YILDIZ TEKNİK ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

İNSAN SESİNİN SESLİLER İÇİN PARAMETRİK ANALİZİ

Elek. Müh. M.Serdar GÜLER

F.B.E. Elektronik ve Hab. Mühendisligi Anabilim Dalı
Elektronik Programında hazırlanan

YÜKSEK LİSANS TEZİ

Tez Danışmanı : Prof. Şefik SARIKAYALAR

İSTANBUL, 1993

1. TEZİN ÖZETİ

İnsanlar arası iletişimin esasını birbirleriyle ***konuşmaları*** tutar.Belki de bu yüzden son yıllardaki en önemli ilgi odaklarından birisi de insan konuşmasının analizi ve sentezidir. Elektronik, bilgisayar ve matematikteki son gelişmeri kullanarak konuşanı ve konuşulanı anlayan makineler yapmak insanlığın yeni hedeflerinin başında gelmektedir. Konuşulanı, yani komutları, anlayan cihazların yapılmasıyla insanın cihazlara uzaktan kumanda yeteneğinin ne kadar gelişece i ortadadır. Ayrıca özürlü insanların hareketlerinde kazanacakları özgürlük ile başkalarına muhtaç olmadan yaşama şansına kavuşacakları düşüncesi tüm bu çalışmaların manevi dinamosudur. Konuşanı ayırdeden cihazlar da belki tüm güvenlik teknolojisini yeni baştan belirleyecektir.

Bu nedenledir ki bu tezin de ilgi konusu insan sesinin ve daha özelde ise Türkçe'deki sesli harflerin parametrik analizini içermektedir. İnsan sesi denilmesine ra men inceleme daha özelde erkek sesinin analizinde yo unlaşmıştır. Çünkü kadın ve çocuk seslerinde harmoniklerin sayısı erkek sesindekinin iki katı kadardır ve ses rezonans frekansları da oldukça yüksek bir bölgeye kaymıştır. Sesin, insanın fizyolojik ses üretme aygıtında oluşumunun açıklanmasıyla başlayan tezin içeri inde, sırasıyla insan ses üretme aygıtının matematiksel modellenmesi, bu modellemeye dayalı genel teori- lerin kısaca açıklanması, bu teorilerin en gelişkini ve şu anda en çok ilgi uyandıranı olan ***Do rusal Öngörüye Dayalı Kodlama*** 'nın açıklanması, bu kodlamanın matematiksel dayanakları ile ispatı, bu teoriye uygun bir elektronik devrenin tasarlanarak bir bilgisayar yardımı ile yapılan analiz sonuçlarının açıklanması yer almaktadır.

Bu tezin gerçekleştirilmesinde bir PC-AT bilgisayardan faydalanılmıştır. Tasarlanan devrede kullanılan elemanlar da Türkiye piyasasından kolayca temin edilebilecek şekilde seçilmiştir.

Bu tezin daha sonraki çalışmalara bir kaynak teşkil etmesi için bizim kullanmadığımız bir takım ses analizi ile ilgili teoriler de zaman zaman açıklanmıştır. İspatlardaki matematiksel formüller ve dayanaklarına da imkan oldukça deteylı değinilmiştir.

2. *SUMMARY*

Talking is the base of communication between the humans. So, recently one of the most important studying subjects is analyzing and synthesizing the human voice. Making intelligent machines, which understand who is talking and what is said, by using last technologies on the electronic,computers, and mathmetical sciences and is the new goal of scientists. It is very clear that making intelligent machines which understand what is said, will increase the facility of remote control of a man. On the other hand injured men can live without help of the others. This thought is also a very important moral power for the scientists. And the machines which understand who is talking, of course will change all of the security technology against the theves.

Because of these reasons, the subject of this thesis also the human voice and specially voice of the Türkish wovels. Although we said the human voice, the main subject is the voice of the men. Because the harmonics of the women and child are twice of the men's and resonance frequencies of them take place on a very high region. This thesis begins with how does produced the voice in the human voice system, and goes respectively modelling of human voice system, general teories about this model, linear predictive coding which is the most new and most studying teory, mathematical basis of this teori, producing an electronic circuit which converts speech into digital form for a com- puter, results of the analisis achieved by using a computer.

While analyizing the results of the circuit, we used a PC-AT. All of the selected components easyly can be found in the Turkey market.

It is explaned some speech analysis teories that we didn't use because we hope that this thesis becomes a source for later studies. And we also try to explaine as much as possible the mathematical teories we used.

İÇİNDEKİLER DİZİNİ :

1. GİRİŞ

Ses, hava ortamındaki boyuna titreşimlerden (akustik dalgalardan) meydana gelir. Akustik anlamda ses, hava basıncındaki hızlı ve kararlı dalgalanmalardan ibarettir. Bu basınçlar, insanlarda ki fizyolojik ses üretme aygıtınca üretilir ve çevreye yayılır. İnsanın fizyolojik ses üretme aygıtı, esas olarak, ses bölgesi (ağız bölgesi bşluğu ile yerleşik kütleler ile burun bölgesi boşluğu) ve bu bölgenin akustik filtreleme davranışından ibarettir. Şekil 1.1 'de insan kafatasının X ışınlarıyla çekilmiş bir görüntüsü vardır. Bu görüntü ses aygıtını ve onun temel bileşenlerini (çene, dil, küçük dil, gırtlak ve ciğerler) açıkça göstermektedir. Nefes borusu düzgün, şekli değişebilen, 17 cm uzunluğunda akustik bir tüptür. Bir ucu ses telleriyle diğer ucu ise dudaklarla sonlandırılmıştır. Nefes borusunun geçiş bölgesi kesit açıklığı dudaklar, çene, dil, ve küçük dilin yerleşiminee belirlenir .ve 0 ile 20 cm^2 arasında değişir. Seslilerin üretilmesinde önemli rolü olan ses tellerinin titreşen kirişleri yaklaşık 18 mm. uzunluğundadır. Yardımcı bir boşluk olan burun bölgesi boşluğu, küçük dilin kapı görevi gören hareketiyle ağız bölgesi boşluğuna bağlanır. Burun bölgesi küçük dilden başlar ve burun deliklerinde biter. İnsanlarda bu boşluk yaklaşık olarak 12 cm. uzunluğunda ve 60 cm^3 hacmindedir. Burunla ilgili olmayan sesler boyunca, küçük dil burun boşluğunu mühürler ve burun deliklerinden dışarıya ses yayılmaz.

Ses sistemindeki üretimine göre sesler esas olarak iki temel gruba ayrılır :

A) Sesliler (a,e,ı,i,o,ö,u,ü) ;

Sesliler için, hava basıncı ciğerlerde yükseltilip, ses tellerine doğru zorlanarak ses tellerinin titreştirilmesi sağlanır. Ses tellerini uyaran bu akış sonucunda peryodik benzeri, yaygın spektrumlu darbeler ortaya çıkar. Ses bölgesi belirli akustik rezonans frekanslarını sağlayacak bir biçimde, ses tellerinden gelen bu peryodik benzeri darbeleri bir çeşit filtrelemeden geçirerek doğru sesliyi üretir. Sesliler için önemli olan bu akustik rezonans frekanslarına ***formant frekansları*** denilir. Seslerin anlaşılır olmasında özellikle ilk üç formant frekansı belirleyicidir. Çünkü işaretin enerjisinin önemli bir kısmı bu ilk üç frekansta toplanmıştır. Daha yüksek dereceden formant frekansları ise sesin kalitesi üzerinde etkilidir. Tablo 1.1 'de sesliler için yaklaşık olarak ilk üç formant frekansı ve Şekil 1.2 'de ise seslilerin ağızdan çıkışı esnasında ağız bölgesinin aldığı şekil görülmektedir. Ancak verilmiş olan bu formant frekansları kesin değerler olmayıp kişiden kişiye ve kullanılan çıkarım yöntemine göre değişebilmektedir. Ses tellerinin ürettiği darbeler arası peryoda ***perde peryodu*** denilir. Perde peryodu işaretin temel frekansını yani perdesini verir.

B) Sessizler (b,c,ç,d,f,g,h,j,k,l,m,n,p,r,s,ş,t,v,y,z) ;

Ses bölgesi sessizlerin üretiminde ses tellerini kullanmaz. Sessizleri de ses bölgesindeki üretilişlerine göre kendi içerisinde dört alt gruba ayırabiliriz :

I) Sürtme sesi çıkaran sesler (frikatif sesler = f,v,s,z) , ses bölgesinin bazı noktalarında boğazlar oluşturulup, çalkantı (türbülans) teşkil edecek bir biçimdeki havanın oluşturulan bu boğazlardan geçmeye zorlanmasıyla meydana gelir.

II) Patlama yapan sesler (p,ç,t,k,b,c,d,g), ses bölgesininin bir noktada tamamen kapatılıp, bu kapamanın ardındaki basıncın arttırılması ve kapamanın birdenbire serbest bırakılmasıyla meydana gelir.

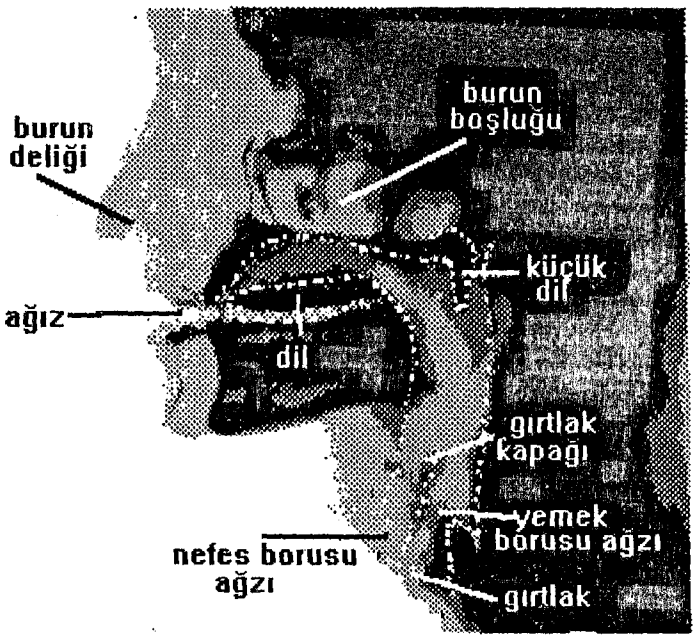
III) Burun boşluğunu kulananlar (m,n), küçük dilin müsaade etmesiyle sesin, ağız bölgesine yardımcı bir boşluk olan burun boşluğundan da yayılmasıyla oluşturulurlar.

IV) Sesli benzeri sessizler (l,r,y), diğer sessizlerden farklı olarak bu sessizler peryodik benzeri bir davranış sergilerler.

2. İNSAN SES ÜRETME SİSTEMİNİN MODELLENMESİ

Yukarıda da anlatıldığı gibi seslilerin uyarım kaynağı darbeli ve peryodiktir. Harmonikleri ise $1/f^2$ oranıyla genliği azalan bir çizgisel spektruma sahiptir. Ses sistemi, darbeli uyarım kaynağına bu formantlara uygun rezonans karakteristiğini yüklemek için zamanla değişen bir filtre gibi davranır ve filtrenin transfer fonksiyonunda sadece kutuplar vardır.

Sessizler nispeten düzgün bir spektruma sahip bir gürültü kaynağıyla uyarılırlar. Sessizler için bu uyarım kaynağına uygulanan transfer fonksiyonu yaklaşık olarak bir çift kutup ve bir sıfırdan ibarettir. Frikatif seslerin çıkarılmasında hem darbeli uyarım kaynağı ve hem de beyaz gürültü kaynağı birlikte kullanılır. Sesli ya da sessiz, çevreye yayılan ses transfer fonksiyonunun kendisine yükleyeceği rezonansları da beraberinde taşıyacaktır. Şekil 2.1 'de görüldüğü gibi ses bölgesini uyarım kaynağı ve filtre olarak ayırmak başlangıç yaklaşımımız olacaktır. Buna göre ağızdan çıkan ses $\{s(t)\}$, uyarım kaynağı $\{g(t)\}$ ile ses bölgesi transfer fonksiyonunun $\{h(t)\}$ bir konvolüsyonu olarak düşünülebilir.

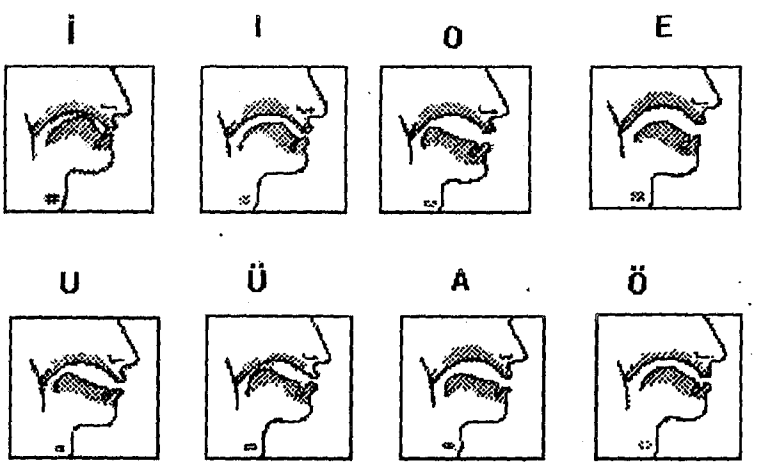


ŞEKİL 1.1 : İnsan ses bölgesinin x ışınları ile çekilmiş görüntüsü.

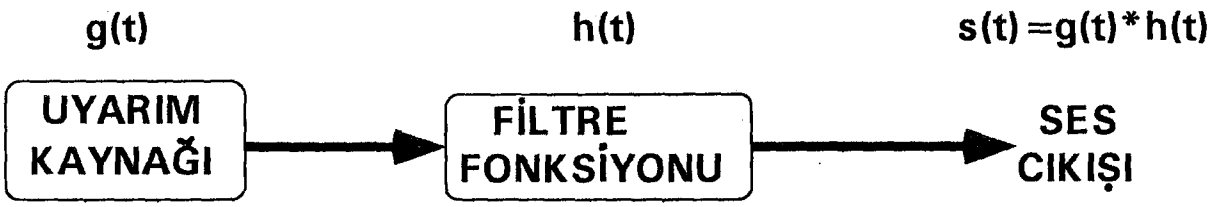
TABLO 1.1 : Sesliler için üç temel formant frekansı

(Hz)

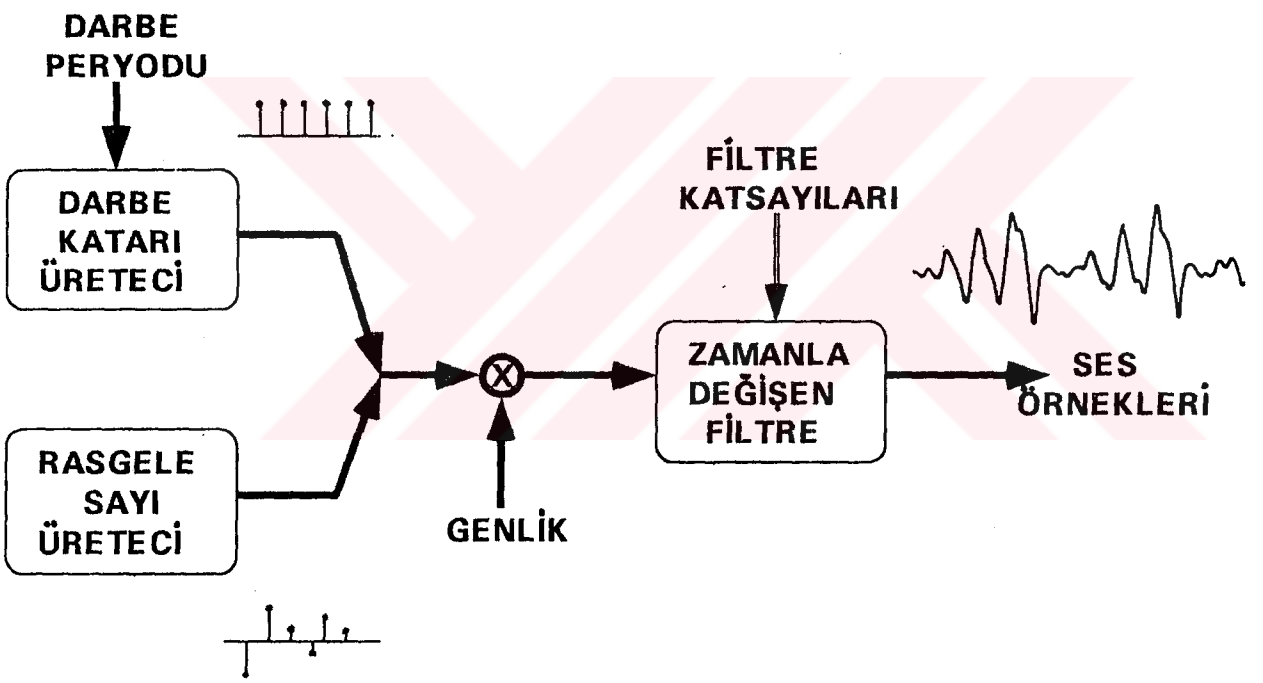
	F_1	F_2	F_3	F_4	F_5
/a/	813	1313	2688	3438	4438
/i/	375	2188	2938	3438	4438
/u/	375	1063	2188	3438	4438
/e/	438	1813	2688	3438	4438
/o/	438	1063	2688	3438	4438



ŞEKİL 1.2 : Seslilerin çıkışında ağzın aldığı şekil.



