

YILDIZ TEKNİK ÜNİVERSİTESİ * FEN BİLİMLERİ ENSTİTÜSÜ

Bilgisayarla Ayrık Sözcük Tanıma Sistemi

A. Levend Abay

Yüksek Lisans Tezi

YILDIZ TEKNİK ÜNİVERSİTESİ
KÜTÜPHANE DOKÜMANTASYON
DAİRE BAŞKANLIĞI

	R 368
Kot	: 48
Alındığı Yer	: Fen Bil. Ens.
	:
Tarih	: 13.4.95
Fatura	:
Fiyatı	: 13.4.95
Ayniyat No	: 1-4
Kayıt No	: 51026
UDC	:
Ek	:



YILDIZ ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

BİLGİSAYARLA AYRIK SÖZCÜK TANIMA SİSTEMİ



YÜKSEK LİSANS TEZİ
BİLGİSAYAR MÜH. A.LEVEND ABAY

İSTANBUL, 1988

ÖNSÖZ

Günümüzde insanoğlu, yaptığı işleri bilgisayarlara yaptırma çabasıdadır. Son yıllarda uzman sistemler kurarak, insanın unutabileceği küçük ayrıntıları dahi bilgisayarlara yükleyerek, güvenilir uzman bilgisayarlar yaratmaktadırlar.

Yapay Zeka konusundaki çalışmalardan biri insanın beş duyusunu bilgisayarla gerçekleştirmektir. Bilgisayara görme, koku alma, işitme, dokunma hissi ve konuşma yeteneklerini kazandırma çalışmaları son 20-30 yıldır sürmektedir. Tam olarak bir sonuca varılamasa da bu tür çalışmalar insanın kendisini tanımaya yardımcı olmakta ve insanoğlunun ne kadar karmaşık bir yapıya sahip olduğunu göstermektedir.

Bu projede amaç, insan sesini incelemek ve analizini yapıp sesdeki sözcüğü ayırmak ve bir sözcük dağılımı oluşturmaktır. Daha sonra bir ses girildiğinde bilgisayar, bu sözcüğün kendi dağılımındaki hangi sözcük olabileceğini bulacaktır.

Bu projenin hazırlanmasında değerli yardımlarından dolayı Sayın Prof. M.Yahya KARSLIĞIL'e ve Sayın Doç. Kırkor HARUTUNYAN'a, tezimin yazılmasında emeği geçen tüm arkadaşlarıma teşekkür ederim.

YILDIZ ÜNİVERSİTESİ

A.Levend ABAY

İSTANBUL, 1988

İÇİNDEKİLER

ÖZET

ABSTRACT

1. MICROLINK ARABİRİMİ

1.1 Giriş	1-1
1.1.1 IEEE-488 Bus'ı	1-2
1.1.2 Cihazın Özellikleri	1-2
1.1.3 Denetim Hatları	1-3
1.1.4 El sıkışma Hatları	1-3
1.1.5 IEEE-488 Konnektörü	1-4
1.2 MICROLINK'teki IEEE-488 Standart'ının Tanıtımı.	1-5
1.2.1 MICROLINK Ana Kartı	1-5
1.2.1.1 POWER Modülü	1-6
1.2.1.2 CONTROL Modülü	1-6
1.3 Projede Kullanılan Modüller	1-7
1.3.1 A12D Modülü	1-7
1.3.1.1 Çeviricinin Teknik Özellikleri ..	1-8
1.3.2 AN1DS Modülü	1-8
1.4 Sistem Yazılımı	1-9
1.4.1 Tüm Sistemin Çalışma Şekli	1-10
1.4.2 Fonksiyonlar	1-10
a) OPENGPIB	1-11
b) READCOM	1-11
c) GPIBCODE	1-13
d) WRITECOM	1-13
e) SENDCOM	1-15
f) READDAT	1-15
g) WRITEDAT	1-16
h) CLOSEGPIB	1-16

1.5 Çeviri Programları	1-17
a) BYT2INT	1-17
b) INT2BYT	1-17
c) WRD2BYT	1-17

2. İNSANDA KONUŞMA SİSTEMİ

2.1 Giriş	2-1
2.2 Seslerin Çıkarılması, Ses Yolu, Ses Cihazı	2-2
2.2.1 Ses Geçidi	2-2
2.2.2 Ses Telleri	2-3
2.2.3 Küçük Dil, Ağız Yolu, Burun Yolu	2-3
2.3 Oluşum Noktası	2-5
2.4 Sedalı Sesler, Sedasız Sesler	2-6
2.5 Vokal	2-8
2.6 Konsonant	2-9
2.6.1 Konsonantların Oluşum Noktaları	2-10
2.6.2 Konsonantların Temas Dereceleri.....	2-10
2.6.3 Nazal Olup Olmama	2-11
2.6.4 Konsonantların Özellikleri	2-11

3. SES TANIMA

3.1 Giriş	3-1
3.2 Ses İşareti	3-2
3.3 İşaret İşleme	3-3
3.4 Sayısal Ses İşleme	3-4
3.5 Ses İle Haberleşme Uygulamaları	3-5
3.6 Konuşma Tanımada Karşılaşılan Sorunlar	3-6

4. SÖZCÜĞÜN UÇ NOKTALARININ BELİRLENMESİ

4.1 Giriş	4-1
4.2 EXPLICIT ve IMPLICIT Uç Nokta Taraması	4-4
4.3 HYBRID Uç Nokta Taraması	4-4
4.4 Enerji ve Sıfır Geçiş Oranlarının Hesaplanması	4-7
4.5 Sıfır Geçiş Oranlarının Hesaplanması	4-8

5. AYRIK SÖZCÜK TANIMA SİSTEMİ

5.1 Giriş	5-1
5.2 A/D Çevirici	5-3
5.3 Ses Enerjisi Hesabı	5-3
5.4 Sözcüğün Uç Noktalarının Belirlenmesi	5-3
5.5 Doğrusal Tahminli Kodlama Analizi	5-5
5.5.1 Durbin's Recursive Procedure	5-7
5.5.2 Hamming Window Kullanımı	5-9
5.6 Doğrusal Zaman Normalizasyonu	5-10
5.7 Şablonların Diske Saklanması	5-13
5.8 Dinamik Zaman Normalizasyonu	5-13
5.8.1 Dynamic Time Warping	5-14
5.8.2 Yerel Yol Sınırlaması (Local Path Constraint)	5-18
5.8.3 Dal Ağırlığı	5-20
5.8.4 Optimum Yolun Uzunluk Normalizasyonu	5-20
5.8.5 Fark Hesabı (Distance Measure)	5-21
5.8.6 DTW Kullanarak Ayrık Sözcük Tanıma	5-21

6. SÖZCÜĞÜN UÇ NOKTALARININ BELİRLENMESİ

6.1 Giriş	6-1
6.2 ÖĞRENME: Sözcük Tanıtma	6-1
6.3 TANIMA: Mikrofondan Girilen Sözcüğün Tanınması	6-2

6.4 Uygulama Programının Ayrık Sözcük Tanıma

Sistemiyle Kullanılması İçin Yapılması Gereken

İşlemler 6-4

SONUÇ

PROGRAMLAR

SPEECH.H	Pr1
OPEN.C	Pr2
EPD.C	Pr3
AUTO.C	Pr4
LPC.C	Pr5
OGREN.C	Pr6
TANI.C	Pr7
LCPIB.ASM	Pr8
GPIBC.H	Pr9

KAYNAKLAR

ÖZGEÇMİŞ

ÖZET

Konusu Ayırık Sözcük Tanıma olan bu projede ana hedef, mikrofondan alınan analog ses işaretini sayısal işarete dönüştürerek, daha önce aynı sistemde oluşturulan sözcük dağarcığı içinde bulup tanımaktır.

Sistem iki ana bölümden oluşmaktadır: TANIMA ve ÖĞRENME. Sesin dış ortamdan alınıp, kendisine ait şablon oluşturuluncaya kadar ki aşamalar her iki bölümde de ortaktır. Analog ses, sayısal sese dönüştürüldükten sonra içindeki sözcük, dış ortamın gürültüsünden ayrılır. Buna End-Point Detection denir.

Ses işareti içindeki başlangıç ve bitiş anları belirlendikten sonra bu sözcüğe ait şablon oluşturulur. Sözcüğü karakterize edecek en iyi ve günümüzde en çok kullanılan yöntem Linear Predictive Coding Technique'dir. Bu teknikte, belirli uzunlukta çerçevelere ayrılan sözcüğün her çerçevesi için bir vektör oluşturulur. Bu vektörün boyu, sistemin duyarlılığı ile doğru orantılıdır ve 10 eleman olarak alınmıştır. Bu elemanların her birine Linear Prediction Coefficient denir.

Daha sonra aynı sözcüğün bir diğer söylenişindeki farklılığı minimuma indirmek için iki aşamalı normalizasyon yapılır. İlki Linear Time Normalization veya Linear Time Warping'dir. Burada sözcük uzunlukları, sabit bir uzunluğa getirilir. Böylece, sözcük karşılaştırma işleminde sözcüklerin farklı uzunlukta olması bir dezavantaj

oluşturmaz.

Bu aşamadan sonra sistem daha önce de belirtildiği gibi ikiye ayrılır. ÖĞRENME bölümünde buraya kadar oluşan bilgi, diske saklanır.

İkinci olarak Dynamic Time Warping veya Dynamic Time Normalization denilen ve telaffuz farklılıklarını ortadan kaldıran yöntem uygulanır. Bu yöntemde, sözcüklerin hangi çerçevelerinin karşılıklı benzerliğe sahip olduğu bulunur. Bu bölüm, TANIMA bölümüdür.

Sözcük dağarcığındaki her sözcük ile bilinmeyen ve ne olduğu tanınmaya çalışılan sözcük arasında DTW uygulandıktan sonra, oluşan benzerlik katsayılarının en küçüğüne sahip olan şablon tanınan sözcüğe aittir. Bu son karar verme yöntemine de Nearest Neighbour Rule denir.

ABSTRACT

The main aim of this project whose subject is The Isolated Word Recognition is; to find then to recognize the analog sound signal received via microphone -in the vocabulary which is formed in the same system before- by transforming into the digital sound signal.

The system is made up of two parts: The Recognition and Training. The phases which last from receiving of the sound external medium up to be formed its pattern, are shared by both of these two parts. The word which is contained by the analog sound is seperated from the noise of the external medium, after the analog sound has been transformed into the digital sound. It is isolated. This process is called End-Point Detection.

After determination of start and end point of the word in the sound signal, the pattern which belongs to this word is formed. The method that is the best one for characterized the word and also the most common is called Linear Predictive Coding Technique. In this technique; the word is distinguished to the certain lengths frames, which a vector is formed in each. The length of vector is direct proportional with the precision of the system and it is taken as a vector with ten elements. Each of these elements is called Linear Prediction Coefficients.

Furthermore, the normalization is done within two phases to minimize the difference which occurs because of the

pronunciation of the same word in other manner. The first one is called Linear Time Normalization or Linear Time Warping. In this phase; the lengths of the word are brought in the certain length. In this manner, being the words in different lengths in the comparison of words doesn't cause a disadvantageous situation.

After this phase the system is divided into the two parts as pointed out before. The information which is made up of until this phase in The Training Part, is stored into the disk.

Secondly, a method which removes the differences of pronunciation, is applied and it is called Dynamic Time Normalization or Dynamic Time Warping. This phase is called The Recognition Phase.

After DTW has applied between all the words that are in the vocabulary and the one that is unknown and it is tried to be recognized what it is, and the whole pattern which has the smallest coefficient that occurs amongs the coefficients belongs to the recognized word. This last method of decision is also called Nearest Neighbour Rule.



1.MICROLINK ARABİRİMİ

1.1.Giriş

MICROLINK arabirimi, laboratuvar cihazları ve çeviriciler ile mini veya mikrobilgisayarlar arasında veri iletişimini IEEE-488 bus'ı üzerinden sağlamak amacıyla tasarlanmıştır.Standart IEEE-488 port'u olmayan bilgisayarlarda kullanılabilmesi için IEEE-488 kartı ilave edilmelidir.Bu projede kullanılan Apricot Xen (IBM PC uyumlu değil) için Xen IEEE-488 arabirim kartı (Biodata) kullanılmıştır.

Sistem,IEEE-488 bus işlemlerinin gerçekleştiği donanım ve bus ile dış dünya arasındaki veri iletişimini gerçekleştiren bir takım modüllerden oluşmaktadır.Bu modüller, analog giriş ve çıkış, sayısal giriş ve çıkış gibi işlemleri gerçekleştirecek ayrı ayrı kartlardır.

MICROLINK arabiriminin arka tarafında Standart IEEE-488 konnektörü vardır.Bu konnektör ile MICROLINK arabirimi IEEE-488 ağına bağlanabilir.

1.1.1.IEEE-488 Bus'ı

IEEE-488 bus'ı uluslararası standarttır.Bu sebeple bilgisayarlara, elektronik test ve ölçme aletlerine, bilgisayar çevre birimlerine kolayca bağlanabilir.

Bus, 16 hattan oluşmaktadır.Bunların 8 tanesi veri iletişimi için kullanılır.Diğer 8 hattın 3'ü cihazın el sıkışmalı (handshake) çalışmasında (asynchronous) kullanılır, 5'i ise kontrol için kullanılır.

1.1.2.Cihazın Özellikleri

Hatlardaki düşük gerilim TRUE, yüksek gerilim FALSE olarak kabul edilmiştir.Bus'a bağlı birimler üç ayrı amaçla çalışabilir:

LISTENER	-Diğer birimlerden veri almak
TALKER	-Diğer birimlere veri göndermek
CONTROLLER	-Bus'taki veri transferini denetlemek.
	CONTROLLER, bus'taki diğer birimleri
	adresleyebilir ve onlara komut
	gönderebilir.Bir nevi bilgisayar gibi
	davranır.

Herhangibir anda sadece bir cihaz CONTROLLER veya TALKER olabilir. Ama birden fazla cihaz aynı anda LISTENER olabilir.

1.1.3.Denetim Hatları

5 denetim hattı vardır:

ATN -Attention- Eğer ATN hattı TRUE ise denetimci aktiftir ve diğer birimler komut beklerler.ATN, TRUE iken diğer birimler LISTENER veya TALKER olarak adreslenebilir.

EOI -End Or Identify- Bu hat TRUE ise veri transferi tamamlanmış demektir.

IFC -Interface Clear- Bus'ı initialise eder.TALKER veya LISTENER modundaki birimler inactive duruma geçerler.Bu hattı sadece denetimci TRUE konumuna geçirebilir.

REN -Remote Enable- Sistem deneticisi, bu hattı TRUE yapar ve cihazları LISTENER olarak adresleyebilir.Bu cihazlar uzaktan denetlenebilir.

SRQ -Service Request- Servis isteyen cihaz tarafından TRUE yapılır.

1.1.4.Elsıkışma Hatları

3 hattan oluşmaktadır:

NRFD -Not Ready For Data- LISTENER veri almaya hazır olduğunda bu hat FALSE konumundadır.Bütün

LISTENER'lar hazır ise hattın gerilimi HIGH olacaktır.Yani TALKER'a bütün LISTENER'ların hazır olduğunu bildirir.

DAV -Data Valid- TALKER veriyi BUS'a koyduğu zaman bu hat TRUE olur. LISTENER'lar bu hat TRUE olmadan veriyi bus'tan almaz.

NDAC -Not Data Accepted- LISTENER veriyi bus'tan alınca bu hattı FALSE yapar.Bütün LISTENER'lar veriyi alınca, hat HIGH konuma geçer. Yani TALKER'a verinin bus'tan alındığını bildirir.

1.1.5.IEEE 488 Konnektörü

<u>Bacak Numaraları</u>			
Veri Hattı	1	13	Veri Hattı
Veri Hattı	2	14	Veri Hattı
Veri Hattı	3	15	Veri Hattı
Veri Hattı	4	16	Veri Hattı
EOI	5	17	REN
DAV	6	18	DAV için toprak
NRFD	7	19	NRFD için toprak
NDAC	8	20	NDAC için toprak
IFC	9	21	IFC için toprak
SRQ	10	22	SRQ için toprak
ATN	11	23	ATN için toprak
Koruma (shield)	12	24	İşaret toprağı

1.2.MICROLINK'teki IEEE-488 Standart'ının Tanıtımı

Sistem, bus üzerinde tek bir adresi (PRIMARY ADDRESS) olan tek bir cihaz olarak görünür.Bir mikrobilgisayara birden fazla MIKROLINK sistemi bağlanabilir.

MICROLINK'in üzerindeki denetim modülü, cihazın PRIMARY adresini tanır.PRIMARY adres, 0 ile 30 arasında bir sayıdır.MICROLINK arabirimi üzerinde LISTENER ile TALKER'ın PRIMARY adresleri aynıdır. Diğer modüllere SECONDARY adres ile erişilir.Örneğin bir modüle veri veya komut göndermek için bilgisayar 2 byte LISTEN adresini gönderir.Bunun ilk byte'ı MICROLINK'in PRIMARY adresi, ikinci byte'ı ise gönderilen modülün adresidir.Aynı şekilde veri almak için de TALK adresi gönderilir.MICROLINK'in asıl kullanılışlığı bu SECONDARY adres özelliğinden gelmektedir.

MICROLINK'in ana fonksiyonu, PRIMARY adresi tanıyıp, veri veya komutları IEEE-488 bus'ından alıp SECONDARY adresteki modüllere dağıtmaktır.Bu modüllere sadece kendilerini ilgilendiren veri veya komutlar ulaşır.

1.2.1.MICROLINK Ana Kartı

Ana kart, bir panel ile 2 modülden oluşmaktadır.Bu modüller sol taraftaki POWER ve CONTROL modülleridir.Ana kart, güç kaynağı, adres kod çözücü ve dahili bus denetim devresi ile IEEE-488 bus bağlantılarından oluşmaktadır.

1.2.1.1.POWER Modülü

Cihazın güç kaynağını içerir.Modül üzerinde gerilim seviyelerini gösteren 3 LED vardır.Normalde sistem açıldığında bu 3 LED de yanar.

1.2.1.2.CONTROL Modülü

Bu modül, MICROLINK arabirimini IEEE-488 bus'ına bağlayan devre ile dahili veri akışını kontrol eden devreden oluşmaktadır.Modül üzerinde PRIMARY adresi belirlemek için 5 anahtar ile LISTENER veya TALKER modlarından hangisinde çalıştığını göstermek için 2 LED vardır. Cihaz fabrikadayken PRIMARY adresi 7 olarak tanımlanmıştır.

<u>Anahtar</u>	<u>Pozisyon</u>
1	Sağ
2	Sağ
4	Sağ
8	Sol
16	Sol

1.3.Projede Kullanılan Modüller

MICROLINK sistemi üzerinde, her türlü isteğe cevap verecek türden çeşitli modüller hazırlanmıştır.Daha önce de bahsedildiği gibi bazı modüllerin SECONDARY adresi vardır.Bu adres modülün ön tarafında kırmızıyla yazılıdır.Ayrıca modüllerin üzerinde seçilip seçilmediğini gösteren bir LED bulunur.

