BİLGİSAYAR ORTAMINDA SESLİ KOMUTLARI TANIMA: ÖRÜNTÜ TANIMA YÖNTEMİ

Halil İbrahim BÜLBÜL
G.Ü. Endüstriyel Sanatlar Eğitim Fakültesi, Bilgisayar Eğitimi, Ankara.
Abdulkadir KARACI
Kastamonu Üniversitesi, Eğitim Fakültesi, B.Ö.T.E.B., Kastamonu.

Özet

Bu çalışmada, Örüntü tanıma yöntemini kullanarak ses tanıyan bir bilgisayar yazılımı hazırlanmış ve ses tanımayla ilgili temel ilkeler sistematik bir içerikte anlatılmıştır. Sesli komutları tanımada çok önemli olan özellik vektörlerinin çıkarılması ve ses üzerinde yapılacak olan ön işlemler ayrıntılı bir şekilde verilmeye çalışılmıştır. Ayrıca geliştirilen programda ses tanıma denemeleri yapılmış ve sonuçlara ilişkin bilgiler özetlenmiştir..

Anahtar Kelimeler: *Ses tanıma, Örüntü Tanıma, İnsan Bilgisayar Etkileşimi, Sinir Ağları*

SPEECH COMMAND RECOGNITION IN COMPUTER: PATTERN RECOGNITION METHOD

Abstract

In this study, a software recognizing voice has been prepared using pattern recognition method. The basic principles about voice recognition have also been given systematically. Character vector's sounding, which is very important in vocal commands and voicing, has been analysed in detail. In addition, voice recognition essays were done on the developed program and the results have been given.

Keywords: *Speech Recognition, Pattern Recognition, Computer and Human Interaction, Neural Network*

1. Giriş

Bu çalışmada sesli ifade tanıma yöntemlerinden olan örüntü tanıma yönteminin ilkeleri ortaya konularak bu yöntemin uygulandığı bir bilgisayar programının geliştirilmesi amaçlanmaktadır. Sesli ifade tanıma insan-bilgisayar arası iletişim için önemli yararlar sağlamaktadır. Sesli ifade verilerinin elde edilmesi çok kolaydır. Klavye ve diğer veri giriş yöntemlerini kullanmak gibi özel bir yetenek gerektirmez. Sesli ifade kullanarak metinlerin elektronik ortama yazı olarak aktarılması oldukça hızlı olabilmektedir. Kullanıcıya hareket serbestliği ve ellerini kullanabilme imkanı sağlamaktadır **(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.

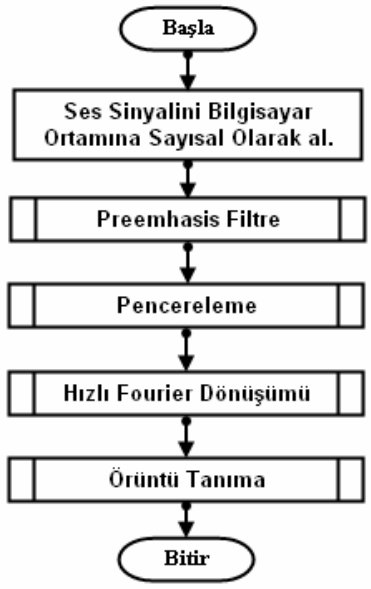
Ses sinyalinin parametreleri yapılacak filtreleme ve analizler sonucunda elde edilirler. Ses tanıma seçilen ses tanıma yöntemlerinden biriyle gerçekleştirilir. Ses tanımada belirgin olarak aşağıdaki yöntemler kullanılmaktadır **(2).**

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

Örüntü tanıma yönteminde özellikleri çıkarılmış 2 ses vektörü vardır. Bu vektörlerin birbirine olan benzerlikleri veya zıtlıkları çeşitli uzaklık ölçüm yöntemleri kullanılarak belirlenir. Örüntü tanıma işleminde ilk yapılması gereken işlenmemiş ham ses verisine bazı önişlemler uygulamaktır. Bu ön işlemler aşağıda anlatılmaktadır.

2. Ses Tanımada Ön İşlemler

Ses tanıma işleminin 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 1'de görülmektedir **(2)**.