ŞEKİL 2.1 : Ses bölgesi davranışının basit blok gösterimi



ŞEKİL 2.2 : İnsan ses üretim sisteminin temel yaklaşım modeli

O halde kullanılacak modelde, seslileri üretmek için o seslinin perde peryodunca uyarılan peryodik bir darbe kaynağı ve sessizleri üretmek için de düz bir spektral zarfa sahip, rasgele sayı üreten bir beyaz gürültü kaynağı kullanılır. Frikatif sesleri çıkartmada bu iki kaynak beraberce kullanılır. Ayrıca, uyarım kaynağı işaretine insan ses üretme sisteminin transfer fonksiyonu özelliklerini verebilmek için bir spektral şekilleme filtresi kullanılmalıdır. Çıkış işaretinin zarfı bu filtrenin spektrumunun şeklini alacaktır.

3.2) AYRIK FOURIER ANALİZİ :

Eğer işaret sayısallaştırılırsa, Ayrık Fourier Analizi denilen bir teknikle frekans analizine tabi tutulur. Ses işareti tam anlamıyla peryodik değildir. Ancak bir peryottan diğerine çok fazla değişikliğe de uğramaz. İşaret bir dikdörtgen pencereden geçirilerek ardışıl N değeri alınır. Pencerenin dışındaki tüm değerler sıfırdır. Bu pencerelemede pencerelemenin kenarlarında spektrumda süreksizlikler oluşur. Bu nedenle dikdörtgen pencerenin yerine artan kosinüs ya da Hamming penceresi kullanılır. Analizde ihtiyaç duyulan her bir frekanstaki enerjidir. Bu, güç spektrumunca verilir.

İnsan kulağı gelen işaretin fazına karşı duyarsızdır. Bu yüzden faz spektrumuna ihtiyaç duyulmaz.

N noktalı bir Ayrık Fourier Analizi için yaklaşık N^2 işleme ihtiyaç duyulur. Bu yüzden işlem açısından daha etkili bir yöntem olan Hızlı Fourier Analizi yöntemi geliştirilmiştir. Bu yöntemde sadece N log N işleme ihtiyaç duyulur.

3.3) FİLTRE DİZİSİ :

Konuşma bir dizi band geçiren filtreden geçirilir ve her bir filtre çıkışındaki enerji miktarı ölçülür. Çıkışlardaki, zamanın bir fonksiyonu olan enerji sayısallaştırılarak bilgisayarda depolanır. Temel frekans öngörümü yapılarak gelen ses işaretinin sesli ya da sessiz olduğu saptanır. Yeterli bir analiz için 10 ile 20 arasında band geçiren filtreye ihtiyaç duyulur.

3.4) TEMEL FREKANSIN (PERDE PERYODUNUN) ÇIKARIMI :

Seslilerde temel frekansın (perde peryodunun) bulunması, otokorelasyon fonksiyonu kullanılarak yapılabilir. Bu fonksiyon bir işareti kendisinin geciktirilmiş haliyle çarpmakla elde edilir. İşaretin durağan olmayan doğasından dolayı, toplama sınırı N noktayla sınırlandırılmalıdır. Eğer işaret pencerelenirse buna ihtiyaç duyulmayabilir. Peryodik bir fonksiyon için, otokorelasyon fonsiyonu, peryodikliğe ve onun çarpanlarına karşılık gelen bir dizi tepeler ortaya çıkarır. Bir konuşma işareti için tepeler temel frekans ve onun harmoniklerindedir.

Otokorelasyon fonksiyonu, bir sesli işaretin peryodunun tahmini için kullanılabildiği gibi işaretin sesli mi yoksa sessiz mi olduğunun karar verilmesi için de kullanılabilir. Peryodik bir dalga şekli otokorelasyon fonksiyonunda tepeler verirken, rasgele bir dalga şekli ise düz bir otokorelasyon fonksiyonu sergiler.

Seslinin perde peryodunun çıkarımı için frekans boyutunda olduğu gibi zaman boyutunda da bir takım algoritmalar vardır. Sıfır geişlerinin algılanması için yapılan algoritmalar bunların başında gelir. Gürültü gibi etkiler nedeniyle, sıfır geçişi yerine bu tür algoritmalarda belli bir eşik seviyesinin geçilmesinin kontrolü de bazen tercih edilir.

Sürekli bir konuşmada ses bölgesinin şeklinin değişmesi nedeniyle filtre katsayıları (transfer fonksiyonu parametreleri) kayar. Fakat dil, çene, ve dudaklar ses bölgesinde önemli kütleler teşkil ettiğinden, bu kütlelerin kasların zorlamasıyla çok hızlı bir değişim göstermeleri beklenemez. Hızlanmadaki bu fiziksel kısıtlamalar nedeniyle parametrelerde meydana gelecek olan kaymalarında oldukça yavaş olması beklenebilir.

Tersine bir düşünüşle bir ses işareti verildiğinde işaretin spektral zarfı, kısa süreli bir ters filtrelemeden geçirilirek elde edilebilir. Ters filtrenin parametreleri ise, işaretin düz bir spektral zarfa sahip olması sağlanacak bir şekilde ayarlanmalıdır. İşte bu düşünce analizimizin temelini oluşturacaktır.

ŞEKİL 2.3 : Ses analizi için ters filtre kullanılması

3. SES ANALİZİNDE KULLANILAN YÖNTEMLER

3.1) KISA SÜRELİ FREKANS ANALİZİ :

Konuşma işaretinin doğasındaki değişimler nedeniyle, zaman boyutunda çalışmanın çok fazla yardımı olamaz. Frekans boyutunda ise zaman değişkeninden bağımsız çalışırız. Bu yöntemde bir dalganın frekans içeriğinin analizi Fourier analiziyle yapılır. Bu teorem der ki; herhangi bir peryodik işaret , sınırlı sayıdaki harmoniğinin seri açılımlarının birbiriyle toplanmasıyla elde edilebilir. Burada, işaretin önceki pencerelenmiş (belli bir aralık içerisinde tanımlı) değerlerinin toplamının yer aldığı spektrum üzerinde çalışılır. Eğer pencere transfer fonksiyonu bir alçak geçiren filtrenin cevabıysa, karşımıza Fourier dönüşümü yapılabilecek bir işaret elde edilir.

Bu yöntemle işaret analizi ses spektrogramı ya da sonograflarda kullanılır.

3.5) CEPSTRAL İŞLEME :

Bu teknik sesi, uyarım işareti ile filtre kısmına ayırmak için kullanılır.Bu yöntem hem uyarım frekansının ve hem de formant frekansının öngörülmesini daha basitleştirir.

Ses işaretinin, uyarım kaynağı ile ses bölgesi transfer fonksiyonunun konvolüsyonu olarak düşünülebileceğini belirtmiştik. Önce konuşma işaretinin ayrık fourier analizi hesaplanır. Frekansın hesaplanacak logaritmik spektrumuna cepstrum denilir. Cepstrum, temel frekansa karşılık gelen büyük bir tepe içerir.

3.6) FORMANT İZLEME ALGORİTMALARI :

Bu algoritmalar sayısallaştırılan işaretin bilgisayara kaydedilmesinden sonra uygulanır. Ancak algoritmaların çalışmasında karşılaştığı dört temel zorluk vardır :

i) İki formant yan yana geldiğinde spektrumda tek bir tepe oluşturacak bir biçimde birleşirler. En düşük frekanslardan yukarıya doğru tepelei sayan ve bunları formantlara atayan bir algoritma bu tek tepeyi tek bir formant olarak algılayabilir. Böylece formantlara yanlış ad verilir.

ii) Bazı sessizlerde, özelliklede burunsal olanlarda, spektrumda formant rezonansları (kutuplar) kadar rezonans dışılıklar (sıfırlar) meydana gelir. Bunlar bazen frekans olarak formantlarla çakışabilirler ve bu noktalardaki enerjiyi azaltırlar. Bu da tepenin maskelenmesine neden olur.

iii) Birçok seslinin, özellikle de açık seslilerin, spektrumunda ses kaynağının ses bölgesi filtrelemesinden daha etkili olduğu alçak frekans bölgesinde zaman zaman bir tepe görülür.

iv) Formantların yoğunluğu arttıkça, özellikle sesizler boyunca, bazı tepeler diğerlerine göre daha önemsizleşir. Bu nedenle bir tepe toplama algoritması başarısız olabilir.

4. DOĞRUSAL ÖNGÖRÜ MODELLEMESİ VE ANALİZİ

Doğrusal öngörü modellemesi ve buna dayalı analiz, ses analizinde çalışanlarca son yıllarda en çok ilgi gösterilen konulardan birisidir. Doğrusal öngörüye dayalı analize Atal (1970) ve Itakura öncülük etmiş, daha sonra Markel (1971) ve Makhoul tarafından geliştirilmiştir. Her geçen gün daha da geliştirilmektedir. Doğrusal öngörü modellemesinin temeli akustik tüp modeline dayanır. Bu tüpün girişi darbeler ya da düşük seviyeli gürültüdür ve tüp matematiksel olarak kolayca işlenebilir olma özelliğine sahiptir, yani, tüpün önemli parametreleri çıkışlarından bulunabilir. Görüldüğü gibi bu model bizim şu ana kadar açıklamaya çalıştığımız insan ses üretim modelinin aynısıdır.

Doğrusal öngörüye dayalı analize şöyle bir başlangıç yapmak yerinde olur :

Bir s(n) işareti bir dizi bilinmeyen u(n) girişi ile sistemin bir dizi çıkışından ibaret olarak düşünülebilir.

$$(b_0 = 1) \quad \text{için} \qquad s(n) = -\sum_{k=1}^{p} a_k s(n-k) + G\sum_{l=0}^{q} b_l u(n-l)$$

Burada G kazancı temsil eder. Bu eşitlik der ki, s(n) çıkışı geçmiş çıkışlar ile o anki ve daha önceki girişlerin doğrusal bir fonksiyonudur. Bunu z dönüşümüyle frekans boyutunda gösterirsek ;

$$H(z) = \frac{S(z)}{U(z)} = G\frac{1 + \sum_{l=1}^{q} b_l z^{-l}}{1 - \sum_{k=1}^{p} a_k z^{-k}} \qquad \text{elde ederiz.}$$

Bu H(z) 'e genel kutup-sıfır model 'i denilir. Bu modelin ilgilenilen iki özel durumu vardır. :

a) Tüm sıfır modeli : $a_k = 0 \quad , \quad 1 \le k \le p$

b) Tüm kutup modeli : $b_l = 0 \quad , \quad 1 \le l \le q.$

Tüm sıfır modeli istatiksel sözlükte hareketli ortalama (Moving Avarage : MA) olarak, ve tüm kutup modeli ise otomatik gerileyen ortalama (autoregressive ; AR) olarak bilinir. Bizim ilgileneceğimiz model tüm kutup modeli olacaktır. O halde ;

$$\left.\begin{array}{l} s(n): Örneklenmiş.veri \\ \hat{s}(n): Öngörülmüş.veri \\ e(n): Öngörü.Hatası.... \\ a_i: Öngörücü.Katsayıları \end{array}\right\} \text{iken ;}$$

Her n örnek indisi için, s(n) örneği kendinden önceki M adet örnekten doğrusal bir kombinasyonla ***öngörülebilir***. Bu yüzden bu modellemeye ***Do rusal Öngörü Modellemesi*** adı verilmiştir. s(n) örneklenmiş bilgi ile $\hat{s}(n)$ öngörü bilgisi arasında bir e(n) hatası olacaktır.

$$s(n) = \hat{s}(n) + e(n) \qquad (1)$$

$$\hat{s}(n) = \sum_{i=1}^{M} -a_i\, s(n-i) \qquad (i = 1,2,...,M) \qquad (2)$$

İşaretin seçimi tamamıyla keyfidir. Farklı bir değişken $b_i = -a_i$ olarak kolayca tanımlanabilir. (2) eşitliğinin frekans domenindeki gösterimi ise dönüşümü alınırsa ;

$$\hat{S}(Z) = \underbrace{-\sum_{i=1}^{M} a_i Z^{i}}_{F(Z)} . S(Z) = F(Z) . S(Z) \qquad \text{olacaktır.} \qquad (3)$$

Burada F(Z) doğrusal öngörücü filtreyi tanımlamaktadır. (1) eşitliğinden ;

$$E(Z) = S(Z) - \hat{S}(Z) = S(Z) - F(Z) . S(Z) = S(Z) . \underbrace{[1 - F(z)]}_{A(Z)}$$

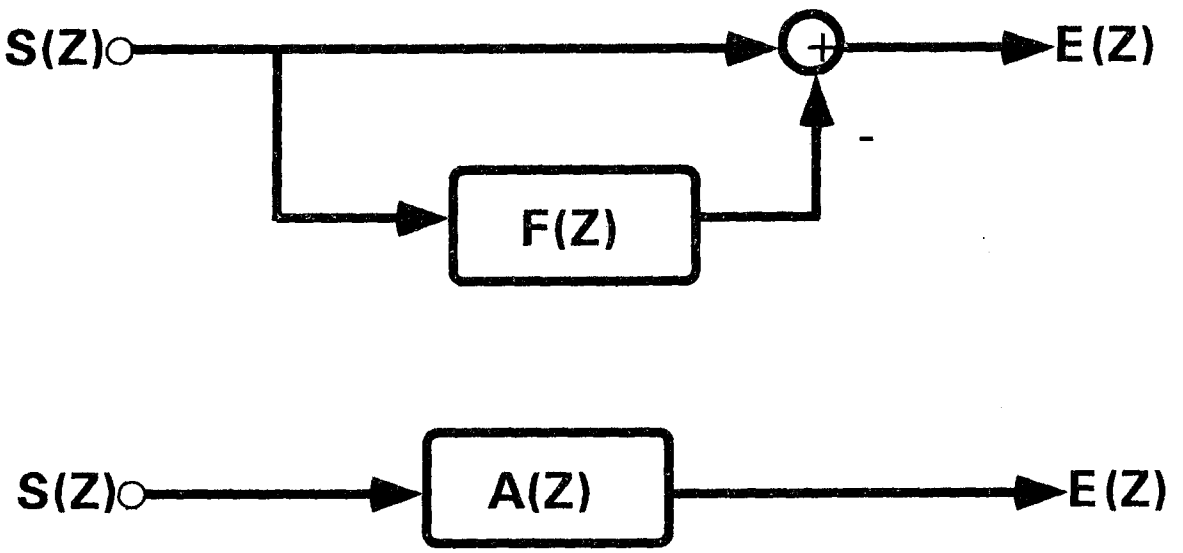
$$E(Z) = S(Z) . A(Z) \qquad (4)$$

O halde

$$A(Z) = 1 - F(Z) \qquad \text{ve}$$

$$A(Z) = 1 + \sum_{i=1}^{M} a_i Z^{-i} \qquad (5)$$

A(Z) ters filtre fonksiyonudur.