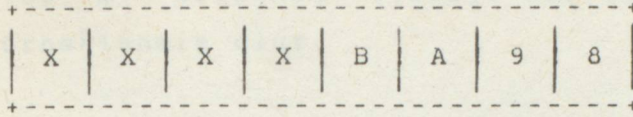
Modüllerin baskılı devresi üzerinde SECONDARY adreslerini belirlemek için 5 anahtar vardır.Anahtarı olmayan modüllerin SECONDARY adresleri yoktur.İlişkili oldukları modüllerle beraber çalışırlar ve kendileri adreslenemezler.

1.3.1.A12D Modülü

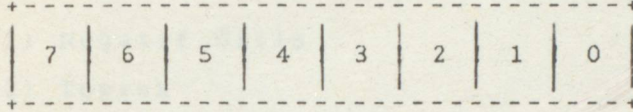
Bu modül 12-bit analog/sayısal çeviricidir.Bu modül, MICROLINK sistemine AN1DS modülüyle bağlanır.AN1DS modülü, analog işaret giriş modülüdür.AN1DS ile alınan analog işaret, A12D modülüyle sayısala dönüştürülür ve sonuç IEEE-488 bus'ına gönderilir.A12D modülü kendi başına adreslenemediğinden SECONDARY adresi yoktur.

Analog/sayısal çeviricinin çevirme süresi 25 mikrosaniyeden küçüktür. Doğrusallık hatası ± 0.5 L.S.B. dir.

Kullanılan sistemde veriler 2 byte olarak alınır.İlk byte'ın düşük anlamlı 4 biti, 12 bitlik esas verinin yüksek anlamlı 4 bitidir. İkinci byte ise esas verinin düşük anlamlı 8 bitidir.Bu sebeple alınan verilerin bu tersliği sonradan düzeltilmelidir.



Düşük anlamlı byte



Yüksek anlamlı byte

1.3.1.1.Çeviricinin Teknik Özellikleri

MICROLINK sistemindeki bütün analog/sayısal çeviricilerin giriş işareti 0 ile 10.24 V arasında değişebilir. Analog giriş 12-bit çeviricide 0 ile 4095 arasında ölçeklendirilir. Bu durumda her bir örnek $(10.24/4095)$ 25 mV'a karşılık düşer. Böylece 1.25 mV'un altı 0, 10.236 mV'un üstü 4095 olarak kodlanır. (Max. sınırlar)

1.3.2.AN1DS Modülü

Bu modül, tek kanallı, diferansiyel girişli analog giriş modülüdür. Bu modülün üzerindeki SECONDARY adres anahtarları ile adresi belirlenebilir. Modülün giriş empedansı 100 Mohm., giriş gürültüsü 0.8 mikroV. (peak to peak , 0.01 - 10Hz) dir.

GAIN ve OFFSET vidalarıyla örnekleme şekli tespit edilebilir. Projede V anahtarı 0.1'e getirilmiştir. Yani giriş işareti 0 ile 100mV (veya ses işaretinin -50 mV ile

+50 mV arasında olacağı düşünülürse 0 V) 2048 olarak örneklenmiş olur.

Modülün işaret girişinde 3'lü DIN soket vardır:

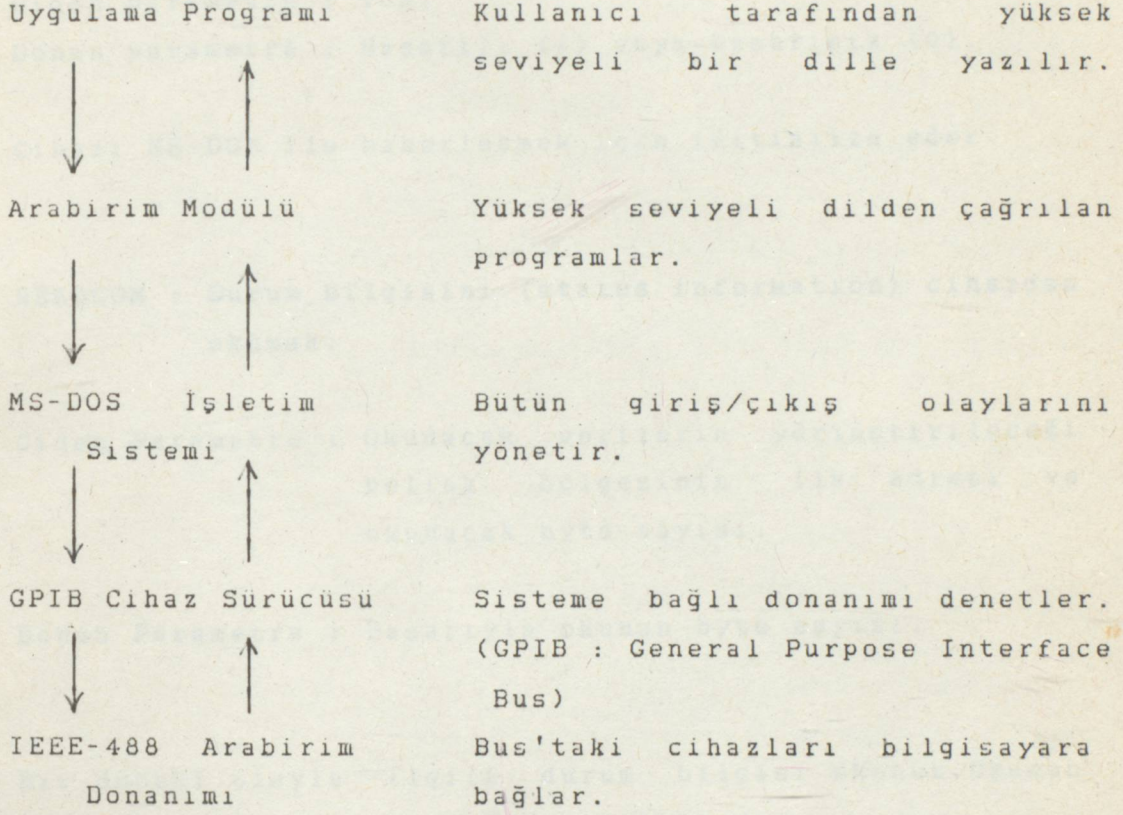
- 1) Negatif Giriş
- 2) Toprak
- 3) Pozitif Giriş

Projede, Pozitif girişten ses işareti verilmiştir. Negatif giriş topraklanmış ve Toprak girişi boş bırakılmıştır.

1.4. Sistem Yazılımı

Arabirim bus'ına bağlı bir cihazla haberleşileceği zaman ileride açıklanacak fonksiyonlardan biri çağrılır. Bu fonksiyonlar, yazılım arabirim modülündedir. Fonksiyonlar, gerekli parametreleri bir giriş/çıkış isteği gibi MS-DOS'a geçirir. İşletim sistemi de bu parametreleri uygun cihaza iletir. Sonuçlar tekrar cihazdan yazılım arabirim modülüne geri döner.

1.4.1.Tüm Sistemin Çalışma Şekli



1.4.2.Fonksiyonlar

Aşağıdaki fonksiyonlar her dil ile kullanılabilir.Ancak Link edilirken o dile ait fonksiyonları içeren .OBJ programı kullanılmalıdır.Elimizdeki yazılımda MSC (MicroSoft C) için bu .OBJ programı olmadığından LGPIB.OBJ adında yeniden yazılıp oluşturulmuştur.

MICROLINK sisteminde kullanılan ve her dil için geçerli olan fonksiyonlar ve açıklamaları aşağıda verilmiştir:

a) OPENGPIB : Cihaz ile bağlantıyı kurmak.

Giden parametre : Yok.

Dönen parametre : Başarılı (1) veya başarısız (0).

Cihazı MS-DOS ile haberleşmek için initialize eder.

b) READCOM : Durum bilgisini (status information) cihazdan okumak.

Ciden Parametre : Okunacak verilerin yerleştirileceği bellek bölgesinin ilk adresi ve okunacak byte sayısı.

Dönen Parametre : Başarıyla okunan byte sayısı.

Bir önceki olayla ilgili durum bilgisi okunur. Okunan ilk 5 byte faydalı bilgiler taşımaktadır.

Byte 0 : En son Parallel Poll cevabı.

Byte 1 : Tamamlama kodu. En son işlenen komutun nasıl tamamlandığı hakkında bilgi içerir. Dönüş parametreleri ve taşıdığı anlamlar arka sayfadadır:

<u>Sayısal Deęeri</u>	<u>Açıklama</u>
0	Normal; hata yok.
1	CPIBCOM'a anlamsız komut gönderildi.
2	TCSET komutundan sonra deęeri belirtilmemiş.
3	TOSET komutundan sonra deęeri belirtilmemiş.
4	IEEE bus'ı üzerinde hazır konumda cihaz yok.
5	Gelen verinin bittiğini EOI işareti bildiriyor.
6	Gelen verinin bittiğini sonlandırma karakteri bildiriyor.
7	Byte beklerken Time Out oldu.
8	Read Handshake Cycle'ındayken Time Out oldu.
9	Byte gönderirken Time Out oldu.
10	Write Handshake Cycle'ındayken Time Out oldu.
11	Kullanımda IEEE arabirim donanımı yok veya başka bir donanım var.

Byte 2 : Current Status Byte

<u>Bit</u>	<u>Değeri '1' ise anlamı</u>
0	TC (Terminating Char) taraması izinli
1	Time Out izinli
2	Son byte olarak EOI gönderilecek (EOIEN)
3	EOI bus hattı TRUE
4	IFC bus hattı TRUE
5	SRQ bus hattı TRUE
6	ATN bus hattı TRUE
7	REN bus hattı TRUE

Byte 3 : Kullanılan sonlandırma karakteri (TCSET)

Byte 4 : Time Out değeri (TOSET)

Diğer byte'lar tanımsızdır.

c) GPIBCODE : Sonuncu tamamlama kodunu almak.

Giden Parametre : Yok.

Dönen Parametre : Tamamlama kodu.

GPIBCODE, READCOM'un basit bir halidir. READCOM ile alınan bilgiler içinde en önemli olan tamamlama kodu (byte 1) olduğundan sadece bunu elde etmek için bu fonksiyon kullanılır.

d) WRITECOM : Option seçmek.

Giden Parametre : Komut kodlarının bulunduğu belleğin başlangıç adresi

Dönen Parametre : Başarıyla işlenen byte sayısı.

Komutlar sırayla bellekten alınır ve işlenir. Herhangibir komut başarısızlıkla sonuçlanırsa işlem durur ve o ana kadar işlenen byte sayısı geri döner.

İki tür komut vardır. Bunlar en anlamlı bitin değerine göre birbirinden ayrılır.

- i) Local komutlar : MSB=1
- ii) IEEE-488 bus komutları : MSB=0

i) Local Komutlar ve kısa açıklamaları

- SIFC : Interface Clear işaretini gönderir.
- RENT : REN hattını TRUE yapar.
- RENF : REN hattını FALSE yapar.
- EXPP : Bir Parallel Poll'ü işletir.
- TCSET : Sonlandırma işaretini belirler.
- TCEN : Giriş süresince TC taramasına izin verir.
- TCDIS : TC tarama iznini kaldırır.
- TOSET : Time Out periyodunu belirler.
- TOEN : Transfer süresince Time Out cevabına izin verir.
- TODIS : Time Out iznini kaldırır.
- EOIEN : EOI option'ı izinli.
- EOIDIS : EOI option'ı izinsiz.
- SRQWT : SRQ işareti bekler.
- ATNF : ATN hattını FALSE yapar.

- ii) Projede kullanılan IEEE-488 bus komutları ve açıklamaları

<u>Hex.</u>	<u>İsim</u>	<u>Açıklama</u>
20		PRIMARY LISTEN adresi 0
.	.	.
.	.	.
3E		PRIMARY LISTEN adresi 30
3F	UNL	Unlisten
40		PRIMARY TALK adresi 0
.	.	.
.	.	.
5E		PRIMARY TALK adresi 30
5F	UNL	Untalk
60		SECONDARY adresi 0
.	.	.
.	.	.
7E		SECONDARY adresi 31

- e) SENDCOM : Bir byte (tek) komut göndermek.

Giden Parametre : Komut byte'ı.

Dönen Parametre : Başarılı (1) veya başarısız (0).

SENDCOM, WRITECOM'un basitleştirilmiş formudur. SIFC veya UNTALK gibi tek komutlar istendiğinde WRITECOM'un yerine kullanılır. Komut uzunluğu otomatik olarak 1 dir.

- f) READDAT : Bus'tan veri bloğu okumak.

Giden Parametre : Okunacak verinin yerleştirileceği belleğin başlangıç adresi ve okunacak byte sayısı.

Dönen Parametre : Okunan byte sayısı.

READDAT fonksiyonu ATN hattını FALSE yapar (data mode).

Aşağıdaki koşullarda işlem durur:

- 1) İşlenen sayıda byte okununca veya
- 2) Verinin sonunu gösteren EOI işaretine rastlayınca veya
- 3) Sonlandırma karakteri okununca veya
- 4) Bus hatası oluşunca (Time Out gibi).

g) WRITEDAT : Bir veri bloğunu bus'a yazmak.

Giden Parametre : Verilerin bulunduğu belleğin başlangıç adresi ve gönderilecek byte sayısı.

Dönen Parametre : Başarıyla gönderilen byte sayısı.

Data mode'u olduğunu göstermek için ATN, FALSE yapılır. Bellekteki byte'lar bus'taki cihaza gönderilir. Verinin kontrolü veya çevirilmesi yapılmaz. Gönderme işlemi, belirtilen byte sayısı kadar transfer yapıldığı veya Time Out vb. gibi bir bus hatası olduğunda sona erer. Hata olduğunda, hata yakalandığı zamana kadar ki byte sayısı dönüş değeri olur. Son byte doğru olarak gönderilemeyebilir.

EOI option'ı geçerli ise EOI hattı, gönderilen son byte'mış gibi kabul edilir.

h) CLOSEGPB : Cihaz ile bağlantıyı koparmak.

Giden Parametre : Yok.

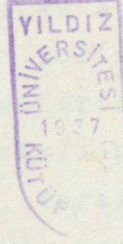
Dönen Parametre : Başarılı (1) veya başarısız (0).

Komut ve veri kanallarını kapatır.Eğer cihaz açık değilse dönüş değeri 0'dır.Yoksa 1'dir.

1.5.Çeviri Programları

Kullanıcı programı tarafından tanımlanan programlar:

- 1) BYT2INT : Birer byte'lık veri dizisini 2'şer byte 0-255 arası tamsayılara dönüştürür.
- 2) INT2BYT : 2 byte'lık tamsayıların düşük anlamlı 8 bitini alarak 1 byte'lık elemanlardan oluşan yeni bir dizi oluşturur.
- 3) WRD2INT : Tamsayı dizisinin her elemanının yüksek ve düşük anlamlı byte'larının yerini değiştirir.



2.İNSANDA KONUŞMA SİSTEMİ

2.1.Giriş

İnsanın ses organları doğuştan sadece şekilsiz sesler, doğal sesler çıkarma özelliğine sahiptir.Çocuk dünyaya geldikten sonra zamanla ses organları şekilli sedalar çıkarma alışkanlığını kazanırlar.Şekilli sedalar çıkarmada ve konuşmadaki doğallık şüphesiz alışkanlığa dayanan ve daha çok görünüşte olan bir doğallıktır.Aslında insan konuşurken, arka arkaya şekilli sedalar çıkarırken farkına varmadan özel ve yapay bir çaba harcar.Çünkü ses organları şekilli seda yapma yeteneğini sonradan kazanmışlardır ve sesleri çıkarmak için normal durumlarının dışına çıkan özel hareketler yapmak zorundadır.Bu sebeptendir ki insan alıştığı seslerden başka olan sesleri, kendi dilinde bulunmayan sesleri çıkarmakta çok güçlük çeker.

2.2.Seslerin Çıkarılması, Ses Yolu, Ses Cihazı

2.2.1.Ses Geçidi

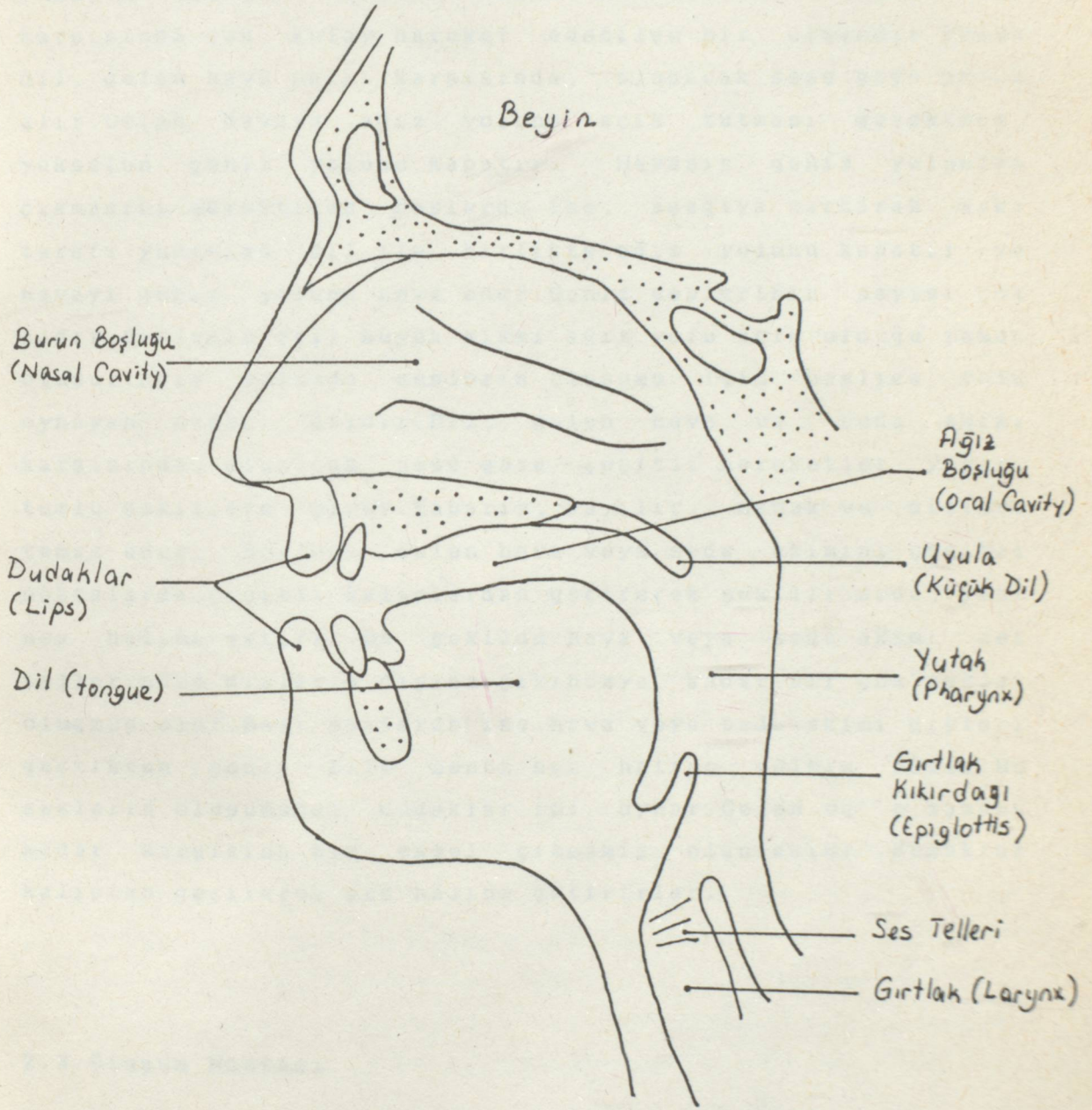
Sesler akciğerden başlayıp ağız ve burunda sona eren solunum yolları ile bu yollar üzerinde sıralanmış organlar tarafından meydana getirilir. Seslerin söylenmesinde akciğerden itibaren sırayla nefes borusu (trachea), gırtlak (larynx), ağız boşluğu (oral cavity) ve burun yolu (nasal cavity) bir ses geçidi olarak vazife görürler. Bu ses geçidinin (vocal tract) çeşitli noktalarında hareket edebilen, birbirine yaklaşıp uzaklaşabilen organlar vardır. Bu organların da yardımı ile ses geçidi açılıp kapanmak, daralıp genişlemek suretiyle değişik şekillere girebilir. Ciğerlerden yükselen hava akımı dışarı çıkmak üzere ses geçidinden geçerken geçidin içindeki organları da harekete getirerek açılıp kapanması ve daralıp genişlemesi sonucunda sesler oluşur. Sıkışıp açılan geçitte hava akımı yol bulmak için karşısına dikilen organları zorlar, onlara çarpıp, sürtünür, böylece seda haline geldikten sonra son olarak bir noktada karşısına çıkan bir kalıptan, bir engelden geçer ve ses haline gelerek dışarı çıkar. Dışarı çıkmak isteyen havanın ses geçitinde çarptığı ve sürtüldüğü yol ve organlar gırtlak, boğaz (throat), ağız ve burun boşlukları ile ses telleri (vocal folds), küçük dil (uvula), damak (hard palate), dil (tongue), dişler (tooth) ve dudaklar (lip) dir.