Şekil 1:Ses tanımaya genel bakış

2.1 Preemhasis Filtre

Filtereleme işlemi zaman ve sıklık evreninde ayrı, ayrı ele alınabilir. Ön filtreleme işlemi, çoğu kez zaman evreninde uygulanmaktadır **(3).** Sıradaki ses sinyali örneğinin, sabit bir katsayı ile çarpılmış bir önceki örnekten farkının alınmasıyla elde edilir **(2).** Preemhasis filtrenin fonksiyonu eşitlik 1'deki gibidir **(4).**

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

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

2.2. Pencereleme

Sinyal işleme aşamasında, genellikle sesli ifadeleri temsil eden sözcüklerin tümünü bir seferde ele alma imkanı bulunmaz. Bu nedenle anlamlı uzunluklara bölünmesi gerekir. Bir seferde işleme alınacak sözcük kümesi, sinyal içinde bir pencere olarak tanımlanır **(2).** Pencereleme teorisi digital sinyal işlemede eskiden beri araştırılan bir konudur. Pencerelemenin Rectangular, Hamming, Hanning, Blackman, Barlett, Kaiser gibi birçok tipi vardır. Hamming window çok kullanılır **(5).**

2.3. 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 spektrumunun kestirimini yapmaktadır **(2).** Fourier dönüşüm fonksiyonu eşitlik 2'deki gibidir **(6).**

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

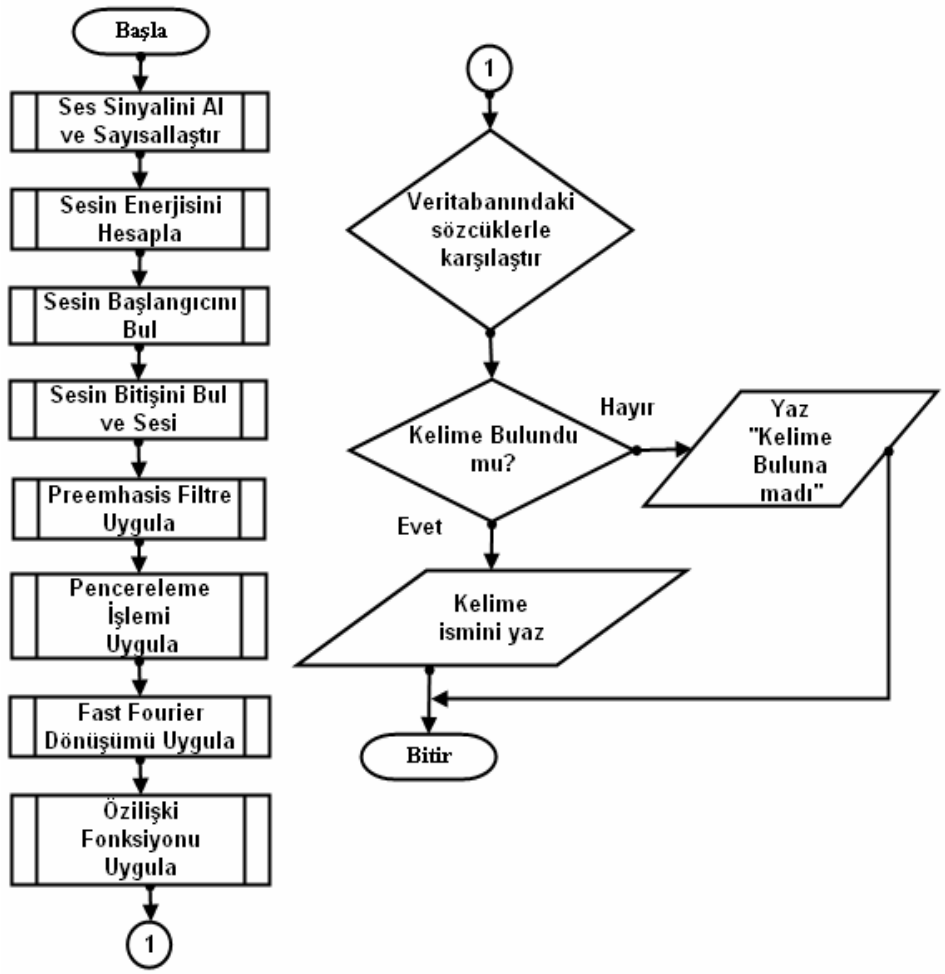
2.4. Sesin Enerjisi

Sesin başlangıç ve bitişinin tespit edilebilmesi için şiddetinin yani enerjisinin belirlenmesi gerekir. Enerji parametreleri bitiş noktasını belirlemede 1970'den beri kullanılmaktadır. Ses sinyalinin enerjisinin hesaplama diğer işlemlerle karşılaştırıldığında basit bir işlemdir. Ses sinyalini işlemede sıfırdan geçiş oranı (Zero Crossing Rate-ZCR) ve kısa zaman enerjisi (short time energy) olmak üzere iki temel parametre vardır. Genellikle bu iki parametre diğerlerinden daha önce hesaplanır. Yanlış tanıma sonuçlarının yarıya yakını bitiş noktasının yanlış tespitindendir **(7).**

