



Şekil 4.15. Kare pencereleme akış şeması

Kare pencereleme işlemini gerçekleştiren alt programın kaynak kodları aşağıda gösterilmektedir.

```
procedure rectang(data:pointer;windowsize:integer);
 var
 veri:pointerintegerdizi;
 i:integer;
 w:real;
  begin
        veri:=data;
        for i:=0 to windowsize-1 do begin
          w:=1;
          veri^[i]:=round(veri^[i]*w)//kare pencereleme fonksiyonunu uygula
        end;///for
  end;//procedure
```

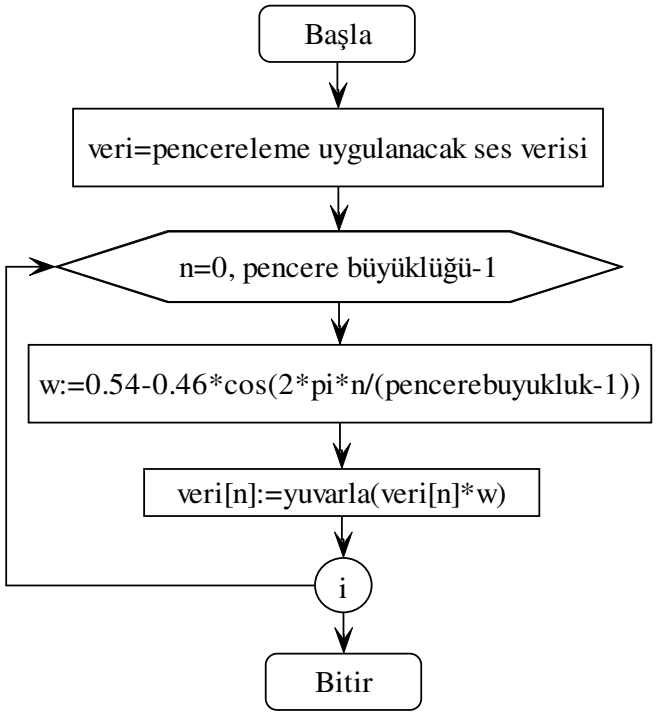
4.7.2. Hamming pencereleme

Hamming pencerelemenin fonksiyonun formülü aşağıdaki gibidir [44].

$$w[n] = \begin{cases} 0{,}54 - 0{,}46\cos\dfrac{2\Pi n}{N}, 0 \le n \le N-1 & (4.7) \\ 0, \quad \text{otherwise} & (4.8) \end{cases}$$

Hamming pencereleme fonksiyonunu uygulayan alt programda w:=0,54-0,46*cos(2*pi*n/(pencerebuyukluk-1)) satırı ile hamming pencereleme fonksiyonu hesaplanmakta ve bu fonksiyon tüm kaynak ses verisine uygulanmaktadır**.**

Hamming pencereleme işlemini gerçekleştiren alt programın akış şeması Şekil 4.16'da gösterilmektedir.



Şekil 4.16. Hamming pencereleme akış şeması

Hamming pencereleme işlemini gerçekleştiren alt programın kaynak kodları aşağıda gösterilmektedir.

```
procedure Hamming(data:pointer;pencerebuyukluk:integer);
  var
  veri:pointerintegerdizi;
  n:integer;
  w:real;
begin
    veri:=data;
    for n:=0 to pencerebuyukluk-1 do begin
      w:=0.54-0.46*cos(2*pi*n/(pencerebuyukluk-1));//hamming fonksiyonu hesapla
      veri^[n]:=round(veri^[n]*w);   //hamming fonksiyonu uygula
    end;
end;
```

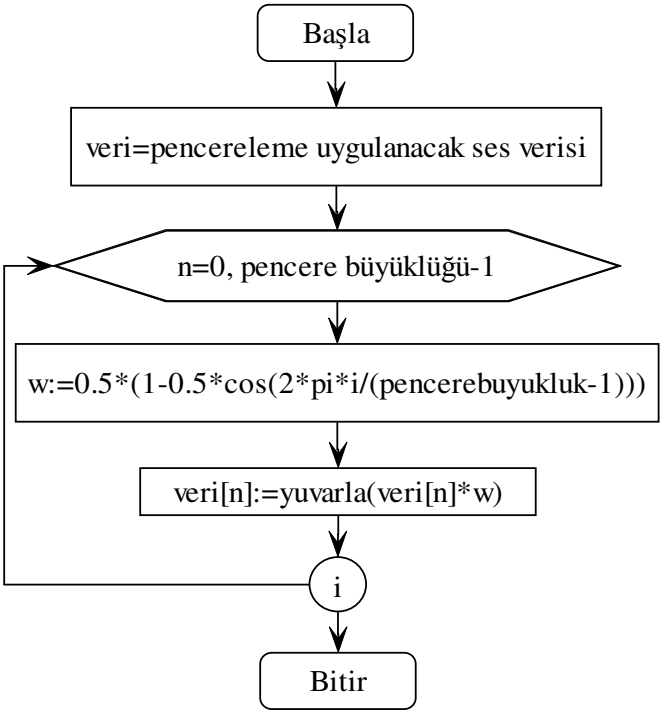
4.7.3. Hanning pencereleme

Hanning pencereleme fonksiyonun formülü aşağıdaki gibidir [45].

$$w[n]=0,5(1-\cos(2\Pi n/(N-1))) \quad (4.9)$$

Hanning pencereleme fonksiyonunu uygulayan alt programda w:=0,5*(1-0,5*cos(2*pi*i/(pencerebuyukluk-1))) satırı ile hanning pencereleme fonksiyonu hesaplanmakta ve bu fonksiyon tüm kaynak ses verisine uygulanmaktadır.

Hanning pencereleme işlemini gerçekleştiren alt programın akış şeması Şekil 4.17'de gösterilmektedir.



Şekil 4.17. Hanning pencereleme akış şeması

Hanning pencereleme işlemini gerçekleştiren alt programın kaynak kodları aşağıda gösterilmektedir.

```
procedure Hanning(data:pointer;pencerebuyukluk:integer);
  var
  veri:pointerintegerdizi;
  n:integer;
  w:real;
begin
    veri:=data;
    for n:=0 to pencerebuyukluk-1 do begin
w:=0.5*(1-0.5*cos(2*pi*i/(pencerebuyukluk-1)));
       veri^[n]:=round(veri^[n]*w);
    end;
end;
```

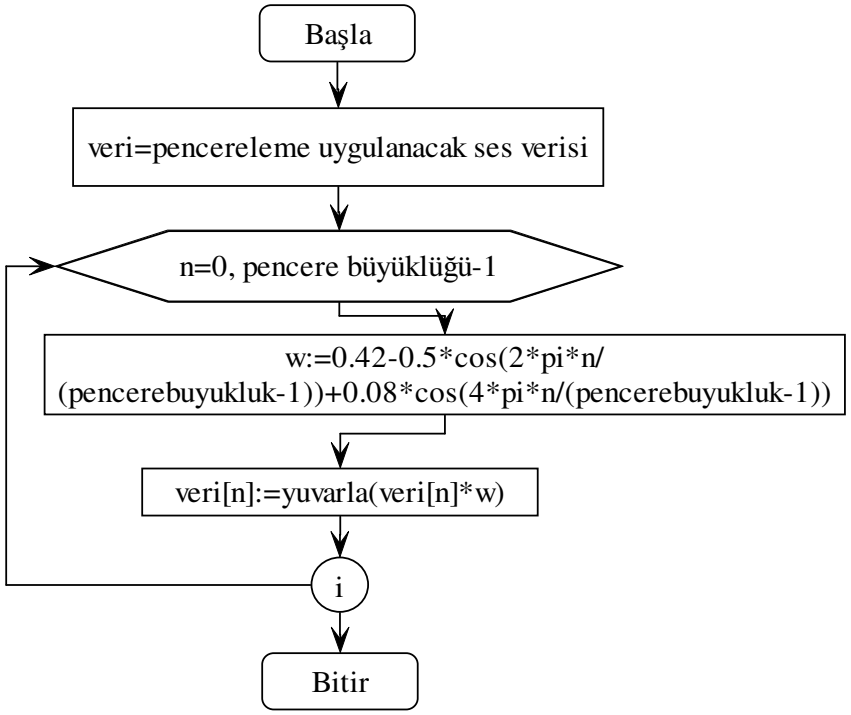
4.7.4. Blackman pencereleme

Blackman pencereleme fonksiyonun formülü aşağıdaki gibidir [46].

$$w(n)=0,42-0,5\cos(2\Pi n/N-1)+0,08\cos(4\Pi n/N-1) \quad (4.10)$$

Blackman pencereleme fonksiyonunu uygulayan alt programda w:=0,42-0,5*cos(2*pi*n/(pencerebuyukluk-1))+0,08*cos(4*pi*n/(pencerebuyukluk-1)) satırı ile Blackman pencereleme fonksiyonu hesaplanmakta ve bu fonksiyon tüm kaynak ses verisine uygulanmaktadır.

Blackman pencereleme işlemini gerçekleştiren alt programın akış şeması Şekil 4.18'de gösterilmektedir.



Şekil 4.18. Blackman pencereleme akış şeması

Blackman pencereleme işlemini gerçekleştiren alt programın kaynak kodları aşağıda gösterilmektedir.

```
Procedure BlackMan(data:pointer;pencerebuyukluk:integer);
  var
  veri:pointerintegerdizi;
  n:integer;
  w:real;
begin
    veri:=data;
    for n:=0 to pencerebuyukluk-1 do begin
      w:=0.42-0.5*cos(2*pi*n/
       (pencerebuyukluk-1))+0.08*cos(4*pi*n/(pencerebuyukluk-1));
       veri^[n]:=round(veri^[n]*w);
    end;
end;
```

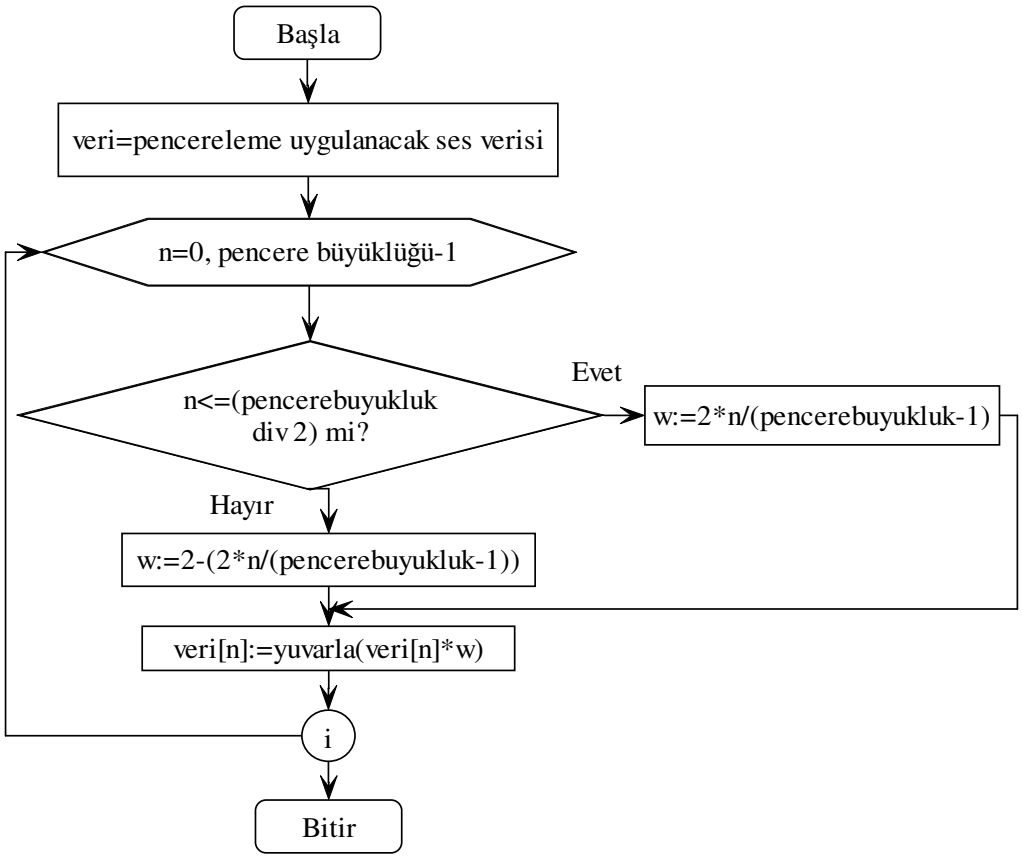
4.7.5. Barlett pencereleme

Barlett pencereleme fonksiyonun formülü aşağıdaki gibidir [43].

$$W(n)=\begin{cases} 2n/(N-1) & ; 0 \le n \le (N-1)/2 \quad (4.11) \\ 2-(2n/(N-1)) & ; (N-1)/2 \le n \le N-1 \quad (4.12) \\ 0 & ; \text{otherwise} \quad (4.13) \end{cases}$$

Barlett pencereleme fonksiyonunu uygulayan alt programda if n<=(pencerebuyukluk div 2) then w:=2*n/(pencerebuyukluk-1) else w:=2-(2*n/(pencerebuyukluk-1)) satırı ile Barlett pencereleme fonksiyonu hesaplanmakta ve bu fonksiyon tüm kaynak ses verisine uygulanmaktadır.

Barlett pencereleme işlemini gerçekleştiren alt programın akış şeması Şekil 4.19'da gösterilmektedir.



Şekil 4.19. Barlett Pencereleme akış şeması

Barlett pencereleme işlemini gerçekleştiren alt programın kaynak kodları aşağıda gösterilmektedir.

```
procedure Barlett(data:pointer;pencerebuyukluk:integer);
  var
  veri:pointerintegerdizi;
  n:integer;
  w:real;
begin
    veri:=data;
    for n:=0 to pencerebuyukluk-1 do begin
      if n<=(pencerebuyukluk div 2) then w:=2*n/(pencerebuyukluk-1)
      else w:=2-(2*n/(pencerebuyukluk-1));
```

```
      veri^[n]:=round(veri^[n]*w);
    end;
end;
```

4.8. Fast Fourier Transform (FFT)

Hızlı Fourier Dönüşümü (FFT-Fast Fourier Transform) ses sinyalini meydana geldiği frekanslara ayırır. Normal bir ses sinyali de renklerin farklı farklı renklerin birleşmesiyle oluşması gibi çeşitli frekanslardaki seslerin birleşmesi ile oluşur. Ses sinyalinin ham şekli insan aynı sözcüğü söyleyese de farklı görüntüler almaktadır. Dolayısıyla ses sinyalinin ham hali (zaman ve genlik) ile algılamak zordur. Bu nedenle çeşitli şekillerde ses sinyali işlenir ve bu şekilde tanınmaya çalışılır. FFT bu tekniklerden biridir. Aynı sözcüğü ifade eden iki ses sinyaline ham hali ile bakıldığında sinyallerin benzerliği zor görünür. Ama sinyallerin FFT'sine bakıldığında benzerliği daha kolay görünür. Ancak FFT'si alınan sinyaller de bazı ufak farklılıklar içerirler [21].

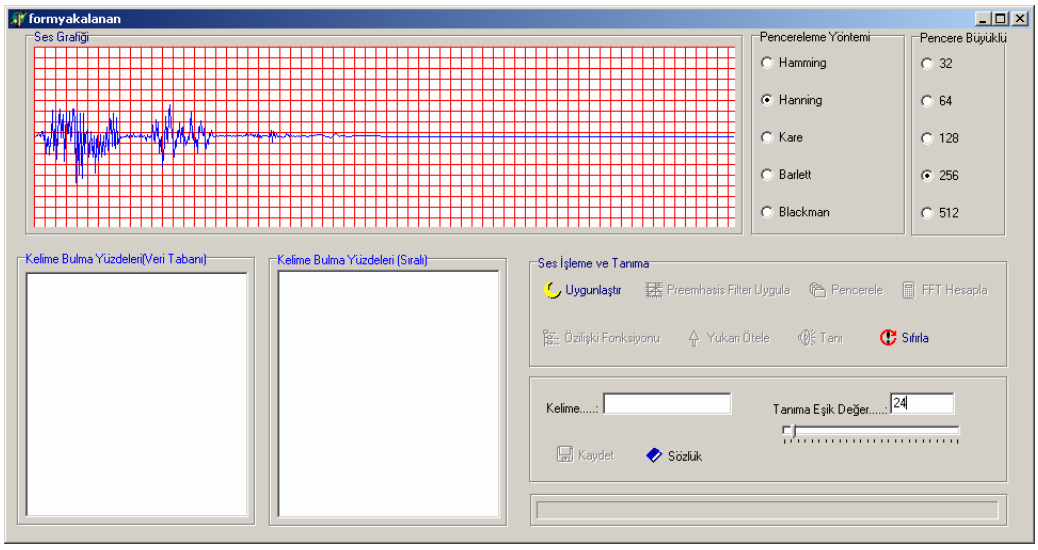
Fast Fourier Transform'un uzun ve enteresan bir tarihi vardır. Orjinali Gauss tarafından keşfedilmiş ve daha sonra Cooley ve Tukey tarafından yeniden keşfedilerek herkes tarafından bilinir hale gelmiştir. FFT, Discrete Fouriror Transform'u (DFT) hızlı bir şekilde hesaplayan bir algoritmadır [47]. DFT dijital sinyal işlemenin temel işlemlerinden biridir. DFT genellikle bir çok uygulamada kullanılan Linear Filtrelemede kullanılır. Bununla birlikte sonlu büyüklükteki ses sinyalinin tamamı için DFT'de gerekli olan hesap diğerlerinden çok fazladır [48]. FFT Discrete Fourior Transform'un(DFT) hesaplanması için etkili bir metottur. Bu metodun etkili olmasının sebebi birçok problemin çözümünde kullanılabilmesidir. Bu yüzden bu teknik çok ilgi görmüştür. FFT DFT'den daha hızlı işlem yapmaktadır. FFT N örnek için $N\log_2 N$ kadar işlem yapar. DFT ise N^2 kadar işlem yapar. Örneğin N=1024 ise FFT $N\text{Log}_2 N$=10 240 işlem yapar. DFT ise N^2=1 048 576 işlem yapar. [49]

DFT fonksiyonu Eş. 4.14 deki gibidir [50].