2.2.2.Ses Telleri

Seslerin oluşumu için herşeyden önce ciğerlerden bir hava akımının yükselmesi gerekir.Ciğerlerin sıkışması ile yükselen bu hava akımı nefes borusundan geçerek o borunun sonunda, seslerin oluşumu bakımından ilk önemli noktaya, seslerin oluşumunda çok büyük rolü olan bir noktaya gelir.Bu noktada ses telleri bulunur.Ses telleri adını verdiğimiz organ karşıdan karşıya atılmış teller halinde olmayıp, gırtlığın sonunda karşı karşıya gelen yarım daire şeklinde iki dudakçıktan ibarettir. Bu dudakçıklar ön tarafta daima bitişik bulunurlar.Arka tarafta ise normal solunum ve bir çok seslerin oluşumu sırasında aralıklı bulunurlar.Yani bir kısım seslerin oluşumuna ses telleri iştirak etmez, hava akımı arkadaki aralıktan geçerek yoluna devam eder.Diğer bir kısım seslerin oluşumuna ise ses telleri büyük ölçüde katılırlar.Ses tellerinin katılması gereken sesler çıkarılacaksa derhal dudakçıkların arka tarafı da kapatılır.Ciğerlerden gelen hava birleşmiş olan dudakçıklar arasından kendisine bir yol açmak üzere onları iter.Bu itme sonucunda hava akımının karşısına çıkmış olan ses telleri titrer, hava akımı onları titreterek aralarından geçer.Bu titreşimden bir kısım seslerin oluşumu için zorunlu olan seda meydana gelir.Ses telleri bazan da kapandıktan sonra titreşmez ve hava akımına yol vermezler.Hava onlara çarparak sesi meydana getirir.

2.2.3.Küçük Dil, Ağız Yolu, Burun Yolu

Seslerin oluşumunda ciğerlerden yükselen hava akımının ses geçidinde ses tellerinden sonra uğradığı ikinci önemli



ŞEKİL 2.1 Ses Organlarını Gösteren İnsan Kafası Kesiti

nokta ağız ve burun yollarının birleştiği noktadır. Bu noktada küçük dil bulunur. Küçük dil damağın arka tarafına, yumuşak damağın sonuna asılı bulunan ve hava akımı karşısında çok kolay hareket edebilen bir organdır. Küçük dil, gelen hava akımı karşısında, oluşacak sese göre durum alır. Gelen havaya ağız yolunu açık tutması gerekince, yükselen geniz yolunu kapatır. Havanın geniz yolundan çıkmasını gerektiren seslerde ise, aşağıya sarkarak arka tarafı yükselen dil ile birlikte ağız yolunu kapatır ve havayı geniz yoluna sevk eder. Geniz seslerinin sayısı çok azdır. Seslerin asıl büyük kısmı ağız yolu açık olduğu zaman oluşur. Ağız yolunda seslerin oluşumu için başlıca rolü oynayan organ dildir. Dil, gelen hava ve seda akımı karşısında, oluşacak sese göre çeşitli hareketler yapar, türlü şekillere girer. Kabarır, yayılır, damak ve dişlere temas eder. Böylece gelen hava veya seda akımını çeşitli noktalarda çeşitli kalıplardan geçirerek şekilli seda, yani ses haline getirir. Bu şekilde hava veya seda akımı ses tellerinden dişlerin dışına çıkıncaya kadar bir çok sesler oluşmuş olur. Bazı seslerde ise hava veya seda akımı dişleri geçtikten sonra bile henüz ses haline gelmiş olmaz. Bu seslerin oluşumunda dudaklar rol oynar. Gelen ve o zamana kadar karşısına bir engel çıkmamış olan akımı dudaklar kalıptan geçirerek ses haline getirirler.

2.3.Oluşum Noktası

Gırtlaktan dudaklara kadar olan geçitte hava veya seda akımının ses haline gelmek için kalıptan geçtiği, boğumlandığı çeşitli noktalar vardır. Bu noktalara seslerin oluşum noktaları denilir. Bir sesi ses haline getiren, onu

seda ve gürültüden farklı, şekilli bir seda yapan şey oluşum noktasıdır. Hava ve seda akımının önüne oluşum noktasından sonra artık hiçbir engel çıkmaz. Ses önündeki açık yoldan rahatlıkla dışarı çıkar.

2.4. Sedalı Sesler, Sedasız Sesler

Seslerin duyulmasını sağlayan sedayı, bir kısım seslerde ses telleri meydana getirir. Oluşumlarında ses telleri titreşmeyen sesleri duyuran seda ise hava akımının gırtlaktan dudaklara kadar ses geçidinde karşısına çıkan engellere çarpması veya sürtünmesinden ileri gelir. Böyle olduğu için sedasını ses tellerinin titreşiminden alan seslerin oluşumu sırasında hem hava akımı hafif, hem de ses geçidinin havanın karşısına çıkardığı engeller zayıf olur; buna karşılık, sedasını çarpma ve sürtünmeden alan seslerin oluşumunda hem hava akımı, hem de ses geçidinin bu akım karşısına çıkardığı engeller kuvvetli olur. Engellerin zayıf veya kuvvetli olması demek ses yolunun az veya çok kapalı olması, hava akımı için rahat veya rahat olmayan bir geçidin bulunması, ses geçidindeki organların normal durumlarından az veya çok farklı hale geçmeleri, gergin veya gevşek olmaları demektir. Şüphesiz sedalarını ses tellerinden alan sesler de, çarpma veya sürtünmeden alan sesler de kendi aralarında bu bakımdan farklıdırlar. Bu farklar çarpma ve sürtünme ile oluşan seslerde az, oluşumlarında ses telleri titreşen seslerde ise çok fazladır. Öyle ki titreşimli seslerin bir kısmı diğer kısmından çok büyük ölçüde ayrılarak, organların gergin veya gevşek olması bakımından aralarında çok fark olduğu halde, çarpma ve sürtünmeyle meydana gelen sesler sınıfına

girer.

Böylece sesler her şeyden önce, titreşimli veya titreşimsiz olarak değil, ses geçidinin açıklık ve kapalılığına, engellerin kuvvetli veya zayıf olmalarına, organların gergin veya gevşek bulunmalarına göre ikiye ayrılırlar.Yani, sedasını ses tellerinden alan seslerin bir kısmının oluşumunda hava ve seda akımının önüne hiçbir engel çıkmaz.Seda, iyice açık olan yolun bir noktasında boğumlandıktan sonra ses halinde dışarı çıkar.Sedasını ses tellerinden alan seslerin diğer kısmı ile sedasını sürtünme ve çarpmadan alan seslerde ise hava ve seda akımının önüne belirli engeller çıkar, ses geçidindeki organlar türlü hareketler yaparak geçidi çeşitli şekillerde daraltır veya kapatırlar.Yalnız bu daralma ve kapanma titreşimlilerde daha zayıf, sürtünme ve çarpma ile seslenenlerde daha kuvvetlidir.Organların birbirlerine yaklaşmaları veya birbiriyle birleşmeleri birincilerde gevşek, ikincilerde ise gergindir.Birincilerde seda ses tellerinde temin edildiği için oluşum noktalarında sadece boğumlanmayı yapacak kadar bir enerji sarfedilir. İkincilerde ise hem seda çıkaracak, hem de boğumlanma yapabilecek kadar kuvvetli bir enerjiye ihtiyaç vardır.Bu yüzden birincilerde hava akımı zayıf, organların teması gevşek; ikincilerde ise hava akımı kuvvetli, organların teması gergin ve sıkı olur.

Takıntıya uğrayarak çıkan seslerin bir kısmı ile takıntıya uğramadan çıkan sesler sedalarını ses tellerinden, takıntıya uğrayarak çıkan seslerin geri kalan kısmı da sedalarını sürtünme ve çarpmadan alırlar.

Sesler için gerçek seda ses tellerinden çıkan sedadır.Ancak bu çeşit sedanın değişik ses perdeleri bulunur, konuşmada

veya melodide ses perdesi ancak bu seda ile yükselip alçalabilir.Çünkü ses perdesinin değişmesi ses tellerindeki titreşimin frekansına, ses tellerinin saniyedeki titreşim sayısına bağlıdır.Ses tellerinin titreşiminden değil de çarpma ve sürtünmeden doğan seslerin perdesi değiştirilemez. O halde gerçek sedalı sesler sedalarını ses tellerinden alan seslerdir. Bu nedenle oluşumlarında ses telleri titreyen seslere sedalı, titremeyen seslere sedasız denir.Seslerin bu sedalı, sedasız özellikleri onların sınıflandırılmasında ve kelime bünyesindeki çeşitli ses olaylarında çok önemlidir.

2.5.Vokal

Oluşumları sırasında ses geçidinde hiçbir engelle karşılaşmayan seslere vokal denir.Sadece, yolun açıklık derecesi vokallerin çeşitlerine göre değişir.Bu değişiklik çenelerin az veya çok açılması, dilin kabarıp alçalması ve dudakların yuvarlaklaşıp düzleşmesi ile sağlanır.Vokallerin oluşumunda oluşum noktalarına göre dilin arkası, ortası veya ucu kabarır fakat ses geçidi tıkanmaz.Vokallerin dışındaki seslerde boğumlanmayı sağlamak için oluşum noktalarında organlar, dil, damak, dişler, dudaklar vb. zayıf veya kuvvetli olarak birbirleriyle temas ederler.Gerçekte oluşum noktaları da bu temas ve yaklaşım noktalarının zirveleridir.Vokallerin oluşumunda ise organlar temas etmezler, açık dudaklar da düz veya yuvarlak şekiller alırlar.Bu düzlük veya yuvarlaklık da vokallere göre dar veya geniş olur.Vokallerin oluşumunun başlangıcında ses telleri vokal için gelen havanın karşısında, normal durumda aralık bulunan arka

tarafklarından birleşerek yolu gevşek bir şekilde kapatırlar. Gelen hava çarparak onların arasından kendine bir yol açar. Bu sırada ses telleri titreşerek seda meydana getirirler. Bu andan itibaren ses geçidi bir daha tıkanmaz. Vokallerin de oluşumunda genellikle ağız yolu açık bulunur. Fakat bazı dillerde nazal vokaller de vardır. Bu vokallerin de oluşumunda ise geniz yolu açık tutulur. Geniz yolunu açık tutmak için, küçük dil aşağı sarkarak dilin kabaran kökü ile birleşir. Geniz yolunu kapayıp ağız yolunu açık tutmak için de küçük dilin kökü yukarı yükselir ve geriye çekilir. İşte vokallerin oluşumunda başlıca ses organlarının hareketleri bunlardır.

2.6. Konsonant

Vokaller dışında kalan diğer seslere konsonant adı verilir. Konsonantların oluşumunda hava ve seda akımı ses geçidinde belirli bir takıntıya uğrar. Ses yolunda belirli bir darlık, gevşek veya gergin bir kapantı olur. Konsonantların bir kısmı sedalı, bir kısmı sedasızdır. Sedalı ve sedasız konsonantlar arasındaki fark sadece bir seda farkı değil, aynı zamanda bir kuvvet farkıdır. Sedalı konsonantlarda ses tellerinin titreşimi için harcanan kuvvet oluşum noktalarında tasarruf edilir. Sedasız konsonantlarda ise oluşum noktalarında hem boğumlanma yapmak, hem de konsonantın duyulması için gerekli sesi çıkarmak üzere daha fazla bir kuvvet harcamak gerekir. Hava akımı sedalılarındakinden daha kuvvetli, ses yolundaki daralma ve kapantı daha sıkı olur. Sedasız konsonantların sedalıları içinde birer karşılıkları vardır. Bu karşılıklı durumu oluşum noktalarının aynı olması temin eder. Aynı noktada

oluşan iki konsonanttan biri sedalı, biri sedasız olursa bunlar karşılıklı konsonant sayılırlar. Karşılıklı konsonantlar, gerekince, birbirlerinin yerlerini kolaylıkla alabilirler. Sedalı konsonantların geri kalan kısmının ise sedasız karşılığı yoktur.

2.6.1. Konsonantların Oluşum Noktaları

Konsonantlar gırtlığın sonundan başlayarak dudaklara kadar ses yolunun çeşitli noktalarında oluşurlar. Oluşum noktası ses tellerinden önce olan konsonantlar çok nadirdir ve ancak bazı dillerde bulunurlar. Ses tellerinden sonra gırtlakta, boğazda, dudaklara kadar ağız boşluğunun çeşitli noktalarında konsonantların oluşum noktaları bulunur. Konsonantların oluşumunda en büyük rolü oynayan organ, dildir. Konsonantların çoğu onun çeşitli hareketleri ile, ağız boşluğunda çeşitli noktalara yaklaşması ve teması ile meydana gelir. Hava ve seda akımına en çok engel çıkaran odur. Özet olarak, konsonantlar gırtlaktan dudaklara kadar ses geçidinde daralma ve kapantılar meydana getiren organların yaklaşma ve temas noktalarında oluşurlar. Bu oluşum noktaları bazan aynı yerde, bazan da grup grup çeşitli bölgelerde toplanır.

2.6.2. Konsonantların Temas Dereceleri

Konsonantların oluşumunda organların temas derecesi her zaman aynı olmaz. Temasa gelen, hava akımına engel çıkaran organlar bazı konsonantlarda tam bir kapantı meydana

getirirler, bazıları ise daha geniş bir geçit bırakırlar.

2.6.3.Nazal Olup Olmama

Konsonantların çoğunda hava ağız yolundan, bir ikisinde ise geniz yolundan dışarı çıkar.Havası geniz yolundan çıkan nazal konsonantların oluşum noktaları da genellikle yine ağız yolundadır.

2.6.4.Konsonantların Özellikleri

Her dilde konsonantların sayısı vokallerinkinden çok fazladır.Tek başlarına, daha doğrusu vokalsiz kelime ve hece oluşturamazlar, kök olamazlar. Tek başlarına söylenince başlarına veya sonlarına daima bir vokal eklenir.Engellerden, kapantılardan doğan sesler oldukları için aralarında ve yanlarında vokal olmadan hece yapamazlar, söylenemezler. Çünkü iki konsonantta ayrı ayrı hareketler yapan organların bu hareketlerini birbirine bağlamak için onların vokallere ait olan sükun halinden geçmeleri, sükun hali ile başlamaları veya bitmeleri gerekir. Kelime içinde tek hecede yan yana gelmeleri de yine bu yüzden güçtür.

Konsonantlar takıntılı oluşumları yüzünden vokallerden sert bir söylenişe sahiptirler.Bazılarında ise bu sertlik kelime içinde açıkça hissedilir.Boyle olanların kelimelerde arka arkaya gelmesi kulağı şiddetle tırmalayan bir ses çirkinliği meydana getirir.

Konsonantların vokaller gibi çeşitli uzunlukları da yoktur. Gerçi oluşumlarında tam bir kapantı olmayanlar tek başlarına sürekli olarak söylenebilirler. Fakat bunlar da ses taklidi bazı sözler dışında, normal kelime bünyesinde sürekli şekilleri ile değil, bir defada çıkan normal süreksiz şekilleri ile kullanılırlar.



3.SES TANIMA

3.1.Giriş

Konuşma tanıma genel olarak, ses işaretlerinin (analog) sayısala dönüştürölüp, elde edilen sayısal verilerden işaretin yazım diline çevirilmesiştir.

Ses, günümüzde en hızlı ve verimli haberleşme ortamlarından biridir. İnsanların konuşarak daha iyi anlaşılabilecekleri açıktır. Ancak bilgisayarla haberleşmede en yaygın ortam olan klavye (keyboard) hem yavaş hem de kullanılması zor olduğundan son yıllarda büyük miktartlı veri girişlerini hızlandırmak için çalışmalar yapılmaktadır. Bu çalışmaların başında da ses (konuşma) gelmektedir. Sesle bilgi girişi klavye kullanımından iki kat daha hızlıdır.

3.2.Ses İşareti

Ses ile haberleşmede, insan beyninde mesajın bir özeti olduğu düşünülebilir. Daha sonra bu mesaj, ses üretmek için gerekli akustik işaretlere dönüştürülür.

Konuşulacak mesaj, ilk önce konuşma mekanizmasını kontrol eden sinirsel işaretlere dönüştürülür. Konuşma mekanizması dil, dudak, ses telleri v.b. organların birlikte hareket etmesidir. Sinir sisteminin uyarılarıyla bu mekanizma hareket ederek, orijinal mesajdaki bilgiyi taşıyan akustik dalga şekli oluşturulur.

Ses üzerinden iletilen bilgi, ayırık özellikler taşımaktadır. Bir ses işaretini sonlu sayıda sembollerle gösterebiliriz. Her ayırık sese bir sembol düşer ki bu ayırık seslere PHONEME denir. Her konuşma dilinin kendine özgü 30 ile 50 arasında PHONEME kümesi vardır.