Ses tanıma sistemlerinin 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 **(2).** Ses sinyalinin sıfırdan geçiş sayısı zero crossing rate olarak bilinir. Sesli ifadelerin yer aldığı kesimlerde zero crossing rate olarak bilinen bu değer düşük olmaktadır. Bu özellik, sesli ifadelerin başlangıç ve bitiş noktalarını belirlemede kullanılmaktadır **(3).**

3. Ses Tanıma Yazılımının Tasarlanması

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. Eğer bulunan kelimeye bağlı çalıştırılması istenen bir komut yada program varsa onu çalıştırır. Geliştirilen yazılımın en genel akış şeması aşağıda gösterilmektedir.



Şekil 2:Geliştirilen yazılımın genel akış şeması

3.1 Sesin Bilgisayar Ortamına Alınması

Ses bilgisayar ortamına mikrofon aracılığıyla alınmaktadır. Sesi bilgisayar ortamına almak için audio1 isimli bileşenin record olayı kullanılmaktadır. Record olayının kodları aşağıda görülmektedir.

```
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));
   gercekses:=lp; Veriyikutupla(lp,sonsize);
  if kutupkont.Checked=true then  Veriyigoruntule (sayisalsessinyali, image1, sonsize);
Enerji;  sesyakala;  end;
```

3.2 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 **(2).**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. Veriyi kutuplama ile ilgili alt yordam aşağıda verilmektedir.

```
procedure veriyikutupla(data:pointer;size:integer);
var
  p:array[0..50000] of integer; b:pbytedizi; 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 akk
      for i:=size-epenbuyukluk to (size-1) do  //yeni verinin son penceresi onceki
oncekiveri[i-(size-epenbuyukluk)]:=p[i];  // veri olarak saklanıyor.
   sayisalsessinyali:=@p; for z:=1 to 50000 do pheryerde[z]:=p[z];end;
```

3.3. Sesin Enerjisini Hesaplama İşlemi

Ses sinyalinin en basit temsillerinden biri onu enerjisi ile göstermektir **(8).** 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 **(2).** 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 **(9).** Bu enerji kısa zaman enerji formülü ile hesaplanmaktadır.

3.3.1. Kısa Zaman Enerji

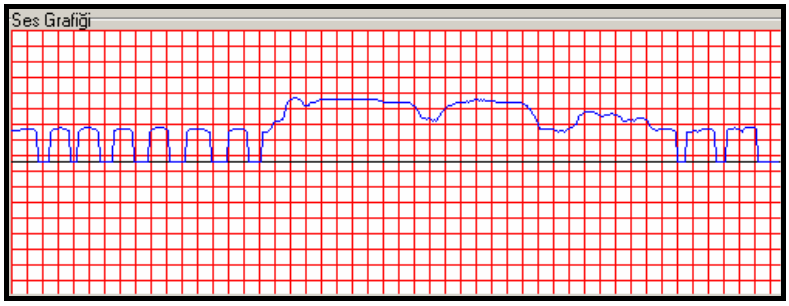
Ses sinyalinin belirli bir zamanda sahip olduğu enerjiyi hesaplamak için kullanılır. Kısa zaman enerji hesabının 3 farklı tanımlaması vardır **(7).**

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

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

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

Geliştirilen yazılımda kare toplam enerji (sum of square energy) fonksiyonu kullanılmıştır. Şekil 3'de kaydet kelimesinin program tarafından hesaplanan enerjisi gösterilmektedir.



Şekil 3: Kaydet Kelimesinin Enerjisi

3.4. 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. **(10)**. Ses tanıma sisteminde sesin bitiş noktasının yanlış belirlenmesinin iki negatif etkisi vardır: Birincisi ses sınırı yanlış belirlendiğinden tanıma hatası ortaya çıkar. İkincisi sesin sınırı yanlış belirlendiğinde ses olmayan bölümlerde devreye gireceği için hesaplama işlemi artar **(11)**. Ses sinyalinin bitişini belirlemek için çeşitli yöntemler vardır. Kelimenin sınırlarını belirlemek için en yaygın, esnek ve doğru yöntem ses sinyalin enerjisini kullanmaktır **(10)**.