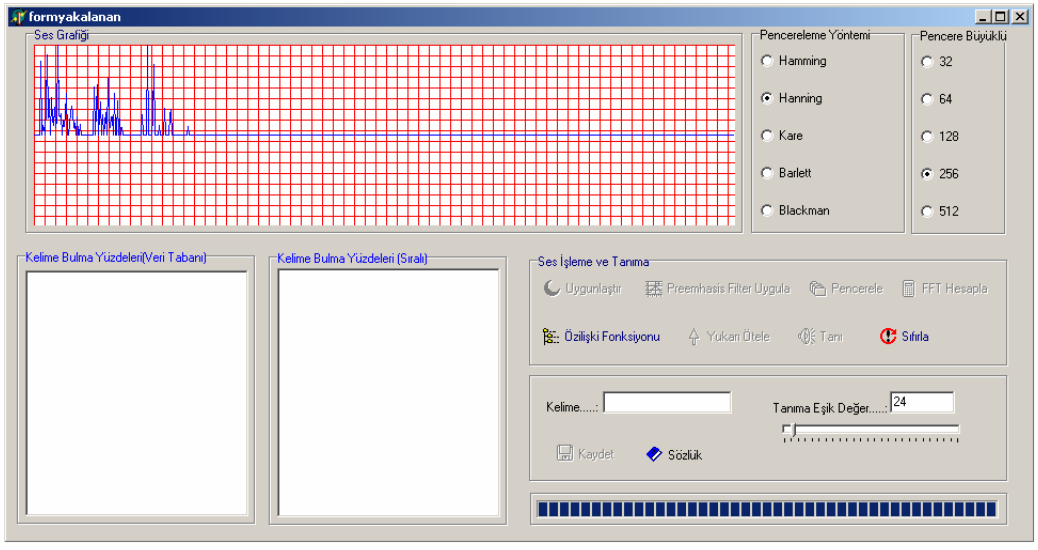
$$X(j)=\sum_{k=0}^{N-1} A(k).W^{jk}\ , \qquad j=0,1,...,N-1 \tag{4.14}$$

$$W=e^{2\Pi i/N}$$

Şekil 4.20'de işlememiş ses sinyali ve Şekil 4.21'de ise FFT uygulanmış ses sinyali gösterilmektedir.



Şekil 4.20. İşlenmemiş ses sinyali



Şekil 4.21. FFT uygulanmış ses sinyali

4.9. Özilişki Fonksiyonu (Autocorrelation Function)

FFT fonksiyonunun özilişki fonksiyonu özellikle ana sıklıkları ortaya çıkaran ve tanımayı kolaylaştıran bir fonksiyondur [10]. Gürültülü ortamlarda karakter sıklığını ortaya çıkarma sesi düzeltmek için anahtar tekniktir. Bir çok gelişmiş konuşma sistemi karakter sıklığı bilgisine ihtiyaç duyar. Bu sistemlerde karakter sıklığını çıkarmanın doğruluğu direkt olarak düzeltme işleminden sonraki sesin kalitesine bağlıdır. Genellikle karakter sıklığını çıkarma yöntemleri 3 kategoride sınıflandırılır; 1. Waveform Processing 2. Spectral Processing 3. Correlation Processing. Üçüncü kategori gürültüye karşı diğerlerine göre daha güçlüdür. Özilişki fonksiyonu (autocorrelation function) bu kategori içindedir ve gürültülü çevrelerde en iyi performansı sağlar [51].

Çeşitli özilişki uzaklıkları vardır. En çok kullanılanlar Bartlett ve Blackman-Tukey'dir. Bunlar aşağıdaki gibi tanımlanır.

X(i) gerçek veri kayıtları, i=0,1,2,…N-1, N veri boyutu olmak üzere Bartlett özilişki uzaklığı "Eş. 4.15" deki gibidir.

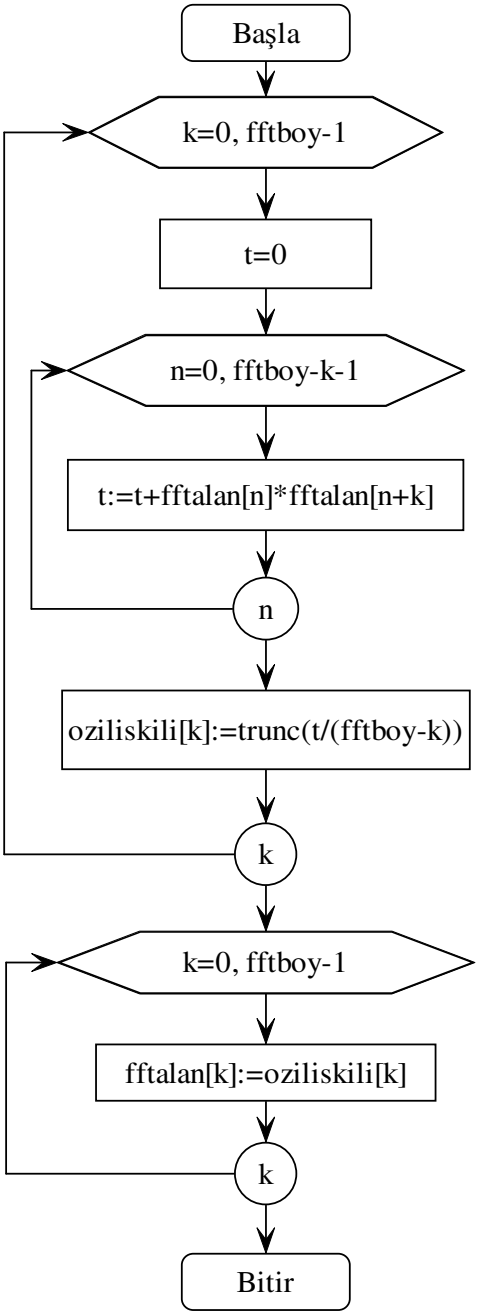
$$R(k) = (1/N) \sum_{i=0}^{N-1-k} x(i).x(i+k) \quad , \quad \text{k=0,1,…N-1} \tag{4.15}$$

Blackman-Tukey özilişki uzaklığı ise " Eş. 4.16" deki gibidir [52].

$$R(k) = (1/(N-k)) \sum_{i=0}^{N-1-k} x(i).x(i+k) \tag{4.16}$$

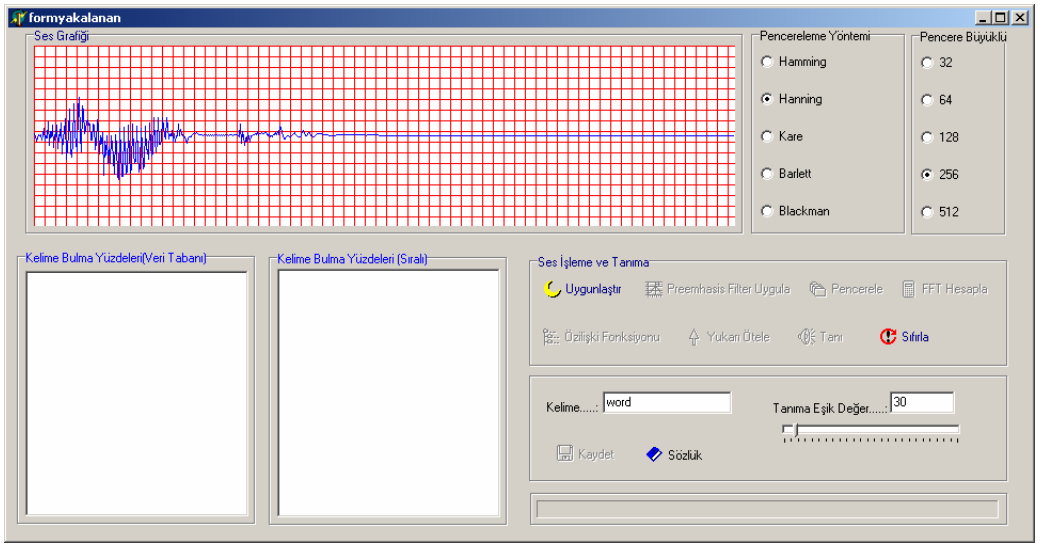
Ses tanımayla ilgili geliştirilen programda Balckman-Tukey özilişki uzaklığı kullanılmaktadır.

Özilişki fonksiyonunu uygulayan alt programın akış şeması Şekil 4.22'de gösterilmektedir.

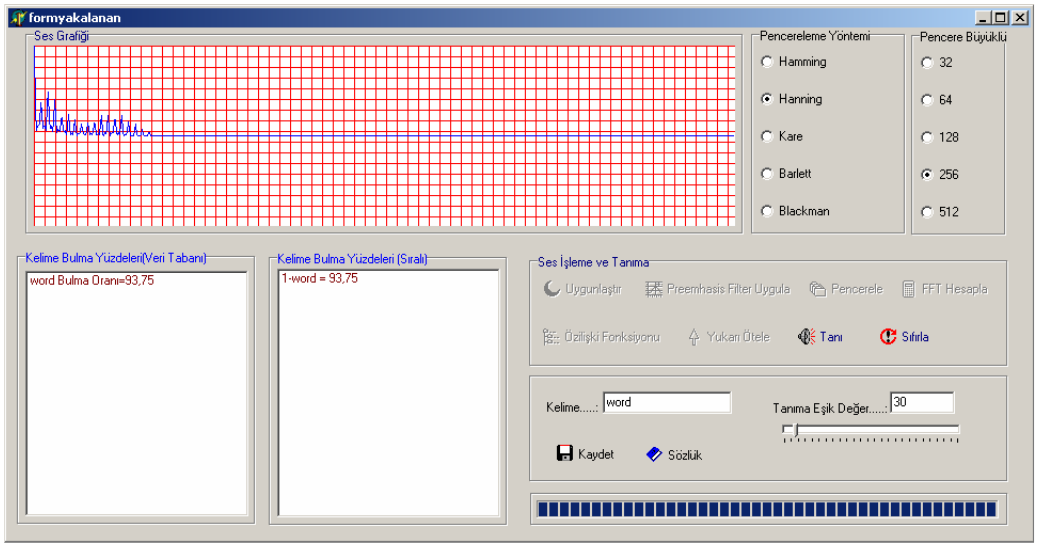


Şekil 4.22. Özilişki fonkisyonu akış şeması

Şekil 4.24'de Özilişki fonksiyonu uygulanmış ses sinyali gösterilmektedir.



Şekil 4.23. İşlenmemiş ses sinyali



Şekil 4.24. Özilişki fonksiyonu uygulanmış ses sinyali

4.10. Tanıma

Veri tabanına daha önceden işlenerek kaydedilen kaynak sinyallerle, konuşmacının seslendirmiş olduğu işlenmiş ses sinyalinin karşılaştırıldığı bölümdür. Bu bölümde veritabanındaki bütün referans sinyallerle kullanıcının seslendirdiği tanınacak ses sinyali karşılaştırılır ve en yakın referans sinyali bulunmaya çalışılır. Eğer bulma

oranı %85'den daha az ise kelime bulunamamış demektir. Programda özilişki fonksiyonu uygulanmış ses sinyali normal pencere sayısından sekiz kat daha fazla pencereye bölünerek veritabanındaki seslere yakınlığı hesaplanır.

Fark alma işlemi (delta), özellik vektörlerinin herhangi bir biçimde farklarının alınmasıdır. Fark vektörleri sesli ifadenin kısa süreli değişimlerinin açığa çıkmasını sağlayan özellik vektörleridir. Fark alma işlemi sonucu elde edilen vektörler zaman boyutunda özellik vektörlerinin benzerliğini açığa çıkarır [31]. Bu hesaplamada kullanılabilecek çeşitli uzaklık ölçüleri bulunmaktadır. Geliştirmiş olduğumuz programda öklid uzaklığı kullanılmaktadır.

4.10.1 Hamming uzaklığı

Yöntemlerin en basitidir. Bu yüzden de sıklıkla kullanılan bir yöntemdir.
$X=(x_1,x_2,\ldots.)$ ve $Y=(y_1,y_2,\ldots.)$
gibi iki vektör için Hamming uzaklığı,

$$d_H = \sum(|x_i - y_i|) \tag{4.17}$$

şeklinde tanımlanır. Bu yöntem, sıklıkla ikili sayı tabanındaki vektörlerin karşılaştırılması için kullanılır.

4.10.2. Öklid uzaklığı

En çok kullanılan yöntemdir. Dik koordinat sisteminde iki nokta arasındaki geometrik uzaklığın bulunmasıdır. Öklid uzaklığı

$$d(X,Y)_{öklid} = \sqrt{\sum_{i=1}^{n}(x_i - y_i)^2} \tag{4.18}$$

şeklinde tanımlanır. Geliştirmiş olduğumuz programda ses tanıma bölümünde benzerliği tespit etmek için öklid uzaklığı kullanılmaktadır.

4.10.3. Manhattan uzaklığı

Hamming uzaklığı yönteminin örneksel vektörler için uygulaması denebilir. Hesap yükü Öklid uzaklığı yöntemine göre çok daha azdır.

$$d_M=\sum_n |x_i - y_i| \tag{4.19}$$

4.10.4. Kare uzaklığı

Öklid uzaklığı yöntemini daha da basitleştirerek (ve böylece daha çok hata payı ekleyerek) ortaya çıkarılan bu yöntem; karşılaştırılacak her iki vektörün, karşılık düşen elemanları arasında ayrı, ayrı ölçülen farkların en büyüğü olarak tanımlanabilir: [3]

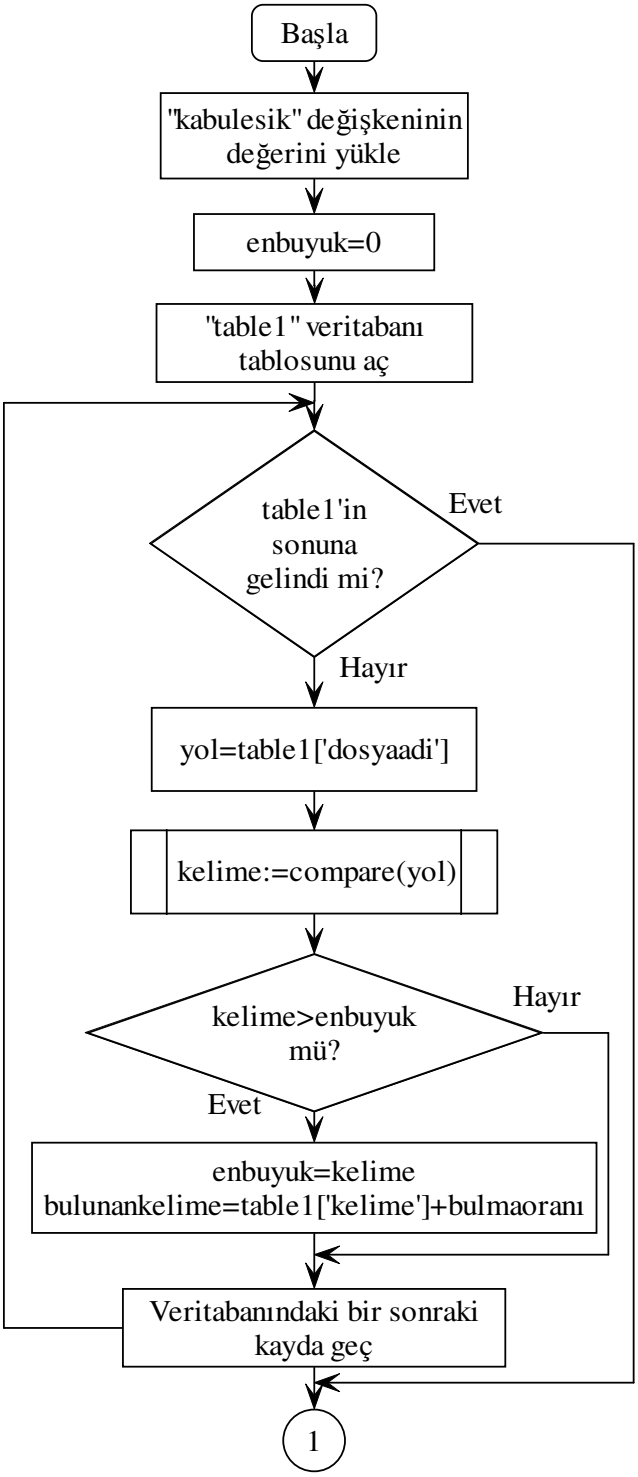
$$D_{kare=en\_yüksek} |x_i - y_i| \tag{4.20}$$

4.10.5. Cepstral uzaklık ölçümü

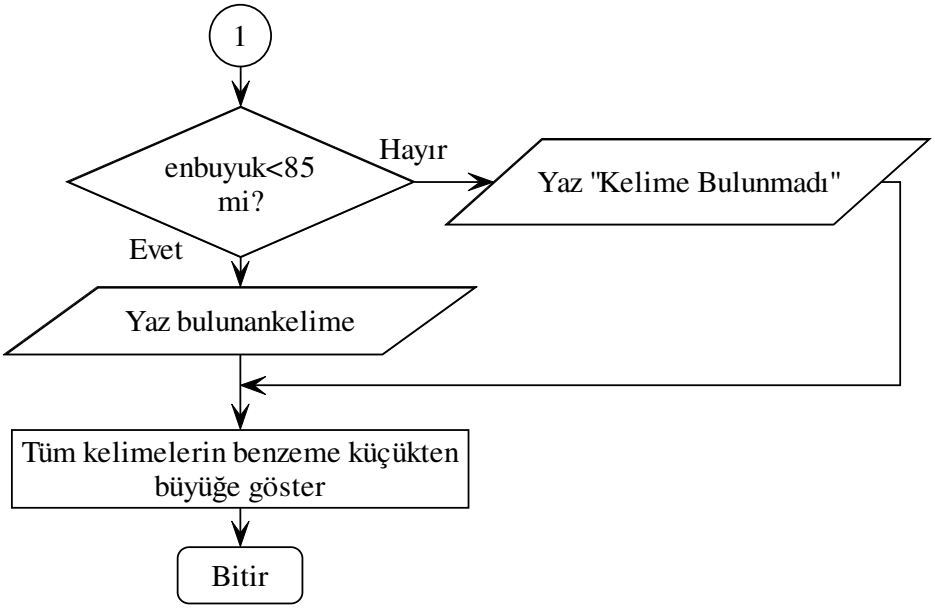
Cepstral uzaklığı ile Öklid uzaklığı benzerdir ve "Eş. 4.21" deki gibi tanımlanır [53].

$$d_{CEP}=\sum_{i=1}^{n} (c_t(i) - c_r(i))^2 \tag{4.21}$$

Programın ses tanıma kısmı iki alt programdan oluşmaktadır. Birinci kısım veritabanındaki referans ses sinyalini çağırır ve ikinci kısım olan karşılaştırma alt programına gönderir. Bu şekilde kullanıcının mikrofona söylemiş olduğu kelime ile veritabanındaki tüm ses verileri ayrı, ayrı karşılaştırılarak en fazla benzeyen kelime bulunmaya çalışılır. Eğer veritabanındaki kelimelerin hiç biri %85'den daha fazla benzemiyorsa kelime bulunamamış demektir. Birinci alt program olan "Tformyakalanan.recognizeClick" alt programının akış şeması Şekil 4.25'de gösterilmektedir.