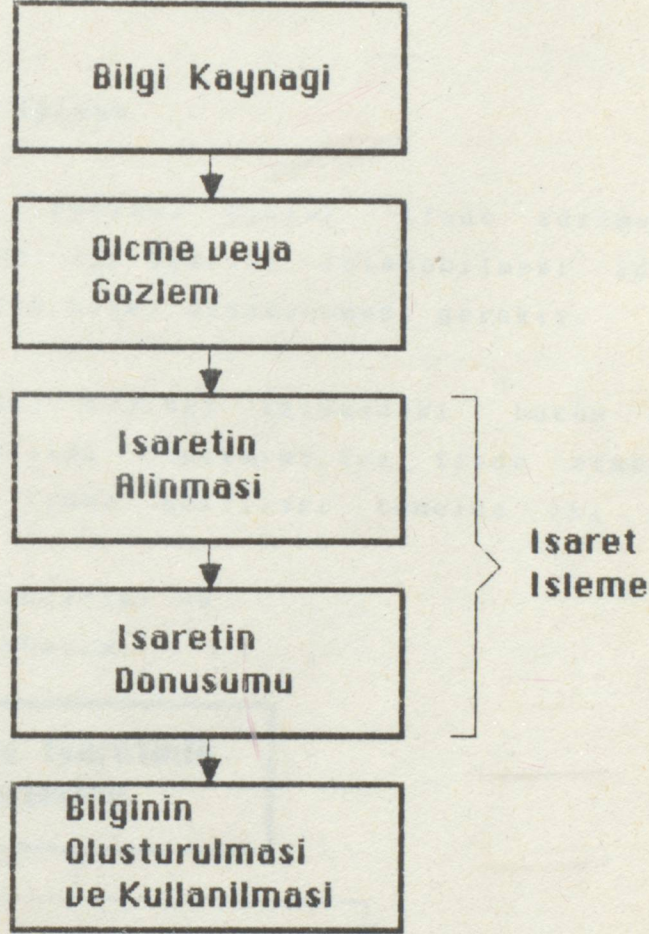
Sesle haberleşme sistemlerinde, ses işareti alınır, saklanır ve bir çok şekilde işlenir. Ses işareti genelde iki şekilde ifade edilir :

- a) Ses işaretindeki mesaj içeriği korunur.
- b) Ses işareti taşıma ve saklamaya uygun veya daha sonraki herhangi bir değişiklikte mesajın içeriğinde önemli bir tahribat yaratmayacak bir biçimde ifade edilir.

Ses işaretini ifade etmek için saniyede 500-1000000 bit bilgi gereklidir. Bu ifadelerdeki tasarımda, işaret işleme yöntemleri önemli rol oynamaktadır.

3.3.İşaret İşleme:

Aşağıdaki şekilde veri işleme olayının genel şekli görülmektedir :



Ses işareti olayında, bilgi kaynağı insandır. Veri işleme olayının son aşaması olan 'bilginin oluşturulması ve kullanılması' ya insan tarafından ya da makineler tarafından yapılır. Bir grup konuşmacı içinden tek bir konuşmacıyı tanıma sisteminde ses işaretinin zamana bağlı spektral ifadesi kullanılabilir. Bu sistemde işarettaki bilgi konuşmacının kimliğidir.

Böylece ses işareti işlemenin iki görevi vardır. İlki ses

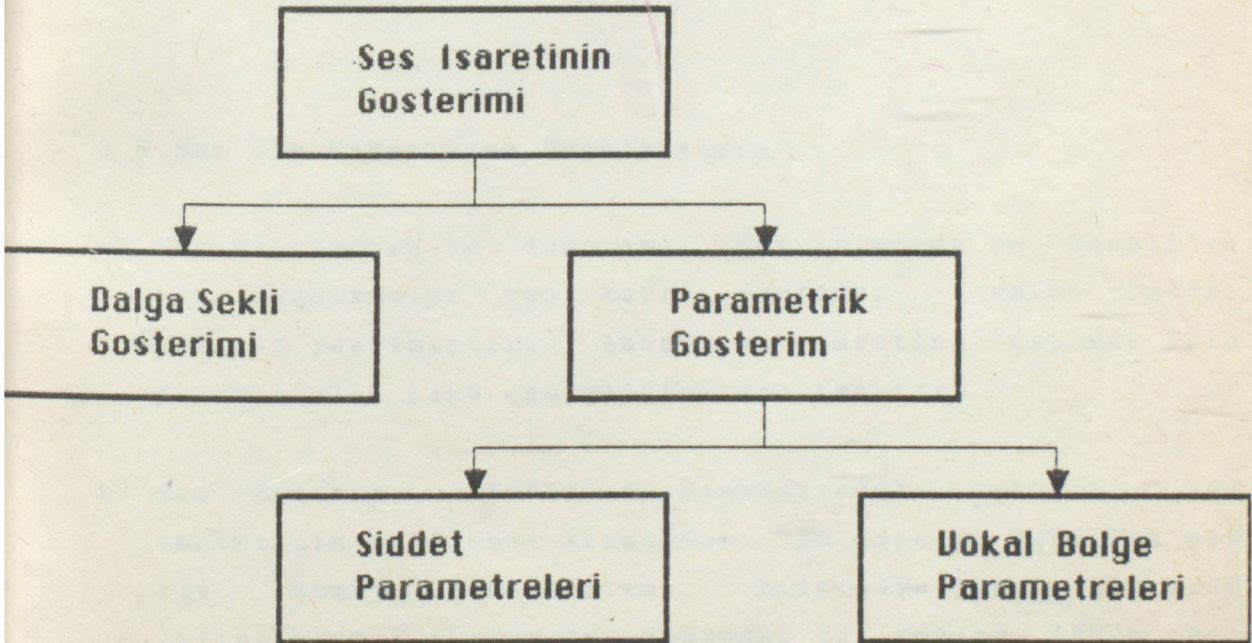
işaretinin, dalga şekli veya parametrik şekliyle ifade edilmesidir.İkincisi, işaret bilgisinin, gerçek olmayan fakat bazı uygulamalara müsait dönüşümünü yapmaktır.

3.4.Sayısal Ses İşleme

Ses işaretinin sayısal olarak ifade edilmesi temel konudur.Sesin en iyi şekilde işlenebilmesi için mümkün olduğu kadar yüksek hızda örneklenmesi gerekir.

Örnekleme işlemi, sayısal işlemedeki bütün teorem ve uygulamaların temelini oluşturur.Sesi ifade etmenin birçok yolu vardır.Bu ifade şekilleri temelde iki ana gruba ayrılır.Yani

- a) Dalga şekli gösterimi ve
- b) Parametrik gösterim.



Dalga şekli gösterimi, ses işaretinin kendisidir. (Analog veya sayısal). Parametrik gösterim ise bir ses üretim modelinin çıkışı gibi ifade edilebilir. Parametrik ifadenin çıkarılmasındaki ilk aşama, sayısal dalga şekli ifadesidir. Yani önce ses örneklenir. Daha sonra ses üretimi için gerekli parametreler çıkarılır. Bu parametreler iki grupta toplanabilir:

- a) Şiddet Parametreleri (Sesin kaynağı ile ilgili)
- b) Vokal Bölge Parametreleri (Ayrık sesle ilgili)

Parametrik gösterimde örnekleme hızı daha düşüktür. Saniyede 15000 bit ve daha aşağısı parametrik gösterim için yeterlidir. 15000-200000 bit arası ise dalga şekli gösterimi için uygundur.

Ses işlemede en önemli nokta, işaret gösteriminin ve sayısal işaret işleme yöntemlerinin seçilmesidir.

3.5. Ses ile Haberleşme Uygulamaları

- a) Sayısal taşıma ve depolama: Ses işlemede en önemli ve ilk uygulamalar ses kayıt (vocoder - voice coder) üzerine yapılmıştır. Amaç, ses işaretini taşımak için gerekli olan bant genişliğini azaltmaktır.
- b) Ses Sentezi: Sesin depolanmasındaki ekonomi, ses sentezinin en önemli konusudur. Bu amaçla, daha çok ses ile komut edilebilen bilgisayarlar üzerinde çalışılmıştır. Ayrıca bu konudaki çalışmalar insan sesi üzerindeki araştırmalarda da büyük rol oynamıştır.

c) Konuşmacıyı tanıma sistemleri: Bazı gizli bilgilere erişmede kişinin sesi parola olarak kullanılabilir.

d) Konuşma Tanıma Sistemleri: Konuşmanın yazılı hale getirilmesidir. Bu tanıma olayı en çok konuşma ortamına, konuşulan mesaja ve konuşmadaki sınırlamalara bağlıdır. Uygulama alanları olarak:

- Bilgisayara veri girişinde elin ve/veya gözün meşgul olduğu zamanlarda verilerin sözle girilmesi,
- Uçak rezervasyonları,
- Endüstri robotlarının ve makinalarının sesle kontrolü,
- Veri Tabanı Yönetimi (Sorgulamaların sözle yapılması),
- Ses kontrollü daktilo (automated dictation)...

gösterilebilir.

3.6. Konuşma Tanımadaki Karşılaşılan Sorunlar

Konuşma tanıma sırasında karşılaşılan sorunlar dört ana grupta toplanır:

1. Ses işaretleri normalde sürekli işaretlerdir. Yani konuşulan cümle içersinde ne kelimeler arasında düzenli bir duraksama ne de kalimenin sınırlarını belirtecek akustik bir işaret vardır. Tanıma algoritması, bu durumda cümle içersine gömülmüş sözcükleri de tanıyabilmelidir. Bunun en kolay yolu konuşmacıya, sözcükler arasında suni boşluklar bıraktırmaktır.

2. Ses işaretleri kişinin durumuna göre büyük farklılıklar

gostermektedir.Örneğin bir kişinin sesi diğerinden cinsiyetine, yaşına veya aksanına bağlı olarak çok değişebilir.Ayrıca bir insanın kendi sesi bile farklı zamanlarda büyük değişiklikler gösterebilir. Örneğin fısıltıyla veya gülererek, soğuk veya sıcak bir ses tonuyla konuşabilir. Bu da sesi etkiler. Gerçekte koşullar aynı olsa bile peşpeşe söylenen iki sözcük hiçbir zaman birbirinin tam olarak aynısı değildir.Fakat benzerdir.

Sesin sürekliliği değişkenliğin diğer bir kaynağıdır. Sözcükler cümle içinde genelde birbirlerine yumuşak geçişler yaptıklarından sözcüklerin başlangıç ve sonları yanıltıcı ölçüde değişebilirler.Örneğin, 'Hep sözlük al' cümlesi hızlı söylenirse 'Hep sözlü kal' olarak anlaşılabilir.

Sesin bu doğal değişkenliği, ses işaretinin kendisini esas alan tekniklerde çok büyük sorunlar ortaya çıkarmaktadır.Böylece tanıma algoritması sesin kendisi ile ilgileneneğine karakteristik özellikleriyle ilgilenmelidir.

3. Üçüncü sorun belirsizliktir. Örneğin İngilizce'deki 'to','too' ve 'two' sözcükleri arasında fazla bir akustik fark yoktur. Benzer şekilde 'grey tape' ile 'great ape' hemen hemen aynı şekilde söylenirler. Bu durumda, tanıma algoritması konuşma içindeki ayırık sözcükleri de tanıyabilmelidir.

4. Konuşma, insan dilinin karmaşık sistemi içinde sadece küçük bir bölümdür.Genelde mesajın kendisinden ziyade, taşıdığı anlam önemlidir. Örneğin yolda birine

'Saatiniz var mı?' diye sorduğumuzda gerçek niyetimiz saatin kaç olduğunu sormaktır. Bu soruya cevap ta 'Evet var.' değil, '16:05' olmalıdır.

Bu sorunlara bakıldığında otomatik konuşma tanıma olayı, hemen hemen erişilmez bir hedef gibi görülür. Oysa önceden yapılan bir takım kabullerle bu sorunların büyük bir kısmı halledilir.



4.SÖZCÜĞÜN UÇ NOKTALARININ BELİRLENMESİ

4.1.Giriş

Söylenilen bir sesin hangi kelime olduğunu anlamak için içerisinden anlam taşıyan bölümü ayırmak gerekir.Çünkü ayrık sözcük tanıma olayında sözcük söylenirken mikrofon belli bir süre (2 sn.) açık kalmaktadır. Söylenen sözcük bu 2 sn.lik sürenin herhangi bir anında başlayıp bir süre devam eder.Sözcük süresi, başlama anı, doğal olarak her sözcükte farklı olacaktır.Bu sebeple ses bilgisi içerisinden sözcük bilgisi sözcüğe zarar vermeden ayrılmalıdır.

Konuşmas tanıma olayında, sözcüğü ortamdan doğru olarak izole etmek çok önemlidir. Düşünülebilir ki, konuşma olmadığı zaman ses genliği sıfırdır. Fakat bu, çok ideal bir ortamda çalışılırsa doğrudur.Örneğin ses geçirmeyen, yalıtılmış bir odada çalışmak gerekir.Oysa günlük hayatta

daima etrafımızda gürültüler vardır. Konuşmacının sesini bu gürültülerden ayırmak için öncelikle bu gürültünün seviyesi hesaplanmalıdır. Kabaca denilebilir ki bu seviye aşıliyorsa anlamlı sözcük başlamıştır. Yine denilebilir ki bir kere bu eşik seviyesini aşan ses tekrar bu seviyenin altına düşerse anlamlı sözcük tamamlanmıştır.

Genel olarak sözcüklerin ön ve arkalarında yukarıda tanımlanan sessizlik veya arka plan gürültüsü 'background noise' vardır. Yani kısaca, bir sözcük söylendiğinde, bu sözcük sessiz bölgelerle sınırlanmıştır. İşte, sözcüğü bu ses dizisinden ayırma işlemine UÇ NOKTA TARAMASI denir.

Ayrık sözcük tanıma sistemlerinde, uç nokta taramasının duyarlı olması iki sebepten dolayı önemlidir:

- 1) Güvenilir bir sözcük tanıma yapılabilmesi için uç noktaların duyarlı olarak tespit edilmesi gerekir.
- 2) Eğer uç noktalar duyarlı olarak tespit edilirse sesin işlenmesi için gerekli zaman ve işlem sayısı en aza indirilir.

Uç noktaların belirlenmesinde ortaya çıkacak problemler konuşmacıdan kaynaklanabileceği gibi, ses taşıma ve kayıt sistemlerinden de kaynaklanabilir. Bir takım arka plan gürültüleri oluşabilir ki bu, uç noktaların duyarlı olarak tespit edilmesini zorlaştırır. Örneğin; konuşmacının ağız ve diş yapısına bağlı olarak sözcüklerin başlangıcında veya bitiminde belirsizlikler oluşabilir. Eğer bir konuşmacı sözcüğü söylemeden önce derin bir nefes alıyorsa bu ne gürültüdür ne de sözcüktür. Veya konuşmacının ön dişleri aralıksa 'SEKİZ' gibi sözcüklerin başlangıç ve bitim

noktalarının tespiti zor olur.Çünkü bu gibi durumlarda /s/ ve /z/ seslerinin süresi çok yayvandır.

Benzer şekilde konuşma sırasında bir kapının açılıp kapanması veya bir sandalyenin sürtülmesi yine problem yaratmaktadır.

Bu tür etkileri yok etmenin bir yolu mikrofona yakın konuşmak ve gürültü kesici mikrofon kullanmaktır.

Yapılan çalışmaların büyük bir bölümünde sözcüğün belli bir aralıkta söylendiği varsayılmıştır.Bu projede sözcüğün söylenmesi için 2 sn.lik süre tutulmuştur.Dolayısıyla bu tür çalışmalar 'non-real time' sözcük tanıma sistemi olarak adlandırılır.Gerçek zamanlı çalışmalarda ise daha sözcük bitmeden başlangıcı belirlenmelidir.

Sözcük tanıma sistemlerinin ana bölümleri, özellik oluşturma, şablon karşılaştırma ve karar vermedir.

Uç nokta tanıma işlemi üç tip mantıkla yapılır:

- 1) EXPLICIT: Sözcük tanıma işleminde ayrı bir bölüm, uç nokta tanıma bölümü olarak geçer.Bu bölüm sözcük tanıma ve karar verme aşamalarından bağımsız çalışır.
- 2) IMPLICIT: Ayrık sözcüğün uç noktaları, sözcük tanıma ve karar aşamaları sırasında belirlenir.Ayrı bir bölüm yoktur.
- 3) HYBRID: Önceki iki yöntemin karışımı bir mantıktır.Önce uç noktalar için birden fazla tahmin yapılır.Daha sonra tanıma fazındaki duruma göre bu tahminlerden en iyisi

seçilir.

4.2.EXPLICIT VE IMPLICIT Uç Nokta Taraması

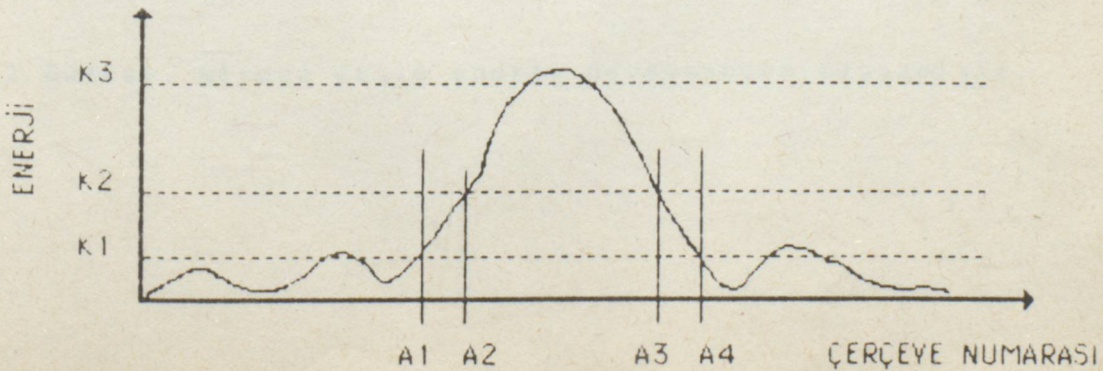
Kaydedilen ses işaretinin enerji eğrisi ve buna bağlı olarak tespit edilen eşik seviyeleri kullanılarak sözcüğün başlangıç ve bitimi tahmin edilebilir.

Sıfır-geçiş eğrisiyle (Zero-crossing contour) f,s,z,v gibi sürtmeli (fricative) seslerle başlayan veya biten sözcüklerin uç noktaları tespit edilebilir.Sıfır-geçişin kullanılması EXPLICIT için uygun değildir.

4.3.HYBRID Uç Nokta Taraması

Tarayıcının giriş bilgisi $R_l(0)$, $l=1,2,\dots,L$ enerji dizisidir. Burada L , kayıt aralığındaki çerçeve sayısıdır.Tarayıcının çıkışı başlangıç $B(m)$ ve bitim $E(m)$ dizileridir.Şablon benzetme ve karar verme aşamalarında, bu uç nokta çiftlerini kullanarak en yakın sözcük bulunur.Bir uç nokta çiftleri arasındaki uzaklık yeterli derecede küçükse diğer çiftlerle ilgilenilmez.Aksi takdirde diğer çiftlerin incelenmesine devam edilir.

Son nokta taramasına, aşağıdaki şekilde de görüldüğü gibi 2 enerji eşiğinin (K_1, K_2) tespitiyle başlanır.



Bu eşiklerin tespitinin sebebi, ses benzeri enerji patlamalarının tanımlanmasıdır. Ses içerisinde bu enerji patlamalarından birden fazla bulunabilir. Asıl problem bunlardan hangisinin konuşulan sözcüğe ait olduğuna karar vermektir.