3.4.1. Sesin Başlangıcını Tespit Etme İşlemi

Sesin başlangıcını tespit etmek için hesaplanan enerji değerini 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 artarda 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. Ses başlangıcı tespit edildikten sonra ses bitişini bulmak için "sesbitara" alt programı çağrılır.

3.4.2 Sesin Bitişini Tespit Etme İşlemi

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.

3.5. Preemhasis Filtre

Sesli ifadeler düşük sıklıklar için yüksek enerji değerlerine sahip olmaktadır. Düşük sıklıktaki bileşenlerin yüksek sıklıktakileri maskelememesi için preemhasis adı verilen filtre kullanılmaktadır **(3).**

Genellikle preemhasis filtre aşağıdaki bağıntıyla ifade edilir.

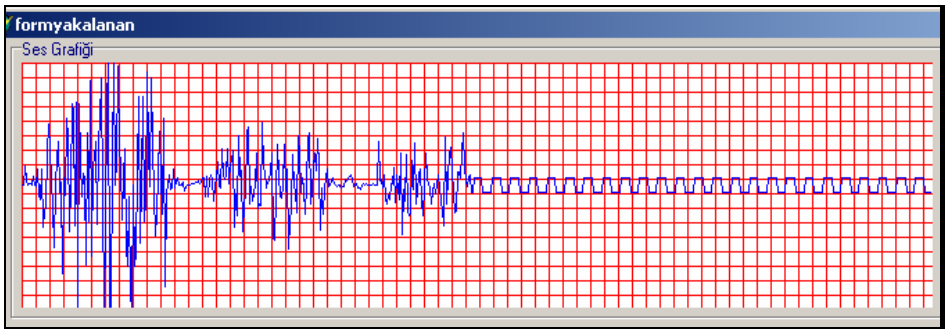
$$P(z)=1-\mu z-1$$

μ katsayısı 0.9 ile 0.95 arasında değişmektedir. (12). z-1 ifadesi standart gecikmeyi temsil eder (2).

Aşağıda Preemhasis filtre ile ilgili yazılmış olan program kodları görülmektedir.

```
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;
    for i:=0 to size-1 do begin
      preemhasisli^[i+1]:=round(preemhasisli^[i+1]-Katsayi*preemhasisli^[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. Kaydet kelimesine Preemhasis filtreleme uygulanmış ses sinyali Şekil 4'de görülmektedir.



Şekil 4: Kaydet Kelimesinin Preemhasis Filtre Uygulanmış Hali

3.6. 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 **(2).**

Sayısallaştırılan sinyaller bilgisayar ortamında ikili sözcükler olarak tutulurlar. Sinyal işleme aşamasında, genellikle sesli ifadeleri temsil eden sözcüklerin tümünü bir seferde ele alma imkanı 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 **(3).**

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.

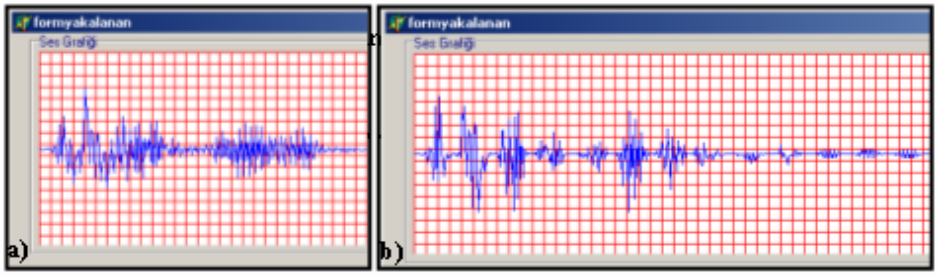
Pencerelemenin anlamı, y(n) ses sinyalinin sonlu süreli bir W(n) ile çarpılmasıdır **(8).**

Pencereleme işlemi, bir pencere bir öncekinin bitme noktasında başlatılarak örtüşmesiz yapılabildiği gibi daha geride başlatılarak pencereler arası örtüşme sağlanabilir. Bu sayede işaretteki kesin kesin sınırlar yumuşatılabilir **(13).**

Geliştilen programdaki Pencereleme ile ilgili program kodları aşağıda görülmektedir.