Şekil 4.25. Referans sinyalleri okuyan alt programın akış şeması



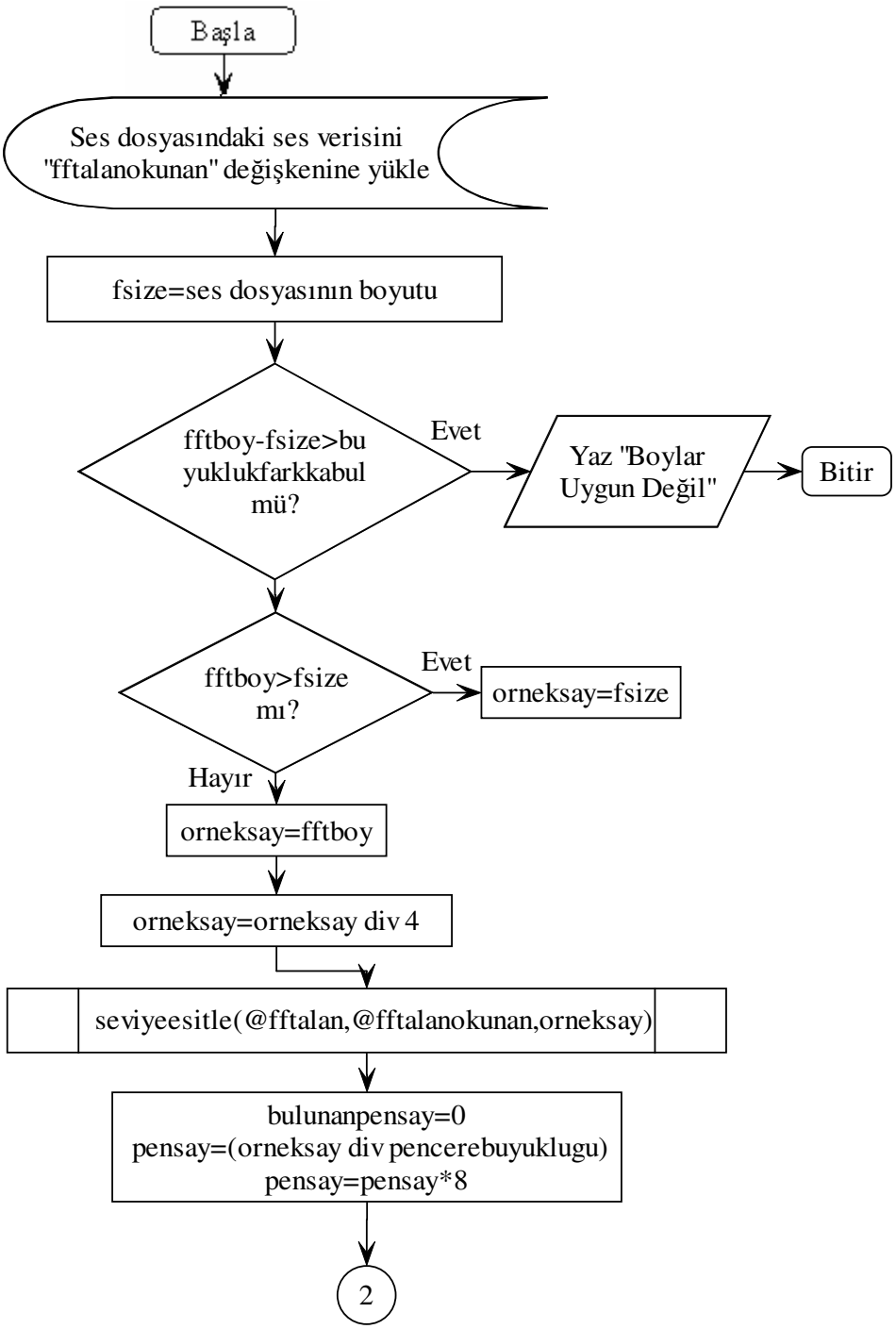
Şekil 4.25. (Devam) Referans sinyalleri okuyan alt programın akış şeması

Veritabanındaki referans sinyali ile tanınacak olan ses sinyalini karşılaştırıp belirli bir tanıma yüzdesi bulan alt programın kodları ise aşağıda verilmektedir. Bu alt program iki ses sinyalini normal pencere sayısından 8 kat daha fazla pencereye bölerek her pencere için öklid uzaklığını hesaplar. Hesaplanan öklid değeri program içinde belirlenen kabul eşik değerinden küçükse bulunan pencere sayısı bir artırılır. Bu işlem her pencere için yapılır. En son olarak

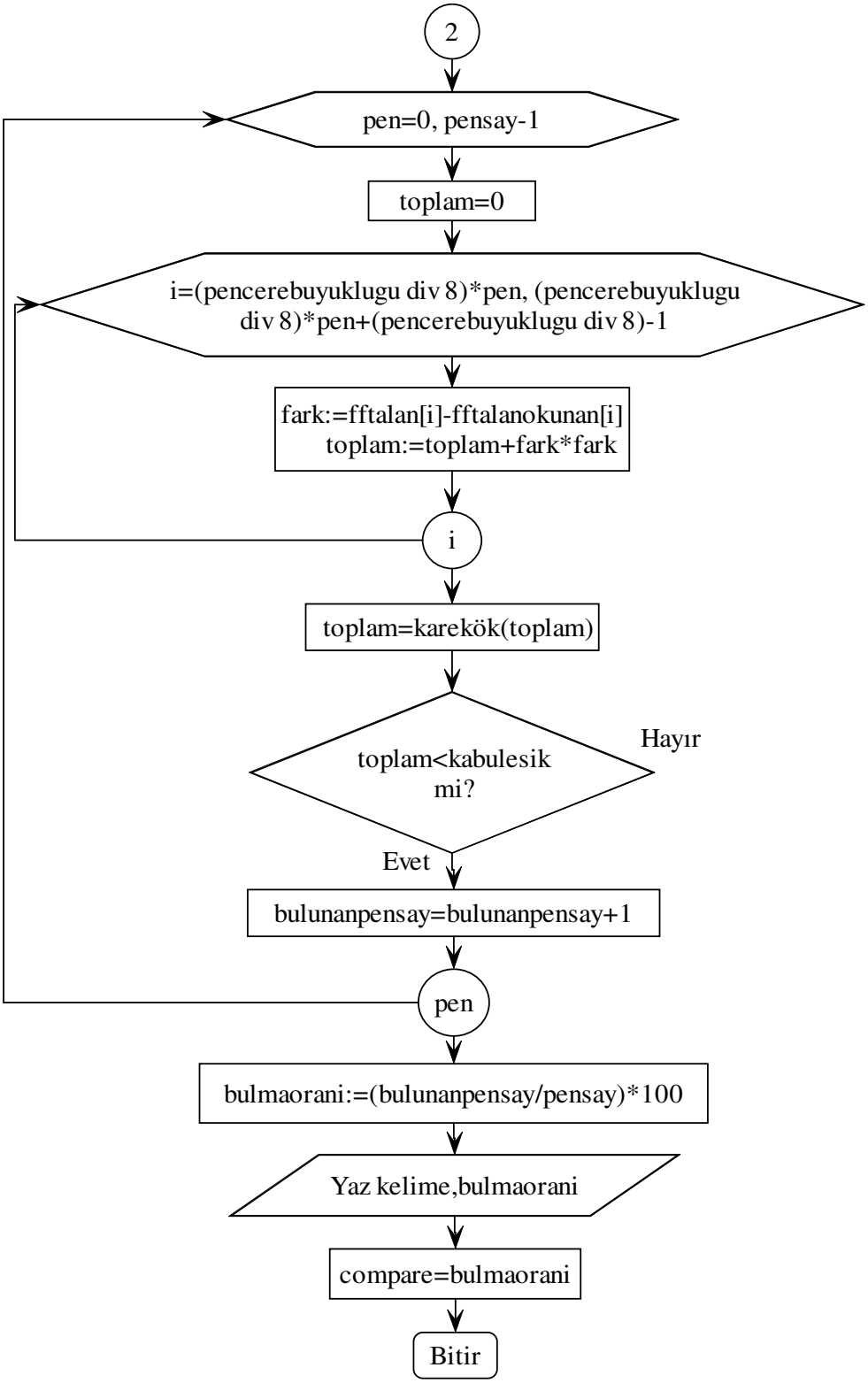
$$\text{Benzerlik Yüzdesi} = \frac{\text{Bulunan Pencere Sayısı} * 100}{\text{Toplam Pencere Sayısı}} \qquad [10] \qquad (4.22)$$

Eş. 4.22 deki formülle benzerlik yüzdesi hesaplanır. Bu benzerlik yüzdesi ilk alt programa geri gönderilir. Bulunan bu benzerlik yüzdesi bir önceki kelimenin benzerlik yüzdesinden büyükse bulunan kelime yerine bu kelime geçer. Bu işlemler veritabanında kayıtlı tüm kelimeler için tekrarlanır.

Tanıma işlemi için ikinci alt program olan “Tformyakalanan.compare” alt programının akış şeması Şekil 4.26’da gösterilmektedir.



Şekil 4.26. Tanıma işlemini gerçekleştiren alt programın akış şeması

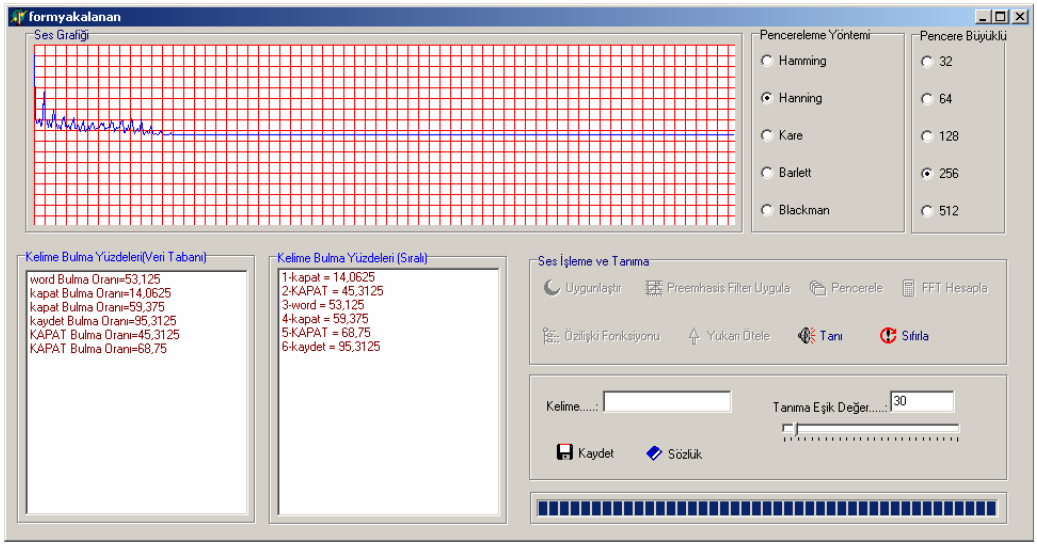


Şekil 4.26. (Devam) Tanıma işlemini gerçekleştiren alt programın akış şeması

Tanıma işlemi için ikinci alt program olan “Tformyakalanan.compare” alt programının kaynak kodları aşağıda verilmektedir.

Bu işlemlerden sonra program sesin tanınıp tanınmadığını tanındıysa yüzde kaç oranında ve hangi kelimenin tanındığını verir. İstenirse bulunan kelimeye bağlı bir komut çalıştırılabilir. Örneğin word kelimesi ile word programı çalıştırılabilir. Ayrıca veritabanında kayıtlı her referans için yüzde kaç oranında benzediğini listeler.

Şekil 4.27. Tanıma işlemi sonuç penceresi



Şekil 4.28. Ses tanıma penceresi

5. SONUÇ VE ÖNERİLER

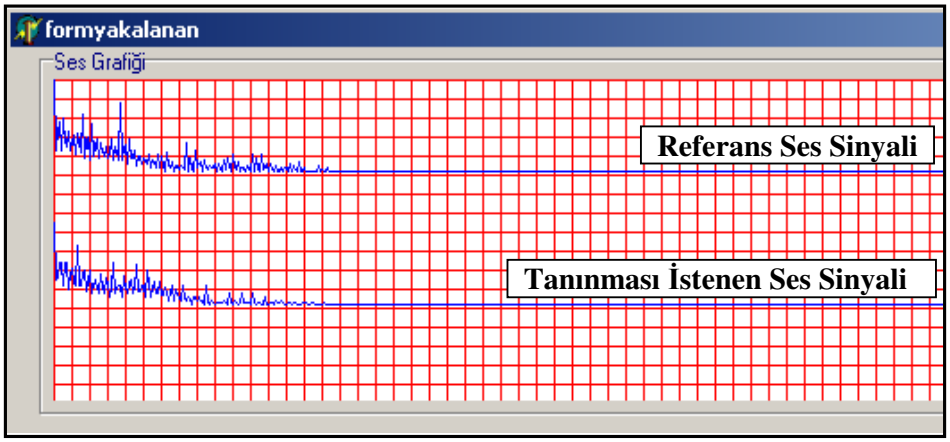
Bu çalışma örüntü tanıma siteminin bir uygulamasıdır. Yapılan birçok deneme sonucunda geliştirilen yazılımın, sesli ifade tanıma doğruluk oranı yüksek çıkmıştır. Ancak sesli ifade tanımayı olumsuz etkileyen bazı faktörlerin olduğu yapılan denemeler sonucunda ortaya konulmuştur. Bunlardan en önemlisi gürültüdür.

Gürültü özellikle sesli ifadenin başlangıç, bitişinin doğru bir şekilde belirlenmesini engellemektedir ve söylenen kelime doğru bir şekilde yakalanamamaktadır. Bu durum da doğal olarak tanıma performansını etkilemektedir. Sesli ifade tanımayı etkileyen diğer bir unsur da sesli ifadelerin ilk söyleniş tarzına benzer bir tarzda söylenmesinin gerekli olmasıdır. Her iki söyleyiş tarzı birbirine ne kadar yakınsa öklid uzaklığı o kadar küçük olmakta ve kelimelerin benzeme oranı artmaktadır. Söyleyiş tarzının negatif etkisini en aza indirmek için aynı kelimenin örnek sayısı artırılabilir. Ayrıca mikrofonun ve ses kartının kalitesi, konuşmacının mikrofona olan uzaklığı da tanıma performansını etkilemektedir.

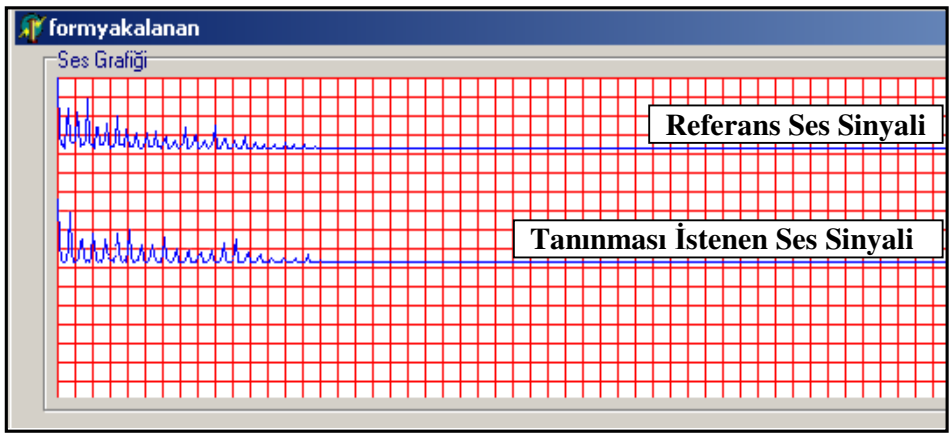
Sesli ifade tanımada en iyi performansı elde etmek için sesli ifade tanımayı etkileyen parametreler iyi bir şekilde ayarlanmalıdır. Gürültünün fazla olduğu ortamlarda ses yakalama enerji eşik seviyesi yüksek tutulmalıdır. Böylece kelimenin başlangıcı daha sağlıklı olarak elde edilebilmektedir. Sesli ifade tanıma işleminde en iyi sonuçlar Hanning pencereleme kullanılarak elde edilmiştir. Uzun kelimeler ve birden fazla kelime yakalama işleminde yazılımın performansı düşüktür. Bu tür uygulamalarda ses bitiş örnek sayısı yüksek seçilmelidir. Aksi takdirde söylenen kelimelerin hepsi program tarafından yakalanamayacaktır.

Örüntü tanıma sistemini kullanarak geliştirilen yazılım kişiye bağımlı olarak kısıtlı sayıdaki kelimeyi başarılı olarak tanımaktadır.

Tanınmış “İleri” kelimesinin veritabanındaki referans sinyali ile karşılaştırma grafiği Şekil 5.1’de, “Word” kelimesinin karşılaştırma grafiği ise Şekil 5.2’de gösterilmektedir.

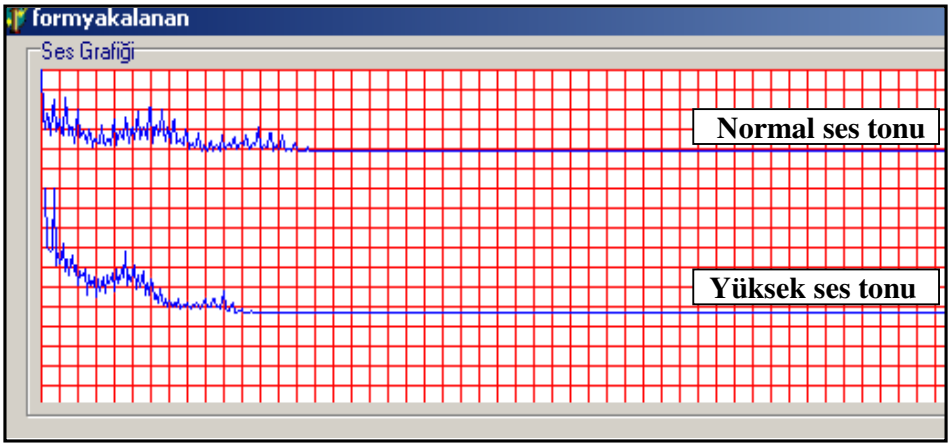


Şekil 5.1. %97,5 benzerlik oranında tanınmış "İleri" kelimesinin referans sinyali ile karşılaştırılması



Şekil 5.2. %100 benzerlik oranında tanınmış "Word" kelimesinin referans sinyali ile karşılaştırılması

Kelimelerin yüksek ve düşük tonda söylenmesi de tanımayı etkilemektedir. Şekil 5.3'de "Kaydet" kelimesinin normal ve yüksek tondaki grafiği gösterilmektedir.