Enerji darbelerini tarama soldan sağa doğru yapılır. $R_1(0)$ değerleri taranır ve ilk K_1 eşiğini aşan enerji çerçevesinin numarası (A_1) kaydedilir. Enerji seviyesi, K_1 'in altına düşmeden daha yüksek bir K_2 eşiğini aşıyor ise enerji seviyesinin başlangıcı tespit edilir. Başlangıç noktası A_1 olarak seçilir. Eğer $A_2 - A_1$ süresi çok uzunsa o zaman başlangıç noktası A_2 olarak seçilir. Bitim noktası buna benzer bir şekilde K_1 ve K_2 eşikleri kullanılarak tespit edilir. Eğer A_3 ile A_4 arasındaki zaman farkı çok uzunsa A_3 noktası bitim noktası olarak kabul edilir.

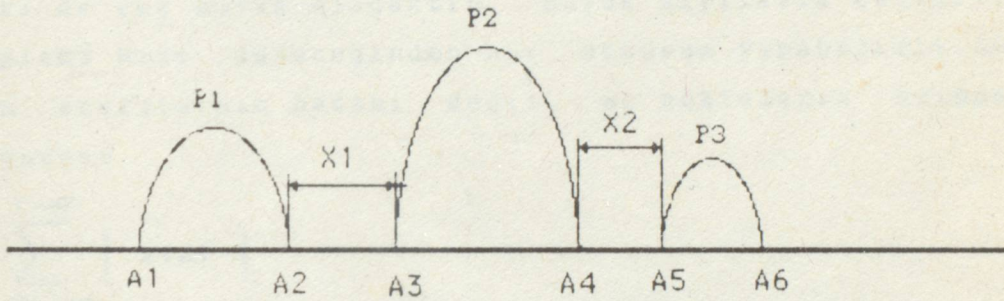
Her taranan enerji darbesi için iki test yapılır. Darbenin tepe enerjisi ölçülür. Eğer bu enerji K_3 eşliğinden küçükse bu darbe atılır. Aynı zamanda darbe süresi çok küçükse (75 ms.'den küçükse), o darbe de atılır.

Enerji darbe tarayıcısının çıkışı, darbe başlangıç noktaları $P_B(m)$ ve darbe bitim noktaları $P_E(m)$ dizileridir ($m=1,2,\dots,M$, M taranan darbe sayısı). $M=0$ bulunursa kayıt iptal edilir.

Daha sonra bu uç nokta çiftlerinden hangisinin uygun olduğuna karar verilir. Karar verilirken aşağıdaki durumlar gözönünde bulundurulur:

- 1) Sözcük, birden fazla enerji darbesinden oluşabilir.

- 2) Sözcük içerisinde daima maksimum enerjiyi içeren bir çerçeve bulunmalıdır.
- 3) İki enerji darbesi arasında bir durma boşluğu vardır.
- 4) Eğer durma boşluğu 150 ms'yi aşarsa bunun kelimeye ait olmadığı düşünülür.



Yukarıdaki şekilde 3 enerji darbesi görülmektedir. Bunlar birbirlerinden $X1$ ve $X2$ çerçeve kadar ayrılmışlardır. Eğer $X1$ ve $X2$, 150 ms.'den küçükse ilk uç nokta çifti ($A1, A6$) çifti seçilir. İkinci çift eğer $X1 > X2$ ise ($A3, A6$) seçilir. Üçüncü çift ($A1, A4$) seçilir. Dördüncü çift olarak ($A3, A4$) seçilir.

Eğer $X1 > 150$ ms. ve $X2 < 150$ ms. ise seçilen sıralı çiftler ($A3, A6$), ($A3, A4$), ($A1, A4$) dır.

Eğer $X1$ ve $X2$, 150 ms.'den büyükse ($A3, A4$), ($A3, A6$) ve ($A1, A4$) seçilir.

4.4.Enerji ve Sıfır Geçiş Oranlarının Hesaplanması

Enerjinin genel tanımı

$$E = \sum_{n=-\infty}^{+\infty} x^2(n)$$

dir. $x(n)$, örneklenmiş sestir.Burada $x(n)$ değerlerinin kareleri oldukça büyük değerler olabilir.Dolayısıyla E değeri de çok büyük olacaktır. Büyük sayılarla çalışılırsa hesaplama hızı düşeceğinden bir dönüşüm yapabiliriz.Amaç, sesin enerjisinin hesabı değil, uç noktaların bulunması olduğundan

$$E = \sum_{n=-\infty}^{+\infty} |x(n)|$$

ifadesi kullanılabilir.Buradaki E değeri, uç noktaların bulunmasında bize yardımcı olabilecek bir büyüklüktür.

n değerinin pratikte sonsuz değerler alamayacağı açıkça ortadadır. Bu durumda, kayıt edilen ses belli uzunluklarda çerçevelere ayrılır ve her çerçeve için ayrı enerji hesaplanır. Böylece,

$$E(m) = \sum_{n=0}^{N-1} |x(m-n)*w(n)|, m=1,2,\dots,M$$

ifadesi kullanılabilir.Burada, N çerçeve uzunluğu, M çerçeve sayısıdır. $w(n)$ çerçeve fonksiyonudur.

Çeşitli çerçeveleme fonksiyonları (Windowing Function) vardır. Bunlardan ikisi:

a) Hamming Window:

$$w_H(n) = 0.54 - 0.46 \cdot \cos\left(2 \pi n / (N-1)\right), \quad 0 \leq n \leq N-1$$

$$= 0, \quad \text{diğer yerlerde}$$

b) Rectangular Window:

$$w_R(n) = 1, \quad 0 \leq n \leq N-1$$

$$= 0, \quad \text{diğer yerlerde}$$

Bu çerçeveleme yöntemlerinden hangisinin tercih edileceği ve analizi sonra yapılacaktır. Fakat yukarıdaki enerji ifadesinde $w_R(n)$ kullanılırsa m. çerçevenin enerjisi

$$E(m) = \sum_{n=0}^{N-1} |x(m-n)|, \quad \text{olarak sadeleşebilir.}$$

4.5. Sıfır-geçiş Oranının Hesaplanması

$$zcr(m) = \sum_{n=0}^{N-1} | \operatorname{sgn}(x(m-n+1)) - \operatorname{sgn}(x(m-n)) | * w(n)$$

$$\operatorname{sgn}(x(n)) = 1 \quad x(n) \geq 0$$

$$\operatorname{sgn}(x(n)) = -1 \quad x(n) < 0$$

$$w(n) = 0.5 \quad 0 \leq n \leq N-1$$

$$w(n) = 0 \quad \text{diğer yerlerde}$$

$z_{cr}(m)$ fonksiyonu, m . çerçeveye ait sıfırdan geçiş sayıdır. Bu sayı, sesin o çerçeveye ait frekansını temsil eder. Anlaşılacağı gibi $z_{cr}(m)$ 'in büyük olması sessizlik bölgesini veya gürültüyü; küçük olması ise sesli bölgeyi gösterir.



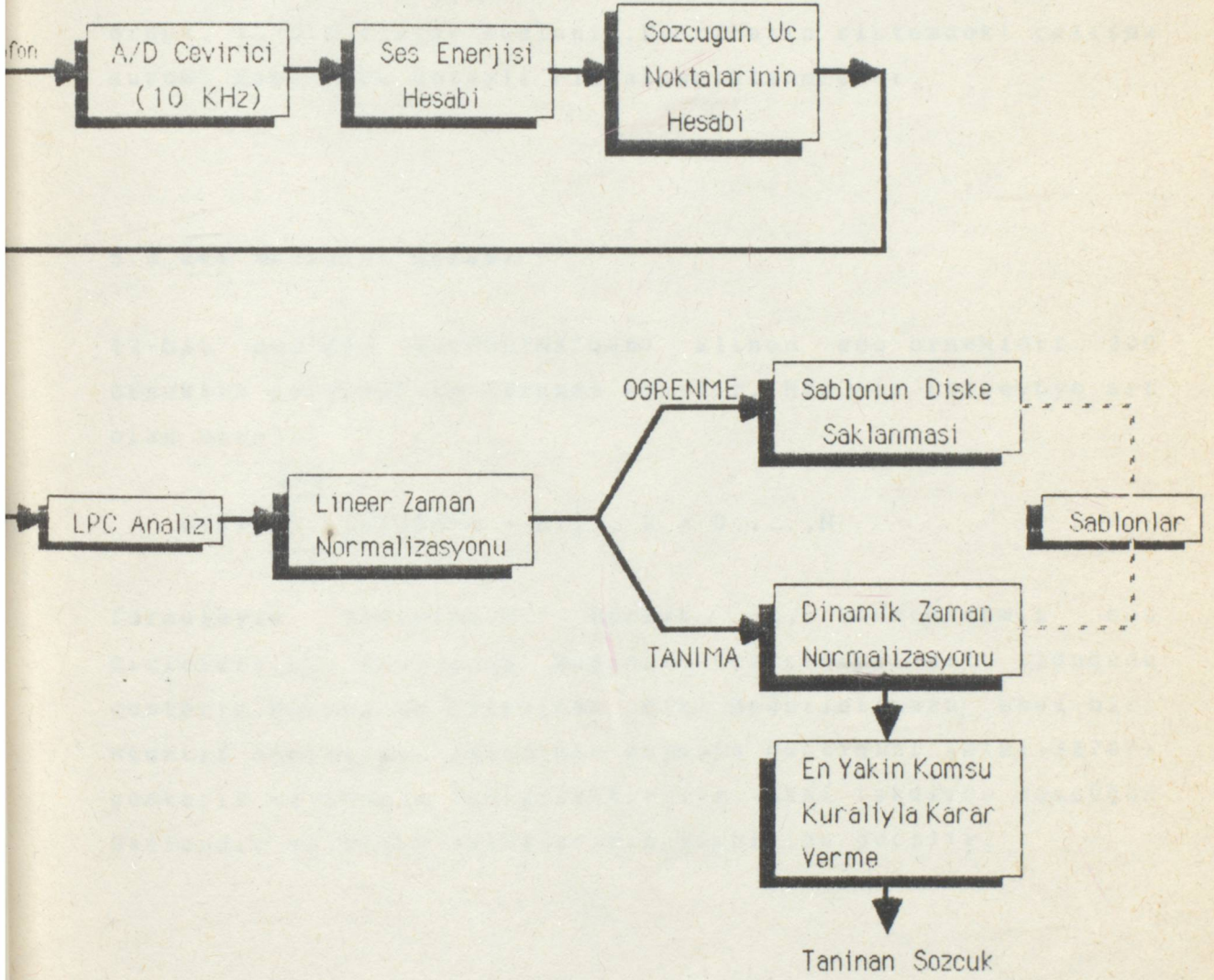
5. AYRIK SÖZCÜK TANIMA SİSTEMİ

5.1. Giriş

Günümüzde en çok kullanılan sözcük tanıma sisteminin blok diyagramı ŞEKİL 5.1 de görülmektedir. Bu şemada da görüldüğü gibi sistem temelde iki modda çalışmaktadır: ÖĞRENME ve TANIMA.

ÖĞRENME modunda, mikrofondan alınan sözcüğü karakterize edecek bir şablon çıkarılır ve bu şablon daha sonra diske kaydedilir.

TANIMA modunda ise, mikrofondan girilen ve o ana kadar ne olduğu sistem tarafından bilinmeyen sözcüğün ÖĞRENME modunda olduğu gibi şablonu çıkarılır ve bu şablon diskteki diğer şablonlarla karşılaştırılarak benzerliği en fazla olan şablonun ait olduğu sözcük tanınan sözcük olarak kabul edilir.



ŞEKİL 5.1 Ayrık Sözcük Tanıma Sisteminin Blok Diyagramı

5.2.A/D Çevirici

Sistemde kullanılan ADC, daha önce de bahsedildiği gibi MICROLINK arabirimidir.Örnekleme hızı 10KHz.'dir.Her bir örnek, 12-bit olarak kodlanır.Bu cihazın sistemdeki çalışma durumu daha önce detaylı olarak açıklanmıştır.

5.3.Ses Enerjisi Hesabı

12-bit ADC'den (MICROLINK'den) alınan ses örnekleri 200 örneklilik çerçevelere (frame) ayrılır. Her bir çerçeveye ait olan enerji,

$$e(k) = \sum_{n=0}^{200} |x(200*k + n)|, \quad k = 0, \dots, N$$

formülüyle hesaplanır. Burada x, örneklenmiş ses dizisidir.k, enerjinin kaçınıcı çerçeveye ait olduğunu gösterir.Hesaplama sırasında e(k) değerlerinden enaz biri negatif olursa bu, enerjinin doymaya geçtiğini (e(k)>32767) gösterir ve sözcük tekrarlattırılır. Aksi takdirde sözcüğün başlangıç ve bitiş noktalarının tespitine geçilir.

5.4.Sözcüğün Uç Noktalarının Belirlenmesi

Burada bulunacak olan şey, sözcüğün uç noktalarını içeren çerçevelerin numarasıdır. Çerçeve uzunluğu 200 örnek olduğundan doğal olarak sözcüğün başladığı ve bittiği noktaların tespitinde 200 örneklilik bir hata yapılır. Bu ise o kadar önemli değildir. Çünkü sözcükleri sessizlik anından

ayırarak tam ve kesin bir sınır yoktur. Şekil olarak bile bunu tespit etmek kolay değildir. Ancak hatayı azaltmak için çerçeve uzunluğunu mümkün olduğu kadar küçük tutmak gerekir. Fakat fazla da küçültürsek işlem sayısı artacak ve hız düşecektir.

Uç noktaları hesaplamada kullanılan yöntem daha önce detaylı olarak açıklanmıştır. İlk önce, gürültü seviyesini tutturmak için kayıt aralığının son 20 çerçevesinin enerji ortalaması hesaplanır. Bu bölge sessizlik olarak kabul edildiğinden sözcüğün mutlaka sonunda yeterli derecede bir boşluk bırakılmalıdır. Daha sonra uç noktaların tespitinde gerekli olan alt ve üst enerji seviyeleri hesaplanır.

```
alt_enerji = 0.03*(max_enerji - ort_enerji) + ort_enerji
alt_enerji = min(alt_enerji , 4*ort_enerji)
üst_enerji = 5*alt_enerji
```

Enerjisi alt seviyeyi geçen çerçevenin, enerji hiç bu seviyenin altına düşmeden üst seviyeyi de geçiyorsa sözcüğün başlangıcını içerdiği kabul edilir. Aynı şekilde bitişi içeren çerçeve de hesaplanır. Bu projede kullanılmamasına rağmen heceler de aynı mantıkla diğerlerinden ayrılmakta ve sözcüğün kaç heceli olduğu kolayca hesaplanabilmektedir.

5.5.Doğrusal Tahminli Kodlama Analizi

Konuşma tanıma olayında en çok kullanılan özellik (feature) analizi Itakura'nın önerdiği DTK (LPC - Linear Predictive Coding) analizidir. DTK'nın anafikri, belli bir andaki konuşma işaretini kendinden önceki konuşma işaretlerinin doğrusal kombinasyonu olarak ifade edebilmektir. Sonlu bir aralıkta gerçek örneklerle, doğrusal olarak tahmin edilenler arasındaki farkların karelerinin toplamını minimize ederek, belirli bir tahmin katsayısı kümesi hesaplanabilir. Bu katsayılara Doğrusal Tahmin Katsayıları (Linear Prediction Coefficients) denir. DT Kodlama metodu, zamanla değişen doğrusal bir sistemi karakterize eden parametreleri çıkarmak için sağlam, güvenilir ve duyarlı bir metoddur.

$$S'(n) = \sum_{k=1}^P a(k) * S(n-k)$$

Yukarıdaki ifadede $a(k)$, $k=1,2,\dots,p$ Doğrusal Tahmin Katsayılarıdır. S dizisi orijinal ses işaretlerini içerir, S' dizisi ise Doğrusal Tahmin Katsayılarıyla oluşacak yeni dizidir. Bu durumda

$$e(n) = S(n) - S'(n)$$

$$e(n) = S(n) - \sum_{k=1}^P a(k) * S(n-k)$$

hatasını minimum yapacak $a(k)$ değerleri bulunmalıdır. Toplam hata E ;

$$E = \sum_n e(n)^2 = \sum_n \left(S(n) - S'(n) \right)^2$$

$$E = \sum_n \left(S(n) - \sum_k a(k) * S(n-k) \right)^2$$

Şimdi bütün problem bu E değerini minimize eden $a(k)$ 'leri bulmaktır. Bu durumda E'nin, a 'nın her bir elemanına göre kısmi türevini 0'a eşitlemek gerekir.

$$\begin{aligned} \partial E / \partial a(i) &= \sum_{n=-\infty}^{+\infty} 2 * \left(S(n) - \sum_{k=1}^P a(k) * S(n-k) \right) * (-S(n-i)) = 0 \\ \sum_{n=-\infty}^{+\infty} S(n) * S(n-i) - \sum_n \sum_{k=1}^P a(k) * S(n-k) * S(n-i) &= 0, \quad 1 \leq i \leq p \\ \sum_{k=1}^P a(k) * \sum_n S(n-k) * S(n-i) &= \sum_n S(n) * S(n-i), \quad 1 \leq i \leq p \end{aligned}$$

Burada $m=n-i$ koyarsak, ve $-\infty < n < +\infty$ olarak kabul edilirse; $n=m+i$ olacaktır.

$$\sum_{k=1}^P a(k) * \sum_{m=-\infty}^{+\infty} S(m) * S(m+i-k) = \sum_{m=-\infty}^{+\infty} S(m) * S(m+i), \quad 1 \leq i \leq p \quad (1)$$

$S(n)$ 'in otokorelasyon fonksiyonu $R(i)$,

$$R(i) = \sum_{n=-\infty}^{+\infty} S(n) * S(n+|i|), \quad 1 \leq i \leq p \quad \text{dir.}$$

$$R(-i) = R(i) \quad (\text{çift fonksiyon})$$

Böylece (1) ifadesi,

$$R(i) = \sum_{k=1}^P a(k) * R(|i-k|), \quad i=1,2,\dots,p \quad \text{haline gelir.} \quad (2)$$

$S(n)$, çerçeve dışında (Hamming Window kullanılmıştır) yani $n < 0$ ve $n > N$ ($N=400$) aralığı dışında sıfır kabul edildiğinden

$$R(i) = \sum_{n=0}^{N-1-|i|} S(n) * S(n+|i|) \quad \text{haline gelir.}$$

(2) ifadesini matris gösterimi ile ifade edecek olursak

$$\begin{bmatrix} R(0) & R(1) & R(2) & \dots & R(p-1) \\ R(1) & R(0) & R(1) & \dots & R(p-2) \\ R(2) & R(1) & R(0) & \dots & R(p-3) \\ \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot \\ R(p-1) & R(p-2) & R(p-3) & \dots & R(0) \end{bmatrix} \begin{bmatrix} a(1) \\ a(2) \\ a(3) \\ \cdot \\ \cdot \\ \cdot \\ a(p) \end{bmatrix} = \begin{bmatrix} R(1) \\ R(2) \\ R(3) \\ \cdot \\ \cdot \\ \cdot \\ R(p) \end{bmatrix}$$

Boyle bir matrisi çözecek en hızlı algoritma Durbin's Recursive Procedure'üdür.