```
Procedure windowing(data:pointer;veriboyutu:integer;pencerebuyukluk:integer);
 var
   gecici:^integer; veri:^integer; i:integer; sifirsay:integer; artanverisayisi:integer;
    begin
       penceresayisi:=veriboyutu div pencerebuyukluk;veri:=data;
       artanverisayisi:=veriboyutu mod pencerebuyukluk; //pencereden artan veri
       if artanverisayisi>0 then begin
          inc (penceresayisi);  sifirsay:=pencerebuyukluk-artanverisayisi;
            veriboyutu:=veriboyutu+artanverisayisi;
            gecici:=data; inc(gecici,veriboyutu+1);
           for i:=0 to sifirsay-1 do begin //boş olan yerleri 0'la doldur
             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 Kare(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; end;   //proc end
```

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 5: (a) Sayısallaştırılmış Sinyal (b)Hanning Pencereleme Uygulanmış Sinyal

3.6.1. Kare Pencereleme

Kare pencereleme fonksiyonunun formülü **(14).**

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

3.6.2. Hamming Pencereleme

Hamming pencerelemenin fonksiyonun formülü aşağıdaki gibidir **(15).**

$$w[n] = \begin{cases} 0.54 - 0.46\cos\dfrac{2\Pi n}{N}, 0 \le n \le N-1 \\ 0, \qquad\qquad \text{otherwise} \end{cases}$$

Hamming pencereleme fonksiyonunu uygulayan alt program kodları aşağıda verilmektedir. Bu 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**.**

```
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;
```

3.6.3. Hanning Pencereleme

Hamming pencereleme fonksiyonun formülü aşağıdaki gibidir **(16)** .

$w[n]=0.5(1-\cos(2\Pi n/(N-1)))$

Hanning pencereleme fonksiyonunu uygulayan alt program kodları aşağıda verilmektedir. Bu 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.

```
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;
```

3.6.4. Blackman Pencereleme

Blackman pencereleme fonksiyonun formülü aşağıdaki gibidir **(17).**

$w(n)=0.42-0.5\cos(2\Pi n/N-1)+0.08\cos(4\Pi n/N-1).$

3.6.5. Barlett Pencereleme

Barlett pencereleme fonksiyonun formülü aşağıdaki gibidir **(14).**

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

Barlett pencereleme fonksiyonunu uygulayan alt program kodları aşağıda verilmektedir. Bu 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.

3.7. Fast Fourier Transform (FFT)

Hızlı Fourier Dönüşümü (FFT-Fast Fourier Transform) ses sinyalini meydana geldiği frekanslara ayırır. Normal bir ses sinyali renklerin farklı, farklı renklerin birleşmesiyle oluşması gibi çeşitli frekanslardaki seslerin birleşmesi ile oluşur. Ses sinyalinin ham şekli insan aynı sözcüğü söylese 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 **(18).**

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

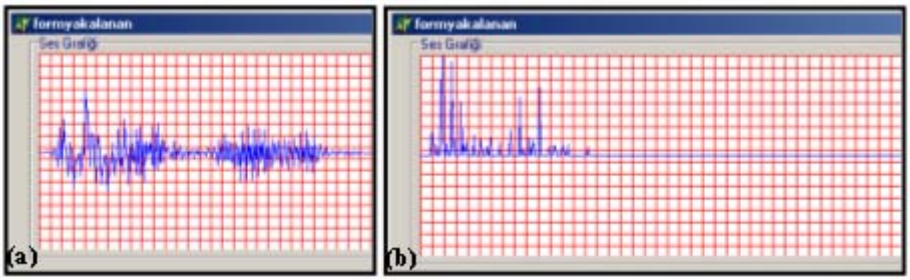
bir şekilde hesaplayan bir algoritmadır. **(19).** DFT dijital sinyal işlemenin temel işlemlerinden biridir. DFT genellikle bir çok uygulamada kullanılan Linear Filtrelemede kullanılır. Bununla birlikte sonlu büyüklükteki ses sinyalinin tamamı için DFT' de gerekli olan hesap diğerlerinden çok fazladır **(20).** FFT Discrete Fourior Transform'un(DFT) hesaplanması için etkili bir metottur. Bu metodun etkili olmasının sebebi bir çok 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 **(21).**

DFT fonksiyonu eşitlik 3'deki gibidir **(22).**

$$X(j)=\sum_{k=0}^{N-1} A(k).W^{jk} \quad , \qquad j = 0,1,...,N-1 \qquad \textbf{(3)}$$

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

Aşağıda işlememiş ses sinyali ve FFT uygulanmış ses sinyali gösterilmektedir.



Şekil 6: (a) İşlenmemiş Ses Sinyali (b) FFT Uygulanmış Ses Sinyali

3.8. Özilişki Fonksiyonu (Autocorrelation Function)

Özilişki fonksiyonu özellikle ana sıklıkları ortaya çıkaran ve tanımayı kolaylaştıran bir fonksiyondur **(2).** 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 Özilişki fonksiyonu (autocorrelation function) gürültülü çevrelerde en iyi performansı sağlar **(23).**

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 eşitlik 4'deki gibidir.

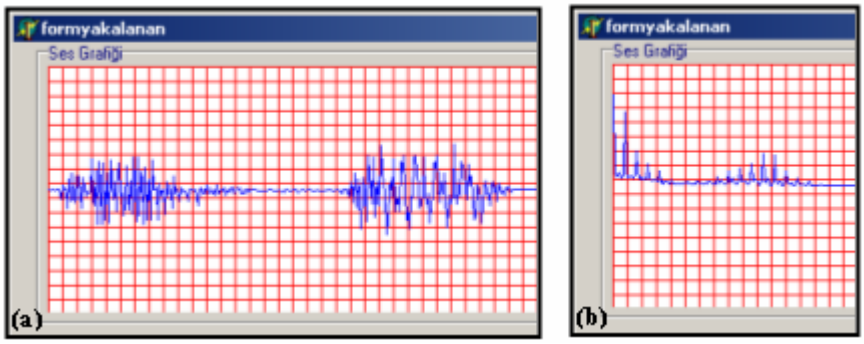
$$R(k) = (1/N)\sum_{i=0}^{N-1-k} x(i).x(i+k) \quad , \quad \text{k=0,1,…N-1} \qquad \textbf{(4)}$$

Blackman-Tukey özilişki uzaklığı ise eşitlik 5'deki gibidir **(24)**.

$$R(k) = (1/(N-k)) \sum_{i=0}^{N-1-k} x(i).x(i+k) \qquad \textbf{(5)}$$

Ses tanımayla ilgili geliştirilen programda Blackman-Tukey özilişki uzaklığı kullanılmaktadır. Özilişki fonksiyonunu uygulayan alt programın kodları aşağıda verilmektedir.