Şekil 5.3. "Kaydet" kelimesinin yüksek ve normal ses tonlu karşılaştırma grafikleri

Çizelge 5.1, Çizelge 5.2, Çizelge 5.3 ve Çizelge 5.4'de verilen tablolarda geliştirilen ses tanıma yazılımında yapılmış olan deneme sonuçları, benzeme oranları ve tanıma yüzdeleri olarak yer almaktadır. Benzeme oranı %85'in üzerinde ise ses tanınmış demektir. Denemeler kadın, erkek sesine ve 1, 2 örneklemeye göre ayrı, ayrı yapılmıştır. Sonuçta tanıma yüzdesi kadın sesinde en düşük %70 erkek sesinde ise %80 çıkmıştır. Ayrıca veritabanındaki örnekleme arttıkça tanıma yüzdesi ve benzerlik oranları artmaktadır. Bu da tanıma performansını artırmaktadır.

Çizelge 5.1. Tek örnekleme, bayan sesine göre yazılımda yapılan deneme sonuçları

Deneme Sayısı	Bayan Sesi Tek Örnekleme			
	Kelimelerin Benzeme Oranları(%)			
	Kapat	Göster	Kaydet	Word
1	89	92,5	90,2	90,13
2	69	77,94	59,72	89,20
3	85,15	97,65	61,11	86,02
4	75	90,44	86,11	93,05
5	89	97,7	100	78,8
6	74,2	97,11	51	88,97
7	74,2	94,79	69,4	75
8	85,7	97,3	97,2	86,8
9	87,5	91,6	91,6	100
10	85,2	85	86,11	91,9
11	88,4	71,87	81,9	89,58
12	89,06	91,07	92,7	90,38

Çizelge 5.1. (Devam) Tek örnekleme, bayan sesine göre yazılımda yapılan deneme sonuçları

13	99,1	82,5	91,6	90,83
14	98	85,15	81,6	84,16
15	92,18	75	85,1	92,30
16	85,15	68	91,3	91,07
17	74	98	85,15	71,5
18	88,33	89,2	71,8	63,3
19	88,4	88,3	91,6	89,58
20	87,5	97,3	81,25	88,46
Tanıma Yüzdesi	%75	%80	%70	%80

Çizelge 5.2. Tek örnekleme, erkek sesine göre yazılımda yapılan deneme sonuçları

Erkek Sesi Tek Örnekleme				
Kelimelerin Benzeme Oranları(%)				
Deneme Sayısı	Kapat	Göster	Kaydet	Word
1	87,5	85,93	89,06	100
2	81,25	96,8	92,18	75
3	85,4	83,3	100	100
4	87,5	83,3	100	100
5	91,6	89,06	100	100
6	93,75	85,7	100	100
7	93,75	93,75	100	100
8	87,5	92,18	85,93	95,3
9	93,75	87,5	96,8	93,75
10	89,5	80	92,18	96,8
11	95,8	89,06	100	96,8
12	89,5	81,25	100	100
13	95,8	94,6	100	100
14	91,6	85,7	100	10
15	93,75	89,06	87,5	91,07
16	91,6	91,07	100	100
17	82,5	87,5	100	100
18	93,75	95,3	93,75	96,5
19	87,5	93,75	97,5	64,5
20	79,1	87,5	100	91,6
Tanıma Yüzdesi	%85	%80	%100	%95

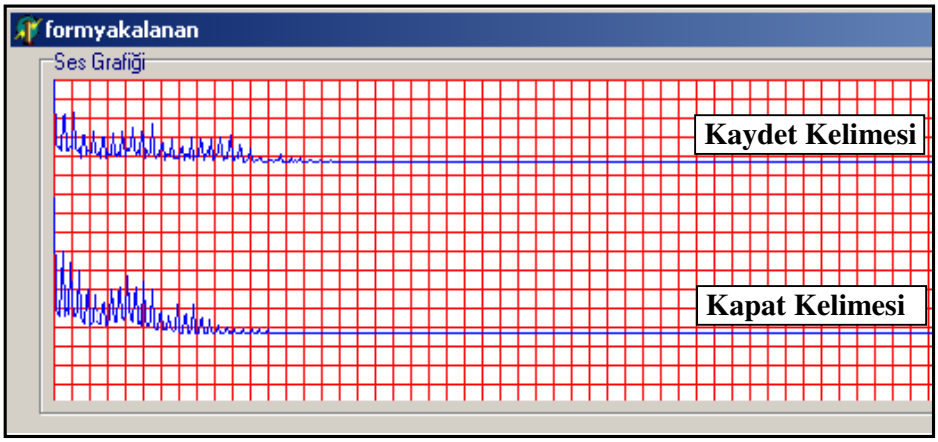
Çizelge 5.3. İki örnekleme, bayan sesine göre yazılımda yapılan deneme sonuçları

Deneme Sayısı	Bayan Sesi 2 Örnekleme			
	Kelimelerin Benzeme Oranları(%)			
	Kapat	Göster	Kaydet	Word
1	91,6	98,2	91	88,5
2	89,5	97,3	89,1	91,07
3	86,5	96,42	85,93	84,8
4	87,5	95,5	72	70
5	82,5	80,5	77,5	90,6
6	87,5	89,7	93,75	79,8
7	86,5	94	91,6	85,8
8	85	88	68,75	87,5
9	87,5	90	84,3	88,75
10	85,2	92	92,04	90,6
Tanıma Yüzdesi	%90	%90	%70	%80

Çizelge 5.4. İki örnekleme, erkek sesine göre yazılımda yapılan deneme sonuçları

Erkek Sesi 2 Örnekleme				
Kelimelerin Benzeme Oranları(%)				
Deneme Sayısı	Kapat	Göster	Kaydet	Word
1	100	100	100	93,75
2	94,4	100	100	92,5
3	96,8	87,5	100	100
4	93,05	93,75	100	100
5	80	94,6	100	100
6	87,5	89,5	100	100
7	93,05	96,8	100	95,8
8	94,4	97,9	92,5	97,7
9	93,75	100	96,8	98,2
10	87,5	92,5	98,4	98,43
Tanıma Yüzdesi	%95	%100	%100	%100

Söyleyiş tarzları birbirine yakın olan, farklı kelimeler de tanımayı olumsuz yönde etkilemektedir. Özellikle kelimelerin birbirine karıştırılmasına sebep olmaktadır. Bu durum karıştırılan kelimelerin veritabanından silinerek tekrar girilmesiyle veya söyleyiş tarzının düzeltilmesi ile giderilebilir. “Kaydet” ve “Kapat” kelimelerinin grafiksel olarak karşılaştırılması Şekil 5.4’de gösterilmektedir.



Şekil 5.4. “Kaydet” ve “Kapat” ses sinyallerinin grafiksel karşılaştırılması

Sesli ifade tanımada başarılı bir yöntem olan HMM kullanılabilir. HMM yöntemi sesli ifade tanımaya istatistiksel olarak yaklaşır. HMM sesli ifade tanıma için istatistiksel bir çerçevede güçlü bir algoritma sağladığı için ve yapı olarak bizim ses sistemimize uygun bir model olarak görüldüğü için daha güçlü bir yöntemdir. HMM yöntemi yutulan sesleri dahi gramer yapısını kullanarak tahmin edebilir. Konuşmada gereksiz kullanılan uzatmaları, “hımm”, humm” gibi sesleri eleyebilir. Ayrıca bu yöntem kullanılarak birden fazla kelime tanınabilir. Bu yöntem kullanılarak bu tez kapsamında tam olarak gerçekleştirilemeyen birden fazla kelimenin tanınması işlemi gerçekleştirilebilir.

Son yıllarda popülerliğini iyice artıran diğer bir yöntem ise yapay sinir ağlarıdır. Yapay sinir ağlarında doğadaki canlılar ve tabii temel olarak insanın beyin ve sinir yapısı taklit edilmektedir. Temel olarak ses tanıma daha çok insana özgü olduğundan bu yöntem daha akılcı ve umut vericidir. Bu tez kapsamında geliştirilen yazılım kişiye bağımlı olarak çalışmaktadır. Bu işlem kişiye bağımsız olarak gerçekleştirilmek istenirse yapay sinir ağları kullanılabilir.

KAYNAKLAR

1. Polur, P. B. D., ”Development and Optimization Of a HMM-ANN Hybrid Structure For Robust Recognition of Impaired Speech Signals”, Doktora Tezi, ***Virginia Commonwealth University***, United States, 1 (2005).

2. Burrows,L.T., “Speech Processing With Linear And Neural Network Models”, Doktora Tezi, ***Cambridge University Engineering Department***, England, 21-22 (1996)

3. Akçay, B., “Yapay Sinir Ağları İle Türkçe Konuşma Tanıma”, Yüksek Lisans Tezi, ***Hacettepe Üniversitesi Fen Bilimleri Enstitüsü***, Ankara, 2,22,42 (1994).

4. Artuner, H., “Bir Türkçe Fonem Kümeleme Sistemi Tasarımı ve Gerçekleştirimi”, Doktora Tezi, ***Hacettepe Üniversitesi Fen Bilimleri Enstitüsü***, Ankara, 4-7,25,31, 47,48,60 (1994)

5. Hansen ,J., C., “Modulation Based Parameters For Automatic Speech Recognition”, Yüksek Lisans Tezi**,** ***University Of Rhode Island Department of Electrical Engineering***, England, 1 (2003).

6. Walker, S., L., “Wavelet-based feature extraction for robust speech recognition”, Doktora Tezi, ***University Of Florida State Department Of Electrical & Computer Engineering***, United States, 8 (2003).

7. Özet, M., Arpacı, O., “Biyoloji 2”, ***Zambak Basım Yayım,*** İstanbul, 93-96 (2000).

8. Keeton, W., T., Could, J., L., “Biyological Science”, Ali Demirsoy, İsmail Türkan, ***Norton&Company,*** New York, 1045 (2000).

9. Karahan,F., “Fusing Length And Voicing Information, And HMM Decision In Speaker Dependent Isolated Word Recognition Systems”, Yüksek Lisans Tezi , ***Middle East University Electrical And Electronics Engineering Department***, Ankara, 24-25 (2000)

10. Doğan, S., “Pc Ortamında Sesli Komutları Tanıma”, Yüksek Lisans Tezi, ***Marmara Üniversitesi Fen Bilimleri Enstitüsü***, İstanbul, 1,8,9,10,12,14-17,21-24,26,27, 36,63,67 (1999)

11. Picone, J.,W., “Signal Modeling Techniques In Speech Recognition”, ***Proceedings of the IEEE***, 81 (9) :1219 (1993)

12. Internet:Wikipedia The Free Encylcopedia. http://en.wikipedia.org/wiki/ Fourier_series (2006).

13. Qiang,H., Youwei, Z.,"On Prefiltering And Endpoint Detection Of Speech Signal", , ***Signal Processing Proceedings ICSP '98. 1998 Fourth International Conference on,*** Beijing, China, 1: 749,750 (1998).

14. Neben, G., McAulay, R., Weinstein, C., "Experiments In Isolated Word Recognition Using Noisy Speech"***, Acoustics, Speech, and Signal Processing, IEEE International Conference on ICASSP '83***, Boston, USA***,*** 8: 1156 (1983).

15. Martin, T.B., "Practical Applications Of Voice Input To Machines", ***Proceedings of the IEEE***, 64 (4): 487 (1976).

16. Sauter, L.," Isolated word recognition using a segmental approach***", Acoustics, Speech, and Signal Processing, IEEE International Conference on ICASSP '85,*** Tampa, Florida, USA, 10: 850 (1985)

17. Mclnnes, F., R., Jack, M., A., Laver, J., "Template Adaptation In An Isolated Word-Recognition System"***, IEE Proceedings***,136 (2): 119 (1989).

18. Welch, J., Oxenberg, S., "Reduction of minimum word-boundary gap lengths in isolated word recognition", ***Acoustics, Speech, and Signal Processing, IEEE International Conference on ICASSP '80***, Denver, Colorado, USA, 5: 190 (1980).

19. Ling, Y., "Keyword Spotting In Continuous Speech Utterances ", Yüksek Lisans Tezi, ***, University of McGill University School Of Computer Science***, Canada, 3 (1999).

20. Mangu, L., L., "Finding Consensus In Speech Recognition", Doktora Tezi, ***University Of Johns Hopkins***, United States, 3 (2000).

21. Aydın, Ö., "Yapay Sinir Ağlarını Kullanarak Bir Ses Tanıma Sistemi Geliştirilmesi", Yüksek Lisans Tezi, ***Trakya Üniversitesi Fen Bilimleri Enstitüsü***, Edirne, 7,50 (2005)

22. Chen, Z., "Handgrip pattern recognition", Doktora Tezi, ***New Jersey Institute of Technology***, United States, 38 (2003).

23. Miki,K. Nishiura, T., Nakamura, S.,Shikano, S., " Speech Recognition Based on HMM Decomposition and Composition Method with a Microphone Array in Noisy Reverberant Environments", ***Electronics and Communications in Japan, ,***85 (2): 17 (2002).

24. Zhu, J., "Pattern Modeling And Classification In Vision Systems", Doktora Tezi, ***University of Princeton Department of Electrical Engineering ",*** United States, 1 (2005).

25. Youssif, R., S., ”Hybrid Intelligent Systems for Pattern Recognition and Signal Processing”, Doktora Tezi, ***University of Cincinnati Department of Electrical & Computer Engineering and Computer Science of the College of Engineering***, United States, 7,8 (2004).

26. Meloni, L., G., P., “Learning Discrete Hidden Markov Models”, ***Computer Applications in Engineering Education, John Wiley & Sons***, 8 (3): 1 (2000).

27. Moscovich,L., G., “Learning Discrete Hidden Markov Models From State Distribution Vectors”, Doktora Tezi, ***Louisiana State University and Agricultural & Mechanical College The Department of Computer Science***, United States, 1 (1998).

28. Weingessel, A., “Speech recognition”, Master's thesis, ***Institut für Statistik und Wahrscheinlichkeitstheorie, Technische Universität***, Munchen, 39 (1994).

29. Brenner, A.,R., Eck, K., Engelhard, G., Noll, T.,G., “Phase Aberration Correction Using Dynamic Time Warping”, ***Ultrasonics Symposium, 1995. Proceedings. IEEE,*** Seattle,USA, 2: 1361 (1995).

30. Youssef, A.,M., Abdel-Galil, T., K., El-Saadany, E., F., Salama, M., M., A., “Disturbance Classification Utilizing Dynamic Time Warping Classifier”, ***IEEE Transactıons On Power Delivery***, 19(1) : 272 (2004).

31. Mengüşoğlu, E.,”Bir Türkçe Sesli İfade Tanıma Sisteminin Kural Tabanlı Tasarımı Ve Gerçekleştirimi”, Yüksek Lisans Tezi***, Hacettepe Üniversitesi Fen Bilimleri Enstitüsü***, Ankara, 1,7,8,42 (1999).

32. Curtis ,B.,A., “Isolated Word Recognition For Digits Zero Through Nine”, Yüksek Lisans Tezi**, *University Of Florida Atlantic University Department of College Engineering***, Florida, 3-9 (1987).

33. Dreyfus-Graf, J., Sonograph and sound mechanics," ***Journal of the Acoustical Society of America***, 22 (6) : 731 (1950).

34. Olson, H., F., Belar, H.,” Phonetic Typewriter”, ***Journal of the Acoustical Society of America,*** 28 (6) : 1072-1081 (1956).

35. Olson, H., F., Belar, H.,” Phonetic Typewriter III”, ***Journal of the Acoustical Society of America***, 33 (11) : 1610-1615 (1961).

36. Shearme, J., Leach, P., “Some experiments with a simple word recognition system”, ***Audio and Electroacoustics, IEEE Transactions on,*** 6 (2) ***:*** 261 (1968).

37. Purton, R., “Speech recognition using autocorrelation analysis”, ***Audio and Electroacoustics, IEEE Transactions on,*** 16 (2): 235,239 (1968).

38. Rajakumari,T., "Speech Recognition System Under Noisy Environment***",*** Yüksek Lisans Tezi***, University of Hyderabad Department of Computer/Information Sciences,*** India, 6,14,15 (2005).

39. İkizler, N.,"Türkçe'de Konuşmacıdan Bağımsız Hece Tanıma Sistemi", Doktora Tezi, ***Karadeniz Teknik Üniversitesi Fen Bilimleri Enstitüsü***, Trabzon,5, 6, 22,54 (2003).

40. Ganapathiraju, A., Webster, L., Trimble, J., Bush, K., Kornman, P., "Comparison Of Energy-Based Endpoint Detectors For Speech Signal Processing", ***Bringing Together Education, Science and Technology, Proceedings of the IEEE***, 500 (1996)

41. Ying, G.S., Mitchell, C.D., Jamieson, L.H., "Endpoint Detection Of Isolated Utterances Based On A Modified Teager Energy Measurement", ***Acoustics, Speech, and Signal Processing ICASSP-93.1993 IEEE International Conference on,*** Minneapolis, Minnesota, USA, 2: 732 (1993)

42. Gökhan, A., "Yapay Sinir Ağları İle Ayrık Türkçe Sözcüklerin Tanınması", ***Fırat Üniversitesi Fen Bilimleri Enstitüsü***, Elazığ, 23 (1997)).

43. Kondoz, A. M., "Digital Speech", B. Evans, G. Pujolle, A. Danthine, O. Spaniol, ***John Wiley & Sons***, Newyork, 36,37 (1994)

44. Kinnunen, T., "Spectral Features For Automatic Text-Independent Speaker Recognition", Licentiate's Thesis, ***University of Joensuu Department of Computer Science***, Finland, 53 (2003).