ŞEKİL 4.1 : Doğrusal Öngörü Modeli'nin Ters Filtre Yaklaşımı

A(Z) 'in parametreleri, doğrudan doğruya konuşmadan ve (1) eşitliğine ***En Küçük Kareler Optimizasyon*** kriterinin uygulanmasıyla elde edilebilir. Optimizasyon kriteri, hata örneklerinin belirli sayısının karelerinin toplamının katsayılara bağlı olarak en aza indirgenmesidir. e(n) hatayı tanımladığından, a_i katsayılarını e(n) 'i en aza indirgeyecek bir biçimde seçmek tatmin edicidir.

Bu optimizasyon kriterinin seçiminin başlıca sebepleri, sonuç eşitliklerinin doğrusal, kolay işlenir olması ve bu kriterin ses analizinde mükemmel sonuç vermesidir.

Analiz iki çizgide yürüyecektir. Önce s(n) işaretinin bilinen (deterministik) bir işaret olduğunu farzedeceğiz, ve sonra s(n) 'i raslantısal bir işlemden bir örnek farzederek analog türetimler vereceğiz.

4.1) BİLİNEN İŞARET :

Toplam karesel hata , E, şöyle tanımlanır :

$$E=\sum_{n=n_0}^{n_1} e^2(n) \qquad \text{ve} \qquad (1) \text{ ve } (2) \text{ eşitliklerinden ;}$$

$$e(n) = s(n) - \hat{s}(n) = s(n) - \left[- \sum_{i=1}^{M} a_i s(n-i) \right] = \sum_{i=0}^{M} a_i s(n-i) \quad \text{olduğundan ,}$$

$$E = \sum_{n=n_0}^{n_1} \left(\sum_{i=0}^{M} a_i s(n-i) \right)^2 \qquad \text{bulunur.} \qquad (6)$$

istenilirse (6) eşitliğinden ;

$$E = \sum_{n=n_0}^{n_1}\left[\sum_{i=0}^{M} a_i s(n-i)\right]^2 = \sum_{n=n_0}^{n_1}\sum_{i=0}^{M}\sum_{j=0}^{M} a_i s(n-i)a_j s(n-j)$$ yazılabilir

Burada n_0 ve n_1 hatanın minimizasyonu üzerinde geçerli indis limitleridir. Ve şu an için önemsizdir. Toplam karesel hata quadratik (ikinci dereceden) yapıdadır. Yani çoğunlukla katsayılar ikinin katları şeklindedir. Toplam karesel hatanın minimizasyonu katsayılara bağlı kısmi türevin sıfıra eşitlenip çözümüyle elde edilir.

$$\frac{\partial E}{\partial a_i} = 0 \quad , \qquad 1 \le i \le p \qquad (7)$$

Bu yüzden (6) ve (7) eşitliğinden ;

$$\sum_{k=1}^{p} a_k \sum_{n} s(n-k)s(n-i) = -\sum_{n} s(n)s(n-k) \quad , \quad 1 \le i \le p \qquad (8)$$

(8) eşitliklerine en küçük kareler terminolojisinde ***Normal Eşitlikleri*** denir. s(n) şaretinin tanımı için, p bilinmeyenli p denklemli bir set biçimindeki (8) eşitliklerinin (6) eşitliğindeki E 'yi minimize edecek bir biçimde a_i ($1 \le i \le p$) öngörücü katsayıları için çözülmesi gerekir.

E_p 'ye minimum toplam karesel hata denilirse, bu hata (6) 'nın kullanılması ve (8) 'de yerine kullanılmasıyla çıkarılabilir :

$$E_p = \sum_{n} s(n)^2 + \sum_{k=1}^{p} a_k \sum_{n} s(n)s(n-k) \qquad (9)$$

Şimdi (6), (8) ve (9) eşitlikleindeki n toplam aralığı üzerinde duralım. Geçerli iki farklı aralık kabulü için iki ayrı öngörü parametresi çözümleme yöntemi vardır :

a) Otokorelasyon Yöntemi (sonsuz toplam aralığı $-\infty < n < \infty$ için),
b) Koveryans Yöntemi. (sonlu toplam aralığı için $0 < n < N-1$ için),

4.1.a) OTOKORELASYON YÖNTEMİYLE ÇÖZÜM :

Bu yöntemde (6) eşitliğindeki hatanın $-\infty\langle n\langle\infty$ sınırsız aralığında minimizasyonundan elde edileceği düşünülür. Bu durumda (8) ve (9) eşitlikleri şu hale indirgenebilir :

$$\sum_{k=1}^{p} a_k R(i-k) = -R(i) \quad , \quad (1 \le i \le p) \qquad (10)$$

$$E_p = R(0) + \sum_{k=1}^{p} a_k R(k) \qquad (11)$$

Burada

$$R(i) = \sum_{n=-\infty}^{\infty} s(n)s(n+i) \qquad (12)$$

Bu s(n) işaretinin ***otokorelasyon fonksiyonu'***dur. Buradaki R(i) 'nin çift fonksiyon olduğuna dikkat edilmelidir. Yani ;

$$R(i) = R(-i) \qquad (13)$$

R(i-k) katsayıları ***otokorelasyon matrisi*** biçiminde olduğundan , bu yönteme ***otokorelasyon yöntemi*** denilir. Otokorelasyon matrisi simetrik bir ***Toeplitz Matrisi*** 'dir. Toeplitz matrisi her bir diagonaldaki tüm elemanları eşit olan bir matristir.

Pratikte s(n) işareti sadece sınırlı bir aralıkta bilinir, ya da sadece sınırlı bir aralıktaki işaretle ilgileniriz. En yaygın yöntemlerden biri s(n) işaretini bir w(n) pencere fonksiyonu ile çarpmaktır. Böylece belli bir aralığın , $0 \le n \le N-1$, dışında sıfır olan bir başka $s'(n)$ işareti elde edilir :

$$s'(n) = \begin{cases} s(n)w(n), \ldots\ldots 0 \le n \le N-1 \\ \\ 0, \ldots\ldots\ldots\ldots\ldots i \ge 0 \end{cases} \qquad (14)$$

Otokorelasyon fonksiyonu bu durumda şöyle verilebilir :

$$R(i) = \sum_{n=0}^{N-1-i} s'(n)s'(n+i) \quad , \quad i \ge 0 \qquad (15)$$

Pencere fonksiyonunun biçimi önemsizdir.

4.1.b) KOVERYANS YÖNTEMİYLE ÇÖZÜM :

Bu yöntemde E hatasının, sonlu bir aralıkta $0\langle n\langle N-1$ minimize edilebileceği düşünülür. Sonuçta (8) ve (9) eşitlikleri şu halde getirilir :

$$\sum_{k=1}^{p} a_k \varphi_{ki} = -\varphi_{0i} \quad , \quad 1 \le i \le p \qquad (16)$$

$$E_p = \varphi_{00} + \sum_{k=1}^{p} a_k \varphi_{0k} \qquad (17)$$

burada

$$\varphi_{ik} = \sum_{n=0}^{N-1} s(n-i)s(n-k) \qquad (18)$$

Bu s(n) işaretinin verilen aralıktaki koveryansıdır. (16) 'daki φ_{ki} katsayıları bir koveryans matrisi biçimindedir., ve bu yüzden bu yönteme koveryans yöntemi denilir. (18) eşitliğinden kolayca görülebilir ki φ_{ki} koveryans matrisi simetriktir , yani $\varphi_{ki} = \varphi_{ik}$. Bununla beraber otokorelasyon matrisinden farklı olarak her bir diagonaldeki elemanlar birbirinden farklıdır.

4.2) RASLANTISAL İŞARET :

Eğer s(n) işareti raslantısal bir işlemin bir örneği olarak farzedilirse, (6) eşitliğindeki e(n) hatası da raslantısal işlemin bir örneğidir. En küçük kareler yönteminde hatanın karesinin beklendik değerini minimize ederiz. Böylece ;

$$E = \varepsilon(e^2(n)) = \varepsilon\left(s(n) + \sum_{k=1}^{p} a_k s(n-k) \right)^2 \qquad (19)$$

(7) eşitliğini (19) eşitliğine uygulayarak, normal eşitliklerini elde ederiz :

$$\sum_{k=1}^{p} a_k \varepsilon\big(s(n-k)s(n-i)\big) = -\varepsilon\big(s(n)s(n-k)\big) \quad , \quad 1 \le i \le p \qquad (20)$$

Bu durumda minimum ortalama hata şöyle verilir ;

$$E_p = \varepsilon\big(s^2(n)\big) + \sum_{k=1}^{p} a_k \varepsilon\big(s(n)s(n-k)\big) \qquad (21)$$

s(n) işaretinin durağan olması ya da olmamasına göre (20) ve (21) eşitliklerinin beklendiği alınır.

4.2.a) DURAĞAN DURUM :

s(n) durağan işlemi için ;

$$\varepsilon\{s(n-k)s(n-i)\} = R(i-k) \qquad (22)$$

burada R(i) işlemin otokorelasyonudur. (21) ve (22) eşitlikleri, sırasıyla (10) ve (11) eşitliklerine benzer eşitliklere indirgenebilir. Tek fark otokorelasyonun bilinen işaret yerine durağan raslantısal bir işaret olmasıdır. Durağan bir işlem için otokorelasyon bir zaman ortalaması olarak hesaplanabilir.

4.2.b) DURAĞAN OLMAYAN DURUM :

Durağan olmayan bir işlem için ;

$$\varepsilon\{s(n-k)s(n-i)\} = R(n-k,n-i) \qquad (23)$$

Burada $R(t,t')$, t ve t' süreleri arasındaki durağan olmayan otokorelasyondur. R(n-k,n-i), n zaman indisinin bir fonksiyonudur. Biz a_k parametrelerinin öngörülmesinde n = 0 durumuyla ilgileneceğiz . Bu durumda (20) ve (21) eşitlikleri şöyle indirgenebilir ;

$$\sum_{k=1}^{p} a_k R(-k,-i) = R(0,-i) \qquad (24)$$

$$E'_p = R(0,0) + \sum_{k=1}^{p} a_k R(0,k) \qquad (25)$$

s(n) işaretinden durağan olmayan otokorelasyon katsayılarının öngörülmesinde, durağan olmayan işlemin ergotik olmadığına ve genel ortalama yerine bir zaman ortalaması kullanılamayacağına dikkat edilmelidir. Bununla birlikte yöresel olarak durağan işlem denilen belli bir gurup durağan olmayan işlem için, otokorelasyon fonksiyonunu zamanda bir noktaya bağlı olarak kısa süreli bir ortalama olarak öngörebiliriz. Konuşma durağan olmayan işleminin yöresel olarak durağan gibi düşünülmesi bunun bir örneğidir.

4.3) ÖNGÖRÜCÜ PARAMETRELERİNİN HESAPLANMASI

4.3.1) DOĞRUDAN HESAPLAMA YÖNTEMİ :

Her iki yöntemde de öngörücü katasayıları a_k , $1 \le k \le p$, p bilinmeyenli p eşitliğin çözümünden elde edilir. Bu eşitlikler otokorelasyon (durağan) yöntemi için (10) eşitliği ve koveryans (durağan olmayan) yöntemi için (16) eşitliğidir. Gerekli eşitlik çözümlemesi için bir kaç standart yöntem vardır (Gauss indirgemesi veya eliminasyonu, ve Crout indirgemesi yöntemi gibi). Bu genel yöntemler $p^3/3+O(p^2)$ işleme (çarpma ya da bölme) ve p^2 depolama alanına ihtiyaç duyar. Bununla beraber (10) ve (16) eşitliklerine dikkat edilirse matris katsayılarının koveryans matrisi olduğu görülür. Koveryans matrisleri simetriktir ve genelde pozitif yarı-tanımlıdırlar. Bu yüzden (10) ve (16) eşitlikleri karekök ya da Cholesky ayrıştırma yöntemiyle etkin bir biçimde çözülebilir. Bu yöntem genel yöntemin yarısı kadar işleme $,p^3/6+O(p^2),$ ve yarısı kadar $,p^2/2,$ depolama alanına ihtiyaç duyar.

Depolama alanında ve işlem süresinde daha fazla indirgeme sağlanması (10) otokorelasyon normal eşitliklerinin çözümünde bunların özel durumlarından faydalanmakla olabilir. (10) eşitliği matris biçiminde şöyle genişletilebilir :

$$\begin{bmatrix} R_0 & R_1 & R_2 & \dots & R_{p-1} \\ R_1 & R_0 & R_1 & \dots & R_{p-2} \\ R_2 & R_1 & R_0 & \dots & R_{p-3} \\ \cdot & \cdot & \cdot & \dots & \cdot \\ R_{p-1} & R_{p-2} & R_{p-3} & \dots & R_0 \end{bmatrix} \begin{bmatrix} a_1 \\ a_2 \\ a_3 \\ \cdot \\ a_p \end{bmatrix} = - \begin{bmatrix} R_1 \\ R_2 \\ R_3 \\ \cdot \\ R_p \end{bmatrix} \qquad (26)$$

p*p otokorelasyon matrisinin simetrik olduğu ve her hangi bir diagonaldeki elemanların da benzer olduğu buradan görülmektedir. Bu çeşit bir eşitliği çözmek için ***Levinson*** üstün bir tekrarlamalı (rekürsif) işlem çıkarmıştır. Bu işlem daha sonra ***Robinson*** tarafından formül haline getirilmiştir. Levinson 'un yöntemi (26) matrisinin sağ tarafındaki sütun vektörünü genel bir sütun vektörü gibi farzeder. Bu kolon vektörünün otokorelasyon matrisinde bulunan elemanları içerdiği gerçeğinin kullanılmasıyla, ***Durbin*** , Levison yönteminden iki kat daha hızlı bir yöntem yayınlamıştır. Yöntem sadece 2p depolama alanına ve $p^2+O(p)$ işleme ihtiyaç duyar.

Durbin 'in tekrarlamalı işlemi şöyle açıklanabilir :

$$E_0 = R(0) \qquad (27a)$$

$$k_i = -\left[R(i) + \sum_{j=1}^{i-1} a_j^{(i-1)} R(i-j) \right] \Big/ E_{i-1} \qquad (27b)$$

$$a_i^{(i)} = k_i$$

$$a_j^{(i)} = a_j^{(i-1)} + k_i a_{i-j}^{(i-1)} \quad , \quad 1 \le j \le i-1 \qquad (27c)$$

$$E_i = (1 - k_i^2) E_{i-1} \qquad (27d)$$

(27b) ve (27d) eşitlikleri i = 1,2,...,p için tekrarlamalı olarak çözülür. Sonuç çözümü ise

$$a_j = a_j^{(p)} \quad , \quad 1 \le j \le p \qquad (27e)$$

ile verilir. p. dereceden bir öngörücü için çözümü elde ederken, p 'den daha küçük olan tüm dereceler için çözümler hesaplanır. Belirtilmiştir ki bu çözüm sayısal olarak göreceli olarak kararsızdır. Bununla beraber araştırmacıların çoğu bunu pratikte bir problem olarak görmemiştir.