```
Procedure TformYakalanan.oziliski;
  var
  oziliskili:array[0..50000] of integer;
  t:real; index:integer; n,k:integer;
  begin
    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.
      oziliskili[k]:=trunc(t/(fftboy-k)); //özilişki tamponuna aktar
    end; //for k
    for  k:=0 to fftboy-1 do fftalan[k]:=oziliskili[k]; end;//proc
```



Şekil 7: (a) İşlenmemiş Ses Sinyali (b) Özilişki Fonksiyonu Uygulanmış SesSinyali

3.9. 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.**(1)** Bu hesaplamada kullanılabilecek çeşitli uzaklık ölçüleri bulunmaktadır. Geliştirmiş olduğumuz programda öklid uzaklığı kullanılmaktadır.

3.9.1. Öklid Uzaklığı

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

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

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

3.9.2. 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 eyrı, ayrı ölçülen farkların en büyüğü olarak tanımlanabilir: **(9)**

$$D_{kare=en\_yüksek}\left|x_i - y_i\right|$$

3.9.3. Cepstral Uzaklık Ölçümü

Cepstral uzaklığı ile Öklid uzaklığı benzerdir ve aşağıdaki şekilde tanımlanır. **(25)**

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

Programın ses tanıma kısmı iki ana alt programdan oluşmaktadır. Birinci kısım veritabanındaki referans ses sinyalini çağırır ve ikinci kısım olan karşılaştırma alt programına gönderir. Birinci alt programın kodları aşağıda verilmektedir.**(26)**

```
procedure Tformyakalanan.recognizeClick(Sender: TObject);
var
  enbuyuk:real; kelime:double; yol:string;  bulunankelime:string;
  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);
bilgi.Items.Clear; enbuyuk:=0;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); //ikinci alt yordam
  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);
     komut:='Yok';
```