45. Sherlock, B.G.," Windowed discrete Fourier transform for shifting data", ***Submitted to Elsevier Science on Signal Processing,*** 74***:*** 177 (1999)

46. Sherlock, B.G., Kakad,Y.P.," Windowed discrete cosine and sine transforms for shifting data", ***Submitted to Elsevier Science on Signal Processing,*** 81: 1472 (2001).

47. Rockmore, D.N.," Some Applications Of Generalized FFTs ", ***Dimacs Series in Discrete Mathematics Theoretical Computer Science ,*** 00 (*19):*1 (1991)

48. Nguyen,T. Q. ,"Integer Fast Fourier Transform", ***IEEE Transactions On Signal Processing,***50 (3): 607 (2002).

49. Cochran, W. T., Cooley, J. W., **"**What Is the Fast Fourier Transform?", ***Proceedings Of The IEEE,*** 55 (10):1664,1667 (1967).

50. Cooley, J. W., Tukey, J. W., ***"***An algorithm for the machine calculation of complex Fourier series***", Math. Comput.,*** 19: 297 (1965).

51. Kobayashi, H., Shimamura, T., **"**A Weighted Autocorrelation Method For Pitch Extraction Of Noisy Speech", ***Acoustics Speech and Signal Processing IEEE International Conference***, İstanbul, Türkiye, 3:1307 (2000)

52. Dendrinos, M., Carayannis, G., "Spectrum Analysis Using A New Autocorrelation Measure", ***Acoustics, Speech, and Signal Processing***, 4:2468 (1998)

53. Tohkura,Y., "A Weighted Cepstral Distance Measure For Speech Recognition", ***Acoustics, Speech, and Signal Processing***, 11: 761 (1986).

EKLER

EK-1 Sesli ifade tanımayla ilgili olarak geliştirilen programın kaynak kodları

```
program Project1;
uses
  Forms,
  sessayisal in 'sessayisal.pas' {Form1},
  seslib in 'seslib.pas',
  sinyalisle in 'sinyalisle.pas',
  goruntu in 'goruntu.pas',
  uyakalanan in 'uyakalanan.pas' {formyakalanan},
  ffthesapla in 'ffthesapla.pas',
  tani in 'tani.pas' {Ftani},
  sozluk in 'sozluk.pas' {formsozluk},
  sesyakalaayar in 'sesyakalaayar.pas' {fyakalaayar};
{$R *.RES}
begin
  Application.Initialize;
  Application.CreateForm(TForm1, Form1);
  Application.CreateForm(Tformyakalanan, formyakalanan);
  Application.CreateForm(TFtani, Ftani);
  Application.CreateForm(Tformsozluk, formsozluk);
  Application.CreateForm(Tfyakalaayar, fyakalaayar);
  Application.Run;
end.
//-----------------------------------------------------------------------------------------------
//Bu unitte öncelikle ses sayısal olarak bilgisayar ortamına alınmaktadır.
//Bilgisayar ortamına alınan ses
//1. Verikutupla ile kutuplanmakta.
//2. Enerji ile enerjisi hesaplanmakta
//3. Enerji ile de enerjisi hesaplanmaktadır.
unit sessayisal;
```

EK–1 (Devam) Sesli ifade tanımayla ilgili olarak geliştirilen programın kaynak kodları

```
interface
uses
  Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms, Dialogs,
  Audio, StdCtrls, ExtCtrls, Buttons, ComCtrls;
type
  TForm1 = class(TForm)
    Audio1: TAudio;
    GroupBox1: TGroupBox;
    Image1: TImage;
    GroupBox2: TGroupBox;
    kutupkont: TCheckBox;
    enerjikont: TCheckBox;
    StatusBar1: TStatusBar;
    bilgi: TPanel;
    GroupBox3: TGroupBox;
    Kaydet: TSpeedButton;
    cal: TSpeedButton;
    caldurdur: TSpeedButton;
    yakalanangoster: TSpeedButton;
    GroupBox4: TGroupBox;
    Memo1: TMemo;
    liste1: TListBox;
    SpeedButton1: TSpeedButton;
    //image1:timage;
    procedure Audio1Record(Sender: TObject; LP, RP: Pointer;
      BufferSize: Word);
    procedure yenidenClick(Sender: TObject);
    procedure enerji;
    procedure sesyakala;
```

EK–1 (Devam) Sesli ifade tanımayla ilgili olarak geliştirilen programın kaynak kodları

```
    procedure yakalanangosterClick(Sender: TObject);
    procedure calClick(Sender: TObject);
    procedure caldurdurClick(Sender: TObject);
    procedure KaydetClick(Sender: TObject);
    procedure SpeedButton1Click(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
  end;
var
  Form1: TForm1;
      LeftStream,RightStream:TMemoryStream;
      temp:array of pchar;
      boyut:array of word;
      topla:pointer;
      toplaboy:real;
      dinamik:byte;
      k:word;
      bsize:longword;
      kacsefer:byte;
      //sestam:boolean=true;
      kontrol:byte;
      defa:integer;
implementation
uses sinyalisle,seslib, uyakalanan,sesyakalaayar;
{$R *.DFM}
procedure TForm1.Audio1Record(Sender: TObject; LP, RP:pointer;BufferSize:
Word);
```

EK–1 (Devam) Sesli ifade tanımayla ilgili olarak geliştirilen programın kaynak kodları

```
  begin
 defa:=defa+1;
    if defa=1 then liste1.items.add('Başla') else  liste1.items.add(inttostr(buffersize));
if bitpersamples=16 then sonsize:=buffersize div 2 else sonsize:=buffersize;
gercekses:=lp;
```

EK-1 (Devam)

```
  Veriyikutupla(lp,sonsize);        //sinyalisle unitinde
  if kutupkont.Checked=true then  Veriyigoruntule(sayisalsessinyali,image1,sonsize);
//ses lib unnitinde
  Enerji;
  sesyakala;
 end;
procedure TForm1.yenidenClick(Sender: TObject);
begin
 if not(Audio1.Recorder.Stop) then begin
    Memo1.Lines.Add(Audio1.ErrorMessage);
  end;
end;

procedure tform1.sesyakala;
begin
//sesenerji.kaynakbuyukluk:=form1.Audio1.Recorder.NoSamples;
//  if not(boxenergy.checked and yakalabox.checked) then exit;
case sesyakaladurum of
  _baslangicbul:begin sesbasara;end;
  _bitisbul:begin sesbitara;end;
end;
 if sesyakaladurum=_sesyakalandi then begin
```

EK–1 (Devam) Sesli ifade tanımayla ilgili olarak geliştirilen programın kaynak kodları

```
    bilgi.Caption:='Yakalandı';
     audio1.Recorder.Active:=false;
end;
end;
procedure tform1.enerji;
var
temp:integer;
begin
  if enerjikont.checked then begin
       // liste.items.add(inttostr(form1.Audio1.Recorder.NoSamples));
   ekaynakbuyukluk:=form1.Audio1.Recorder.NoSamples;
   ekaynak:=sayisalsessinyali;
   kisazamanenerji;
   enerjigoruntule(@ehedef,form1.Image1,ekaynakbuyukluk-epencerebuyukluk);
 end;
end;
procedure TForm1.yakalanangosterClick(Sender: TObject);
begin
 formyakalanan.show;
  yakalanangoruntule(@yakalanansesbuf,baslangicbitisyer,formyakalanan.image1);
end;
procedure TForm1.calClick(Sender: TObject);
var
i:byte;
begin
liste1.Items.Clear;
   LeftStream.Free; RightStream.Free;
   LeftStream := nil; RightStream:=nil;
   LeftStream :=TMemoryStream.Create;
```

EK–1 (Devam) Sesli ifade tanımayla ilgili olarak geliştirilen programın kaynak kodları

```
    RightStream:=TMemoryStream.Create;
     LeftStream.Position:=0;
for k:=1 to kaceleman do begin
if bitpersamples=8 then LeftStream.Write(pyakala[k],1);
   liste1.items.add(inttostr(pyakala[k]));
  end;
if (LeftStream<>nil) and (LeftStream.Size>0) then begin
    LeftStream.Position:=0;
    if RightStream.Size>0 then RightStream.Position:=0
    else begin
      RightStream.Free;
      RightStream:=nil;
    end;
    Audio1.Player.Play(LeftStream,RightStream,0);
  end;
end;
procedure TForm1.caldurdurClick(Sender: TObject);
begin
 Audio1.Player.Stop;
end;
procedure TForm1.KaydetClick(Sender: TObject);
begin
form1.Refresh;
liste1.Items.Clear;
if not(Audio1.Recorder.Stop) then begin
    Memo1.Lines.Add(Audio1.ErrorMessage);
  end;
  defa:=0;
sesyakaladurum:=_baslangicbul;   //verileri sıfırla
```

EK–1 (Devam) Sesli ifade tanımayla ilgili olarak geliştirilen programın kaynak kodları

```
kaceleman:=0;                    //sonraki işlem için
oncekiveribosalt;
baslangicbitisyer:=0;
k:=0;
  Audio1.Recorder.Triggered:=false;
  bsize:=0;
  if not(Audio1.Recorder.Start) then Memo1.Lines.Add(Audio1.ErrorMessage);
  if LeftStream <> nil then begin
    LeftStream.Free; RightStream.Free;
    LeftStream := nil; RightStream:=nil;
  end;
  LeftStream :=TMemoryStream.Create;
  RightStream:=TMemoryStream.Create;
end;
procedure TForm1.SpeedButton1Click(Sender: TObject);
begin
fyakalaayar.show;
end;
end.
//-------------------------------------------------------------------------------------------------
unit sinyalisle;
interface
uses seslib,sessayisal,uyakalanan;
{Type
 Enerji=class
       pencerebuyukluk:integer;
       kaynakbuyukluk:integer;
       kaynak:pointer;
       hedef:pointer;
```

EK–1 (Devam) Sesli ifade tanımayla ilgili olarak geliştirilen programın kaynak kodları

```
      logenable:boolean; }
 const
   Penyontem:array[0..4] of string
          =('Kare','Balckman','Hanning','Hamming','Barlett');
   var
     //*****Kısa Zaman Enerji Parametreleri
      epencerebuyukluk:integer;
      ekaynakbuyukluk:integer;//Sayısal hale getirilen örnek sayısı
      ekaynak:pointerintegerdizi;   //Sayısal hale getirilmiş ve kutuplanmış ses
sinyalini enerji hesabı için tutar.
      ehedef:array  [0..50000] of integer;//enerjisi hesaplanmış sayısal ses sinyallerini
tutar.
      logenable:boolean=true;
      epenbuyukluk:integer=256; //Enerji hesaplanırken dizi üzerindeki aralık
//****Pencereleme Parametreleri**********
    penceresayisi:integer;
    pencerelemeyontemi:string;
    pencerebuyuklugu:integer=512;
    pencerekayma:integer=0;
    pheryerde:array[0..50000] of integer;
//*****************************************
 procedure veriyikutupla(data:pointer;size:integer);
  procedure preemhasisfiltre(veri:pointer;size:integer;katsayi:real);
  Procedure windowing(data:pointer;veriboyutu:integer;pencerebuyukluk:integer);
  procedure Hamming(data:pointer;pencerebuyukluk:integer);
  procedure BlackMan(data:pointer;pencerebuyukluk:integer);
  procedure rectang(data:pointer;windowsize:integer);
  procedure Barlett(data:pointer;pencerebuyukluk:integer);
  procedure Hanning(data:pointer;pencerebuyukluk:integer);
```

EK–1 (Devam) Sesli ifade tanımayla ilgili olarak geliştirilen programın kaynak kodları

```
  procedure kisazamanenerji;
  procedure sesbasara;
  procedure sesbitara;
 implementation
uses math,sysutils;
procedure veriyikutupla(data:pointer;size:integer);
var
   p:array[0..50000] of integer;//pintegerdizi;
   b:pbytedizi;  //seslibde
   w:pworddizi;  //seslibde
   i:integer;
   z:integer;
begin
  if bitpersamples=8 then begin
         b:=data;
         for i:=0 to epenbuyukluk-1 do  //Önceki veri bloğunun son
             p[i]:=oncekiveri[i];         //penceresini başa ekle.
         for i:=epenbuyukluk to size-1 do
           p[i]:=b^[i-epenbuyukluk]-127;  //yeni veriyi ekle akk
         for i:=size-epenbuyukluk to (size-1) do  //yeni verinin son penceresi onceki
veri olarak
            oncekiveri[i-(size-epenbuyukluk)]:=p[i];  //saklanıyor. akk
  end
  else if bitpersamples=16 then begin
       w:=data;
       for i:=0 to epenbuyukluk-1 do //Önceki veri bloğunun son
             p[i]:=oncekiveri[i];          //penceresini başa ekle
       for i:=epenbuyukluk to size-1 do
```

EK–1 (Devam) Sesli ifade tanımayla ilgili olarak geliştirilen programın kaynak kodları

```
          if w^[i-ependbuyukluk]>=32768 then p[i]:=w^[i-ependbuyukluk]-65536  //yeni veriyi ekle akk
           else   p[i]:=w^[i-ependbuyukluk];
      for i:=size-ependbuyukluk to (size-1) do  //yeni verinin son penceresi onceki veri olarak
           oncekiveri[i-(size-ependbuyukluk)]:=p[i];  //saklanıyor. akk
  end;
  sayisalsessinyali:=@p;
  for z:=1 to 50000 do pheryerde[z]:=p[z];
end;
{constructor enerji.create;
begin
 pencerebuyukluk:=256;
 kaynak:=nil;
 logenable:=true;
 getmem(hedef,200000);
end;}
{destructor enerji.Destroy;
begin
hedef:=nil;
end;}
procedure kisazamanenerji;
var
baslangic,bitis:integer;
i,j:integer;
toplam:real;
veri:integer;
begin
ekaynakbuyukluk:=form1.Audio1.Recorder.NoSamples;
```

EK–1 (Devam) Sesli ifade tanımayla ilgili olarak geliştirilen programın kaynak kodları

```
baslangic:=0;
 bitis:=ekaynakbuyukluk-epenbuyukluk;
 i:=baslangic;
while i<=bitis do begin
    toplam:=0;
     for j:=0 to epenbuyukluk-1 do begin
        if bitpersamples=16 then veri:=ekaynak^[i+j] div 256
        else veri:=ekaynak^[i+j];
          toplam:=toplam+veri*veri;
     end;
   if logenable then if toplam<1000 then toplam:=0 else toplam:=10*log10(toplam);
      ehedef[i]:=trunc(toplam);
 i:=i+1;
 end; //while
end;
Procedure sesbasara;
var
    baslangic,bitis:integer;
    i,j,k:integer;
    gecerliorneksayisi:integer;
    Begin
       ekaynakbuyukluk:=form1.Audio1.Recorder.NoSamples;
      gecerliorneksayisi:=0;
      for i:=0 to ekaynakbuyukluk-1 do begin
        if abs(ehedef[i])>sesyakalaesikdeger then begin
           inc(gecerliorneksayisi);
           if gecerliorneksayisi>=sesyakalaorneksayisi then begin
              sesyakaladurum:=_bitisbul;
              kontrol:=1;
```

EK–1 (Devam) Sesli ifade tanımayla ilgili olarak geliştirilen programın kaynak kodları

```
          //tasinacakveribuyukluk:=ekaynakbuyukluk-(i-gecerliorneksayisi);
          baslangicbitisyer:=i-gecerliorneksayisi;//tasinacakveribuyukluk;
          form1.bilgi.caption:='Başlangıç Tamam';
              for k:=0 to baslangicbitisyer do begin
                yakalanansesbuf[k]:=sayisalsessinyali^[i-gecerliorneksayisi+k];
           pyakala[k]:=pheryerde[i-gecerliorneksayisi+k]+127;//yakalanan gerçek ses
                kaceleman:=kaceleman+1;
              end;//for
               baslangicbitisyer:=i+1;
          break;
        end;
      end else gecerliorneksayisi:=0;
    end;//for
  End;
procedure sesbitara;
var
  baslangic,bitis:integer;
  i,j,k:integer;
  nerden,gecerliorneksayisi:integer;
  tasinacakveribuyukluk:integer;
begin
  ekaynakbuyukluk:=form1.Audio1.Recorder.NoSamples;
  gecerliorneksayisi:=0;
 if kontrol=1 then begin   //ses başlangıcı bulunmuşsa ilk buffer için işlem yap
  for i:=baslangicbitisyer to ekaynakbuyukluk-1 do begin
    if ehedef[i]<sessusesikdeger then begin
      inc(gecerliorneksayisi);
      if gecerliorneksayisi>=sessusorneksayisi then begin
          sesyakaladurum:=_sesyakalandi;
```

EK–1 (Devam) Sesli ifade tanımayla ilgili olarak geliştirilen programın kaynak kodları

```
          form1.bilgi.caption:='Yakalandı';
          tasinacakveribuyukluk:=i-gecerliorneksayisi;
          //---------ses bitişi yakalanmışsa işlem yap
           for k:=baslangicbitisyer to tasinacakveribuyukluk do begin
              yakalananSesbuf[k-
baslangicbitisyer+1]:=sayisalsessinyali^[k];//integerenerjibuf^[k];//integerseskaynak[
k];
               pyakala[k]:=pheryerde[i-gecerliorneksayisi+k]+127;
               kaceleman:=kaceleman+1;
           end;
              baslangicbitisyer:=baslangicbitisyer+tasinacakveribuyukluk;
          exit;
       end;
     end else gecerliorneksayisi:=0;
   end;//for
//-------ses bitişi yakalanmamışsa işlem yap---------
         tasinacakveribuyukluk:=ekaynakbuyukluk;
         for k:=baslangicbitisyer to tasinacakveribuyukluk do begin
             yakalananSesbuf[k-baslangicbitisyer]:=sayisalsessinyali^[k];
             pyakala[k]:=pheryerde[i-gecerliorneksayisi+k]+127;
             kaceleman:=kaceleman+1;
         end;
        baslangicbitisyer:=tasinacakveribuyukluk+1;
        kontrol:=0;
//-------------------------------------------
 end  // if Kontrol=1 'in end'i
  else begin //ses yakalama işlemi bitmemişse sonraki buffrelar için işlem yap
    for i:=0 to ekaynakbuyukluk-1 do begin
       if ehedef[i]<sessusesikdeger then begin
```