Eğer tüm otokorelasyon katsayıları bir sabitle ölçeklenirse, (26) matrisinin çözümü bundan etkilenmez. Özellikle, eğer tüm R(i) 'ler R(0) ile bölünerek normalize edilirse, ***normalize otokorelasyon katsayıları ,r(i),*** olarak bilinenler elde edilir ;

$$r(i) = \frac{R(i)}{R(0)} \qquad (28)$$

Normalize otokorelasyon katsayılarının özelliği $|r(i)| \le 1$ olmasıdır.

(26) matrisinin çözümünün bir yan ürünü her adımda minimum toplam hatanın $,E_i$ hesaplanmasıdır. Kolayca gösterilebilir ki öngörücü katsayılarının derecesi arttıkça minimum hata da artar (ya da aynı kalır). E_i karesel bir hata olduğundan, hiçbir zaman negatif olmaz. Bu yüzden,

$$0 \le E_i \le E_{i-1} \quad , \quad E_0 = R(0) \qquad (29)$$

İstenirse kazanç da E_p cinsinden yazılabilir :

$$G^2 = E_p = R(0) + \sum_{k=1}^{p} a_k R(k) \qquad (30)$$

Eğer otokorelasyon katsayıları (28) eşitliğindeki gibi normalize edilmişse, bu durumda en küçük (minimum) hata da , E_i , R(0) ile bölünür. Sonuç niceliğine ***normalize hata*** ,V_i , denilir :

$$V_i = \frac{E_i}{R(0)} = 1 + \sum_{k=1}^{i} a_k r(k) \qquad (31)$$

(31) eşitliğinden açıktır ki ;

$$0 \le V_i \le 1 \quad , \quad i \ge 0 \qquad (32)$$

(27d) ve (31) eşitliklerinden de, sonuç normalize hatası , V_p ,

$$V_p = \prod_{i=1}^{p} (1 - k_i^2) \qquad (33)$$

k_i ara nicelikleri, $1 \le i \le p$, ***yansıma katsayıları*** olarak bilinir. İstatiksel literatürde bunlar ***kısmi korelasyon katsayıları*** olarak bilinirler. k_i ; s(n+1),...,s(n+i-1) sabit tutularak ve s(n) ile s(n+i) arasındaki (negatif) kısmi korelasyon olarak yorumlanabilir. Buradan k_i şöyle verilebilir ;

$$k_i = \frac{Z_{i+1} - Z_i}{Z_{i+1} + Z_i} \qquad (34)$$

4.3.2) TEKRARLI HESAPLAMA YÖNTEMİ :

Eşzamanlı doğrusal eşitliklerin çözümünde doğrusal yöntemlerden başka, bir gurup tekrarlı çözüm yöntemi de vardır. Bu yöntemlerde çözüm için bir başlangıç tahmini ile işe başlanır. Çözüm, sonra, bazı hata kriterlerinin gradiyentine dayanan bir düzeltme teriminin eklenmesiyle güncellenir. Genelde tekrarlı yöntemler istenen derecedeki yaklaşımı başarmak için, doğrudan hesaplama yöntemlerinden daha fazla sayıda hesaplamaya ihtiyaç duyar.

4.4) KUTUP SAYISININ EN UYGUN SEÇİMİ :

Tüm kutuplu modellerde verilmesi gereken en önemli kararlardan birisi de uygun kutup sayısının belirlenmesidir. Kutup sayısı ,p, arttıkça modelin uygunluğu da gelişir. Sorun nerede durulacağıdır. Açıktır ki ele alınan konu için uygun gelen en az değerdeki p alınmalıdır. Çünkü, p azaldıkça hem yapılacak işlem sayısı azalacak ve hem de hatalı durum olabilirliği (V_p azalırken p ile birlikte arttığından) en aza indirgenecektir.

Eğer işaretin spektrumu, p_0 kutuplu tüm-kutuplu bir spektrum ise, bu durumda biliyoruz ki $V_p = V_{p_0}$, $p \geq p_0$, ve $k_p = 0$, $p \rangle p_0$ 'dir, yani , $p \rangle p_0$ için hata eğrisi düzgün kalır. Bu yüzden, eğer işaret spektrumunun tüm-kutuplu bir spektrum olduğunu tahmin ediyorsak, en uygun p 'yi bulmak için ne zaman hata eğrisinin düzgün hale geldiğini kontrol edecek bir test yapılmalıdır. Son zamanlarda *Akaike* tarafından bir test işlemi yayınlanmıştır. Akaike, ortalama logaritmik olabilirlik öngörümüne karşılık gelen bir bilgi kriterinin kullanılmasını önermiştir. Bu şöyle verilir:

$$I(p) = -2\log(maksimum\_olasılık) + 2p \qquad (35)$$

I(p) 'yi en az yapacak p değeri en uygun değer olarak ele alınır. Bizim tüm kutuplu modelimizde, eğer işaretin bir Gauss olasılık dağılımı olduğunu farzedersek (35) eşitliği (toplanacak sabitleri ihmal ederek ve N ile bölerek) şu hale indirgenebilir :

$$I(P) = \log V_p + \frac{2p}{N_e} \qquad (36)$$

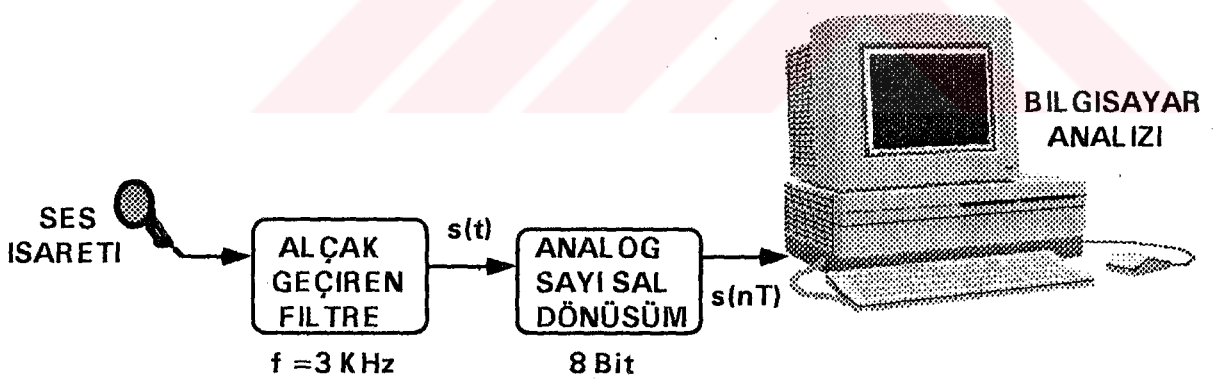
Burada N_e işaretteki veri noktalarının etkin sayısıdır. Ses işareti, yaklaşik Laplace dağılımına sahip olduğundan, olabilirlik fonksiyonu kestirilmeden önce işaret Hamming penceresi ile çarpılır ve daha sonra gerekli ayarlamalar yapılırsa N_e =0.4N için daha doğru olan aşağıdaki sonuç elde edilir :

$$I(P) = 10\log\left(V_p / R(0)\right) + (8.686p/0.4N) \qquad (37)$$

Model kutup derecesi seçimi için, Akaike 'nin yöntemine ek olarak en küçük tanımlama uzunluğu yöntemi geliştirilmiş ve daha iyi bir istatistik olduğu görülmüştür. (Wax , 1985) Bu tez için bunların göz önüne alınmasıyla en uygun p kutup değeri 12 olarak bulunmuş ve uygulanmıştır.

5) PROJE ADIMLARININ AÇIKLANMASI

Öncelikle ses işaretinin bir dinamik mikrofon kullanılarak analog işarate dönüştürülmesi gerekmektedir. Ancak elde edilen işaretin genlik düzeyi incelemeye müsait olmadığından hemen mikrofon çıkışındaki işaret milivoltlar mertebesinden voltlar mertebesine yükseltilmelidir. Bunun için basit bir iki katlı tranzistörlü kuvvetlendirici kullanılmıştır. Elde edilen ses analog işareti ±5 V arasında salınan, 8KHz frekanslı bir işarettir. Ancak ses işaretinin enerjisinin büyük bir kısmı 5 KHz 'in altında toplanmıştır. Seslilerin analizi için de bu sını yeterlidir. Çünkü tüm sesliler bu aralıkta yer alır. Eğer fon gürültüsü analizi zorlaştırıcı boyutlardaysa kesim frekansı 80Hz olan bir yüksek geçiren filtre kullanılması tavsiye edilir. Bu işaret örneklenerek sayısallaştırılmadan önce Nyquist kriterinin sağlanabilmesi için bir alçak geçiren filtreden geçirilmiştir. Nyquist kriterine göre örnekleme frekansı örneklenecek işaretin en büyük frekansından iki kat daha büyük olmalıdır. Bu yüzden ses işaretleri için yeterli bir örnekleme olan 10 KHz örnekleme hızı için ses işaretinin 5 KHz 'den daha yüksek harmonik ve bileşenlerden temizlenmesi gerekir. 2. dereceden bir Butterwoth alçak geçiren filtresi kullanıldığından, ve bu filtrenin çok keskin bir filtrelemesi olmadığından filtrenin kesim frekansı 3 KHz olarak seçilmiştir.

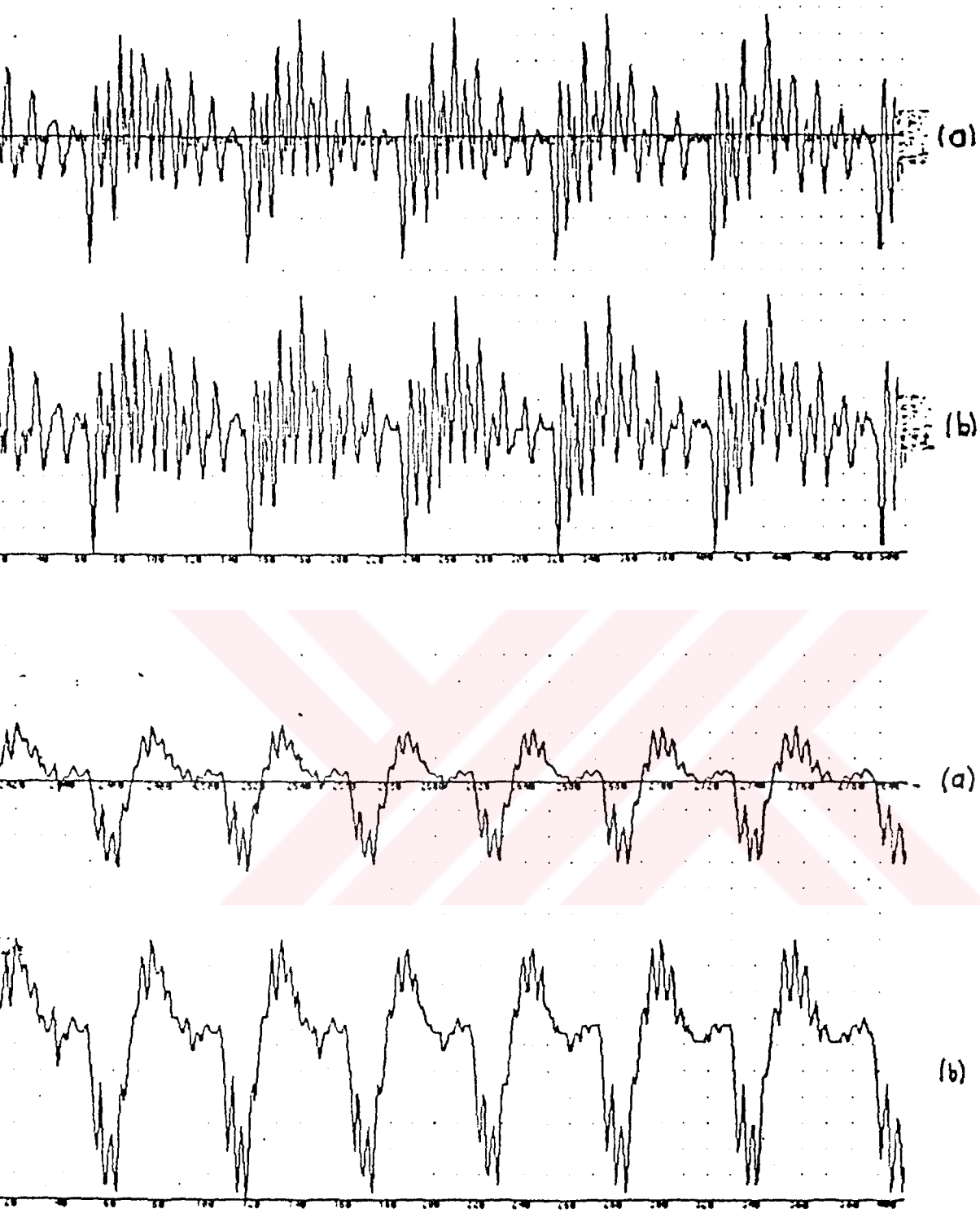


ŞEKİL 5.1 : Ses İşaretinin Bilgisayardaki Analize Hazırlanması

Alçak geçiren filtre çıkışında, filtre kazancını süzen bir AC İşaret Kuvvetlendirici kullanılarak, sonra kullanılmış olan İşlemsel Kuvvetlendiricilerin (OP-AMP) doymaya girmeleri önlenmiştir.

±5 V arasında değişmekte olan analog işaretin Analog-Sayısal Dönüştürücü girişine uygun bir hale, 0 - 5 V' arasına, taşınması için bu işarete 2.5 V 'luk bir DC seviye eklenerek salınım aralığının daraltılması için, sayısal dönüştürücü öncesinde bir Toplayan Kuvvetlendirici kullanılmıştır.

Artık işaret 0 - 5 V aralığında salınan, ADC girişine uygun hale getirilmiştir.