5.5.1.Durbin's Recursive Procedure

$$R(i) = \sum_{k=1}^p a(k) * R(|i-k|)$$

ifadesini kullanarak Tahmin Katsayılarını hesaplamak için anlaşılacağı gibi o çerçeveye ait bölümün önce otokorelasyon katsayıları hesaplanmalıdır. Çerçeve uzunluğu bu kez 400 olarak tespit edildiğinden ilgilenilen çerçevenin dış kısımları

$$S(n) = S(n) * w(n) , w(n): \text{Hamming Window Function}$$

çarpımıyla sıfırlanır. Daha sonra ses işaretinin bu yeni hali üzerinden otokorelasyon katsayıları hesaplanır.

Bu durumda çerçeveye ait otokorelasyon katsayıları bilindiğine göre artık $a(k)$ Doğrusal Tahmin Katsayıları hesaplanabilir. Bu hesap için gerek hız gerek bellek

açısından en uygun algoritmanın Durbin's Recursive Algorithm'i olduğu araştırmalar sonunda bulunmuştur. Çünkü otokorelasyon matrisi simetrik özel bir matristir. Dolayısıyla alışılmış yöntemler bu özel denklem sisteminin çözümünde yavaş kalmaktadır.

Durbin's Recursive Procedure'ü aşağıdadır:

$$E^{(0)} = R(0)$$

$$k(i) = \left(R(i) - \sum_{j=1}^{i-1} a(j)^{(i-1)} * R(i-j) \right) / E^{(i-1)}, \quad 1 \leq i \leq p$$

$$a(i)^{(i)} = k(i)$$

$$a(j)^{(i)} = a(j)^{(i-1)} - k(i) * a(i-j)^{(i-1)}, \quad 1 \leq j \leq (i-1)$$

$$E^{(i)} = (1 - k(i)^2) * E^{(i-1)}$$

Bu eşitlikler rekürsif olarak $i=1,2,\dots,p$ için çözülür. Sonuçta

$$a(j) = a(j)^{(p)}, \quad 1 \leq j \leq p$$

değerleri Doğrusal Tahmin Katsayıları olarak elde edilir. Burada görülen parantez içindeki üstel ifadeler formüllerin kaçınıcı adımdaki hesabı olduğunu gösterir. Örneğin enson yazılan eşitlik $a(j)$ 'lerin p . adımdaki değerinin enson değer olduğunu gösteriyor.

Formüllerdeki p değeri, DT Katsayılarının derecesini gösterir ve bu değer bu projede 10 olarak alınmıştır. Araştırmalar, p değerinin 8' den küçük olduğu durumlarda bu algoritmanın iyi sonuç vermediğini göstermektedir. Eğer p

değeri 12 veya 14'den daha büyük alınırsa veya bu sayı sonsuza büyütülürse sonucun duyarlılığı açısından pek fazla bir değişiklik olmadığı yine bu konuda yayınlanan makalelerde açıklanmaktadır. Bu projede $p=10$ alınmasının sebebi sistemin hızını düşürecek hesap şişmelerinden kurtulmaktır ve duyarlık açısından da iyi sonuç vermiştir.

Yukarıda da anlatıldığı gibi her çerçeve için 10 adet DT katsayısı hesaplanmaktadır. Daha ilerde anlatılacağı gibi sözcük uzunluğu 25 çerçeve ile normalize edildiğinden bir sözcüğü karakterize edecek katsayı sayısı 250 dir. Herbiri 2'şer byte'dan 500 byte yani yaklaşık 0.5 Kbyte büyüklüğünde şablonlar oluşturulmaktadır.

5.5.2. Hamming Window Kullanımı

Daha önce de açıklandığı gibi her çerçeve işleme girmeden önce bir çerçeve fonksiyonundan geçirilir. Eğer Rectangular Window Fonksiyonu kullanılırsa esas eğri sanki bir ekmek dilimi gibi kesilmiş olur. Çünkü;

$$w(n) = 1, 0 < n < N=400$$

$$w(n) = 0, \text{ diğ} \text{er yerlerde}$$

Bu durumda çerçevenin dışındaki bölgeler aniden sıfır olacağından analiz sonucunda yanlış katasayılar oluşacaktır. Oysa Hamming Window Fonksiyonu incelenecek olursa uç noktalara doğru Çan Eğrisi gibi yumuşatılarak sıfırlanır yani aniden sıfır olmaz. Bu da analizdeki hatayı azaltır.

Ancak Hamming Window kullanıldığı takdirde çerçevenin uç noktalarına rastlayan değerler yavaşça sıfırlanacağından bu bölgelerin analizi yapılamaz; daha doğrusu yanlış değerler üzerinden yapılır. Bunu önlemek için peşpeşe gelen çerçeveler ucucuna (Non-Overlapping) değil birbirinin üzerine çakıştırılarak (Overlapping) yerleştirilmelidir. Oluşturulan bu sistemde çerçeveler 200 örnek kaydırılarak hesaba girmektedir. Böylece birinde kaybedilen veri diğer çerçevede hesaba girer. Örneğin 1000 örneklik bir ses bilgisini herbiri 400 örneklik çerçevelere yukarıda açıklanan mantıkla ayıracak olursak çerçeve sınırları şöyle olacaktır:

<u>Başlangıç</u>	<u>Bitiş</u>
0	399
200	599
400	799
600	999

Görüldüğü gibi çerçeve sayısı artmış, ancak böyle yapmakla sistemin duyarlılığı da artmıştır. Örnek olarak 350. örneği göz önüne alalım; bu örnek 1. çerçevenin sonlarına rastladığından çarpım katsayısı küçüktür. Oysa aynı örnek 2. çerçevenin 150. örneği olarak ortalara rastlar ve çarpım katsayısı büyüktür.

5.6. Doğrusal Zaman Normalizasyonu

Söylenen sözcüğün sınırları belirlendikten ve karakteristik özellikleri yani Doğrusal Tahmin Katsayıları çıkarıldıktan sonra sıra Tanıma Aşamasına gelir. Ancak daha önceden de belirtildiği gibi, aynı sözcüğün uzunluğu bile, yani vektör

sayısı (her çerçeve için 10 elemanlık bir vektör oluşturulduğu daha önce açıklanmıştı) farklı zamanlardaki söylenişlerde farklı sayılarda olabilmektedir. Böyle olunca iki şablonu doğrudan correlate etmek iyi sonuç vermeyecektir. Bununla beraber, bütün şablonların başlangıçlarını denk getirerek ve sadece çakışan kısımları correlate ederek uygun bir ölçü elde edilebilir. Veya karşılıklı farkların karelerinin toplamı da bir ölçü olarak alınabilir. Bu yöntem minimum hesap gerektirir. Her şablon için N vektör birbiriyle karşılaştırılır. Burada N, kısa olan şablondaki vektör sayısıdır.

Diğer bir karşılaştırma yöntemi ise şöyledir: Önce her iki şablonun başlangıçları çakıştırılır ve daha sonra sonları çakışınca kadar adım adım kısa olan şablon diğerinin üzerinde kaydırılır. Her adımda çakışan kısımların benzerliği hesaplanır. En yüksek benzerlik ölçüsü, karşılaştırmamanın sonucu olarak kabuledilir. Hesaplama açısından bu yöntem oldukça yüküldür. Her şablon için $M \cdot (N - M)$ vektör karşılaştırması yapılır. Burada N, uzun şablonun; M, kısa şablonun vektör sayısıdır. Bu yöntemde tanıma yüzdesi öncekinden daha yüksektir.

Her iki teknikte de sorun, aynı sözcüğün iki söylenişi arasında uzunluk farkı olabileceğinin dikkate alınmamasından kaynaklanmaktadır. Örneğin, bir 'HELİKOPTER' sözcüğünü üç kere arka arkaya söylesek bunların uzunlukları birbirinden azçok farklı olacaktır. Bu durumda zaman eksenini üzerinde, yani uzunlukta bir optimizasyon yapmak gerekir. Bu tekniğe Doğrusal Zaman Normalizasyonu (Linear Time Normalization veya Linear Time Warping - LTW) denir.

Pratikte ya her referans şablonu, bilinmeyen sözcüğün şablonuna göre ya da bütün şablonlar standart bir uzunluğa göre normalize edilir. Her iki durumda da fark hesabı sabit bir vektör sayısına bağlıdır. Her şablon için N vektör karşılaştırması yapılır. Burada N, bilinmeyen sözcüğün veya standart kabul edilen şablonun vektör sayısıdır.

Projenin deneme aşamasında sözcük uzunluklarının (L), $14 < L < 28$ aralığında değiştiği gözlenmiştir. Bu durumda bütün şablonlar 25 vektör uzunluğuna getirilmiştir. Eğer bilinmeyen sözcüğe göre normalize edilirse, Tanıma Aşamasında her referans şablonu için normalizasyon işlemi yapılması gerekir. Bu da sistemin hızını düşürecektir. Oysa şablonlar son halini almadan önce sabit bir uzunluğa (25 vektör) getirilip saklanırsa bilinmeyen sözcüğün bir kez 25'e normalize edilmesi yetecektir. Normalizasyon Fonksiyonu:

```

oran = INT((bm-1) / (M-1))
ms = 0, ..., bm-1 ve i = 0, ..., p için
    k = ms*oran
    m = INT(k)
    s = k-m
    rs(ms,i) = (1-s)*r(m,i) + s*r(m+1,i)
r(a,10) : Normalize edilmemiş şablon
bm : Normalize edilmemiş şablonun uzunluğu
m : Normalize edilmemiş şablonun indisi
rs(25,10) : Normalize edilmiş şablon
M : Normalize edilmiş şablonun uzunluğu
ms : Normalize edilmiş şablonun indisi
i : Vektör elemanları
a : Vektör sayısı (çerçeve sayısı)

```


5.7.Şablonların Diske Saklanması

Oluşturulan şablonlar, ileride kullanılmak ve kullanıcı programları tarafından rahatça tanınabilmek için sözcüğün kendisiyle adlandırılan file'lara yazılırlar.Örneğin 'BİR' sözcüğü '1. ' isimli file'a yazılır. Böylece bu sözcüğün sayısal karşılığı da program tarafından rahatlıkla değerlendirilebilir. Ayrıca Tanıma programı da her benzerlik hesabında yeni şablonu sözcüğün kendi ismiyle çağırır ki bu, programların hazırlanmasında büyük kolaylıklar sağlamış ve programın anlaşılabilirliğini artırmıştır.

5.8.Dinamik Zaman Normalizasyonu

Büyük kapasiteli sözcük dağarcıklarında Doğrusal Zaman Normalizasyonunun tanıma yüzdesi büyük ölçüde düşebilir. Doğrusal Zaman Normalizasyonu (DZN), şablonları aynı uzunluğa getirir ama hala şablonların ayrıntıları aynı değildir. Doğru bir normalizasyon, sözcüğe ait ilk ve son çerçevelerin duyarlı olarak tespitine bağlıdır. Aksi halde iki sözcük tam olarak çakışmayacak ve sözcüğün özellikleri kayacaktır.

Böyle bir durumda zaman normalizasyon yolunu (Time Normalization Path) otomatik olarak bulacak bir yöntem gerekmektedir. Bu normalizasyon yolu, ilk şablonun hangi çerçevesinin ikinci şablonun hangi çerçevesiyle çakıştığını gösterir. Klasik yöntemlerle bu yolu bulmak için belki milyonlarca olasılığı birbir denemek gerekecektir. Oysa bu problem Dinamik Programlama (DP) ile $N \times M$ işlemde kolayca

çözülebilir. Burada N ve M, şablonların çerçeve sayılarıdır. DP, en iyi çözümü bulmayı garantilemektedir. Bu yöntemin özel bir adı da Dynamic Time Warping (DTW) dir.

ŞEKİL 5.2'de DTW kullanılarak bulunmuş optimum yol görülmektedir.

5.8.1.Dynamic Time Warping

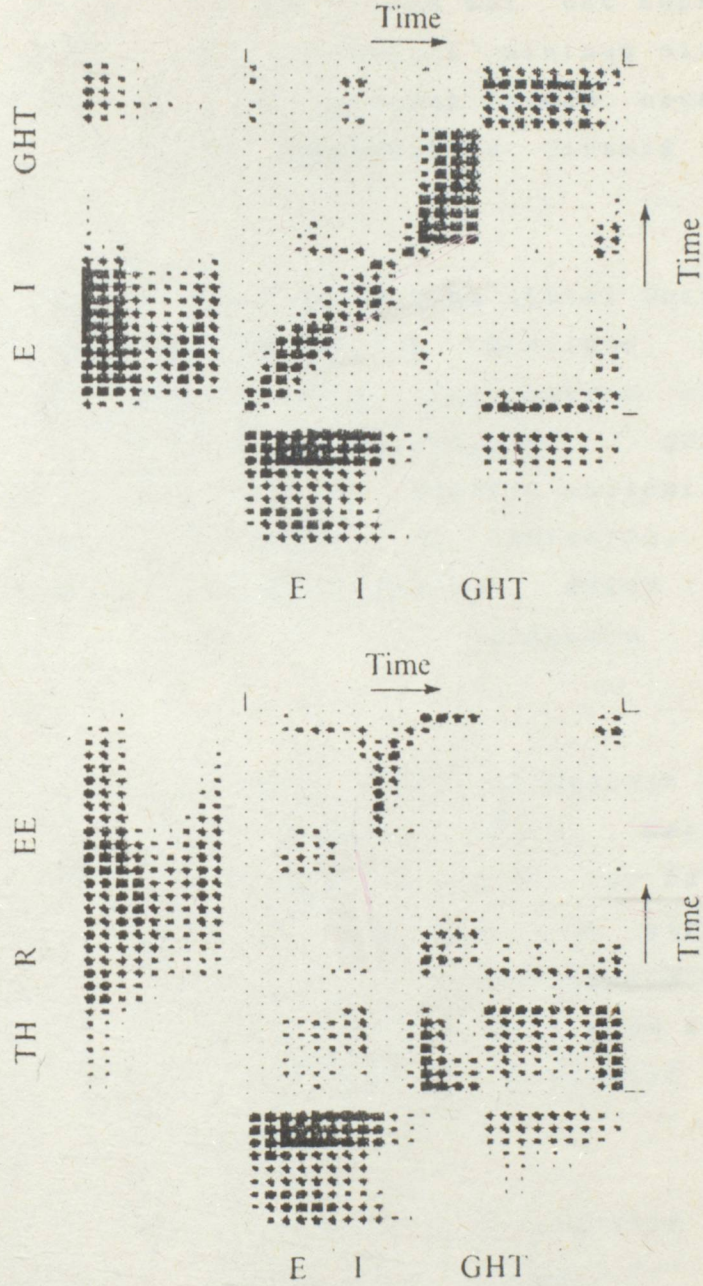
DTW, temelde iki bölümden oluşur: ŞEKİL 5.3, birinci bölümü göstermektedir. Şekilde görüldüğü gibi iki şablondan biri dikey, diğeri yatay olarak yerleştirilir. Şablonların her bir çerçevesinin üç katsayı ile karakterize edildiğini varsayalım. Dikey şablonun dört çerçevesi, yatay şablonun beş çerçevesi olsun. Ortadaki matris Fark Matrisi olarak adlandırılır ve şablonların çerçevelerinin birbirleriyle benzerliklerini ifade eden rakamlardan oluşur.Örneğin sağ üst köşedeki 20, dikey şablonun ilk çerçevesinin, yatay şablonun son çerçevesinden oldukça farklı olduğunu göstermektedir. Benzer şekilde, 2.satır 2.sütundaki 1 de, her iki şablonun 2. çerçevelerinin birbirine çok benzediğini göstermektedir. Bu benzerlikler, her karşılıklı çerçeve çiftinin herbir elemanının fark karelerinin toplamı (Euclidean Distance) olarak hesaplanır:

$$Fark(n,m) = \sum_{i=1}^{10} (R(n,i)-T(m,i))^2, \quad k,m = 1,...,25$$

Fark(25,25) : Fark matrisi

R(25,10) : Referans şablonu

T(25,10) : Test (bilinmeyen) şablonu



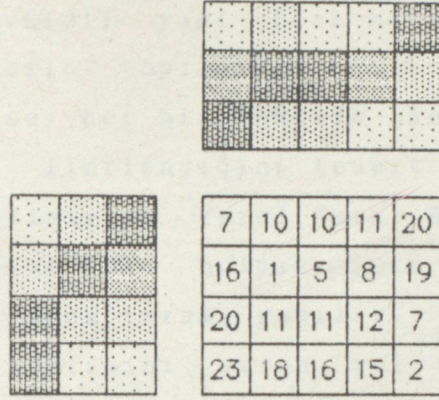
ŞEKİL 5.2 EIGHT sözcüğünün EIGHT ve THREE sözcükleriyle arasındaki Fark Matrisi (Koyu noktalar farkın küçük olduğu bölgeleri gösterir)

İkinci bölümde, Fark Matrisinin sol üst köşesinden sağ alt köşesine giden, fark toplamaları minimum olan optimum yol bulunur. Bu yol, iki şablonun zaman eksenini üzerindeki nonlinear ilişkisini gösterir ve Dinamik Programlamayla hesaplanır.

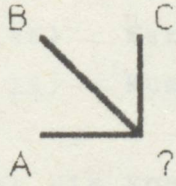
DP ile bir Yerel Karar Fonksiyonu (Local Decision Function) kullanılarak Fark Matrisiyle birlikte, Kümülatif Fark Matrisi adı verilen ikinci bir matris oluşturulur. Bu matrisin nasıl oluşturulduğu ŞEKİL 5.4'de açıklanmıştır. ŞEKİL 5.4-(a), örnekte kullanılan Yerel Karar Fonksiyonunu göstermektedir. Bu fonksiyon, optimum yolun hangi kısmı yollardan (yatay, dikey veya çapraz) ilerleyeceğini gösterir. Bu fonksiyon aşağıdaki gibi uygulanır:

Kümülatif Fark Matrisinin herbir elemanının hesabı için, bu elemana gelinceye kadarki (Yerel Karar Fonksiyonu üzerindeki) en düşük maliyet (cost) ile bu elemanın Fark Matrisindeki karşılığı toplanır. Bu elemana gelinceye kadarki maliyet, Yerel Karar Fonksiyonunun tanımladığı bir önceki değerlerin en küçüğüdür. Bu işlem sol üst köşeden başlayarak peşpeşe sürdürülürse, Kümülatif Fark Matrisinin tüm elemanları hesaplanmış olur.

ŞEKİL 5.4-(b), hesaplanmakta olan Kümülatif Fark Matrisi'ni göstermektedir. Hesaplanacak eleman '?' işaretiyle ve önceki üç eleman da koyu renkle gösterilmiştir. Hesaplanacak elemanın maliyeti Fark Matrisinde görüldüğü gibi 12, bu elemana gelinceye kadarki maliyet ise 19, 13 ve 21 sayılarının en küçüğüdür. Bu durumda Kümülatif Fark Matrisinin '?' işaretli elemanı 25 ($13+12$)'dir.



ŞEKİL 5.3 Fark Matrisinin Oluşturulması



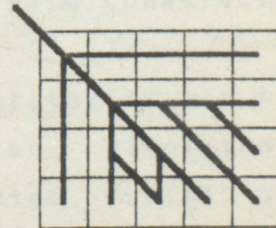
(a)

7	17	27	38	58
23	8	13	21	
43	19	19	?	