EK–1 (Devam) Sesli ifade tanımayla ilgili olarak geliştirilen programın kaynak kodları

```
          inc(gecerliorneksayisi);
          if gecerliorneksayisi>=sessusorneksayisi then begin
             sesyakaladurum:=_sesyakalandi;
              tasinacakveribuyukluk:=i-gecerliorneksayisi;
       //---------------ses bitişi yakalanmışsa-----------
                  for k:=0 to tasinacakveribuyukluk do begin
yakalananSesBuf[baslangicbitisyer+k]:=sayisalsessinyali^[k];//integerenerjibuf^[k];//i
ntegerseskaynak[k];
                     pyakala[k]:=pheryerde[i-gecerliorneksayisi+k]+127;
                     kaceleman:=kaceleman+1;
                  end; //for k end
              baslangicbitisyer:=baslangicbitisyer+tasinacakveribuyukluk;
              exit;
            end; //if  end
        end else gecerliorneksayisi:=0; //if end
      end;  //for end
           tasinacakveribuyukluk:=ekaynakbuyukluk;
   //-----------ses bitişi yakalnmamışsa---------------
           for k:=0 to tasinacakveribuyukluk do begin
            yakalananSesBuf[baslangicbitisyer+k]:=sayisalsessinyali^[k];
            pyakala[k]:=pheryerde[i-gecerliorneksayisi+k]+127;
            kaceleman:=kaceleman+1;
            end;  //for
          baslangicbitisyer:=tasinacakveribuyukluk+1;
   end;
end;
procedure preemhasisfiltre(veri:pointer;size:integer;katsayi:real);
var
 preemhasisli:pointerintegerdizi;
```

EK–1 (Devam) Sesli ifade tanımayla ilgili olarak geliştirilen programın kaynak kodları

```
 i:integer;
 begin
    preemhasisli:=veri;
    if preemhasisli^[0]>10 then preemhasisli^[0]:=0;
         formyakalanan.progressbar1.Position:=0;
         formyakalanan.progressbar1.max:=size;
    for i:=0 to size-1 do begin
      preemhasisli^[i+1]:=round(preemhasisli^[i+1]-Katsayi*preemhasisli^[i]);
      formyakalanan.progressbar1.Position:=i;
    end;
 end;
Procedure windowing(data:pointer;veriboyutu:integer;pencerebuyukluk:integer);
var
  gecici:^integer;//sıfırla doldurmak için pointer
  veri:^integer; //veri adresi baştan sona
  i:integer;  //döngü değişkeni
  sifirsay:integer; //kac tane değer sıfırla dolacak
  artanverisayisi:integer;
   begin
      penceresayisi:=veriboyutu div pencerebuyukluk;
      veri:=data;
      artanverisayisi:=veriboyutu mod pencerebuyukluk;
      if artanverisayisi>0 then begin
           inc (penceresayisi);
           sifirsay:=pencerebuyukluk-artanverisayisi;
           veriboyutu:=veriboyutu+artanverisayisi;//sifirsay;
           gecici:=data;
           inc(gecici,veriboyutu+1);
           for i:=0 to sifirsay-1 do begin
```

EK–1 (Devam) Sesli ifade tanımayla ilgili olarak geliştirilen programın kaynak kodları

```
            gecici^:=0;
            inc(gecici);
          end;  //for end
        end; //if end
         formyakalanan.progressbar1.Position:=0;
        formyakalanan.progressbar1.max:=penceresayisi;
  for i:=0 to penceresayisi-1 do begin
      if pencerelemeyontemi='Kare' then rectang(veri,pencerebuyukluk);
      if pencerelemeyontemi='Blackman' then blackman(veri,pencerebuyukluk);
      if pencerelemeyontemi='Hanning' then hanning(veri,pencerebuyukluk);
      if pencerelemeyontemi='Hamming' then hamming(veri,pencerebuyukluk);
      if pencerelemeyontemi='Barlett' then barlett(veri,pencerebuyukluk);
      inc(veri,pencerebuyukluk);
        formyakalanan.progressbar1.Position:=i;
  end; //for end
end;  //proc end
procedure rectang(data:pointer;windowsize:integer);
 var
 veri:pointerintegerdizi;
 i:integer;
 w:real;
  begin
       veri:=data;
       for i:=0 to windowsize-1 do begin
         w:=1;
         veri^[i]:=round(veri^[i]*w)//kare pencereleme fonksiyonunu uygula
       end;///for
  end;//procedure
procedure Hanning(data:pointer;pencerebuyukluk:integer);
```

EK–1 (Devam) Sesli ifade tanımayla ilgili olarak geliştirilen programın kaynak kodları

```
  var
  veri:pointerintegerdizi;
  n:integer;
  w:real;
begin
    veri:=data;
    for n:=0 to pencerebuyukluk-1 do begin
w:=0.5-0.5*cos(2*pi*n/(pencerebuyukluk-1));
       veri^[n]:=round(veri^[n]*w);
    end;
end;
procedure Hamming(data:pointer;pencerebuyukluk:integer);
  var
  veri:pointerintegerdizi;
  n:integer;
  w:real;
begin
    veri:=data;
    for n:=0 to pencerebuyukluk-1 do begin
       w:=0.54-0.46*cos(2*pi*n/(pencerebuyukluk-1));//hamming pencereleme
fonksiyonu hesapla
       veri^[n]:=round(veri^[n]*w);   //hamming fonksiyonu uygula
    end;
end;
procedure Barlett(data:pointer;pencerebuyukluk:integer);
  var
  veri:pointerintegerdizi;
  i:integer;
  w:real;
```

EK–1 (Devam) Sesli ifade tanımayla ilgili olarak geliştirilen programın kaynak kodları

```
begin
   veri:=data;
   for i:=0 to pencerebuyukluk-1 do begin
     if i<=(pencerebuyukluk div 2) then w:=2*i/(pencerebuyukluk-1)
     else w:=2-(2*i/(pencerebuyukluk-1));
     veri^[i]:=round(veri^[i]*w);
   end;
end;
procedure BlackMan(data:pointer;pencerebuyukluk:integer);
  var
  veri:pointerintegerdizi;
  n:integer;
  w:real;
begin
   veri:=data;
   for n:=0 to pencerebuyukluk-1 do begin
     w:=0.42-0.5*cos(2*pi*n/(pencerebuyukluk-
1))+0.08*cos(4*pi*n/(pencerebuyukluk-1));
     veri^[n]:=round(veri^[n]*w);
   end;
end;
 initialization
end.
//-------------------------------------------------------------------------------------
unit uyakalanan;
interface
uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,
  Dialogs, ExtCtrls, Buttons, StdCtrls, FileCtrl, DB, DBTables, ComCtrls;
```

EK–1 (Devam) Sesli ifade tanımayla ilgili olarak geliştirilen programın kaynak kodları

```
type
  Tformyakalanan = class(TForm)
    pyontem: TRadioGroup;
    pbuyukluk: TRadioGroup;
    Table1: TTable;
    Table1Dosyaadi: TStringField;
    Table1Kelime: TStringField;
    Table1Komut: TStringField;
    Table2: TTable;
    Table2Dosyaadi: TStringField;
    Table2Kelime: TStringField;
    Table2Komut: TStringField;
    Table1Kacinci: TAutoIncField;
    Table2Kacinci: TAutoIncField;
    GroupBox1: TGroupBox;
    Image1: TImage;
    open: TOpenDialog;
    save: TSaveDialog;
    GroupBox2: TGroupBox;
    bilgi: TListBox;
    GroupBox3: TGroupBox;
    uygunlastir: TSpeedButton;
    preemhasis: TSpeedButton;
    pencerele: TSpeedButton;
    fft: TSpeedButton;
    boziliski: TSpeedButton;
    otele: TSpeedButton;
    recognize: TSpeedButton;
    GroupBox4: TGroupBox;
```

EK–1 (Devam) Sesli ifade tanımayla ilgili olarak geliştirilen programın kaynak kodları

```
  Label2: TLabel;
  Edit3: TEdit;
  Label1: TLabel;
  Edit2: TEdit;
  bkaydet: TSpeedButton;
  sozluk: TSpeedButton;
  sifirla: TSpeedButton;
  TrackBar1: TTrackBar;
  GroupBox5: TGroupBox;
  sirali: TListBox;
  GroupBox6: TGroupBox;
  ProgressBar1: TProgressBar;
  Edit1: TEdit;
  Label3: TLabel;
  SpeedButton1: TSpeedButton;
  OpenDialog1: TOpenDialog;
  procedure preemhasisClick(Sender: TObject);
  procedure pyontemClick(Sender: TObject);
  procedure pbuyuklukClick(Sender: TObject);
  procedure uygunlastirClick(Sender: TObject);
  Procedure FFTDataYerlestir(PencereNo:integer);
  Procedure FFTislem;
  procedure fftClick(Sender: TObject);//genisautocorrect;
 Procedure oziliski;
 Procedure upotele;
  procedure bkaydetClick(Sender: TObject);
  Function compare(path:string):real;
  procedure fftsifirla;
  procedure recognizeClick(Sender: TObject);
```

EK–1 (Devam) Sesli ifade tanımayla ilgili olarak geliştirilen programın kaynak kodları

```
   PROCEDURE seviyeesitle(p1:pointer;p2:pointer;size:integer);
   procedure sozlukClick(Sender: TObject);
   procedure FormShow(Sender: TObject);
   procedure pencereleClick(Sender: TObject);
   procedure boziliskiClick(Sender: TObject);
   procedure oteleClick(Sender: TObject);
   procedure TrackBar1Change(Sender: TObject);
   procedure Edit3Change(Sender: TObject);
   procedure sifirlaClick(Sender: TObject);
   procedure FormActivate(Sender: TObject);
   procedure SpeedButton1Click(Sender: TObject);
  private
   { Private declarations }
  public
   { Public declarations }
  end;
var
  formyakalanan: Tformyakalanan;
  enkucukgeneltoplam:real;
  geneltoplam:real;
  kacinci:byte;
implementation
  uses seslib,sinyalisle,math,ffthesapla,sessayisal, tani, sozluk;
{$R *.dfm}
procedure Tformyakalanan.preemhasisClick(Sender: TObject);
var k:integer;
begin
preemhasis.Enabled:=false;
pencerele.Enabled:=true;
```

EK–1 (Devam) Sesli ifade tanımayla ilgili olarak geliştirilen programın kaynak kodları

```
  preemhasisfiltre(@yakalanansesbuf,baslangicbitisyer,preemhasiskatsayi);
  yakalanangoruntule(@yakalanansesbuf,baslangicbitisyer,image1);
  for k:=0 to baslangicbitisyer do yakalanansesbuftut[k]:=yakalanansesbuf[k];
end;
procedure Tformyakalanan.pyontemClick(Sender: TObject);
begin
if  pyontem.ItemIndex=0 then pencerelemeyontemi:='Hamming';
if  pyontem.ItemIndex=1 then pencerelemeyontemi:='Hanning';
if  pyontem.ItemIndex=2 then pencerelemeyontemi:='Kare';
if  pyontem.ItemIndex=3 then pencerelemeyontemi:='Barlett';
if  pyontem.ItemIndex=4 then pencerelemeyontemi:='Blackman';
end;
procedure Tformyakalanan.pbuyuklukClick(Sender: TObject);
begin
if  pbuyukluk.ItemIndex=0 then pencerebuyuklugu:=32;
if  pbuyukluk.ItemIndex=1 then pencerebuyuklugu:=64;
if  pbuyukluk.ItemIndex=2 then pencerebuyuklugu:=128;
if  pbuyukluk.ItemIndex=3 then pencerebuyuklugu:=256;
if  pbuyukluk.ItemIndex=4 then pencerebuyuklugu:=512;
end;
procedure Tformyakalanan.uygunlastirClick(Sender: TObject);  //seviye Uygunlaştır.
  var
  t:real;
  m:real;
  oran:real;
  i:integer;
 begin
preemhasis.Enabled:=true;
uygunlastir.Enabled:=false;
```

EK–1 (Devam) Sesli ifade tanımayla ilgili olarak geliştirilen programın kaynak kodları

```
    t:=0;
    progressbar1.Position:=0;
    progressbar1.max:=baslangicbitisyer div 3;
    for i:=0 to baslangicbitisyer-1 do begin  t:=t+abs(yakalanansesbuf[i]);end;
    m:=t/baslangicbitisyer;
    oran:=20/m;
    for i:=0 to baslangicbitisyer-1 do begin
    yakalanansesbuf[i]:=trunc(yakalanansesbuf[i]*oran);
    if i mod 3=0 then progressbar1.Position:=i;
    end;
    yakalanangoruntule(@yakalanansesbuf,baslangicbitisyer,image1);
 end;
  Procedure TformYakalanan.FFTDataYerlestir(PencereNo:integer);
  var
  i,index,fftpencere:integer;
  r:real;
  begin
    FFTPencere:=pencerebuyuklugu{PencereBuyukluk} div 2;
    index:=PencereNo*FFTPencere;
    for i:=0 to FFTPencere-1 do begin
       r:=round(sqrt(fr[i]*fr[i]+fi[i]*fi[i]));
       if LogFFT then begin
         if r>0 then FFTAlan[i+index]:=round(20*log10(r)) else
FFTAlan[i+index]:=0;
       end
       else FFTAlan[i+index]:=round(r);
    end; //for
  end;//proc
   Procedure TformYakalanan.FFTislem;
```

EK–1 (Devam) Sesli ifade tanımayla ilgili olarak geliştirilen programın kaynak kodları

```
  var
  i,j,yer,w:integer;
  begin
   yer:=0;
     for w:=0 to Penceresayisi-1 do begin
       for i:=0 to Pencerebuyuklugu{Pencerebuyukluk}-1 do begin
        fi[i]:=0;
        fr[i]:=yakalanansesbuf[yer];
        yer:=yer+1;
       end; //for i
     FFT_Hazirla;
     FFTDataYerlestir(w);
    end;//for w
  fftboy:=Baslangicbitisyer; //div 2;
  end;
procedure tformyakalanan.fftsifirla;
var k:integer;
begin
for k:=0 to 32768 do begin    // fft değerlerini
fr[k]:=0;                     //sıfırla
fr[k]:=0;
end;
n:=0;
hicut:=0;
locut:=0;
noise:=0;
lnN:=0;
divn:=0;
fftboy:=0;
```

EK–1 (Devam) Sesli ifade tanımayla ilgili olarak geliştirilen programın kaynak kodları

```
for k:=0 to 50000 do fftalan[k]:=0;
end;
procedure Tformyakalanan.fftClick(Sender: TObject);
var k:integer;
begin
fft.Enabled:=false;
boziliski.Enabled:=true;
fftsifirla;
FFTislem;
yakalanangoruntule(@fftalan,FFTboy,image1);
end;
Procedure TformYakalanan.oziliski;
  var
  ac:array[0..50000] of integer;
  t:real;
  index:integer;
  n,k:integer;
  begin
       formyakalanan.progressbar1.Position:=0;
       formyakalanan.progressbar1.max:=fftboy;
    for k:=0 to fftboy-1 do begin
        t:=0;
          for n:=0 to fftboy-k-1 do  t:=t+fftalan[n]*fftalan[n+k]; //özilişki fonk.
      ac[k]:=trunc(t/(fftboy-k)); //özilişki tamponuna aktar
       formyakalanan.progressbar1.Position:=k;
    end; //for k
    for  k:=0 to fftboy-1 do fftalan[k]:=ac[k];
  end;//proc
Procedure TformYakalanan.upotele; //yukarı ötele
```

EK–1 (Devam) Sesli ifade tanımayla ilgili olarak geliştirilen programın kaynak kodları

```
 var
 eb,i,fark:integer;
begin
  eb:=fftalan[(pencerebuyuklugu div 2)]; //en yüksek enerjinin frekanısnı bul
  for i:=(pencerebuyuklugu div 2) to (pencerebuyuklugu*2)-1 do
      if fftalan[i]>eb then eb:=fftalan[i];  //büyük olanı bul
    if eb<150 then begin //en büyük olanın tepe noktasını 150'ye getir.
      fark:=150-eb;
        for i:=0 to fftboy-1 do fftalan[i]:=fftalan[i]+fark;
    end; //if}
end;
procedure Tformyakalanan.bkaydetClick(Sender: TObject);
var f,fgercekses:file of byte;
veri:^byte;
k:integer;
fftalanokunan:array[0..50000] of integer;
begin
if edit2.Text='' then application.MessageBox('!!!Kelime
Girmediniz!!!','Bilgi',mb_OK);
table1.Active:=true;
table2.Active:=true;
if save.Execute then begin
assignfile(f,save.FileName);
assignfile(fgercekses,save.FileName+'.ger');
rewrite(f);
rewrite(fgercekses);
blockwrite(f,fftalan,fftboy);
blockwrite(fgercekses,pyakala,baslangicbitisyer);
closefile(f);
```

EK–1 (Devam) Sesli ifade tanımayla ilgili olarak geliştirilen programın kaynak kodları

```
closefile(fgercekses);
table1.Append;
table1['dosyaadi']:=save.FileName;
table1['kelime']:=edit2.Text;
if edit1.text<>'' then table1['komut']:=edit1.Text else table1['komut']:=' ';
table1.Post;
table1.Active:=false;
table2.Append;
table2['dosyaadi']:=save.FileName+'.ger';
table2['kelime']:=edit2.Text;
table2.Post;
table2.Active:=false;
end;
{assignfile(f,'26');//.FileName);
reset(f);
blockread(f,fftalanokunan,filesize(f));
for k:=0 to fftboy do begin
 Memo1.Lines.Add(inttostr(k)+'.eleman'+inttostr(fftalan[k]));
end;
for k:=0 to filesize(f) do begin
 Memo2.Lines.Add(inttostr(k)+'.eleman'+inttostr(fftalanokunan[k]));
end;}
end;
procedure Tformyakalanan.recognizeClick(Sender: TObject);
var
  enbuyuk:real;
  kelime:double; //bulma oranı
  yol:string;
  bulunankelime:string;
```

EK–1 (Devam) Sesli ifade tanımayla ilgili olarak geliştirilen programın kaynak kodları

```
  komut:string;
  siralikelime:array[1..200] of string;
  siraliyuzde:array[1..200] of real;
 kackelime:integer;
  i,k:integer;
  temp:real;
  tempkelime:string;
begin
kackelime:=0;
boziliski.Enabled:=false;
bkaydet.Enabled:=true;
kabulesik:=strtoint(edit3.text);
 screen.Cursor:=crhourglass;
bilgi.Items.Clear;
enbuyuk:=0;//Karar değişkeni
 enkucukgeneltoplam:=MAXINT; //uzaklık fark değişkeni
 table1.Active:=false;
 table1.Active:=true;
progressbar1.Max:=table1.RecordCount;
progressbar1.Position:=0;
 while not table1.eof do begin
 kackelime:=kackelime+1;
 yol:=table1['dosyaadi'];
  kelime:=compare(yol);
  siralikelime[kackelime]:=table1['kelime'];
  siraliyuzde[kackelime]:=kelime;
  if kelime>enbuyuk then begin//hangi örnek daha yakın
     enbuyuk:=kelime;
     bulunankelime:=table1['kelime']+'*** Bulma Oranı='+floattostr(kelime);
```