ŞEKİL 5. : a) Alçak geçiren filtre çıkışı

b) DC seviye Eklenerek 0 - 5 V arasına ötelenmiş işaret

ŞEKİL 5.3 : Adımların Şematik Gösterimi

İnsanın duyma üst eşiği 120 dB 'dir. Konuşma işaretinin ise yaklaşık olarak 70 dB ile sınırlı olduğu kabul edilir. Bu işaretin kaç bit ile temsil edilebileceğini bulmak istersek ;

$$20\log_{10} 2^n = 70\ dB \quad \text{ise} \quad n = 12\ bit \qquad (38)$$

O halde 12 bitlik bir dönüştürücü kullanılması doğru olacaktır. Ancak piyasad 8 bitlik ADC 'nin yaygın olarak kullnılması nedeniyle bizim de devremizde 8 bitlik bir ADC

Kullanılacak dönüştürücünün dönüşüm hızı ise

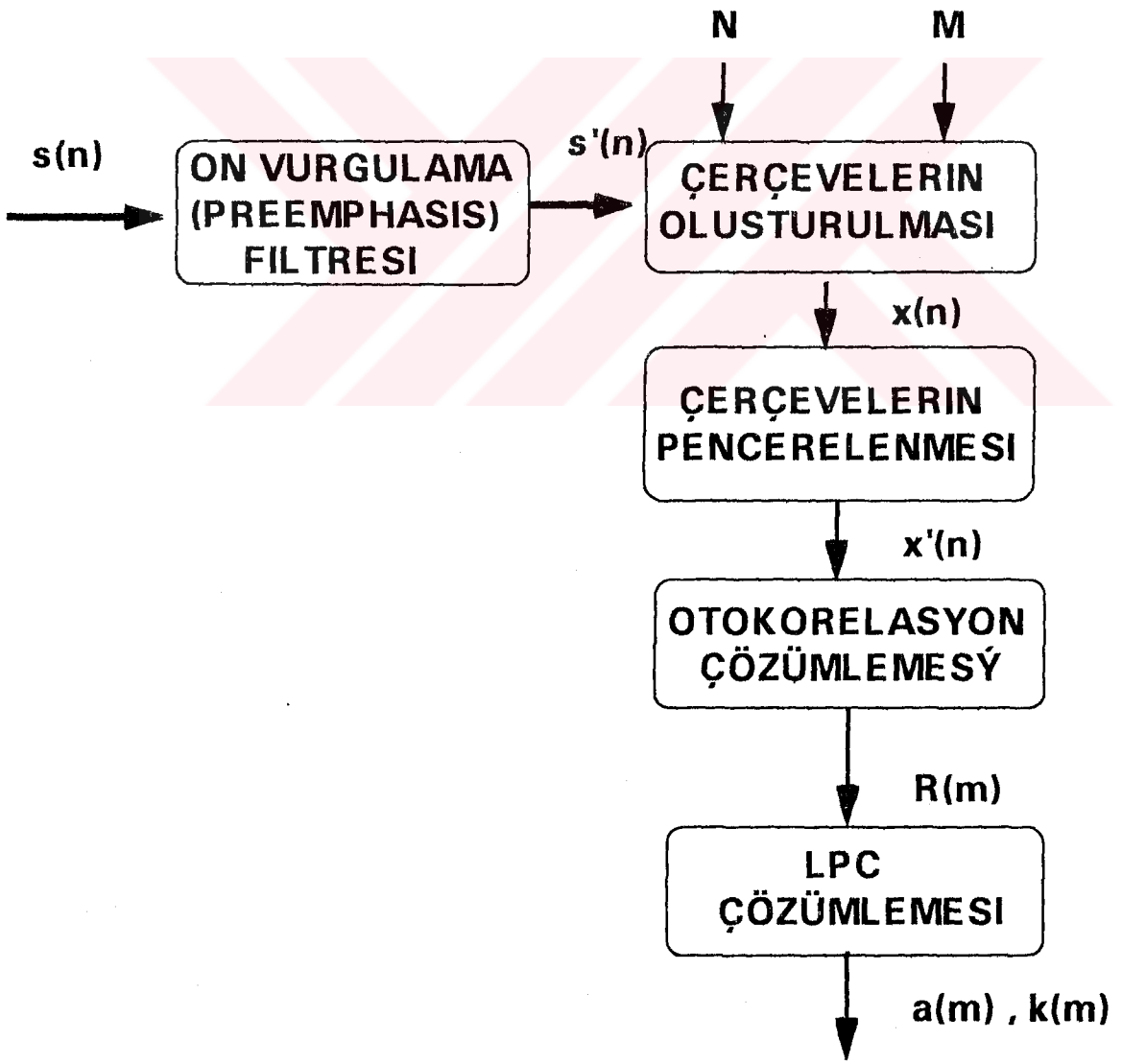
$t = 1 / f = 1 / 10000\ Hz = 100\ \mu sn$ olmalıdır. *(39)*

Yani dönüştürme süresi 0.1 msn 'den daha küçük olan bir ADC seçilmelidir.

Devremizde Ardışıl Yaklaşım Dönüşüm yöntemini kullanan bir ADC800 kullanılmıştır.

5.1) VERİ İŞLEME

Veri işleme bilgi içeren işaretlerden gereken bilginin çıkarılmasıdır. Konuşma işareti üzerinde LPC parametreleri ile özellik çıkarma amacı ile gerçekleştirilen işlemler özet olarak Şekil 6.3 'de verilmiştir.



ŞEKİL 5.4 : LPC Paramatreleri ile özellik çıkartma

Ses işareti, işaret güçü gürültü gücü oranını biraz daha arttırmak için önce bir ön vurgulama (preemphasis) filtresinden geçirilir. Bu filtreleme aynı zamanda, hesaplama yapılan ortamdaki hesaplamaya bağlı kararsızlıkların ortadan kaldırılması için de yararlıdır. Bu aşamada daha sonra sesli-sessiz ayrımı kısa aralıklarla hesaplanan enerjilere göre yapılır. Bu ayrım yapılırken küçük enerjili çerçeveler sessiz olarak, yüksek enerjiye sahip olanlar ise sesli olarak düşünülür. Analiz hedefimiz sesliler olduğundan sessiz çerçeveler konuşma işaretinden çıkarılmaktadır.

Ön vurgulama filtresinin transfer fonksiyonu ;

$$H(z) = 1 - az^{-1} \qquad (40)$$

Burada a 'nın değeri çoğu veri işleme uygulamasında olduğu gibi a=0.9375 alınmıştır. Frekans boyutunda giriş-çıkış ilişkisi yazılacak olursa ;

$$s'(n) = s(n) - as(n-1) \qquad (41)$$

elde edilir. Yani bir sonraki örnek kendinden bir önceki örnekten çıkarılacaktır.

Sayısal işaret özellik ölçümü amacıyla N adet örnek içeren alt bölümlere ayrılır. Tipik çerçeve boyutları 10 ms ile 40 ms arasındadır. 10 KHz 'lik örnekleme frekansı için N, 100 ile 400 arası seçilmelidir. Birbirini izleyen alt bölümler arasındaki örnek aralığı ise M ile gösterilmiştir. Açıktır ki $M<N$ olduğunda komşu alt bölümlerde üst üste binme (overlapping) olacaktır. Böyle bir binme, özellik katsayısı vektörlerinde nispeten bir yumuşama sağlar.

Verinin özelliklerini yumuşatmak için, çerçevenin her iki tarafına uçlardan konuşma örneklarini sıfırlayan yumuşatıcı bir pencere , w(n), uygulanmıştır. Bu pencere bir ***Hamming Penceresi*** 'dir.

$$w(n) = 0.54 - 0.46\cos\left(\frac{2\pi n}{N-1}\right) \quad , \quad 0 \le n \le N-1 \qquad (42)$$

Çerçeveler pencerelenince elde edilen işaret şu şekli alacaktır :

$$x_l'(n) = x_l(n).w(n) \quad , \quad 0 \le n \le N-1 \quad \text{ve} \quad 0 \le l \le L-1 \qquad (43)$$

Burada L indisi alt bölümün numarasını, n ise o bölüm içinde hangi örneğin işlendiğini göstermektedir.

Pencerelenmiş çerçeveler içerisindeki sayısal işaret yeterince durağan kabul edilerek ve doğrudan bir çözümleme yöntemi olan otokorelasyon çözümlemesine tabi tutulur. Otokorelasyon çözümlemesi şu sonucu verecektir :

$$R_l(m) = \sum_{n=0}^{N-1-M} x_l'(n)x_l'(n+m) \quad , \quad m = 0,1,...,p \qquad (44)$$

Burada ki p kutup sayısını göstermektedir ve uygun kutup sayısının bulunmasının anlatıldığı bölümde hatırlacağı gibi, bu değerin p=12 olarak seçilmesi uygun görülmüştü.

Son aşama ise LPC katsayılarının $,a_l(j)$, j = 1,2,...,p , veya , kısmi korelasyon katsayılarının (yansıma katsayılarının) , $k_l(j)$, hesaplanmasıdır. Her bir çerçeve için bu katsayılar Durbin yöntemiyle bulunduktan sonra, özellik vektörü şu şekilde elde edilir :

$$P_a = \langle a_l(j) \rangle_l \quad \text{veya} \quad P_k = \langle k_l(j) \rangle_l \quad l = 0,1,...,L\text{-}1 \ , \ j = 1,2,...,p \qquad (45)$$

Burada $\langle . \rangle_l$ operatörü, l 'ye göre ortalama alma anlamındadır. Bu tez de P_a ve P_k özellik vektörlerinin her ikisi de denenmiş ve benzer sonuçlar elde edilmiştir. Bu nedenle P_k özellik vektörünün kullanılmasının özel bir nedeni yoktur.

5.3) ÖRÜNTÜ (PATTERN) TANIMA :

Ses işaretinin özelliklerinden oluşan örüntülerin tanınması işlemidir. Örüntüler arasındaki benzerliğin bir ölçüsü örüntüler arasındaki uzaklıktır. Bu uzalık değişik biçimlerde tanımlanabilir. Örüntü tanımada yaygın olarak kullanılan uzaklık ölçüleri şunlardır :

a) Euclide,

b) Mahalonobis,

c) Dinamik Zaman Saptırması (Dynamic Time Warping) uzaklık ölçüsü.

5.2.a) EUCLIDE UZAKLIK ÖLÇÜSÜ :

x ve y iki vektör olsun. Bu iki vektör arasındaki Euclide uzaklığı $\left((x-y)^T(x-y)\right)^{1/2}$ olarak bilinir. Bu uzaklığı başvuru , R, ve test , T, örüntü vektörlerine uyarlayacak olursak, iki örüntü arasındaki Euclide uzaklık ölçüsünü

$$\left((R-T)^T(R-T)\right)^{1/2} \qquad (46)$$

olarak elde ederiz. Bu uzaklık ölçüsünü kullanarak ses örüntüsünü tanımada iyi sonuçlar elde etmek olası değildir. Bunun başlıca nedenleri, bu uzaklık ölçüsünün istatiksel veri içermemesi ve ses örüntüsünün aynı kişi için bile zamana göre değişim göstermeşidir.

5.2.b) MAHALANOBİS UZAKLIK ÖLÇÜSÜ :

Euclide uzaklık ölçüsüne göre daha karmaşıktır ve sesin istatiksel verilerini de içerir (Atal, 1971 ve 1976). Bu uzaklık ölçüsünü şöyle tanımlayabiliriz :

x, giriş örüntüsünü temsil eden L boyutlu bir sütun vektör olsun. Burada x 'in k 'ıncı elemanı k 'ıncı ölçümdür. i 'inci konuşmacının ölçümlerinin bileşik olasılık yoğunluk fonksiyonunun, ortalaması m_i ve koveryans matrisi W_i olan çok boyutlu Gauss dağılımı olduğunu farzedersek, x 'in L boyutlu Gauss yoğunluk fonksiyonu aşağıdaki gibi olur:

$$g(x) = (2\pi)^{-1/2}|W|^{-1/2} exp\left[-(1/2)(x-m_i)^T W_i^{-1}(x-m_i)\right] \qquad (47)$$

Burada sorun x 'in hangi konuşmacıya ait olduğunun bulunmasıdır. Buna göre, eğer $p_i g_i(x) \geq p_j g_j \quad , \quad \forall i \neq j$ ise x, i 'inci konuşmacı olarak atanacaktır. (47) eşitliğinin her iki tarafının doğal logaritmasını alıp, x 'e göre değişmeyen terimleri atarsak,

$$d_i = (x-m_i)^T W_i^{-1}(x-m_i) \qquad (48)$$

elde ederiz. Burada

$$m_i = \frac{i}{N_i}\sum_{n=1}^{N_i} x_i(n) \qquad (49)$$

$$W_i = \frac{1}{N_i}\sum_{n=1}^{N_i} x_i(n)x_i^T(n) - m_i m_i^T \qquad (50)$$

Bu eşitlik Mahalanobis uzaklığı olarak bilinir.

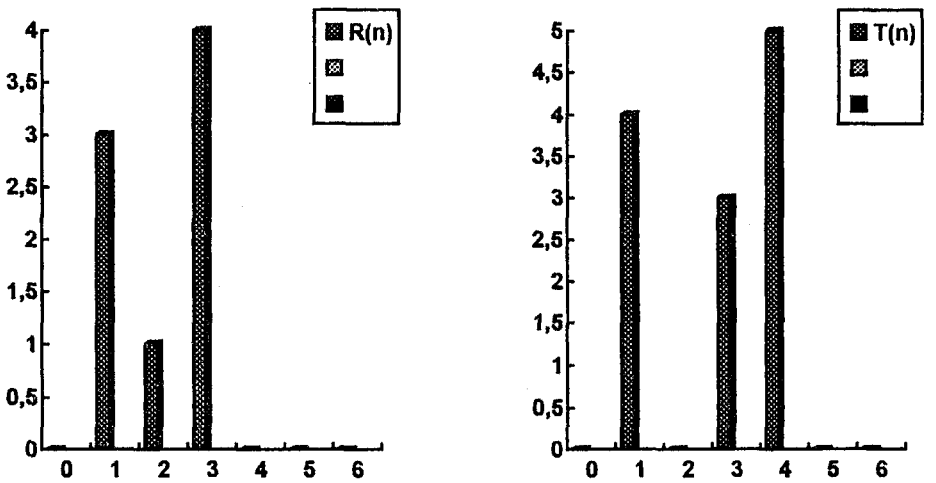
5.2.c) DİNAMİK ZAMAN SAPTIRMA :

Aynı konuşmacıya ait örüntülerin zamana ve genliğe göre sapma göstermesi konuşmacıların ya da konuşulanların tanınması işlemini güçleştirmektedir. Örüntülerin eşzamanlaması yolu ile bu sorunun üstesinden gelmek kısmen de olsa mümkündür. Eşzamanlama işlemi, belirli ses özellikleine göre doğru uzaklık hesaplamasını sağlar.

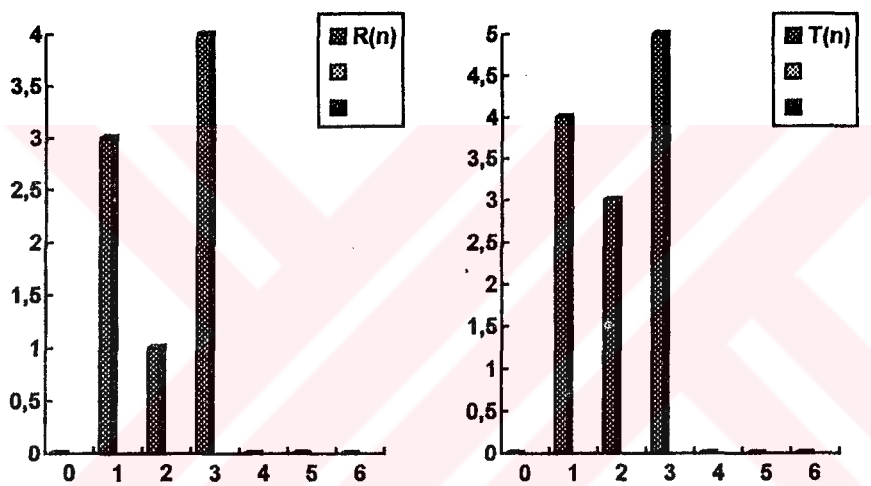
Bir örnek olarak, bir R=(0,3,1,4,0) başvuru örüntüsü ile bir T=(0,4,0,3,5,0) test örüntüsünün eşzamanlaması Şekil 6.5 'deki gibi olur. Şekil 6.5.a 'da görülen test örüntüsü başvuru örüntüsüne göre eşzamanlanmıştır ve eğer burada Euclide uzaklık ölçüsü kullanırsa yanlış bir sonuç elde edilir. Çünkü T örüntüsünün zaman ekseninde, R örüntüsünde mevcut bulunan özelliklere göre bir kayma söz konusudur. T 'deki özellikler Şekil 6.5.b 'deki gibi R 'nin benzer özellikleri ile zaman olarak aynı indise karşılık getirilip gerekli uzaklık hesaplaması yapılırsa daha doğru bir karşılaştırma ölçüsü elde edilmiş olur. Açıktır ki uzaklık ne kadar küçük çıkarsa benzerlik o kadar fazladır. Test örüntüsü eldeki bütün başvuru örüntüleri ile karşılaştırılarak, en küçük uzaklığın elde edildiği başvuru örüntüsünün kimliği, bu test örüntüsüne atanır. Bu koşullarda hesaplama zamanı, kısa eğitme ve kısa test konuşma zamanı söz konusu olduğunda, dinamik zaman saptırma uzaklık ölçüsü, Mahalanobis uzaklık ölçüsüne göre daha verimli bir alternatif oluşturur (Rosenberg, 1975).

5.3) DİNAMİK ZAMAN SAPTIRMASI İLE ÖRÜNTÜ TANIMA

Örüntüler elde edildikten sonraki aşama, örüntüler arası benzerliğin belirlenmesidir. Konuşma şekli zamanla oldukça fazla değiştiğinden, örüntülerin eşzamanlanması gerekmektedir. Şekil 6.6 'da, T(t) ile R(t) örüntüleri arasındaki eşzamanlama işlemi görülmektedir (Rabiner, 1981).