(b)

7	17	27	38	58
23	8	13	21	40
43	19	19	25	28
66	37	35	34	27

(c)



(d)

ŞEKİL 5.4 Kümülatif Fark Matrisinin Oluşturulması ve Yerel Karar Ağacı

ŞEKİL 5.4-(c), Kümülatif Fark Matrisinin tamamlanmış halidir. Sağ alt köşedeki koyu renkli rakam bu iki şablonun kümülatif farkıdır yani Fark Matrisindeki en kısa yol boyunca farkların toplamıdır. En kısa yolu (optimal yolu) bulmak için ise her bir noktada hangi yönde (yatay, dikey veya çapraz) ilerlendiğini tespit etmek gerekir. ŞEKİL 5.4-(d)'de kullanılan Yerel Karar Ağacı görülmektedir. En küçük maliyetli yolu bulmak için ters işlemle sağ alt köşeden başlayarak tarama yapılır. Fark Matrisine bakacak olursak 27 toplamını veren yol $7+1+5+12+2$ 'dir. Diğer yolların kümülatif toplamı 27'den daha büyüktür.

Bu örnekte DP için kullanılan rekürsif ifade;

$$D(i,j) = d(i,j) + \min(D(i-1,j), D(i-1,j-1), D(i,j-1)), \quad 1 \leq i \leq I, \quad 1 \leq j \leq J$$

I, J : Şablonların çerçeve sayıları (25)

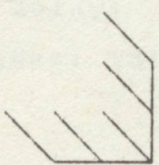
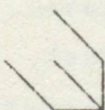
$d(25,25)$: Fark Matrisi

$D(25,25)$: Kümülatif Fark Matrisi

Burada ilk koşul olarak $D(1,1)=0$ alınmıştır ve iki şablon arasındaki kümülatif fark toplamı yani toplam fark $D(I,J)$ 'de oluşur.

5.8.2.Yerel Yol Sınırlaması (Local Path Constraint)

Yukarıda açıklanan örnek, DTW algoritmalarının belki de en basitidir. Bu örnekte kullanılan algoritmaya 'Temel Simetrik Algoritma' denir. Bu yöntem, ismini Yerel Karar Fonksiyonunun şeklinden alır. ŞEKİL 5.5'de çeşitli Yerel

p	Sekilsel ifade	Simetrik/asimetrik	DP-ifadesi $D(i,j)=$
0		Simetrik	min $D(i,j-1)+d(i,j)$ $D(i-1,j-1)+2d(i,j)$ $D(i-1,j)+d(i,j)$
		Asimetrik	min $D(i,j-1)+d(i,j)$ $D(i-1,j-1)+d(i,j)$ $D(i-1,j)+d(i,j)$
1/2		Simetrik	min $D(i-1,j-3)+2d(i,j-2)+d(i,j-1)+d(i,j)$ $D(i-1,j-2)+2d(i,j-1)+d(i,j)$ $D(i-1,j-1)+2d(i,j)$ $D(i-2,j-1)+2d(i-1,j)+d(i,j)$ $D(i-3,j-1)+2d(i-2,j)+d(i-1,j)+d(i,j)$
		Asimetrik	min $D(i-1,j-3)+(d(i,j-2)+d(i,j-1)+d(i,j))/3$ $D(i-1,j-2)+(d(i,j-1)+d(i,j))/2$ $D(i-1,j-1)+d(i,j)$ $D(i-2,j-1)+d(i-1,j)+d(i,j)$ $D(i-3,j-1)+d(i-2,j)+d(i-1,j)+d(i,j)$
1		Simetrik	min $D(i-1,j-2)+2d(i,j-1)+d(i,j)$ $D(i-1,j-1)+2d(i,j)$ $D(i-2,j-1)+2d(i-1,j)+d(i,j)$
		Asimetrik	min $D(i-1,j-2)+(d(i,j-1)+d(i,j))/2$ $D(i-1,j-1)+d(i,j)$ $D(i-2,j-1)+d(i-1,j)+d(i,j)$
2		Simetrik	min $D(i-2,j-3)+2d(i-1,j-2)+2d(i,j-1)+d(i,j)$ $D(i-1,j-1)+2d(i,j)$ $D(i-3,j-2)+2d(i-2,j-1)+2d(i-1,j)+d(i,j)$
		Asimetrik	min $D(i-2,j-3)+2(d(i-1,j-2)+2d(i,j-1)+d(i,j))/3$ $D(i-1,j-1)+d(i,j)$ $D(i-3,j-2)+d(i-2,j-1)+2d(i-1,j)+d(i,j)$

$D(i,j)$: Kumulatif Fark Matrisi
 $d(i,j)$: Fark Matrisi

ŞEKİL 5.5 Simetrik ve Asimetrik DP Algoritmaları
 (Yerel Karar Fonksiyonları)

Karar Fonksiyonları görülmektedir. Temelde Yerel Karar Fonksiyonu, optimum yolun geçebileceği olası yolların şeklini tanımlar. Örneğin Temel Simetrik Fonksiyon kullanıldığında istenmeyen yatay veya düşey uzamalar meydana gelir. Bu durumda Karar Fonksiyonuna extra dallar eklemek mümkündür. Örneğin ŞEKİL 5.5'deki $P=1$ 'de olduğu gibi yatay veya düşey bir hareketten önce mutlaka çapraz bir yol gidilmesi şart koşulabilir. Bu durumda optimum yolun eğimi Yerel Karar Fonksiyonunun şekliyle belirlenir. Bundan dolayı bu özellik Yerel Yol Sınırlaması veya Eğim Sınırlaması (Slope Constraint) olarak bilinir.

5.8.3.Dal Ağırlığı

Yerel Karar Fonksiyonundaki dalların ağırlıkları değiştirilerek yolun akışı yönlendirilebilir. Örneğin, Temel Simetrik Fonksiyonda yolun doğal eğimi daha çok çapraz dallar üzerindedir. Çünkü bu dallarda fark, doğal olarak azdır. Bu, istenmeyen bir durum olabilir. Çapraz üzerindeki dalları 2 ile çarparak eğimin bu yolda olması önlenabilir. Genelde her dal ayrı ayrı ağırlıklandırılabilir.

5.8.4.Optimum Yolun Uzunluk Normalizasyonu

Optimal yol üzerindeki kümülatif fark, yolu oluşturan farkların (dalların) toplam sayısına bağlıdır. Bundan dolayı uzun sözcüklerin fark toplamları kısa sözcüklerinkinden daha büyüktür. Bu sorunu ortadan kaldırmak için kümülatif fark normalize edilir. Böylece

optimum yolun ortalama maliyeti bulunur.

Bu konuda çeşitli normalizasyon yöntemleri kullanılabilir; Sakoe ve Chiba kümülatif farkı, $I+J$ 'ye bölmeyi önermişlerdir. Burada I ve J şablonların uzunluklarıdır (25).

Diğer bir yöntem de kümülatif farkı gerçek yol uzunluğuna bölmektir. Bunu yapabilmek için optimum yolun kümülatif fark toplamını veren dalların sayısını işlem sırasında belirlemek gerekir. Buda haliyle sistemin hızını düşürecektir.

5.8.5.Fark Hesabı (Distance Measure)

Tüm DTW işlemi, iki çerçeve veya vektör arasındaki farkı hesaplayacak uygun yöntemin seçimine bağlıdır. Bu farklar, çeşitli fonksiyonlar ile hesaplanabilir. Bunlardan en çok kullanılan Euclidean Distance'dır. Yani farkların karelerinin toplamıdır. LPC analizinde ise genellikle Itakura'nın Log Likelihood Distance Ratio formülü kullanılır.

5.8.6.DTW Kullanarak Ayırık Sözcük Tanıma

ŞEKİL 5.6, Optimal Nonlinear Zaman Normalizasyonunu kullanarak yapılan Ayırık Sözcük Tanıma bir örnektir. Bu örnekte üç referans şablonu dikey olarak yerleştirilmiştir. 'ONE', 'TWO', 'THREE'. Yatay şablon ise tanınacak sözcüğe

aıttır. (Bu örnekte 'ONE'). Bilinmeyen sözcük yukarıda anlatılan tekniklerle karşılaştırılmış ve üç nonlineer zaman yolu şekilde gösterilmiştir. Aynı zamanda bilinmeyene ait şablon ile diğer üç şablon arasındaki kümülatif farklar da şekilde görülmektedir. En iyi uyuşumda fark, en küçük olacağından bilinmeyen sözcüğün 'ONE' olduğu kolaylıkla tanınmaktadır.

Bu örnekte kullanılan algoritmada, yol normalizasyonu yapılmadan basit simetrik fonksiyon kullanılmıştır. Şekilden de görülebileceği gibi eğer şablon tam olarak birbirleriyle çakışıyorsa yani birbirinin benzeriyse optimal yol hemen hemen bir doğrudur. Aksi takdirde bu yol doğrusal olmayacaktır. Bunun sebebi, iki farklı sözcüğün zaman eksenini üzerinde ciddi bozulmalarla birbirine benzer hale getirilebilmesidir. Aslında bu, Eğim Sınırlamasının bazan tanıma performansını neden artırdığının ifadesidir. Yani farklı iki sözcükten birinin başı diğerinin sonuna benzeyebilir. Bu durumda yol çapraz gitmeyecektir. Bunu önlemek için Eğim Sınırlaması yapılır.

Pratikte, her bir sözcük için birden fazla referans oluşturulur. Bunun sebebi, söyleniş tarzındaki farklılıklara rağmen sözcüğün tanınabilmesini kolaylaştırmaktır. Benzer şekilde, öğrenme aşamasında aynı sözcüğün farklı söylenişlerinin ortalaması alınarak daha uygun bir şablon da oluşturulabilir.

DTW tekniklerini kullanarak yapılan Ayrık Sözcük Tanımada, tanıma yüzdesi Lineer Zaman Normalizasyonu kullanarak yapılan tanımadan daha yüksektir. Böyle bir çalışmada 10 rakam için beklenen duyarlılık %99'dan fazladır.



6.SİSTEMİN ÇALIŞTIRILMASI

6.1.Giriş

Ayrık Sözcük Tanıma Sisteminin tümü, iki bağımsız programdan oluşmaktadır:

- a) öğren.c: Sözcük öğrenme programı.
- b) tanı.c: Sözcük tanıma programı.

6.2.ÖĞRENME: Sözcük Tanıtma

Sisteme yeni bir sözcük tanıtmak istenirse:

A>öğren

yazmak yeterlidir. Bu durumda ekrana:

Seslendirilecek kelimeyi verin

mesajı çıkacaktır. Burada verilecek sözcük, aynı zamanda oluşturulacak şablonun saklanacağı file'in ismi olacaktır.

bilgisayar

yazıldığı zaman 'bilgisayar' sözcüğünün seslendirileceği belirtilmiş olur. Hemen ardından

Seslendirmeden önce ENTER'a basın

mesajı alınır. Bu mesaj çıktığında, program ENTER tuşuna basılmasını bekleyecektir. ENTER'a basıldıktan sonra, sözcüğün mikrofon aracılığı ile bilgisayara aktarılması için 2 sn. süre verilmiştir. Bu 2 sn. sonunda bilgisayar gerekli analizi yapar ve oluşan şablonu 'bilgisayar. ' adlı file'a yazar. Eğer ses girişi yapılırken giriş seviyesi (ses) düşükse 'Ses anlaşılamadı' mesajı alınır. Eğer ses seviyesi yüksekse 'Ses enerjisi çok yüksek' mesajı alınır. Böyle durumlarda konuşmacıya tekrar hakkı tanınmıştır. Ayrıca haberleşme sırasında arabirimle ilgili bir sorun çıktığında bu durum da kullanıcıya ilgili mesaj ile iletilir.

6.3.TANIMA: Mikrofondan Girilen Sözcüğün Tanınması

Öğretilen sözcükler veya uygulama programında kullanılacak sözcükler 'sözlük. ' adlı file'a yerleştirilir. Tanı.c

programı, tanıyacağı sözcüğün şablonu ile bu file'da yazılı olan sözcüklerin şablonlarını karşılaştırır.

Herhangi bir uygulama programında ses yardımıyla giriş yapılabilmesi için

sestani(sözcük);

fonksiyonu çağrılmalıdır. Bu fonksiyonun argümanı boş gönderilir ve dönüşte tanınan sözcüğü taşır.

Bu fonksiyon çağrıldığında

>> (sözcük listesi) <<

Seslendirmeden önce ENTER'a basın

mesajı ekrana gelir.

Sözcük listesi, fonksiyonun her çağrılışında ekrana gelecektir. Bunun amacı, kullanıcıya hangi sözcüklerin bilgisayar tarafından tanındığını hatırlatmaktır. Kullanıcı ENTER'a bastıktan sonra 2 sn. içinde sözcüğü söylemelidir. Bu durumda 'tani.c' programı sesi dış ortamdan alır, şablonunu çıkardıktan sonra diğer şablonlarla karşılaştırır ve tanınan sözcüğü uygulama programına geri gönderir.

st.c programı, bir örnek uygulama programıdır. Bu program 0-9 arası rakamları ve 'çık' sözcüğünü tanımaktadır. Eğer söylenen sözcük 0-9 arasında değilse yani 'çık' ise programın çalışması sona erer.

6.4.Uygulama Programının Ayrık Sözcük Tanıma Sistemiyle Kullanılması İçin Yapılması Gereken İşlemler

Oluşturulan sistem, 'ses.lib' arşivinde bütünleştirilmiştir. sestani(sözcük); fonksiyonunun C dili tarafından tanınabilmesi için derledikten sonra

Link (program adı),,,ses

komutuyla bağ kurulmalıdır.

```

/*****          ST.C
*
*  Bu program bir uygulama programıdır.
*
*****/
main()
{
    char kelime[10];
A:    sestani(kelime);
    if(kelime[0]=='c') exit(0);
    printf("\n ==> %c \n",kelime[0]);
    goto A;
}

```


SONUÇ

Bu projede ana hedef, konuşmacıya bağımlı bir ayrık sözcük tanıma sistemi tasarlamak ve geliştirmektir. Bu hedefe erişmek için son zamanlarda en çok kullanılan Doğrusal Tahminli Kodlama (LPC) Metodu kullanıldı. Projenin her aşamasında, kendi bölümünde en iyi ve standart hale gelmiş yöntemler kullanıldı. Sistemin düşük hızlı bir bilgisayarda kurulması, performansı ters yönde etkilemiştir. Ancak, bu seçimi yapmak, eldeki imkanlardan dolayı (MICROLINK'in arabirim kartının sadece bu bilgisayara uyumlu olmasından dolayı) zorunluydu.

Sistem, her ne kadar konuşmacıya bağımlı olarak tasarlanmışsa da kolayca konuşmacıdan bağımsız hale getirilebilir. Değişik özellikteki konuşmacılardan (yaşlı, genç, kadın, erkek, ...) örnekler alıp her sözcük için birden fazla şablon oluşturulursa sistem, konuşmacıdan bağımsız olarak çalışabilir. Bu şablonlar 'sözlük' file'ına farklı ama ortak özellikleri olan isimlerle yazılmalıdır. Örneğin 'İKİ' sözcüğü için, konuşmacıdan bağımsız çalışan bir sistemde dört ayrı konuşmacıdan alınan örneklerle dört şablon halinde tanımlanacaksa 'İKİ1', 'İKİ2', 'İKİ3', 'İKİ4' isimleri file adı olarak kullanılabilir. Bu durumda sistem bunları farklı sözcükler gibi görecektir. Eğer konuşmacı 'İKİ' derse, bu sözcük 'İKİ3' olarak tanınabilir. Ancak bu durum kullanıcı programı tarafından son karakteri atılarak 'İKİ' olarak düzeltililebilir. Doğal olarak konuşmacıdan bağımsız bir sistemin çalışması daha yavaş olacaktır.

Bu projede 2 sn. içinde söylenen bir sözcük ortalama 25 sn. içinde tanınmaktadır. Proje, 11 sözcük için denenmiştir. Bu

sözcük dağarcığı için hiçbir yanlış tanıma olayıyla karşılaşılmamıştır. Sistem, bu haliyle sesleri birbirine benzeyen 2-3 erkek konuşmacı için denendiğinde de iyi sonuç vermiştir.


```
*****          SPEECH.H
*
*   Ayrik Sozcuk Tanima Sisteminde Kullanilan parametreler.
*
*****/

#define min(x,y) (((x) > (y)) ? (y) : (x))
#define abs(x) (((x) < 0) ? -(x) : (x))
#define sqr(x) (x)*(x)
#define PAUSE while(!kbhit());getch()
#define ZAMAN 20000          /* Ses kayit = 2 sn. 10000 sample/sec */
#define FRAME 200           /* EPD'de cerceve uzunlugu */
#define LIMIT ZAMAN/FRAME   /* EPD'de cerceve sayisi */
#define P 11                /* LPC'de P sabiti */
#define M 25                /* LTW'de normalize edilecek uzunluk */
#define N 400               /* LTW'de cerceve uzunlugu */
```



```

/*****          EPD.C

```

```

*
* Bu program dis ortamdan alinan sayisal ses bilgisi icinde sozcugun
* her hecesinin basini ve sonunu bulur.
*
* dbyte[] : Ham ses bilgisi
* boy      : Dizinin boyu
* endps    : Uc noktalarin bulunduğu çerçevelerin numaralari
*           Çift indisler hece baslari, tek indisler hece sonlaridir
*
*****/

```

```

#include "speech.h"

```

```

int energy[LIMIT],n[10];
int me=0;

```

```

epd(dbyte,boy,endps,adet)
int *dbyte,boy,*endps,*adet;
{
    int e;
    register int i,j,nn=0;
    me=0;
    for(i=0;i<boy;i+=FRAME,nn++)
    {
        e=0;
        for(j=i;j<(i+FRAME);j++)
            e+=abs(dbyte[j]); /* Çerçeve enerjilerini hesapla */
        if(e<0)
        { puts(" Ses enerjisi çok yüksek");
          return(0);
        }
        energy[nn]=e;
        me=(e>me) ? e : me; /* Max. enerjiyi bul */
    }
    *adet=end_pnt_det(endps);
    if(*adet>10 || *adet<2)
    {
        puts(" Sozcuk anlasilamadi");
        return(0);
    }
    return(1);
}

```

```

end_pnt_det(n)
int *n;
{
    int e=0,itl,itu,meane;
    register int i,m=0,k=0;
    for(i=(LIMIT-20);i<LIMIT;i++) e+=energy[i];
    meane=e/20;
    itl=0.03*(me-meane)+meane;
    itl=min(itl,4*meane);
    itu=5*itl;
    BAS:   while(m<=LIMIT)

```



```

        {
            if(energy[m]>itl)
            {
                n[k]=m;
                k++;
                while((energy[m]>itl) && (energy[m]<itu)) m++;
                if(energy[m]>itu) goto SON;
                k--;
            }
            else m++;
        }
SON:    while(m<=LIMIT)
        {
            if(energy[m]<itu)
            {
                while((energy[m]>itl) && (energy[m]<itu)) m++;
                if(energy[m]<itu)
                {
                    n[k]=m+1;
                    k++;
                    goto TEKRAR;
                }
            }
            else m++;
        }
TEKRAR: if(m>LIMIT) return(k);
        goto BAS;
    )

```



```
/* ***** AUTO.C
*
* Bu program &x adresinden baslayip k tane (integer) verinin
* oziliski katsayilarini (autocorrelation coef.) hesaplar.
* Donus olarak r[] dizisini gonderir.
*
* *****/

#include "speech.h"

autoc(x,k,r)
int *x,k;
float *r;
{
    long int t;
    float r1;
    register int i,m;
    for(i=0;i<P;++i)
    {
        t=0;
        for(m=0;m<(k-i);++m) t+=(long)x[m]*(long)x[m+i];
        r[i]=(float)t;
    }
    r1=r[0];
    for(i=0;i<P;++i) r[i]=r[i]/r1;
}
```


/*****

LPC.C

*

* Bu program Durbin's Recursive Algorithm ile r[] dizisinde
 * gonderilen oziliski katsayilarini kullanarak Dogrusal Tahmin
 * Katsayilarini (Linear Prediction Coefficients - LPC) hesaplar.
 * Donus parametresi LPC katsayilarini iceren lpc[] dizisinin
 * baslangic adresidir.