EK–1 (Devam) Sesli ifade tanımayla ilgili olarak geliştirilen programın kaynak kodları

```
      if table1['komut']<>' ' then komut:=table1['komut'];
  end;
 table1.next;
progressbar1.Position:=kackelime;
   end;//while
 if enbuyuk<85 then //bulunan en yakın referans %85'ten az benziyorsa
 application.MessageBox('veri Tabanında Kelime Yok','Uyarı',Mb_Ok)
 else begin //ses tanındı
    if komut<>'' then winexec(pchar(komut),SW_SHOWNORMAL);
    application.MessageBox(pchar(Bulunankelime),'Arama Sonucu',mb_OK);
 end;
 for k:=1 to kackelime-1 do begin
  for i:=k+1 to kackelime do begin
    if siraliyuzde[k]>siraliyuzde[i] then begin
       temp:=siraliyuzde[k];
       siraliyuzde[k]:=siraliyuzde[i];
       siraliyuzde[i]:=temp;
       tempkelime:=siralikelime[k];
       siralikelime[k]:=siralikelime[i];
       siralikelime[i]:=tempkelime;
    end;
  end;
end;
sirali.Items.clear;
for k:=1 to kackelime do begin
  sirali.Items.Add(inttostr(k)+'-'+siralikelime[k]+' = '+floattostr(siraliyuzde[k]));
end;
 screen.Cursor:=crdefault;
 table1.Active:=false;
```

EK–1 (Devam) Sesli ifade tanımayla ilgili olarak geliştirilen programın kaynak kodları

```
end;// proc
Function Tformyakalanan.compare(path:string):real;
var
f:file of byte;
fsize:integer;
p:pointer; //geçici göstergeç
p2:pointer;//geçici göstergeç
pkelime:pointerintegerdizi; //kelimenin yüklendiği adres
fftalanokunan:array[0..50000] of integer;
toplam:real;//Dönüş Değeri
k,i:integer;
fark:integer; //karşılaştırma sonucu
orneksay:integer;//karşılaştırılacak örnek sayısı
m,n:real;
bulunanpensay:integer; //pencere sayısı
Bulmaorani:real; //karşılaştırma sonucu
pensay:integer;//pencere sayısı
pen:integer;
penyer:integer;//Penceredeki pozisyon
begin
  if not fileexists(path) then begin
   application.MessageBox('dosya açılamadı','Arama Sonucu',mb_OK);
    compare:=0;
    exit;
  end;
assignfile(f,path);//.FileName);
reset(f);
fsize:=(filesize(f));
blockread(f,fftalanokunan,fsize);
```

EK–1 (Devam) Sesli ifade tanımayla ilgili olarak geliştirilen programın kaynak kodları

```
//fsize:=fsize div 2;
closefile(f);
toplam:=0;
  if abs(fftboy-fsize)>buyuklukfarkkabul then begin
      application.MessageBox('BOYLAR UYGUN DEĞİL','Arama Sonucu',mb_OK);
     compare:=0;
     exit;
  end;
if fftboy>fsize then orneksay:=fsize else orneksay:=fftboy;
orneksay:=orneksay div 4;
seviyeesitle(@fftalan,@fftalanokunan,orneksay);
bulunanpensay:=0;
geneltoplam:=0;
pensay:=(orneksay div pencerebuyuklugu);
pensay:=pensay*8;
for pen:=0 to pensay-1 do  begin
  toplam:=0;
    for i:=(pencerebuyuklugu div 8)*pen to (pencerebuyuklugu div
8)*pen+(pencerebuyuklugu div 8)-1 do begin
      fark:=fftalan[i]-fftalanokunan[i];
      toplam:=toplam+fark*fark;
  end; //for i
  toplam:=sqrt(toplam);
  geneltoplam:=geneltoplam+toplam;
if toplam<=kabulesik then bulunanpensay:=bulunanpensay+1;
end; //for pen
bulmaorani:=(bulunanpensay/pensay)*100;
bilgi.Items.Add(table1['kelime']+' Bulma Oranı='+floattostr(bulmaorani));
compare:=bulmaorani;
```

EK–1 (Devam) Sesli ifade tanımayla ilgili olarak geliştirilen programın kaynak kodları

```
end; //end function
PROCEDURE Tformyakalanan.seviyeesitle(p1:pointer;p2:pointer;size:integer);
var
ssoylenen:pointerintegerdizi;
sveritabani:pointerintegerdizi;
i,j:integer;
pensay:integer;
topsoylenen,topveritabani:real;
x:integer;
begin
  ssoylenen:=p1;
  sveritabani:=p2;
  pensay:=size div pencerebuyuklugu;
  for i:=0 to pensay-1 do begin
    topsoylenen:=0;
    topveritabani:=0;
    x:=i*pencerebuyuklugu;
      for j:=0 to pencerebuyuklugu-1 do begin
        topsoylenen:=topsoylenen+ssoylenen^[j+x];
        topveritabani:=topveritabani+sveritabani^[j+x];
      end; //for j
    topsoylenen:=topsoylenen/pencerebuyuklugu;
    topveritabani:=topveritabani/pencerebuyuklugu;
    topsoylenen:=topsoylenen-topveritabani;
    for j:=0 to pencerebuyuklugu-1 do begin
      sveritabani^[j+x]:=sveritabani^[j+x]+round(topsoylenen);
    end;//for j
  end;//for i
end;
```

EK–1 (Devam) Sesli ifade tanımayla ilgili olarak geliştirilen programın kaynak kodları

```
procedure Tformyakalanan.sozlukClick(Sender: TObject);
begin
formsozluk.Show;
end;
procedure Tformyakalanan.FormShow(Sender: TObject);
var k:integer;
begin
   for k:=0 to baslangicbitisyer do yakalanansesbuftut[k]:=yakalanansesbuf[k];
  if pencerele.Enabled=true then sifirla.Enabled:=false else sifirla.Enabled:=true;
  pencerelemeyontemi:='Hanning';
  pencerebuyuklugu:=256;
  bilgi.Items.Clear;
  sirali.Items.clear;
  uygunlastir.Enabled:=true;
preemhasis.Enabled:=false;
pencerele.Enabled:=false;
fft.Enabled:=false;
boziliski.Enabled:=false;
otele.Enabled:=false;
recognize.Enabled:=false;
bkaydet.Enabled:=false;
end;
procedure Tformyakalanan.pencereleClick(Sender: TObject);
var k:integer;
begin
pencerele.Enabled:=false;
fft.Enabled:=true;
sifirla.Enabled:=true;
     windowing(@yakalanansesbuf,baslangicbitisyer,pencerebuyuklugu);
```

EK–1 (Devam) Sesli ifade tanımayla ilgili olarak geliştirilen programın kaynak kodları

```
    yakalanangoruntule(@yakalanansesbuf,baslangicbitisyer,image1);
end;
procedure Tformyakalanan.boziliskiClick(Sender: TObject);
begin
boziliski.Enabled:=false;
otele.Enabled:=true;
oziliski;
yakalanangoruntule(@fftalan,FFTboy,image1);
end;
procedure Tformyakalanan.oteleClick(Sender: TObject);
begin
otele.Enabled:=false;
recognize.Enabled:=true;
  upotele;
  yakalanangoruntule(@fftalan,FFTboy,image1);
end;
procedure Tformyakalanan.TrackBar1Change(Sender: TObject);
begin
edit3.Text:=inttostr(trackbar1.position);
end;
procedure Tformyakalanan.Edit3Change(Sender: TObject);
begin
if edit3.text<>'' then trackbar1.Position:=strtoint(edit3.text);
end;
procedure Tformyakalanan.sifirlaClick(Sender: TObject);
var k:integer;
begin
pencerele.Enabled:=true;
sifirla.enabled:=false;
```

EK–1 (Devam) Sesli ifade tanımayla ilgili olarak geliştirilen programın kaynak kodları

```
 for k:=0 to baslangicbitisyer do yakalanansesbuf[k]:=yakalanansesbuftut[k];
 yakalanangoruntule(@yakalanansesbuf,baslangicbitisyer,image1);
end;
procedure Tformyakalanan.FormActivate(Sender: TObject);
begin
  progressbar1.Position:=0;
end;
procedure Tformyakalanan.SpeedButton1Click(Sender: TObject);
begin
 opendialog1.Execute;
  edit1.Text:=opendialog1.FileName;
end;
end.
//---------------------------------------------------------------------------------------------
unit sesyakalaayar;
interface
uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,
  Dialogs, ComCtrls, StdCtrls, Buttons, ExtCtrls;
type
  Tfyakalaayar = class(TForm)
    GroupBox5: TGroupBox;
    Panel1: TPanel;
    GroupBox1: TGroupBox;
    TrackBar1: TTrackBar;
    Edit1: TEdit;
    GroupBox2: TGroupBox;
    Edit2: TEdit;
    TrackBar2: TTrackBar;
```

EK–1 (Devam) Sesli ifade tanımayla ilgili olarak geliştirilen programın kaynak kodları

```
    GroupBox3: TGroupBox;
    Edit3: TEdit;
    TrackBar3: TTrackBar;
    GroupBox4: TGroupBox;
    TrackBar4: TTrackBar;
    Edit4: TEdit;
    SpeedButton1: TSpeedButton;
    SpeedButton2: TSpeedButton;
    procedure TrackBar1Change(Sender: TObject);
    procedure Edit1Change(Sender: TObject);
    procedure TrackBar2Change(Sender: TObject);
    procedure Edit2Change(Sender: TObject);
    procedure TrackBar3Change(Sender: TObject);
    procedure Edit3Change(Sender: TObject);
    procedure Edit4Change(Sender: TObject);
    procedure TrackBar4Change(Sender: TObject);
    procedure SpeedButton1Click(Sender: TObject);
    procedure SpeedButton2Click(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
  end;
var
  fyakalaayar: Tfyakalaayar;
implementation
  uses seslib;
{$R *.dfm}
procedure Tfyakalaayar.TrackBar1Change(Sender: TObject);
```

EK–1 (Devam) Sesli ifade tanımayla ilgili olarak geliştirilen programın kaynak kodları

```
begin
edit1.Text:=inttostr(trackbar1.Position);
sesyakalaorneksayisi:=trackbar1.Position;
end;
procedure Tfyakalaayar.Edit1Change(Sender: TObject);
begin
if edit1.text<>'' then  trackbar1.position:=strtoint(edit1.text) else
trackbar1.Position:=0;
sesyakalaorneksayisi:=trackbar1.position;
end;
procedure Tfyakalaayar.TrackBar2Change(Sender: TObject);
begin
edit2.Text:=inttostr(trackbar2.Position);
sessusorneksayisi:=trackbar2.Position;
end;
procedure Tfyakalaayar.Edit2Change(Sender: TObject);
begin
if edit2.text<>'' then  trackbar2.position:=strtoint(edit2.text) else
trackbar2.Position:=0;
sessusorneksayisi:=trackbar2.Position;
end;
procedure Tfyakalaayar.TrackBar3Change(Sender: TObject);
begin
edit3.Text:=inttostr(trackbar3.position);
sesyakalaesikdeger:=trackbar3.position;
end;
procedure Tfyakalaayar.Edit3Change(Sender: TObject);
begin
```

EK–1 (Devam) Sesli ifade tanımayla ilgili olarak geliştirilen programın kaynak kodları

```
if edit3.text<>'' then  trackbar3.position:=strtoint(edit3.text) else
trackbar3.Position:=0;
sesyakalaesikdeger:=trackbar3.Position;
end;
procedure Tfyakalaayar.Edit4Change(Sender: TObject);
begin
if edit4.text<>'' then  trackbar4.position:=strtoint(edit4.text) else
trackbar4.Position:=0;
sessusesikdeger:=trackbar4.Position;
end;
procedure Tfyakalaayar.TrackBar4Change(Sender: TObject);
begin
edit4.Text:=inttostr(trackbar4.position);
sessusesikdeger:=trackbar4.position;
end;
procedure Tfyakalaayar.SpeedButton1Click(Sender: TObject);
begin
  edit1.Text:='100';
  edit2.Text:='50';
  edit3.Text:='40';
  edit4.Text:='40';
end;
procedure Tfyakalaayar.SpeedButton2Click(Sender: TObject);
begin
fyakalaayar.close;
end;
end.
//----------------------------------------------------------------------------------------------
unit seslib;
```

EK–1 (Devam) Sesli ifade tanımayla ilgili olarak geliştirilen programın kaynak kodları

```
interface
uses graphics,classes,extctrls;
const
  SesAlanBuyukluk=10000;
 type
   PbyteDizi=^DiziByte;
   DiziByte=array[0..50000] of byte;
   PwordDizi=^Diziword;
   Diziword=array[0..50000] of word;
   PointerintegerDizi=^Diziinteger;
   Diziinteger=array[0..50000] of integer;
   veritipi=(_None,_Byte,_Word,_Integer);
   SesYakalaDurum_=(_BaslangicBul,_BitisBul,_SesYakalandi);
   Var
   OncekiVeri:array[0..48000] of integer;
   dosyayolu:string;  //dbpath
   sayisalsessinyali:pointerintegerdizi;
   _sestamponalan:integer;
   baslangicbitisyer:integer;
   yakalanansesbuf:array[0..50000] of integer;//pintegerdizi;
   yakalanansesbuftut:array[0..50000] of integer;
   pyakala:array[0..50000] of integer; //yakalanan gerçek ses
   gercekses:pchar;
   fftalan:array[0..50000] of integer;
   fftboy:integer;
   hammingflag:boolean=false;
   preemhasiskatsayi:real=0.95;
   esik:integer=80000;
   sesyakalaorneksayisi:integer=100; //ses başlangıcı kabulü örnek sayısı
```

EK–1 (Devam) Sesli ifade tanımayla ilgili olarak geliştirilen programın kaynak kodları

```
sessusorneksayisi:integer=50;
sesyakalaesikdeger:integer=40;
sessusesikdeger:integer=40;
bitpersamples:integer=8; //örnekleme çözünürlüğü
sonsize:integer=0;
renkenerji:Tcolor=Clblue;
eskidosya:string;
gurultuesik:integer=10;
logfft:boolean;
kabulesik:integer=30;
buyuklukfarkkabul:integer=32000;
renkcount:integer=clred;
uygulapreempfilter:boolean;
uygulapencere:boolean;
uygulafft:boolean;
uygulaautocorr:boolean;
Uygulagucortala:boolean;
uygulaformant:boolean;
uygulaseviye:boolean;
sesanalizsinyalrenk:TColor;
sesanalizarkarenk:TColor;
sesanalizgrrenk:TColor;
sesyakalasinyalrenk:TColor;
sesyakalaarkarenk:TColor;
sesyakalagrrenk:TColor;
fftsinyalrenk:TColor;
fftgrrenk:TColor;
fftarkarenk:TColor;
sesyakaladurum:sesyakaladurum_ ;
```

EK–1 (Devam) Sesli ifade tanımayla ilgili olarak geliştirilen programın kaynak kodları

```
    kaceleman:integer;
     procedure veriyigoruntule(veri:pointer;image:timage;size:integer);
     procedure enerjigoruntule(veri:pointer;image:timage;size:integer);
     procedure yakalanangoruntule(veri:pointer;miktar:integer;image:timage);
     procedure oncekiveribosalt;
     procedure yakalanangoruntuleson(veri:pointer;miktar:integer;image:timage);
     Function PowerBin(n:integer):integer;
     procedure yakalanangoruntulesozluk(veri:pointer;miktar:integer;image:timage);
implementation
uses sessayisal,goruntu,tani;
procedure veriyigoruntule(veri:pointer;image:timage;size:integer);
begin
  sesgoruntu.onrenk:=clblue;//sesanalizsinyalrenk;
  sesgoruntu.arkarenk:=clwhite;//sesanalizarkarenk;
  sesgoruntu.grrenk:=clred;//sesanalizgrrenk;
  sesgoruntu.ortaeksen:=true;
  sesgoruntu.ustbinme:=false;
  sesgoruntu.kaynak:=veri;
  sesgoruntu.verimiktar:=size;
  sesgoruntu._image:=image;
  sesgoruntu.goruntuleolcekli;  //goruntu unitinede
    sesgoruntu.isaretlisinyal:=true;
  end;
procedure enerjigoruntule(veri:pointer;image:timage;size:integer);
begin
sesgoruntu.onrenk:=renkenerji;
sesgoruntu.ustbinme:=true;
sesgoruntu.kaynak:=veri;
sesgoruntu.verimiktar:=size;
```

EK–1 (Devam) Sesli ifade tanımayla ilgili olarak geliştirilen programın kaynak kodları

```
sesgoruntu._image:=image;
sesgoruntu.goruntuleolcekli;
end;
procedure yakalanangoruntule(veri:pointer;miktar:integer;image:timage);
begin
  sesgoruntu.onrenk:=clblue;//sesyakalasinyalrenk;
  sesgoruntu.arkarenk:=clwhite;//sesyakalaarkarenk;
  sesgoruntu.grrenk:=clred;//sesyakalagrrenk;
  sesgoruntu.ortaeksen:=true;
  sesgoruntu.kaynak:=veri;
  sesgoruntu.verimiktar:=miktar;
  sesgoruntu._image:=image;
    sesgoruntu.isaretlisinyal:=true;
  sesgoruntu.clearcanvas;
   sesgoruntu.drawgrid;
  sesgoruntu.goruntuleolcekli;
end;
procedure yakalanangoruntuleson(veri:pointer;miktar:integer;image:timage);
begin
  sesgoruntu.onrenk:=clblue;//sesyakalasinyalrenk;
  sesgoruntu.arkarenk:=clwhite;//sesyakalaarkarenk;
  sesgoruntu.grrenk:=clred;//sesyakalagrrenk;
  sesgoruntu.ortaeksen:=true;
  sesgoruntu.kaynak:=veri;
  sesgoruntu.verimiktar:=miktar;
  sesgoruntu._image:=ftani.image1;
  sesgoruntu.clearcanvas;
   sesgoruntu.drawgrid;
  sesgoruntu.goruntuleolcekli;
```