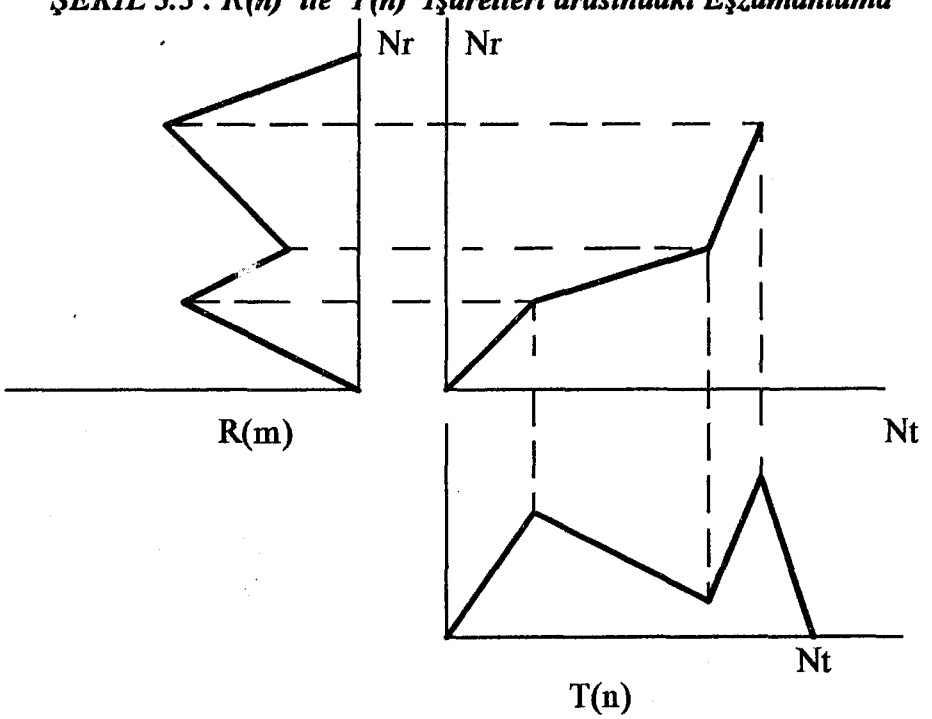


a) Eşzamanlama Öncesinde Durum



b) Eşzamanlama Sonrası Durum

ŞEKİL 5.5 : R(n) ile T(n) İşaretleri arasındaki Eşzamanlama



ŞEKİL 5.6 : R(t) ile T(t) 'nin w(n) fonksiyonu kullanılarak eşzamanlanması

Burada amaç iki örüntü arasındaki uzaklığı , D(T,R) , bir w dönüşümü ile en aza indirgemektir. Öyle bir W(t) bulunmalıdır ki aşağıdaki ifade en küçük olsun :

$$D(T,R) = \int_{t_0}^{t} d(t,w(t))G(t,w(t),\frac{dw(t)}{dt})dt \qquad (51)$$

Bu eşitliğin sürekli zamanda çözümü genelde yoktur. Bu yüzden bu problemin çözümü kesikli zamanda gerçekleştirilir. R ve T 'nin aşağıdaki gibi verildiğini düşünelim :

$$T(T(1), T(2), T(3), ..., T(Nt)) \quad ve \quad R(R(1), R(2), R(3), ..., R(Nr)) \quad (52)$$

En iyi eşzamanlama yolu, başvuru örüntüsü zaman koordinat ekseni, m, ile deney örüntüsü zaman koordinat sistemini , n, birbiri ile ilişkilendiren eğridir ve şu şekilde verilir :

$$m = w(n) \qquad (53)$$

w(n) 'in sınır koşulları ise şöyle verilebilir :

$$w(1) = 1 \quad ve \quad w(Nt) = Nr \qquad (54)$$

w(n) 'in belirlenmesi için bir kaç yol önerilmiştir. Bunlar :

1) Doğrusal Eşzamanlama ;

$$m = (n-1)(Nr-1) / (Nt-1) + 1 \qquad (55)$$

2) Zaman - Olay Eşleme, R ve T 'deki belirgin benzer olayların aynı zamanlara karşılık getirilmesi,

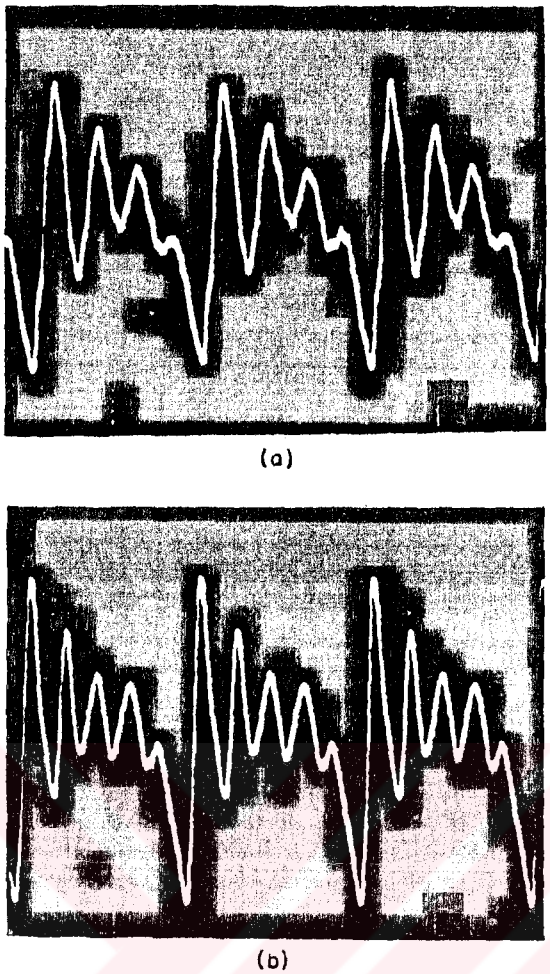
$$m_1 = w(n_1)$$
$$m_2 = w(n_2)$$
$$\vdots$$
$$m_Q = w(n_Q) \qquad (56)$$

3) Korelasyonların en büyütülmesi ; w(n), R ile T arasındaki korelasyonu en büyük yapacak bir şekilde seçilir.

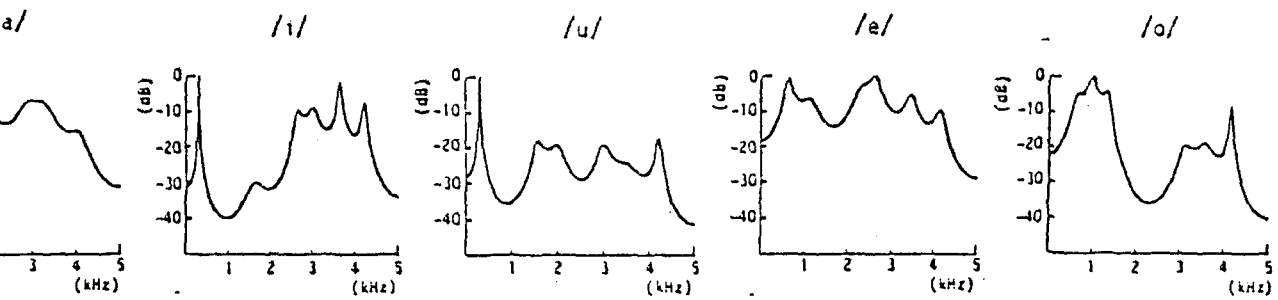
4) Dinamik zaman saptırma, saptırma eğrisi aşağıdaki optimizasyon probleminin çözümü olarak bulunur :

$$D^* = min_{w(n)} \sum_{n=1}^{Nt} d(T(n), R(w(n))) \qquad (57)$$

Burada d(T(n)) ile R(w(n)), deney örüntüsünün n 'inci çerçevesi ile başvuru örüntüsünün w(n) 'inci çerçevesi arasındaki uzaklıktır.



ŞEKİL 5.7 : Bazı Seslilerin Zaman Boyutundaki Dalga Şekilleri
a) "A" seslisine ait b) "O" seslisine ait



ŞEKİL 5.8 : 12 Kutuplu bir Doğrusal Öngörü Kodlaması sonucu bulunacak seslilere ait spektral zarflar

5.4) ÇERÇEVEDEN ÇERÇEVEYE UZAKLIK ÖLÇÜLERİ

(57) eşitli inin optizasyonu için öncelikle d(.) uzaklık ölçüsünün (metri inin) tanımlanması gerekir. Çeşitli uzaklık ölçüleri kullanılabilir. Örne in, basit bir ***Euclide Uzaklık Ölçüsü*** kullanılırsa :

$$d(T,R)=\sum_{i=0}^{p}(T_i-R_i)^2 \qquad (58)$$

Buradaki T_i ve R_i , i 'inci vectör bileşenleridir. Bu uzaklık ölçüsü sık sık kullanılır.

Kullanılabilecek bir başka uzaklık ölçüsü ***Kovaryans A ırlıklı*** uzaklık ölçüsüdür. Bu uzaklık aşa ıdaki gibi ta ımlanır :

$$d(T,R)=(T-R)W^{-1}(T-R)^T \qquad (59)$$

Burada, W^{-1} , kullanılan özelli in koveryans matrisinin tersidir. Bu matrisin beklenen de eri şudur :

$$\varepsilon[W]=(1/N)\sum_{i=1}^{N}(x_i-\varepsilon[x])(x_i-\varepsilon[x])^T \qquad (60)$$

x_i , i 'inci çerçeveye ait özellik kümesi ve N isetoplam çerçeve sayısıdır. Burada kovaryans a ırlı ının kullanılması, özellikler arasındaki korelasyonu uyumlandırır ve uzaklı ın tamamında tüm özelliklere a ırlık verilmesini sa lar.

Spektral uzaklık ölçüsü kullanılırsa, bu ölçü için önce test ve başvuru örüntülerinin logaritmik spektrumu elde edilmelidir ve uzaklık ise şu şekilde verilir :

$$d(T,R)=\int_W\left[\log\left(T(e^{jw})\right)-\log\left(R(e^{jw})\right)\right]^2 dw \qquad (61)$$

Burada integral, ilgilenilen frekans aralı ında alınır.

LPC olabilirlik ölçüsü ise Itakura tarafından geliştirilmiştir (1975) ve LPC parametrelerinden oluşan özellik kümeleri için son derece faydalıdır. Bu uzaklık ölçüsü şu şekilde verilebilir :

$$d(T,R) = log\left[\left(a_R V_T a_R^T\right) / \left(a_T V_T a_T^T\right)\right] \qquad (62)$$

Burada a_R ve a_T sırasıyla çerçevedeki, test ve başvuru örüntülerinin do rusal öngörü katsayılarından oluşan vektörler; V_T ise test örüntüsüne ait korelasyon katsayılarından oluşan matristir.

Herhangi bir uzaklık ölçüsünün en önemli özelliklerinden biri hesaplanabilme hızıdır. Çünkü uzaklık ölçülerinin hesabı örüntü tanıma işleminde en uzun süre alan hesaptır. Bu yüzden yukarıda anlatılmış olan uzaklık ölçüleri için daha hızlı hesaplama yöntemleri geliştirilmiştir. Bu açıdan problemi tekrar ele alırsak aşa ıdaki sonuçları elde ederiz :

1) Euclide uzaklı ı ;
   Bu uzaklık ölçüsünün hesaplanması (p+1) çarpma, (p+1) çıkarma, (p+1) toplama işlemi gerektirir. Farkların kareleri yerine kendileri do rudan toplanırsa çarpma işlemlerinden kurtulunmuş olunur.
2) Koveryans a ırlıklı uz lık;
   Bu uzaklı ın hesaplanması $(p+1)^2$ çarpım ve toplam içerir. p 'nin mantıklı herhangi bir de eri için bu hesaplama oldukça uzun sürer.
3) Spektral uzaklık;
   Bu uzaklık, bir integralin veya kesikli yaklaşımın hesabını gerektirir. Her iki durumda da hesap uzun olacaktır. Bu hesabı kolaylaştırma yolları 1976 yılında Gray tarafından açıklanmıştır. (örne in hiperbolik kosinüs açılımı gibi)

6) GELİŞTİRİLEN BİLGİSAYAR PROGRAMLARI

Türkçe 'deki seslilerin tanınmasını yapacak olan programların yazımında Turbo C++ kullanılmıştır. C programlama dilinin tercih nedeni, derleyicinin hızlı çalışan amaç kod üretebilmesi, hızlı çalışan programların oluşturulabilmesi için kitaplık alt programlarına izin vermesi, programcıyı kısıtlamaması olarak özetlenebilir.

Programlar küçük modüller halinde yazıldı ından ve bu modüller birer kitaplık alt programı haline getirildi inden, programların kullandı ı fonksiyonların kaynak kodlarının verilmesi yeterli görülmüştür. Bu fonksiyonlar şunlardır :

*void preemp(char *f_name) :*

Bu fonksiyon, f_name adlı dosyadaki ses işaretini preemphasis filtresinden geçirir ve sonucu f_name.pr adlı dosyada saklar. Bu filtreleme aşa ıdaki filtre transfer fonksiyonuna göre çalışır.

$$H(z) = 1.0 - a * z^{-1}$$

*void remove_silence(char *f_name) :*

Bu fonksiyon f_name adlı dosyadaki ses işaretinden, ses içermeyen kısımları çıkarır ve sonuçta elde edilen işareti out adlı dosyada saklar. Bu işlem ses işaretinde bloklanan çerçevelerdeki enerjilerin hesaplanması ile başlar ve minimum enerji hesaplanır. Daha sonra 1.035*(minimum enerji) de erinden daha küçük enerjiye sahip çerçeveler dosyadan çıkarılır.

*void frame(char *f_name) :*

Bu fonksiyon f_name adlı dosyadaki ses işaretinden, %50 oranında üst üste binmiş çerçeveler oluşturur. Bir başka deyişle, e er, L tane toplam çerçeve var ise sonuç işaret , 2*L-1 çerçeve içerecektir. Sonuçta elde edilen işaret f_name.fr adlı dosyada saklanacaktır.

*void Hamming(char *f_name) :*

Hamming(.) fonksiyonu, f_name adlı dosyadan aldı ı ses işaretini bir Hamming penceresi ile pencereler. Hamming pencere fonksiyonu şu şekilde verilebilir :

$$w_H(n) = 0.54 - 0.46\cos(2\pi n / (N-1))$$

Burada N çerçeve genişli idir. Sonuçta elde edilen çerçeve genişli i f_name.wn adlı dosyada saklanacaktır.