```
      end; //else //if kelime>
         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
        application.MessageBox(pchar(Bulunankelime),'Arama Sonucu',mb_OK);
      end;
    for k:=1 to kackelime do begin
      sirali.Items.Add(inttostr(k)+'-'+siralikelime[k]+' = '+floattostr(siraliyuzde[k]));
    end;
      table1.Active:=false;end;// proc
```

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

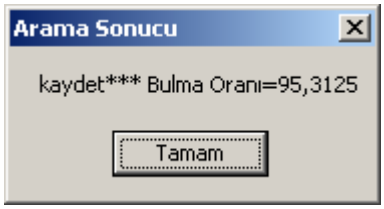
$$\text{Benzerlik Yüzdesi} = \frac{\text{Bulunan Pencere Sayısı}*100}{\text{Toplam Pencere Sayısı}}$$

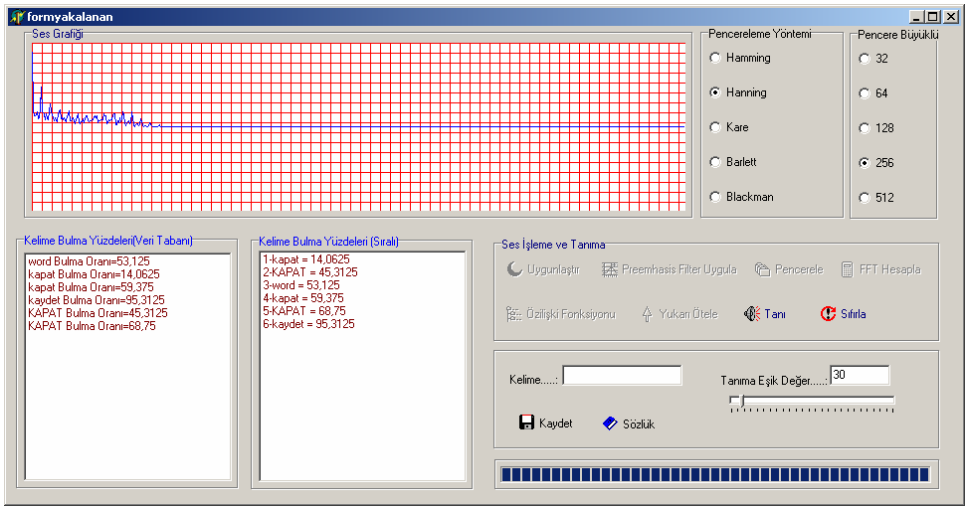
formülüyle benzerlik yüzdesi hesaplanır. Bu benzerlik yüzdesi ilk alt program gönderilir. Bulunan bu benzerlik yüzdesi bir önceki kelimenin benzerlik yüzdesinden büyükse bulunan kelime yerine bu kelime geçer. Bu işlemler veritabanında kayıtlı tüm kelimeler için tekrarlanır.