EK–1 (Devam) Sesli ifade tanımayla ilgili olarak geliştirilen programın kaynak kodları

```
end;
procedure yakalanangoruntulesozluk(veri:pointer;miktar:integer;image:timage);
begin
  sesgoruntu.onrenk:=clblue;//sesyakalasinyalrenk;
  sesgoruntu.arkarenk:=clwhite;//sesyakalaarkarenk;
  sesgoruntu.grrenk:=clred;//sesyakalagrrenk;
  sesgoruntu.ortaeksen:=true;
  sesgoruntu.kaynak:=veri;
  sesgoruntu.verimiktar:=miktar;
  sesgoruntu._image:=image;
  sesgoruntu.ustbinme:=false;
  sesgoruntu.isaretlisinyal:=false;
  sesgoruntu.clearcanvas;
   sesgoruntu.drawgrid;
  sesgoruntu.goruntuleolcekli;
end;
procedure oncekiveribosalt;
begin
 fillchar(oncekiveri,sizeof(oncekiveri),0);
 fillchar(sayisalsessinyali,sizeof(sayisalsessinyali^),0);
end;
Function PowerBin(n:integer):integer; //ikili sistemde kaç basamak var
var
pwr2,cnt:integer;
begin
  pwr2:=1;
  cnt:=0;
  while pwr2<n do begin
    pwr2:=pwr2*2;
```

EK–1 (Devam) Sesli ifade tanımayla ilgili olarak geliştirilen programın kaynak kodları

```
      inc(cnt);
    end;
    powerbin:=cnt;
end;
initialization
oncekiveribosalt;
end.
//---------------------------------------------------------------------------------------------
unit sozluk;
interface
uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,
  Dialogs, DB, DBTables, Buttons, StdCtrls, FileCtrl, ExtCtrls, Audio,
  Grids, DBGrids, Menus;
type
  Tformsozluk = class(TForm)
    Table2: TTable;
    Table2Dosyaadi: TStringField;
    Table2Kelime: TStringField;
    Table2Komut: TStringField;
    Audio1: TAudio;
    DataSource1: TDataSource;
    GroupBox1: TGroupBox;
    DBGrid1: TDBGrid;
    GroupBox2: TGroupBox;
    Image1: TImage;
    GroupBox3: TGroupBox;
    yukle: TSpeedButton;
    Label1: TLabel;
```

EK–1 (Devam) Sesli ifade tanımayla ilgili olarak geliştirilen programın kaynak kodları

```
    Edit1: TEdit;
    cal: TSpeedButton;
    SpeedButton2: TSpeedButton;
    Sil: TSpeedButton;
    Table2Kacinci: TAutoIncField;
    Table1: TTable;
    Table1Dosyaadi: TStringField;
    Table1Kelime: TStringField;
    Table1Komut: TStringField;
    Table1Kacinci: TAutoIncField;
    PopupMenu1: TPopupMenu;
    procedure yukleClick(Sender: TObject);
    procedure calClick(Sender: TObject);
    procedure Button4Click(Sender: TObject);
    procedure DBGrid1DblClick(Sender: TObject);
    procedure FormActivate(Sender: TObject);
    procedure FormClose(Sender: TObject; var Action: TCloseAction);
    procedure SpeedButton2Click(Sender: TObject);
    procedure SilClick(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
  end;
var
  formsozluk: Tformsozluk;
  LeftStream,RightStream:TMemoryStream;
  pyakalasozluk:array[0..50000] of integer; //yakalanan gerçek ses
  kaceleman1:integer;
```

EK–1 (Devam) Sesli ifade tanımayla ilgili olarak geliştirilen programın kaynak kodları

```
implementation
uses seslib;
{$R *.dfm}
procedure Tformsozluk.yukleClick(Sender: TObject);
var
f:file of byte;
veri:array[0..50000] of integer;
dosyaboyu,k:integer;
begin
for k:=0 to 50000 do pyakalasozluk[k]:=0;
assignfile(f,table2['dosyaadi']);//.FileName);
reset(f);
dosyaboyu:=(filesize(f));
edit1.Text:=inttostr(dosyaboyu);
blockread(f,pyakalasozluk,dosyaboyu);
kaceleman1:=dosyaboyu;
closefile(f);
yakalanangoruntulesozluk(@pyakalasozluk,dosyaboyu,image1);
end;
procedure Tformsozluk.Button4Click(Sender: TObject);
begin
 Audio1.Player.Stop;
end;
Procedure Tformsozluk.DBGrid1DblClick(Sender: TObject);
begin
  yukle.click;
  cal.click;
end;
procedure Tformsozluk.FormActivate(Sender: TObject);
```

EK–1 (Devam) Sesli ifade tanımayla ilgili olarak geliştirilen programın kaynak kodları

```
var k:integer;
begin
      table2.Active:=true;
      table1.Active:=true;
for k:=0 to 50000 do pyakalasozluk[k]:=0;
    yakalanangoruntulesozluk(@pyakalasozluk,3000,image1)
end;
procedure Tformsozluk.FormClose(Sender: TObject; var Action: TCloseAction);
begin
 table2.Active:=false;
end;
procedure Tformsozluk.calClick(Sender: TObject);
var
i:byte;
k:integer;
begin
   LeftStream.Free; RightStream.Free;
   LeftStream := nil; RightStream:=nil;
   LeftStream :=TMemoryStream.Create;
   RightStream:=TMemoryStream.Create;
    LeftStream.Position:=0;
for k:=0 to kaceleman1 do begin
if bitpersamples=8 then LeftStream.Write(pyakalasozluk[k],1);
end;
if (LeftStream<>nil) and (LeftStream.Size>0) then begin
   LeftStream.Position:=0;
   if RightStream.Size>0 then RightStream.Position:=0
   else begin
    RightStream.Free;
```

EK–1 (Devam) Sesli ifade tanımayla ilgili olarak geliştirilen programın kaynak kodları

```
     RightStream:=nil;
   end;
   Audio1.Player.Play(LeftStream,RightStream,0);
 end;
end;
procedure Tformsozluk.SpeedButton2Click(Sender: TObject);
begin
Audio1.Player.Stop;
end;
procedure Tformsozluk.SilClick(Sender: TObject);
var
  sonuc:integer;
begin
Table1.Locate('kacinci',table2['kacinci'],[loPartialKey]);
 sonuc:=application.MessageBox('Kaydı Silmek İstediğinize Emin
misiniz?','Uyarı',mb_YESNO);
 if sonuc=6 then begin
  { deletefile('c:\ses tanıma\sesler\7.ger');
     deletefile('c:\ses tanıma\sesler\7');}
   DeleteFile(table2['dosyaadi']);
   DeleteFile(table1['dosyaadi']);
   table1.delete;
   table2.delete;
   application.MessageBox('Kaydınız Silindi','Bilgi',mb_OK);
 end;
 end;
end.
//-------------------------------------------------------------------------------------------
unit goruntu;
```

EK–1 (Devam) Sesli ifade tanımayla ilgili olarak geliştirilen programın kaynak kodları

```
interface
uses extctrls,graphics,windows,seslib;
Type
  goruntuleme=class
            _image:timage;
            kaynak:pointer;  //veri adresi
            verimiktar:integer;  //veri büyüklük
            tip:veritipi;   //verinin temel boyutu
            arkarenk:tcolor; //canvas rengi
            onrenk:tcolor;  //kalemrengi
            _drawgrid:boolean;
            grrenk:tcolor; //grid trengi
            gridx:integer; //grid büyükklük
            gridy:integer;//grid büyüklük
            yariyuk:boolean;//sinyalin y üzerrindeki yeri
            boslukx:integer; //sol ve
            bosluky:integer;//alt boşluk miktarı
            ortaeksen:boolean;
            xaxisrenk:integer;
            ortaaxisrenk:integer;
            isaretlisinyal:boolean;
            ustbinme:boolean;
            procedure clearcanvas;
            procedure drawgrid;
            procedure drawaxis;
            procedure ekranhazirla;
  public
    constructor create;virtual;
    procedure goruntuleolcekli;virtual;
```

EK–1 (Devam) Sesli ifade tanımayla ilgili olarak geliştirilen programın kaynak kodları

```
 end;
var
sesgoruntu:goruntuleme;
implementation
constructor goruntuleme.create;
begin
 _image:=nil;
 kaynak:=nil;
 verimiktar:=0;
 tip:=_Byte;
 arkarenk:=clblue;
 onrenk:=clwhite;
 _drawgrid:=true;
 grrenk:=clgray;
 gridx:=10;
 gridy:=10;
 boslukx:=0;
 bosluky:=0;
 ortaeksen:=true;
 xaxisrenk:=clblack;
 ortaaxisrenk:=clblack;
 isaretlisinyal:=true;
 ustbinme:=true;
end;
procedure goruntuleme.clearcanvas;
var
alan:trect;
begin
_image.Canvas.Brush.Color:=arkarenk;
```

EK–1 (Devam) Sesli ifade tanımayla ilgili olarak geliştirilen programın kaynak kodları

```
alan.left:=0;
alan.top:=0;
alan.Right:=_image.Width;
alan.bottom:=_image.Height;
_image.Canvas.FillRect(alan);
end;
procedure goruntuleme.drawgrid;
var
  width:integer;
  height:integer;
  xadet:integer;
  yadet:integer;
  x,y:integer;
begin
    if not _drawgrid then exit;
    _image.Canvas.Pen.Color:=grrenk; //gridrengi
    width:=_image.Width-boslukx;
    height:=_image.Height-bosluky;
    xadet:=width div gridx+1;
    yadet:=height div gridy+1;
    for x:=0 to xadet-1 do
    begin
      _image.Canvas.moveto(x*gridx+boslukx,0);
      _image.Canvas.lineto(x*gridx+boslukx,height);
    end;
    for y:=0 to yadet-1 do
    begin
     _image.Canvas.moveto(0,y*gridy+bosluky);
     _image.Canvas.lineto(width,y*gridy+bosluky);
```

EK–1 (Devam) Sesli ifade tanımayla ilgili olarak geliştirilen programın kaynak kodları

```
    end;
end;
procedure goruntuleme.drawaxis;
var
width:integer;
height:integer;
orta:integer;
begin
    width:=_image.Width-boslukx;
    height:=_image.Height-bosluky;
    orta:=height div 2;
    if boslukx>0 then begin
      _image.Canvas.Pen.Color:=xaxisrenk;
      _image.Canvas.MoveTo(boslukx,0);
      _image.Canvas.LineTo(boslukx,height);
     end;
     if bosluky>0 then begin
      _image.Canvas.Pen.Color:=xaxisrenk;
      _image.Canvas.MoveTo(boslukx,height);
      _image.Canvas.LineTo(_image.Width,height);
     end;
     if ortaeksen then
     begin
       _image.Canvas.Pen.Color:=ortaaxisrenk;
      _image.Canvas.MoveTo(boslukx,orta);
      _image.Canvas.LineTo(_image.Width,orta);
    end;
end;
procedure goruntuleme.ekranhazirla;
```

EK–1 (Devam) Sesli ifade tanımayla ilgili olarak geliştirilen programın kaynak kodları

```
begin
 clearcanvas;
 drawgrid;
 drawaxis;
end;
procedure goruntuleme.goruntuleolcekli;
var
kaynakdatayer,xoran,yoran:real;
x,y,i,height,tipsize:integer;
bkaynakp:^integer;
grafikveri:pointerintegerdizi;
veriI:integer;
begin
  if not ustbinme then   ekranhazirla;
  if kaynak=nil then exit;
  height:=_image.Height;
  if isaretlisinyal then height:=height div 2;
  kaynakdatayer:=0;   //ses verisi üzerinde belirli aralıklarla veriyi alır.
  //renkleri ver
  _image.Canvas.Brush.color:=arkarenk;
  _image.Canvas.Pen.Color:=onrenk;
  //ölçekleri hesapla
  xoran:=verimiktar/_image.Width; //veri miktarı resim genişliğine bölünerek adım
sayısı belirleniyor.
  if bitpersamples=8 then tipsize:=256 else tipsize:=65536;
  if isaretlisinyal then yoran:=height*2/tipsize else yoran:=height/tipsize;
  grafikveri:=kaynak; //integer pointer
//------Her yeni ses değeri geldiğinde baştan çizilmeye başlanıyor.----
  veriI:=grafikveri^[trunc(kaynakdatayer)];
```

EK–1 (Devam) Sesli ifade tanımayla ilgili olarak geliştirilen programın kaynak kodları

```
  if bitpersamples=16 then
    if veriI>=32768 then veriI:=veriI-65536;
    y:=height-trunc(yoran*veriI); //veri y ekseni üzerindeki yeri
    if y>_image.Height then y:=_image.Height-1;
    if y<0 then y:=0;
    _image.Canvas.MoveTo(0,y);
    for x:=1 to _image.Width-1 do begin
      kaynakdatayer:=kaynakdatayer+xoran; //belirlenen adım aralıklarıyla
      veriI:=grafikveri^[trunc(kaynakdatayer)]; //veriyi al
      if bitpersamples=16 then
          if veriI>=32768 then  veriI:=veriI-65536;
      y:=height-trunc(yoran*veriI);
      if y>_image.Height then y:=_image.Height-1;
      if y<0 then y:=0;
      _image.Canvas.LineTo(x,y);//çiz
     end;
end;
initialization
 sesgoruntu:=goruntuleme.create;
end.
//-----------------------------------------------------------------------------------------------
unit ffthesapla;
interface
 uses math;
type
  gerceldizi=array [0..32768] of real;
var
fr,fi:gerceldizi;
n,hicut,locut,noise,lnN:integer;
```

EK–1 (Devam) Sesli ifade tanımayla ilgili olarak geliştirilen programın kaynak kodları

```
divn:real;
 Procedure FFT_Hazirla;
implementation
uses seslib,sinyalisle;
  Function Power2(i:integer):integer;
     var n,k:integer;
    begin
       n:=1;
       for i:=1 to i do n:=n*2;
       power2:=n;
   end;
   Procedure FFT_Hesapla(var fr,fi:gerceldizi;lnN:integer;sign:integer);
    var
      t,nd2,nm1:integer;
      i,j,k,l,le,le1,ip:integer;
      tmp,ur,ur1,ui,wr,wi,tr,ti,dv:real;
    Begin
       n:=power2(lnN);
       nd2:=N div 2;
       nm1:=n-1;
       j:=1;
       for i:=1 to N-1 do begin//Ters Biti Dizilimi yapılıyor
          if i<j then begin
             tmp:=fr[i];
             fr[i]:=fr[j];
             fr[j]:=tmp;
             tmp:=fi[i];
             fi[i]:=fi[j];
             fi[j]:=tmp;
```

EK–1 (Devam) Sesli ifade tanımayla ilgili olarak geliştirilen programın kaynak kodları

```
 end;//if
 k:=nd2;
 while k<j do begin
   j:=j-k;
   k:=k div 2;
 end;//while
 j:=j+k
end; //for
for l:=1 to lnN do begin //Zaman doneminden frekans donemine geçiliyor
 le:=Power2(l);
 le1:=le div 2;
 ur:=1;
 ui:=0;
 wr:=cos(pi/le1);
 wi:=sign*sin(pi/le1);
 for j:=1 to  le1 do begin
  i:=j;
    while(i<=N) do begin
      ip:=i+le1;
      tr:=fr[ip]*ur-fi[ip]*ui;
      ti:=fr[ip]*ui+fi[ip]*ur;
      fr[ip]:=fr[i]-tr;
      fi[ip]:=fi[i]-ti;
      fr[i]:=fr[i]+tr;
      fi[i]:=fi[i]+ti;
      i:=i+le;
    end;//while
  ur1:=ur*wr-ui*wi;
  ui:=ur*wi+ui*wr;
```

EK–1 (Devam) Sesli ifade tanımayla ilgili olarak geliştirilen programın kaynak kodları

```
          ur:=ur1;
        end;//for j
     end; //for  l
   for i:=1 to N do begin //Düzeltme Faktörünü Uygula
       fr[i]:=fr[i]*divn;
       fi[i]:=fi[i]*divN;
   end;
  end;//Proc
Procedure Init(lnN:integer);
var
i:integer;
begin
  n:=power2(lnN);
  for i:=0 to n do begin
    fi[i]:=0;
    fr[i]:=sin(random(10)*i*pi/180);
  end;//for
  divn:=1/sqrt(n*1.0);
end;
  Procedure FFT_Hazirla;
    var
     PSource:^integer;
     HFFT:^integer;
     i:integer;
     r:real;
   Begin
       n:=PencereBuyuklugu;
       lnN:=PowerBin(n);  //Pencere büyüklüğünün 2'nin üstü biçimi
       divn:=1/sqrt(n*1.0); //düzeltme faktörü
```

EK–1 (Devam) Sesli ifade tanımayla ilgili olarak geliştirilen programın kaynak kodları

```
      // Hicut:=ustkesfrekans; //üst kesim frekansı
       //Locut:=altkesfrekans;//alt kesim frekansı
       Noise:=gurultuesik;//Gürültü Sınırı
       FFT_Hesapla(fr,fi,LnN,-1); //Hızlı Fourier dönüşüm
       for i:=0 to n-1 do begin
         if round(sqrt(fr[i]*fr[i]+fi[i]*fi[i]))<noise then begin//gürültü at
           fr[i]:=0;
           fi[i]:=0;
         end;//if
       end;//for
   End; //Proc
end.
```

ÖZGEÇMİŞ

Kişisel Bilgiler

Soyadı, adı : KARACI, Abdulkadir

Uyruğu : T.C.

Doğum tarihi ve yeri : 27.03.1977 Samsun

Medeni hali : Evli

Telefon : 0 (366) 212 67 86

Faks :

e-mail : akaraci@gazi.edu.tr

Eğitim

Derece	Eğitim Birimi	Mezuniyet tarihi
Lisans	Gazi Üniversitesi/ Bilgisayar Eğitimi Bölümü	2001
Lise	19 Mayıs Lisesi	1994

İş Deneyimi

Yıl	Yer	Görev
2003-	Gazi Üniversitesi Kastamonu Eğitim Fakültesi B.Ö.T.E	Öğretim Görevlisi

Yabancı Dil

İngilizce

Yayınlar

1. Karacı, A.(Ortak), "Bilgisayara Giriş", ***Lisans Yayıncılık***, İstanbul, 2005

Hobiler

Bilgisayar teknolojileri, Bilgisayar programlama