*void Durbin(char *f_name) :*

Bu fonksiyon sesişaretini f_name adlı dosyadan okuduktan sonra, konuşulan sesliye ait özelliklerine Durbin çözümleme algoritmasına göre hesaplar ve sonuçta elde edilen özellikleri f_name adlı dosyada saklar:

void feature(char f_name) :

Bu fonksiyon f_name adlı her bir sesliye ait özellikleri f_name1,...,f_name(n) adlı dosyalardan okuyarak ortalaması alınmış başvuru özellikl vektörünü oluşturur. Sonuç başvuru özellik vektörü feature adlı dosyada saklanır.

void dp_Sakoe(int n) :

Bu fonksiyon, feature.n adlı başvuru dosyasındaki özellik vektörünü, feature.t adlı test işareti özelliklerini içeren dosyadaki özellik vektörü ile eş zamanlayarak karşılaştırır. Fonsiyondan dönen de er iki örüntü arasındaki uzaklıktır.

eu.c :

Bu program, basit olarak iki örüntü arasındaki uzaklı ı Euclide uzaklı ını hesaplar. Hesaplanan euclide uzaklı ı aşa ıdaki gibidir.

$$D = (R-T)^T (R-T)$$

Burada T ile R , sırasıyla karşılaştırılan başvuru ve test örüntüleridir.

maha.c :

Bu program, verilen başvuru ve test örüntüleri arasındaki Mahalanobis uzakı ını hesaplar. Bu uzaklık aşa ıdaki gibi hesaplanır :

$$d(T,R) = (T-R)W^{-1}(T-R)^T$$

Burada, W , test örüntülerinin koveryans matrisidir ve aşa ıdaki gibi tahmin edilir :

$$W = (1/N)\sum_{i=1}^{N}(x_i - \varepsilon[x])(x_i - \varepsilon[x])^T$$

Burada, N test örüntülerindeki toplam çerçeve sayısı ve x_i ise çerçeveye ait örüntülerdir.

cov.c :

Bu program, cov dosya1 dosya2 şeklinde işletilir ve sonuçta dosya1 ile dosya2 arasındaki koveryans matrisini hesaplar. Sonuçta elde edilen koveryans matrisi W adlı dosyada saklanır.

fmi.c :

Bu program, fmi dosya N biçiminde işletilir ve dosyadaki N*N boyutundaki kare matrisin tersi alınır. E er matrisin determinantı $1.0*10^{-15}$ 'den daha küçükse tekil matris tanısı ile birlikte bir hata mesajı verilir ve programın çalışması sona erer. fmi.c programı tamamen paralel olan Modified Csanky Metodunu kullanır ve $(\log_2 N)^2$ işlemde çarpma ve bölme matrisinin tersini alır. İşlem sayısı, Gauss yoketme yöntemi için N^9 , L-U (Lower Upper) ayrıştırma yöntemi (alt ve üst üçgen matrislere ayırma) için yaklaşık N^2 'dir.

7) SONUÇLAR :

Gerçekleştirilen deneylerde 10 KHz örnekleme frekansında, 0.8 msn 'lik konuşma örnekleri, orta kalitede bir dinamik mikrofondan alınarak kaydedilmiştir. Seslerin kaydedildi i işaret gücü-gürültü gücü oranı yaklaşık olarak 30 dB kadardır. Ortam gürültüsü işlem performansı ve sonuç başarımları üzerinde önemli bir etkiye sahiptir.

Deneyde önce, ses kalitesi göz önüne alınmaksızın, başvuru örüntüsünü oluşturacak girişler alınır. Sonra tanıma işlemi için gerekli olan ses özellik vektörleri do rusal öngörü kodlaması teknikleri kullanılarak elde edilir. Verilen test örüntülerinin tanınması için kısa süreli bir e itme kullanılır. Örüntü tanıma işlemi Euclide, Mahalanobis ve Dinamik Zaman Saptırma uzaklıklarının her biri ayrı ayrı kullanılarak sonuçlar araştırılmıştır. Ancak gerek Euclide, gerek Mahalanobis ve gerekse Dinamik Zaman Saptırması kullanılarak yapılan karşılaştırmalarda başarılı sonuçlar elde edilememiştir. Dinamik zaman saptırması yönteminde di erlerine göre do ruya biraz daha yakın sonuçlara rastlanmıştır.

Yeni test girişleri alınmaksızın, daha önce başvuru örüntüsünü oluşturmada kullanılan girişlerden birisi test girişi kabul edilirse; performansta iyileşmeler görülmüştür. Bu durumda her üç uzaklık ölçüsü için de elde edilen başarım oranları şu şekildedir :

	Euclide	Mahalanobis	Dinamik Z.S.
Başarı (%) :	42	61	76

Sonuç performansının daha başarılı sonuçlar vermesi öncelikle alınan kayıtların daha iyileştirilmesine ba lıdır. Buna ek olarak daha uzun süreli e itmenin de tanıma işlemine katkısı vardır. Modellemedeki kutup sayısının ve örnekleme çözünürlü ünün arttırımı da (ses için 12 veya 14 bit/örnek 'lik çözünürlük seçilmesi gibi) performansı arttıracaktır.

KAYNAKLAR :

1) Akaike, H., A New Look at Statistical Model Identification,
IEEE Trans. on Automatic Control , Dec. 1974

2) Atal, B. S. , Automatic Recognition of Speakers From Their Voices,
Proc. IEEE , April 1976

3) Flanagan, J. L., Properties of the Speech Signal,
The Journal of the Acoustical Society of America, vol.51, pp.1375-1387,
Mar. 1972

4) Gray, A. H., et. al., Distance Measures for Speech,
Proc. IEEE Trans. on ASSP, Oct. 1976

5) Itakura, F., Minimum Prediction Residual Principle Applied to Speech Recognition
IEEE Trans. on ASSP, Vol. 23, Feb. 1975

6) Makhoul, J., Linear Prediction : A Tutorial Review,
Proc. IEEE, April 1975

7) Maknoul, J., Spectral Analysis of Speech by Linear Prediction,
IEEE Trans. on Audio and Electroacoustics, June 1973

8) Markel, J. D., et. al., Linear Prediction of Speech ,
Springer-Verlag , Berlin, 1976

9) Miyoshi, Y., el. al., Analysis of Speech Signals of Short Pitch Period by a Sample
Selective Linear Prediction,
IEEE Trans. on Acoustics, Speech and Signal Processing, Sep. 1987

10) Rabiner, L. R., et. al., Digital Processing of Speech Signals,
Prentice-Hall Inc., Englewood Cliffs, New Jersey, 1978

11) Rabiner, L. R., Levinson, S. E., Isolated and Connected Word Recognition
Theory and Selected Applications,
IEEE Trans. on Acoustics Speech and Signal Processing , Feb. 1981

12) Reddy, D. R., Speech Recognition by Machine
Proc. IEEE ,vol.64, pp. 501-531, April 1976

13)Rosenberg, A., et. al., New Techniques for Automatic Speaker Verification,
IEEE Trans. on ASSP, vol.23, pp.169-176 , 1976

13)Schafer, R. W., et. al., System for Formant Analysis of Voiced Speech,
The Journal of the Acoustical Society of America, vol.47, pp.634-648,
Feb. 1970

14) Schafer, R. W., et. al., Digital Represantation of Speech Signals,
Proc. IEEE, vol.63, pp.662-677 , Apr. 1975

ÖZGEÇMİŞ

1968 yılında İstanbul 'da dogdu. İlk ve orta ögrenimini İstanbul 'da tamamladı. İstanbul Ataköy Lisesi Yabancı Dil Kolundan 1985 yılında mezun olarak aynı yıl açılan ÖSYM sınavları sonucunda Yıldız Üniversitesi Elektronik ve Haberleşme Bölümüne girdi. 1990 yılında mezun olup, aynı yıl Yıldız Teknik Üniversitesi 'nde açılan yüksek lisans sınavlarını kazanarak F.B.E Elektronik Anabilimdalında yüksek lisansa başladı.

PREEMP.C

```
/*
This  function  preemphasizes  "f_name"  into  "f_name.pr"
in accordance with simulation of the digital low pas filter
frequency domain transfer function :

                              -1
         H(z)= 1.0 - a * z        ,

where a is chosen as 0.9375  proposedly.
*/

void preemp(f_name)
char f_name[10];

{
 FILE *fp1,*fp2;
 int stream,str,dummy;

 fp1=fopen(f_name,"r");
 strcat(f_name,"pr");
 fp2=fopen(f_name,"w");
 if (!fp1 || !fp2){
         printf(" \n Unable to open ....");
         exit (1);
 }
 fscanf(fp1,"%d",&dummy);
 fprintf(fp2,"%d\n",dummy);
 for(;;) {
         fscanf(fp1,"%d",&stream);
         if(!feof(fp1)){
               str=stream-0.9375*dummy;
               fprintf(fp2,"%d\n",str);
               dummy=stream;
         }
         else break;
  }
 fclose(fp1);
 fclose(fp2);
 }
```

REMOVE_SILENCE.C

```
remove_silence(f_name)
{

 float find_min();
 float energy();
 FILE *fp1,*fp2;
 long int offset;
 float E[200];
 float min_E;
 float frame[200];
 float threshold;
 float temp;
 register int i,j;
 int j_max;
 fp1=fopen(f_name,"r");
 fp2=fopen("out","w");
 if(!fp1||!fp2){
         printf("Unable to open......");
         exit(1);
 }
 for(;;) {
     for(j=0;j<200;j++){
       for(i=0;i<fr_size;i++){
          fscanf(fp1,"%f",&frame[i]);
          if(feof(fp1)) break;
       }
       if(feof(fp1)) break;
       E[j]=energy(frame,fr_size);
       j_max=j;
     }
     if(feof(fp1)) break;
 }
 min_E=find_min(E,j_max);
 fseek(fp1,0L,0);
 threshold=1.035*min_E;
 for (i=0;i<=j_max;i++) {
    for (j=0;j<fr_size;j++){
       fscanf(fp1,"%f",&temp);
       if(E[i]>threshold)
       fprintf(fp2,"%f\n",temp);
    }
 }
 fclose(fp1);
 fclose(fp2);
 }



 float energy(fr,fr_size)

 int  fr_size;

 float fr[];

 {

  register int i;
```

```
float sum=0.0;
for (i=0;i<fr_size;i++)
   sum+=fr[i]*fr[i];
   return sum;
}

float find_min(E,max_index)
float E[200];
int max_index;
{
 register int i;
 float min=E[0];
 for (i=1;i<=max_index;i++)
     min = (min<E[i]) ? min : E[i];
     return min;
 }
```

FRAME.C

```
/*
This function produces overlapped frames from
file "f_name" and stores  the  result  in  the
"f_name.fr" .
*/

 void frame(f_name)
 char f_name[10];
 {
  FILE *fp1,*fp2;
  register int i,j;
  int buf1[200],buf2[200];
  fp1=fopen(f_name,"r");
  strcat(f_name,"fr");
  fp2=fopen(f_name,"w");
  if ( !fp1 || !fp2) {
       printf(" \n   Unable to Open !\n");
       exit(1);
  }
  for (i=0;i<n_frames;i++) {
      for (j=0;j<90;j++)
          fscanf(fp1,"%d",&buf1[j]);
          for (j=90;j<180;j++)
             fscanf(fp1,"%d",&buf2[j-90]);
          if(i!=0){
              for (j=0;j<90;j++)
                 fprintf(fp2,"%d\n",buf1[j]);
          }
          for (j=0;j<90;j++)
                fprintf(fp2,"%d\n",buf1[j]);
          for (j=90;j<180;j++)
              fprintf(fp2,"%d\n",buf2[j-90]);
          if (i!=n_frames-1) {
              for (j=90;j<180;j++)
                fprintf(fp2,"%d\n",buf2[j-90]);
          }
  }
  fclose(fp1);
  fclose(fp2);
  }
```

HAMMING.C

```
/*
 This function  Hamming_windows  the  data  in
 "f_name" and stores the result in "f_name.wn"
 according to the  relationship :
      s'[n]=s[n]*w[n]  ,
 where  w[n] is the Hamming window function and
 given as
     w[n]= 0.54 - 0.46 * cos ( 2*pi*n / (N-1) ) ,
 where N is the window size and n=0,...,N-1.
 */

 void Hamming(f_name)
 char f_name[10];
 {
 FILE *fp1,*fp2;
 float frame[180];
 int w_size=180;
 register int n,nf;
 float pi=3.14159265;
 float  w;
 double temp;
 fp1=fopen(f_name,"r");
 strcat(f_name,"wn");
 fp2=fopen(f_name,"w");
if( !fp1 || !fp2) {
    printf("\n   Unable to open ......\n");
    exit(1);
}
for(nf=0;nf<n_frames;nf++) {
    for (n=0;n<w_size  ;n++)   {
         fscanf(fp1,"%f",&frame[n]);
         if(feof(fp1)) break;
         temp = 2*pi*((float)n /(float)( w_size-1));
         w= 0.54 - 0.46*cos(temp);
         frame[n] *= w;
         fprintf(fp2,"%f\n",frame[n]);
    }
}
fclose(fp1);
fclose(fp2);
}
```

DURBIN.C

```
/*
This function calculates the PARtial CORrelation
coefficients for each frame in file "f_name" and
stores the results in "f_name.du". The algorithm
used is due to Durbin and which is known  to  be
the fastest !
*/

void durbin(f_name)
char f_name[10];
{
 FILE *fp1,*fp2;
 int p=12;
 int nf;
 register int i,j;
 float E[13];
 float R[13];
 float A[13][13];
 float a[13];
 float k[13];
 float sum;

 fp1=fopen(f_name,"r");
 strcat(f_name,"du");
 fp2=fopen(f_name,"w");
 if ( !fp1 || !fp2) {
     printf(" \n   Unable to Open !\n");
     exit(1);
 }
 for(nf=0;nf<n_frames;nf++){
            for (i=0;i<=p;i++)
            fscanf(fp1,"%f",&R[i]);
            k[0]=a[0]=A[0][0]=0.0;
            E[0]=R[0];      /*     initial condition  ...  */
            for(i=1;i<=p;i++) {
                  sum=0.0;
                  for(j=1;j<i;j++)
                  sum+=A[j][i-1]*R[i-j];
                  k[i]=(R[i]-sum)/E[i-1];
                  A[i][i]=k[i];
                  for(j=1;j<i;j++)
                  A[j][i]=A[j][i-1]-k[i]*A[i-j][i-1];
                  E[i]=(1-k[i]*k[i])*E[i-1];
            }
            for(i=1;i<=p;i++)
                a[i]=A[i][p];
            for (i=1;i<=p;i++)
                fprintf(fp2,"%f\n",k[i]);
  }
  fclose(fp1);
  fclose(fp2);
  }
```

FEATURE.C

```
/*
This function averages the partial correlation
coefficients in file "f_name " and stores the
final set in file "feature.i" where i
indicates the i'th speaker .
*/

void feature(f_name,i)
char f_name[10];
int i;
{
FILE *fp1,*fp2;
int p=12;
char alpha_i[10];
char out_f_name[]="feature";
float k[13];
register int i,j,nf;
fp1=fopen(f_name,"r");
if(i) {
        itoa(i,alpha_i,10);
        strcat(out_f_name,alpha_i);
}
else
strcat(out_f_name ,".t");
fp2=fopen(out_f_name,"w");
if ( !fp1 || !fp2) {
       printf(" \n   Unable to Open !\n");
       exit(1);
}
for (i=0;i<13)
      k[0]=0.0;
for (nf=0;nf<n_frames;nf++) {
       for (j=1;j<13;j++) {
           fscanf(fp1,"%f",&stream);
           k[j]+=stream;
       }
}
fclose(fp1);
for (j=1,j<13;j++) {
       k[j]/=n_frames;
       fprintf(fp2,"%f",k[j]);
}
fclose(fp2);
}
```

DP_SAKOE.C

```
/*
This function produces a distance score as a measure
of similarity between two patterns stored in files
"feature.i" and "feature.t" . The algorithm of this
procedure is due to Sakoe and known to be
the last version of the dynamic programming approach
to pattern matching .
*/
float dp_Sakoe(n)
int n;
{
FILE *fp1,*fp2;
int i,j;
int I=12,J=12,N=12;
float R[13],T[13];
int r=6;
float g[13][13];
float c1,c2,c3;
float D;
float temp,dist;
char f_name[]="feature.";
char alpha_i[10];
itoa(n,alpha_i,10);
strcat(f_name,alpha_i);
fp1=fopen(f_name,"r");
fp2=fopen("feature.t","r");
if ( !fp1 || !fp2 ) {
      printf(" \n   Unable to Open !\n");
      exit(1);
}
T[0]=R[0]=0.0;
for(i=1;i<=12;i++) {
     fscanf(fp1,"%f",&R[i]);
     fscanf(fp2,"%f",&T[i]);
}
fclose(fp1);
fclose(fp2);
for(i=0;i<13;i++)       /*  initial condition ... */
    for (j=0;j<13;j++) g[i][j]=0.0;
i=1;
j=1;
g[1][1]=2*(R[1]-T[1])*(R[1]-T[1]);
i=i+1;
for(;;) {
  if(i>j+r) {
       j++;
       if(j>J) {
           /*    T.N distance;    */
           D=g[I][J]/(float)N;
           return D;
       }
       else i=j-r;
  }
  else {
     if(i<0) i++;
     else {
        if(i>I) i++;
        else {
```

```
                /*  D.P.  eqn.  */
                dist=(R[i]-T[j])*(R[i]-T[j]);
                c1=g[i][j-1]+dist;
                c2=g[i-1][j-1]+2*dist;
                c3=g[i-1][j]+dist;
                temp=(c1<c2) ? c1: c2;
                temp=(temp<c3) ? temp :c3;
                g[i][j]=temp;
                i++;
            }
        }
    }
}
```