```
    Function Tformyakalanan.compare(path:string):real;
    var
    f:file of byte;fsize:integer;p:pointer; 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; Bulmaorani:real; //karşılaştırma sonucu
    pensay:integer; 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);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;end; //end function
```

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 8: Seslerin Benzeme Oranları

4. Sonuç

Bu çalışmada sesli ifade tanımayla ilgili bir yazılım geliştirilmiş ve örüntü tanıma yönteminin ilkeleri uygulamalı olarak ortaya konulmuştur. Yapılan denemeler sonucunda yazılımın sesli ifadeleri kişiye bağımlı olarak yüksek oranda doğru olarak tanıdığı tespit edilmiştir.

Geliştirilen yazılımda ses tanımayla ilgili ön işlemler kullanıcı kontrolünde tek, tek uygulanarak her uygulamanın sonucunda çıkan ses sinyalinin grafiği şekil olarak görülebilmektedir. Bu açıdan bakıldığında bu yazılım üniversitelerde ses tanımayla ilgili okutulan derslerde uygulama kaynağı olarak kullanılabilir. Çünkü yazılım sesli ifadeleri tanıma için gerekli işlemleri adım, adım kullanıcı kontrolünde yapmakta ve her adım sonunda grafiksel bilgi vermektedir. Bu da öğrencinin sesli ifadeleri tanımayla ilgili adımları anlamasına yardımcı olabilir.

Yazılımda yapılan denemeler sonucunda ortamdaki gürültünün tanımayı olumsuz yönde etkilediği görülmektedir. Ayrıca kelimelerin ilk söyleniş tarzına benzer şekilde söylenmesi tanıma oranını artırmaktadır. Kelimelerin veritabanındaki örnek sayısı artırıldıkça kelimeleri söyleyiş tarzının tanımayı olumsuz yönde çok fazla etkilemediği görülmektedir.

Geliştirilen ses tanıma yazılımında benzeme oranı %85'in üzerinde ise ses tanınmış demektir. Bu yazılımda kadın, erkek sesine ve 1, 2 örneklemeye göre ayrı, ayrı yapılan 20 deneme sonucuna göre tanıma yüzdesi kadın sesinde en düşük %70 en yüksek %90 erkek sesinde ise en düşük %80 en yüksek % 100 çıkmıştır. Ayrıca veritabanındaki örnekleme arttıkça tanıma yüzdesi ve benzerlik oranları artmaktadır. Bu da tanıma performansını artırmaktadır.

Kaynaklar

1. Mengüşoğlu, E.,"Bir Türkçe Sesli İfade Tanıma Sisteminin Kural Tabanlı Tasarımı Ve Gerçekleştirimi", Yüksek Lisans Tezi,Hacettepe Üniversitesi Fen Bilimleri Enstitüsü, Ankara, s.1,42 (1999).
2. Doğan, S., "Pc Ortamında Sesli Komutları Tanıma", Yüksek Lisans Tezi, Marmara Üniversitesi Fen Bilimleri Enstitüsü, İstanbul, s.1,8,9,10,12,14,36,63,67 (1999)
3. Artuner, H., "Bir Türkçe Fonem Kümeleme Sistemi Tasarımı ve Gerçekleştirimi", Doktora Tezi ,Hacettepe Üniversitesi Fen Bilimleri Enstitüsü, Ankara, s.25,31 (1994)
4. 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, s.24-25 (2000)
5. Picone, J.,W., "Signal modeling techniques ın speech recognition", Proceedings of the IEEE, 81,No:9:1219 (1993)

6. Internet:Wikipedia The Free Encylcopedia. http://en.wikipedia.org/wiki/Fourier_series (2006).
7. Qiang,H., Youwei, Z.,"On prefiltering and endpoint detection of speech signal", Signal Processing Proceedings ICSP '98. 1998 Fourth International Conference on, 1: 749,750 (1998).
8. İkizler, N.,"Türkçe'de Konuşmacıdan Bağımsız Hece Tanıma Sistemi", Doktora Tezi, Karadeniz Teknik Üniversitesi Fen Bilimleri Enstitüsü, Trabzon, S. 22,54 (2003).
9. Akçay, B., "Yapay Sinir Ağları İle Türkçe Konuşma Tanıma", Yüksek Lisans Tezi, Hacettepe Üniversitesi Fen Bilimleri Enstitüsü, Ankara, s.22,42 (1994).
10. 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)
11. 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, 2: 732 (1993)
12. Burrows,L.T., "Speech Processing With Linear And Neural Network Models", Doktora Tezi ,Cambridge University Engineering Department,England, s.21-22 (1996)
13. Gökhan, A., "Yapay Sinir Ağları İle Ayrık Türkçe Sözcüklerin Tanınması", Yüksek Lisans Tezi,Fırat Üniversitesi Fen Bilimleri Enstitüsü, Elazığ, s.23 (1997)).
14. Kondoz, A. M., "Digital Speech", B. Evans, G. Pujolle, A. Danthine, O. Spaniol, England, 36,37 (1994)
15. Kinnunen, T., "Spectral features for automatic text-ındependent speaker recognition", University of JoensuuDepartment of Computer Science, Licentiate's Thesis, S.53 (2003))
16. Sherlock, B.G.," Windowed discrete fourier transform for shifting data", *Submitted to Elsevier Science on Signal Processing,*74*:* 177 (1999)
17. 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).
18. Aydın, Ö., "Yapay Sinir Ağlarını Kullanarak Bir Ses Tanıma Sistemi Geliştirilmesi", Yüksek Lisans Tezi, Trakya Üniversitesi Fen Bilimleri Enstitüsü, Edirne, s. 50 (2005)
19. Rockmore, D.N.," Some applications of generalized ffts ", *Dimacs Series in Discrete Mathematics Theoretical Computer Science ,* 00*,Sayı:19:*1 (1991)

20. Nguyen,T. Q. ,"Integer fast fourier transform", *IEEE Transactions On Signal Processing,50,No:3:* 607 (2002).

21. Cochran, W. T., Cooley, J. W., "What Is the Fast Fourier Transform?", *Proceedings Of The IEEE,* 55, No:10:1664,1667 (1967).

22. *Cooley, J. W., Tukey, J. W., "A*n algorithm for the machine calculation of complex fourier series*", Math. Comput.,* 19: 297 (1965).

23. Kobayashi, H., Shimamura, T., "A weighted autocorrelation method for pitch extraction of noisy speech", Acoustics Speech and Signal Processing IEEE International Conference, 3:1307 (2000)

24. Dendrinos, M., Carayannis, G., "Spectrum analysis using a new autocorrelation measure", Acoustics, Speech, and Signal Processing,4:2468 (1998)

25. Tohkura,Y., "A weighted cepstral distance measure for speech recognition", *Acoustics, Speech, and Signal Processing, 11:761 (1986).*

26. Karacı, A.,"Bilgisayar Ortamında Sesli İfadeleri Tanıma", Yüksek Lisans - Tezi,Gazi Üniversitesi Fen Bilimleri Enstitüsü, Ankara, s.75-82 (2006).