*

*****/

#include "speech.h"

lpc(r,lpc)

float *r;

int *lpc;

{

float k,t,err=r[0];

float a[P],pa[P];

register int i,j;

for(i=1;i<P;i++)

{

t=0;

for(j=1;j<=(i-1);j++)

t+=pa[j]*r[i-j];

k=(r[i]-t)/err;

a[i]=k;

for(j=1;j<=(i-1);j++)

a[j]=pa[j]-k*pa[i-j];

err=(1-k*k)*err;

for(j=1;j<=i;j++) pa[j]=a[j];

}

lpc[0]=(int)(100*err+.5);

for(i=1;i<P;i++) lpc[i]=(int)(100*pa[i]+.5);

}


```

*****          OGREN.C
*
* Bu program disaridan okudugu dbyte[] dizisini analiz ettikten sonra
* hesaplanan parametreleri o sozcugun adini tasiyan file'a saklar.
*
*****/

#include "speech.h"
#include <stdio.h>
extern double cos();
int dbyte[ZAMAN];
int ddbyte[N];
float hw[N];

main()
{
    FILE *fp;
    int basson[10],adet,bas,son;
    register int i,j,k,l;
    static int lp[40][P];
    int lwlp[M][P];
    float x[P];
    char s[10];
    puts(" Seslendirilecek kelimeyi verin");
    gets(s);
    fp=fopen(s,"w");
A1:  puts(" Seslendirmeden once ENTER'a basin");
    PAUSE;
    if(input(dbyte,ZAMAN)==0) goto A1; /* Sesi al */
    hamm(hw); /* Hamming Window carpanlarini hesapla */
    if(epd(dbyte,ZAMAN,basson,&adet)==0) goto A1; /* Uc noktalar ? */
    bas=basson[0]*FRAME;
    son=(basson[adet-1]+1)*FRAME;
    if((son-bas)/200>40) /* Sozcuk uzunlugu 40 cerceveden fazla mi */
    {
        puts(" Sozcuk uzun !");
        goto A1;
    }
    for(i=bas,k=0;i<=(son-N);i+=FRAME,k++)
    {
        for(l=0;l<N;l++) ddbyte[l]=(int)((float)dbyte[i+l]*hw[l]);
        autoc(ddbyte,N,x); /* Cercevenin oziliski kats. hesapla */
        lpc(x,&lp[k][0]); /* Cercevenin LPC kats. hesapla */
        autoc(&lp[k][1],P-1,x); /* LPC kats.'nin oziliski kats. */
        for(j=0;j<P;j++)
            lp[k][j+1]=(int)(100*x[j]+.5); /* 100'e normalize */
    }
    lwarp(lp,lwlp,k); /* Dogrusal Zaman Normalizasyonu */
    for(i=0;i<M;i++)
    {
        for(j=1;j<P;j++)
            fprintf(fp," %4d",lwlp[i][j]);
        fprintf(fp,"\n");
    }
    fclose(fp);
}

```



```

}

hamm(hamm)
float *hamm;
{
    float x,y=2*3.1415926;
    register int n;
    for(n=0;n<N;n++)
    {
        x=y*n;
        hamm[n]=0.54-0.46*cos(x/N);
    }
}

/* Dogrusal Zaman Normalizasyonu :
   r[] : Normalize edilmemis matris.
   bm   : Normalize edilmemis matrisin vektor sayisi
   rs[] : Normalize edildikten sonra olusan matris */

lwarp(r,rs,bm)
int r[][P],rs[][P],bm;
{
    int m;
    register int i,ms;
    float oran,k,s;
    oran=(float)(bm-1)/(float)(M-1);
    for(ms=0;ms<M;ms++)
    {
        k=(float)ms*oran;
        m=(int)k;
        s=k-m;
        for(i=1;i<P;i++)
            rs[ms][i]=(int)((1.0-s)*(float)r[m][i] + s*(float)r[m+1][i])
    }
}

```



```

/*****          TANI.C
*
* Bu program disaridan aldigi ham ses bilgisini analiz ederek olusan
* sablonu disk'teki diger sablonlarla karsilastirir ve karsilastirma
* sonucu en kucuk olan sablonun ait oldugu sozcugu, aranan sozcuk olarak
* belirler.
*
*****/

```

```

#include "speech.h"
#include <stdio.h>
extern double cos();
extern double sqrt();
int dbyte[ZAMAN];
int ddbyte[N];
float hw[N];
int sozcuk;
char s[20][10];

```

```

sestani(kelime)

```

```

char *kelime;

```

```

{
    FILE *fp;
    int basson[10],adet,bas,son;
    register int i,j,k,l,ll;
    char ss[10];
    static int lp[40][P];
    int lwlp[M][P-1];
    float x[P];
    fp=fopen("sozluk","r");
    fscanf(fp,"%d",&sozcuk);
    printf("\n >>");
    for(ll=0;ll<sozcuk;ll++)
    {
        fscanf(fp,"%s",ss);
        strcpy(s[ll],ss);
        printf(" %s",s[ll]);
    }
    printf(" <<\n");
    fclose(fp);
R1:  puts(" Seslendirmeden once ENTER'a basin");
    PAUSE;
    if(input(dbyte,ZAMAN)==0) goto R1;
    hamm(hw);
    if(epd(dbyte,ZAMAN,basson,&adet)==0) goto R1;
    bas=basson[0]*FRAME;
    son=(basson[adet-1]+1)*FRAME;
    if((son-bas)/200>40)
    {
        puts(" Sozcuk uzun !");
        goto R1;
    }
    for(i=bas,k=0;i<=(son-N);i+=FRAME,k++)
    {
        for(l=0;l<N;l++) ddbyte[l]=(int)((float)dbyte[i+l]*hw[l]);
    }
}

```



```

        autoc(ddbyte,N,x);
        lpc(x,&lp[kl[0]]);
        autoc(&lp[kl[1]],P-1,x);
        for(j=0;j<P;j++)
            lp[kl[j+1]]=(int)(100*x[j]+.5);
    }
    lwirp(lp,lwlp,k);
    dtw(lwlp,kelime);
}

hamm(hamm)
float *hamm;
{
    float x,y=2*3.1415926;
    register int n;
    for(n=0;n<N;n++)
    {
        x=y*n;
        hamm[n]=0.54-0.46*cos(x/N);
    }
}

lwirp(r,rs,bm)
int r[][P],rs[][P-1],bm;
{
    int m;
    register i,ms;
    float oran,k,s;
    oran=(float)(bm-1)/(float)(M-1);
    for(ms=0;ms<M;ms++)
    {
        k=(float)ms*oran;
        m=(int)k;
        s=k-m;
        for(i=1;i<P;i++)
            rs[ms][i-1]=(int)((1.0-s)*(float)r[m][i] + s*(float)r[m+1][i]);
    }
}

/* t[] ve r[] vektorlerinin Euclidean Distance hesabi */
tt : Hesaplanan fark

e_dist(t,r,tt)
int *t;
int *r;
int *tt;
{
    unsigned int tot=0;
    register int i;
    for(i=0;i<(P-1);i++)
        tot+=sqr(t[i]-r[i]);
    *tt=(int)sqrt((float)tot);
}

/* Dinamik Zaman Normalizasyonu (Dynamic Time Warping - DTW)

```


x : Taninacak sozcugun sablonu (Fark Matrisi).
 g[] : Kumulatif Fark Matrisi
 kelime : Aranan sozcugun karakter karsiligi

*/

```
dtw(x, kelime)
int x[][P-1];
char *kelime;
{
    FILE *fp;
    int y[M][P-1], l=0, kck=32000, gg;
    register i, j, k;
    static int g[M][M];
    for(k=0; k<sozcuk; k++)
    {
        fp=fopen(s[k], "r");
        for(i=0; i<M; i++)
            for(j=0; j<(P-1); j++)
                fscanf(fp, " %4d", &y[i][j]);
        fclose(fp);
        for(i=0; i<M; i++)
            for(j=0; j<M; j++)
                e_dist(&x[i][0], &y[j][0], &g[i][j]);
        g[0][0]*=2;
        for(i=1; i<M; i++)
            g[0][i]=g[0][i-1];
        for(i=1; i<M; i++)
        {
            g[i][0]=g[i-1][0];
            for(j=1; j<M; j++)
            {
                gg=g[i][j];
                g[i][j]=gg+min3(g[i][j-1], g[i-1][j-1]+gg, g[i-1][j]);
            }
        }
        gg=g[M-1][M-1]/50;
        if(kck>gg)
        {
            kck=gg;
            l=k;
        }
    }
    strcpy(kelime, s[l]);
}

int min3(x, y, z)
int x, y, z;
{
    int k;
    k=min(min(x, y), z);
    return k;
}
```



```

; /*****          LGPIB.ASM

;
_text    segment byte public 'code'

        public  _opengpib
        public  _readcom
        public  _readdat
        public  _writecom
        public  _writedat
        public  _sendcom
        public  _gpibcode
        public  _closegpib
        public  _wr2int

        assume  cs:_text
_wr2int   proc    near
        push    bp
        mov     bp,sp
        mov     bx,[bp+04]
        mov     cx,[bp+06]
        shr     cx,1
LOOP:
        mov     al,[bx]
        xchg    al,[bx+1]
        mov     [bx],al
        add     bx,2
        loop    LOOP
        pop     bp
        ret
_wr2int   endp

_opengpib proc    near
        push    bp
        mov     bp,sp
        push    ds
        mov     dx,cs
        mov     ds,dx
        mov     dx,OFFSET $00F9
        mov     ax,3D02h
        int     21h
        jb     L2A
        mov     CS:$00F5,ax
        mov     dx,OFFSET $0101
        mov     ax,3D02h
        int     21h
        jb     L2A
        pop     ds
        mov     CS:$00F7,ax
        mov     ax,0001h
        pop     bp
        ret
L2A:
        mov     ax,0000h
        pop     ds
        pop     bp

```



```

        ret
_opengpib    endp

_readcom     proc    near
        push    bp
        mov     bp,sp
        mov     bx,CS:$00F5
        jmp     short L42
_readcom     endp

_readdat     proc    near
        push    bp
        mov     bp,sp
        mov     bx,CS:$00F7
L42:        push    ds
        mov     dx,[bp+04]
        mov     cx,[bp+06]
        mov     ax,4402h
        int     21h
        jnb     L53
        mov     ax,0000h
L53:        pop     ds
        pop     bp
        ret
_readdat     endp

_writecom     proc    near
        push    bp
        mov     bp,sp
        mov     bx,cs:$00F5
        jmp     short L68
_writecom     endp

_writedat     proc    near
        push    bp
        mov     bp,sp
        mov     bx,cs:$00F7
L68:        push    ds
        mov     dx,[bp+04]
        mov     cx,[bp+06]
        mov     ax,4403h
        int     21h
        jnb     L79
        mov     ax,0000h
L79:        pop     ds
        pop     bp
        ret
_writedat     endp

_sendcom     proc    near
        push    bp
        mov     bp,sp
        mov     bx,cs:$00F5
        mov     al,[bp+04]

```



```

        push    ds
        mov     dx,cs
        mov     ds,dx
        mov     cs:$0109,al
        mov     dx,OFFSET $0109
        mov     cx,0001
        mov     ax,4403h
        int     21h
        jnb     LA0
        mov     ax,0000
LA0:     pop     ds
        pop     bp
        ret
_sendcom      endp

_gpibcode     proc      near
        push    bp
        mov     bp,sp
        push    ds
        push    cs
        pop     ds
        mov     dx,OFFSET $010a
        mov     cx,0002
        mov     bx,cs:$00F5
        mov     ax,4402h
        int     21h
        mov     al,cs:$010B
        xor     ah,ah
        pop     ds
        pop     bp
        ret
_gpibcode     endp

_closegpib    proc      near
        push    bp
        mov     bp,sp
        mov     bx,cs:$00F5
        mov     ax,3E00h
        int     21h
        jb     LF0
        mov     bx,cs:$00F7
        mov     ax,3E00h
        int     21h
        jb     LF0
        mov     ax,0001h
        mov     cs:$00F5,0abcdh
        mov     cs:$00F7,0bcdeh
        pop     bp
        ret
LF0:     mov     ax,0000h
        pop     bp
        ret
_closegpib    endp

$00F5      DW      0abcdh

```



```
$00F7 DW 0bcdeh
$00F9 DB 'GPIBCOM',0
$0101 DB 'GPIBDAT',0
$0109 DB 0
$010A DB 0
$010B DB 0
*_text ends
end
```



```

/*****      GPIBC.H

```

```

*
*   This header file defines some useful constants for programs
*   in Lattice C which use the GPIB Device Drivers, and the
*   DFLCx interface modules.
*

```

```

*****/

```

```

/* GPIBCOM Local Command Codes */

```

```

#define SIFC 0x80 /* Send InterFace Clear signal */
#define RENT 0x81 /* Set REN line True */
#define RENF 0x82 /* Set REN line False */
#define EXPP 0x83 /* Execute a Parallel Poll */
#define TCSET 0x84 /* Set new Terminating Character */
#define TCEN 0x85 /* Enable T.C. detection on input */
#define TCDIS 0x86 /* Disable T.C. detection */
#define TOSET 0x87 /* Set TimeOut period */
#define TOEN 0x88 /* Enable timeouts during transfers */
#define TODIS 0x89 /* Disable timeouts */
#define EOIEN 0x8A /* Send EOI with last byte output */
#define EOIDIS 0x8B /* Disable sending of EOI */
#define SRQWT 0x8C /* Wait for SRQ signal to arrive */
#define ATNF 0x8D /* Set ATN line False */

```

```

/* Some of the IEEE-488 Bus Command Codes */

```

```

#define UNT 0x5F /* Untalk */
#define UNL 0x3F /* Unlisten */
#define GET 0x08 /* Group Execute Trigger */
#define SDC 0x04 /* Selective Device Clear */
#define DCL 0x14 /* Device Clear */

```

```

/* Addressing Base Values */

```

```

#define TAD 0x40 /* Talk Address */
#define LAD 0x20 /* Listen Address */
#define SAD 0x60 /* Secondary Address */

```


KAYNAKLAR

- A Speaker Independent Isolated Turkish Word Recognition System
Master Thesis
A.F.Ünal
- A Speaker-Independent Speech-Recognition System Based on Linear Prediction
IEEE Vol.ASSP.26, No 1, Feb. 1978
V.N.Gupta, J.K.Bryan, J.N.Cowdy
- An Adaptive, Ordered, Graph Search Technique for Dynamic Time Warping for Isolated Word Recognition
IEEE Vol.ASSP.30, No 4, August 1982
M.K.Brown, L.R.Rabiner
- An Improved Endpoint Detector for Isolated Word Recognition
IEEE Vol.ASSP.29, No 4, August 1981
L.R.Rabiner, F.Lamel
- Analysis of Speech Signals of Short Pitch Period by a Sample-Selective Linear Prediction
IEEE Vol.ASSP.35, No 9, September 1987
Y.Miyoshi, K.Yamato, R.Mizoguchi, M.Yanagida, O.Kakusho
- Automatic Speech Recognition: A Critical Survey and Discussion of The Literature
British Post Office Research Report, 1968
S.R.Hyde

- C Made Easy
McGraw Hill
H.Schildt

- Digital Signal Processing of Speech Signals
L.R.Rabiner, R.W.Schafer

- Distance Measures for Speech Processing
IEEE Vol.ASSP.24, No 5, October 1976
A.H.Gray, J.D.markel

- Dynamic Programming Algorithm Optimization for Spoken
Word Recognition
IEEE Vol.ASSP.26, No 1, February 1978
H.Sakoe, S.Chiba

- Interactive Clustering Techniques for Selecting Speaker
-Independent Reference Templates for Isolated Word
Recognition
IEEE Vol.ASSP.27, No 2, April 1979
S.E.Levinson, L.R.Rabiner, A.E.Rosenberg, J.G.Wilpon

- LPC Distance Measures and Statistical Tests with
Particular Reference to the Likelihood Ratio
IEEE Vol.ASSP.30, No 2, April 1982
P.Souza, P.J.Thomson

- MICROLINK Hardware User Manual

- Minimum Prediction Residual Principle Applied to
Speech Recognition
IEEE Vol.ASSP.23, No 1, February 1975
F.Itakura

- MS GW-BASIC Interpreter Under MS-DOS (User Guide)
Olivetti

- On Creating Reference Templates for Speaker
Independent Recognition of Isolated Words
IEEE Vol.ASSP.26, No 1, February 1978
L.R.Rabiner

- Pitch Detection by Data Reduction
IEEE Vol.ASSP.23, No 1, February 1975
N.J.Miller

- Prosodically Guided Syllable Based Speaker Independent
Isolated Turkish Word Recognizer
Master Thesis in E.E. Boğaziçi University, 1986
C.Ersoy

- Speaker-Independent Isolated Word Recognition for a
Moderate Size (54 Word) Vocabulary
IEEE Vol.ASSP.27, No 6, December 1979
L.R.Rabiner, J.G.Wilpon

- Speaker-Independent Recognition of Isolated Words Using
Clustering Techniques
IEEE Vol.ASSP.27, No 4, Aug. 1979
L.R.Rabiner, S.E.Levinson, A.E.Rosenberg, J.G.Wilpon

- Spectral Linear Prediction: Properties and Applications
IEEE Vol.ASSP.23, No 3, June 1976
J.Makhoul

- Speech Recognition: An Idea Whose Time Is Coming
G.M.White
- Speech Recognition Experiments with Linear Prediction,
Bandpass Filtering, and Dynamic Programming
IEEE Vol.ASSP.24, No 2, April 1976
G.M.White, R.B.Neely
- Speech Syntesis and Recognition
VNR (UK) Co.Ltd., 1988
J.N.Holmes
- Systems For Isolated and Connected Words Recognition
NATO ASI: New Systems and Architectures For Automatic
Speech Recognition and Syntesis,Bonas,France, 1984
R.K.Moore
- Türk Dil Bilgisi
Prof.Dr. M.Ergin

ÖZGEÇMİŞ

A.Levend ABAY, 1965 yılında İstanbul'da doğdu.Liseyi Kadıköy Suadiye Lisesi'nde bitirdi.1986'da Yıldız Üniversitesi Mühendislik Fakültesi Bilgisayar Bilimleri Mühendisliği Bölümünden mezun oldu. Ocak-1987'den buyana mezun olduğu bölümün Yazılım Anabilim Dalında Araştırma Görevlisi olarak çalışmaktadır.