EU.C

```
/*  EU.C */

# include <stdio.h>
# include <stdlib.h>
# include <ctype.h>
# include <process.h>
# include <string.h>
# include <math.h>

main(argc,argv)
int argc;
char *argv[];
{
FILE *fp1,*fp2,*fp3;
int i;
float R[12],T[12];
float D=0;
char *str;
if (argc != 3) {
     printf("\n    Usage :\n");
     printf("\n      eu   ref test \n");
     exit(0);
}
fp1=fopen(argv[1],"r");
fp2=fopen(argv[2],"r");
fp3=fopen("result","a");
if ( !fp1 || !fp2 || !fp3) {
     printf(" \n   Unable to Open !\n");
     exit(1);
}
strcpy (str,argv[1]);
strcat (str,"<--->");
strcat (str,argv[2]);
for(i=0;i<12;i++) {
    fscanf(fp1,"%f",&R[i]);
    fscanf(fp2,"%f",&T[i]);
}
fclose(fp1);
fclose(fp2);
for (i=0;i<12;i++) D+=(R[i]-T[i])*(R[i]-T[i]);
D=sqrt((double)D);
fprintf(fp3,"    %-15s  %-15f\n ",str,D);
fclose(fp3);
return 0;
}
```

MAHA.C

```
/*  MAHA.C */
# include <stdio.h>
# include <stdlib.h>
# include <ctype.h>
# include <process.h>
# include <string.h>
# include <math.h>

main(argc,argv)
int argc;
char *argv[];
{
 FILE *fp1,*fp2,*fp3,*fp4;
 register int i,j,k;
 float R[12],T[12],W[12][12],temp[12];
 float sum,D;
 char *str;

 if (argc != 4) {
     printf("\n    Usage :\n");
     printf("\n      maha ref test cov.name\n");
     exit(0);
 }
 fp1=fopen(argv[1],"r");
 fp2=fopen(argv[2],"r");
 fp3=fopen(argv[3],"r");
 fp4=fopen("result","a");
 if ( !fp1 || !fp2 || !fp3) {
    printf(" \n   Unable to Open !\n");
    exit(1);
 }
 strcpy (str,argv[1]);
 strcat (str,"<--->");
 strcat (str,argv[2]);
 for(i=0;i<12;i++) {
     fscanf(fp1,"%f",&R[i]);
     fscanf(fp2,"%f",&T[i]);
 }
 for(i=0;i<12;i++) {
      for(j=0;j<12;j++) {
         fscanf(fp3,"%f",&W[i][j]);
      }
 }
 fclose(fp1);
 fclose(fp2);
 fclose(fp3);
 for(j=0;j<12;j++) temp[j]=(T[j]-R[j]);
 D=0.0;
 for(i=0;i<12;i++) {
     sum=0.0;
     for(j=0;j<12;j++) {
          sum+=W[i][j]*temp[j];
     }
     D+=temp[i]*sum;
 }
 fprintf(fp4,"    %-15s  %-15f\n ",str,fabs((double)D));
 fclose(fp4);
}
```

COV.C

```
/*  COV.C */

# include <stdio.h>
# include <process.h>
# include <math.h>

main(argc,argv)
int argc;
char *argv[];
{
 FILE *fp1,*fp2,*fp3;
 float x[12],y[12],cov[12][12],temp;
 register int i,j;
 if(argc!=3) {
     printf("\n  Usage cov ref1 ref2\n");
     exit(0);
 }
 fp1=fopen(argv[1],"r");
 fp2=fopen(argv[2],"r");
 fp3=fopen("W","w");
 if(!fp1 || !fp2 || !fp3) {
    printf("\n  Unable to open file ...\n");
    exit(1);
 }
 for(i=0;i<12;i++) fscanf(fp1,"%f",&x[i]);
 for(i=0;i<12;i++) fscanf(fp2,"%f",&y[i]);
 for(i=0;i<12;i++) {
    for(j=0;j<12;j++) {
        cov[i][j]=x[i]*y[j];
        temp=cov[i][j];
        fprintf(fp3,"%f  ",temp);
    }
    fprintf(fp3,"\n ");
 }
 fclose(fp1);
 fclose(fp2);
 fclose(fp3);
 return 0;
 }
```

FMI.C

```
# include <stdio.h>
# include <stdlib.h>
# include <alloc.h>
# include <process.h>
# include <math.h>
# include <conio.h>

int main(argc,argv)
int argc;
char *argv[];
{
   FILE *fpr,*fpw;
   register int i,j;
   float temp,*A,*B;
   int N;
   float *invert();
   if(argc!=3){
        clrscr();
        printf("\n\n    FMI - Fast Matrix Inverter\n ");
        printf("\n    (c) NESET ONCUL 1992, Hacettepe U. " );
        printf("\n\n   Usage: " );
        printf("\n          fmi  [File_name] [Dimension]\n  ");
        printf("\n   File_name  is a DOS file  in  which the  '
        printf("\n   matrix   to be  inverted  is  contained  '
        printf("\n   and Dimension is the number of the rows  '
        printf("\n   or columns of it... The inverse of  the  '
        printf("\n   matrix is stored in file INV.FMI...      '
        printf("\n\n   Example:   fmi matrix 15 \n\n" );
        exit(0);
   }

   fpr=fopen(argv[1],"r");
   if(!fpr){
       printf("\n  Unable to open %s...",argv[1]);
       exit(1);
   }

   N=atoi(argv[2]);
   if(N<=0){
       printf(" Wrong Dimension...\n");
       exit(0);
   }

   fpw=fopen("INV.FMI","w");
   if(!fpw){
       printf("\n  Unable to open INV.FMI to write the result...\n"
       exit(1);
   }

   A=(float *)malloc(N*N*sizeof(float));
   if(!A) exit(1);
```

```
    for(i=0;i<N;i++){
       for(j=0;j<N;j++) {
            fscanf(fpr,"%f",&temp);
            *A=temp;
            A++;
       }
    }
    A-=N*N;

    printf("\n");

    B=invert(A,N);
    printf("\n       Inverse Matrix :  \n\n");

    for(i=0;i<N;i++){
       for(j=0;j<N;j++) {
            printf(" %6.2e ",*B);
            fprintf(fpw," %10.5e ",*B);
            B++;
       }
       printf("\n");
       fprintf(fpw,"\n");
    }

    return 0;
}

float *invert(A,N)
float *A;
int N;
{

    register int i,j,k,l;
    float  *B, *C;
    float *I;
    float *t;
    float *p;
    float *dummy,*res;
    float trace();
    float *add();
    float *mult();
    float *mult_by_scalar();
    float *solve_Toeplitz();
    float *powers_of_A;

    dummy=(float *)malloc(N*N*sizeof(float));
    if(!dummy) {
       printf("\n Memory allocation error..");
       exit(1);
    }

    res=(float *)malloc(N*N*sizeof(float));
    if(!res) {
       printf("\n Memory allocation error..");
       exit(1);
    }
```

```
t=(float *)malloc(N*sizeof(float));
if(!t) {
   printf("\n Memory allocation error..");
   exit(1);
}

I=(float *)malloc(N*N*sizeof(float));
if(!I) {
   printf("\n Memory allocation error..");
   exit(1);
}

powers_of_A=(float *)malloc((N+1)*N*N*sizeof(float));
if(!powers_of_A) {
   printf("\n Memory allocation error..");
   exit(1);
}

for(i=0;i<N;i++) {
   for (j=0;j<N;j++){
     I[N*i+j]=0.0;
     if(i==j) I[N*i+j]=1.0;
   }
}

dummy=&I[0];

for(i=0;i<N*N;i++) {
    *powers_of_A=*dummy;
    dummy++;
    powers_of_A++;
}

C=&I[0];

for(i=1;i<=N;i++){
   dummy=mult(C,A,N);
   *t=trace(dummy,N);
   t++;
   for(j=0;j<N*N;j++){
       *powers_of_A=*dummy;
       dummy++;
       powers_of_A++;
   }
   C=dummy-N*N;
}

powers_of_A-=(N+1)*N*N;
t-=N;

p=solve_Toeplitz(&t[0],N);
if(fabs(p[N-1])<1.0e-15) {
   printf("\n Singular Matrix...Aborting...\n");
   exit(0);
}
```

```
   C=mult_by_scalar(-p[N-2]/p[N-1],I,N);
   powers_of_A+=N*N;

   for(i=1;i<=N-2;i++) {
      dummy=mult_by_scalar(-p[N-i-2]/p[N-1],powers_of_A,N);
      res=add(C,dummy,N);
      C=&res[0];
      powers_of_A+=N*N;
   }

   dummy=mult_by_scalar(-1.0/p[N-1],powers_of_A,N);
   res=add(C,dummy,N);

   return &res[0];
}

float trace(A,N)
float *A;
int N;
{
   float tr;
   register int i;

   tr=0.0;
   for(i=0;i<N;i++){
       tr+=A[N*i+i];
   }
   return (tr);
}

float *add(x,y,N)
float *x,*y;
int N;
{
    float *z;
    register int i,j;

    z=(float *)malloc(N*N*sizeof(float));
    if(!z) {
      printf("\n Memory allocation error..");
      exit(1);
   }

    for(i=0;i<N;i++){
       for(j=0;j<N;j++){
           *z=x[N*i+j]+y[N*i+j];
           z++;
       }
    }

    return (z-N*N);
}

float *mult_by_scalar(scalar,x,N)
float scalar,*x;
int N;
{
```

```
    float *z;
    register int i,j;

    z=(float *)malloc(N*N*sizeof(float));
    if(!z) {
      printf("\n Memory allocation error..");
      exit(1);
    }

    for(i=0;i<N;i++){
       for(j=0;j<N;j++){
           *z=x[N*i+j]*scalar;
           z++;
       }
    }
    return (z-N*N);
}

float *mult(x,y,N)
float *x,*y;
int N;

{
    float *z;
    float sum;
    register int i,j,k;

    z=(float *)malloc(N*N*sizeof(float));
    if(!z) {
       printf(" \n Memory allocation failure....\n");
       exit(1);
    }
    for(i=0;i<N;i++){
          for(j=0;j<N;j++){
               sum=0.0;
               for(k=0;k<N;k++) sum+=x[i*N+k]*y[k*N+j];
               *z=sum;
               z++;
          }
    }
    return (z-N*N);
}

float *solve_Toeplitz(t,N)
float *t;
int N;
{
   float *p;
   float sum;
   register int i,j;

   p=(float *)malloc(N*sizeof(float));
   if(!p) {
       printf("\n Memory allocation error..");
       exit(1);
    }

   p[0]=0.0;
```

```
for (i=0;i<N;i++) {
   sum=0.0;
   for(j=0;j<i;j++)  sum+=p[j]*t[i-j-1];
   p[i]= - (t[i]+sum) / ((float)(i+1));
}
return &p[0];
```

VIEW.C

```
/*  VIEW.C */

# include   <stdio.h>
# include   <stdlib.h>
# include   <process.h>
# include   <ctype.h>
# include   <graphics.h>
# include   <alloc.h>
# include   <conio.h>


main(argc,argv)
int       argc;
char      *argv[];

{
FILE    *fp;
int     g_driver, g_mode;
register int     i,j;
int dum;
int     *y,data_max,n_frames;
int      fr_size=580;
char     cho,c,*str,*fr;

struct points{
    int   x,y;
} old,new;


if(argc!=2){
     printf("usage:    view [source_file]\n");
     exit(1);
}

printf("\n Specify Type of Data File  \n\n [A]scii or [I]nteger File

for(;;){
     cho=tolower( getch() );
     if( cho=='a' || cho=='i')break;
     else printf("\n Try Again ... (Press  I or A  )");
}
printf("\n\n Reading File: %s ...PLEASE WAIT...",argv[1]);

y=(int *)malloc(10000*sizeof(int));
if(!y){
     printf("memory request failed\n");
     exit(1);
}

fp=fopen(argv[1],"r");
if(!fp){
     printf("unable to open file : %s\n",argv[1]);
     exit(1);
```

```
j=0;
      if( cho=='a'){
        while(!feof(fp)){
          fscanf(fp,"%c",&c);
          y[j]=(int)c;
          j++;
        }
      }
      if( cho=='i'){
       while(!feof(fp)){
          fscanf(fp,"%d",&dum);
          y[j]=dum;
           j++;
       }
      }

j/=fr_size;
n_frames=j;

itoa(n_frames,fr,10);
for(j=0;j<n_frames;j++){
        for(i=0;i<fr_size;i++){
              y[j*fr_size+i]=-y[j*fr_size+i]+240;
        }
}


/*      initialize graph mode    */
detectgraph(&g_driver, &g_mode);
initgraph(&g_driver, &g_mode, "..\\bgi");
cleardevice();

settextstyle(TRIPLEX_FONT,HORIZ_DIR,2);
outtextxy(20,30,argv[1]);
settextstyle(DEFAULT_FONT,HORIZ_DIR,1);

outtextxy(0,110," 2.5v");
line(45,113,49,113);
outtextxy(0,176," 1.2v");
line(45,179,49,179);
outtextxy(0,237,"  0 v");
line(45,240,49,240);
outtextxy(0,298,"-1.2v");
line(45,301,49,301);
outtextxy(0,364,"-2.5v");
line(45,367,49,367);
line(50,69,50,410);
line(50,410,639,410);
line(639,410,639,69);
line(639,69,50,69);
old.x=0;
old.y=240;
j=0;
outtextxy(100,10," SPEECH DATA VIEWING PROG. (c) NESET ONCUL, H.U.,

outtextxy(160,30," B: Backward                  # smpls/fr    = 580 ");
outtextxy(160,45," F: Forward    X: eXit        TOTAL FRAMES = ");
outtextxy(505,45,fr);
```

```
  while( j>=0 && j<n_frames ){
       itoa(j+1,str,10);
       setviewport(51,70,638,409,1);
       outtextxy(2,10,"  FRAME #:");
       setbkcolor(BLACK);
       setcolor(WHITE);
       settextstyle(TRIPLEX_FONT,HORIZ_DIR,2);
       if(j==0)    outtextxy(100,0,"FIRST");
       else if(j==n_frames-1) outtextxy(100,0,"LAST");
            else outtextxy(100,0,str);
       settextstyle(DEFAULT_FONT,HORIZ_DIR,1);

       setlinestyle(DASHED_LINE,1,1);
       line(0,170,fr_size+3,170);
       setlinestyle(SOLID_LINE,1,1);
       for(i=0;i<fr_size;i++){
           new.x=i;
           new.y=y[j*fr_size+i];
           line(old.x,old.y-70,new.x,new.y-70);
           old.x=new.x;
           old.y=new.y;
           if(i==fr_size-1)  old.x=new.x-fr_size;
       }
/*
       c=toupper(getch());
       clearviewport();
       if(c=='B')j--;
       if(c=='F')j++;
       if(c=='B'&&j==-1)j++;
       if(c=='F'&&j==n_frames)j--;
       if(c=='X')break;
 */
       for(;;){
           c=toupper(getch());
           if(c=='B'|| c=='F'|| c=='X') break;
       }
       clearviewport();
       if(c=='B')j--;
       if(c=='F')j++;
       if(c=='B'&&j==-1)j++;
       if(c=='F'&&j==n_frames)j--;
       if(c=='X')break;
  }
  closegraph();
  return 0;
}
```