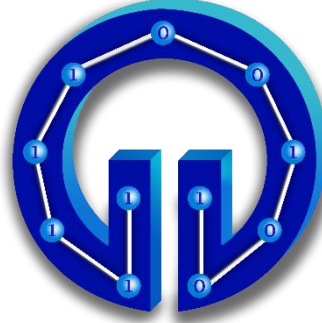


**KARADENİZ TEKNİK ÜNİVERSİTESİ
MÜHENDİSLİK FAKÜLTESİ
BİLGİSAYAR MÜHENDİSLİĞİ BÖLÜMÜ**



**ÇOK ÖZELLİKLİ ALGORİTMA İLE COPY-MOVE SES
SAHTECİLİĞİ TESPİTİ**

BİTİRME PROJESİ

**İsmail KOL
Metehan COŞKUN
Enes KARALI
Ozan PEKTEZEL**

**2020-2021 BAHAR DÖNEMİ
KARADENİZ TEKNİK ÜNİVERSİTESİ
MÜHENDİSLİK FAKÜLTESİ
BİLGİSAYAR MÜHENDİSLİĞİ BÖLÜMÜ**

**ÇOK ÖZELLİKLİ ALGORİTMA İLE COPY-MOVE SES
SAHTECİLİĞİ TESPİTİ**

BİTİRME PROJESİ

**İsmail KOL
Metehan COŞKUN
Enes KARALI
Ozan PEKTEZEL**

2020-2021 BAHAR DÖNEMİ



IEEE Etik Kuralları IEEE Code of Ethics



Mesleğime karşı şahsi sorumluluğumu kabul ederek, hizmet ettiğim toplumlara ve üyelerine en yüksek etik ve mesleki davranışta bulunmaya söz verdiğimi ve aşağıdaki etik kurallarını kabul ettiğimi ifade ederim:

1. Kamu güvenliği, sağlığı ve refahı ile uyumlu kararlar vermenin sorumluluğunu kabul etmek ve kamu veya çevreyi tehdit edebilecek faktörleri derhal açıklamak;
2. Mümkün olabilecek çıkar çatışması, ister gerçekten var olması isterse sadece algı olması, durumlarından kaçınmak. Çıkar çatışması olması durumunda, etkilenen taraflara durumu bildirmek;
3. Mevcut verilere dayalı tahminlerde ve fikir beyan etmelerde gerçekçi ve dürüst olmak;
4. Her türlü rüşveti reddetmek;
5. Mütenasip uygulamalarını ve muhtemel sonuçlarını gözeterek teknoloji anlayışını geliştirmek;
6. Teknik yeterliliklerimizi sürdürmek ve geliştirmek, yeterli eğitim veya tecrübe olması veya işin zorluk sınırları ifade edilmesi durumunda ancak başkaları için teknolojik sorumlulukları üstlenmek;
7. Teknik bir çalışma hakkında yansız bir eleştiri için uğraşmak, eleştiriyi kabul etmek ve eleştiriyi yapmak; hatları kabul etmek ve düzeltmek; diğer katkı sunanların emeklerini ifade etmek;
8. Bütün kişilere adilane davranmak; ırk, din, cinsiyet, yaş, milliyet, cinsi tercih, cinsiyet kimliği, veya cinsiyet ifadesi üzerinden ayrımcılık yapma durumuna girişmemek;
9. Yanlış veya kötü amaçlı eylemler sonucu kimsenin yaralanması, mülklerinin zarar görmesi, itibarlarının veya istihdamlarının zedelenmesi durumlarının oluşmasından kaçınmak;
10. Meslektaşlara ve yardımcı personele mesleki gelişimlerinde yardımcı olmak ve onları desteklemek.

IEEE Yönetim Kurulu tarafından Ağustos 1990'da onaylanmıştır.

ÖNSÖZ

Bu proje, “Copy-move detection of digital audio based on multi-feature decision” adı altında yayınlanmış yöntemin gerçekleştirilmesidir.

“Ses sahteciliğinin tespiti” adlı bu çalışma, Karadeniz Teknik Üniversitesi, Bilgisayar Mühendisliği Bölümü’nde bahar dönemi bitirme projesi olarak hazırlanmıştır. Bu proje ile adli davalarda sıklıkla kullanılan ses sahteciliğini ve bu sahteciliğin nasıl tespit edileceği konularında araştırmalarda bulunulmuştur.

Bu projenin yapımında bize yardımcı olan GÜZİN ULUTAŞ hocamıza saygılarımızı sunup teşekkür ediyoruz. Ayrıca her zaman yanımızda olan ve desteklerini, sevgilerini bizden hiçbir zaman esirgemeyen ailelerimize her şey için çok teşekkür ediyoruz.

İsmail KOL
Metehan COŞKUN
Enes KARALI
Ozan PEKTEZEL

Trabzon 2021

İÇİNDEKİLER

	Sayfa No
IEEE ETİK KURALLARI	II
ÖNSÖZ	III
İÇİNDEKİLER	IV
ÖZET	V
1. GENEL BİLGİLER	1
1.1. Giriş	1
1.2. Ses Sahteciliği	1
1.3. Copy-Move	2
1.4. Çok Özellikli Tespit Yöntemi	3
2. YAPILAN ÇALIŞMALAR	5
2.1. Ön İşleme	5
2.1.1. Ses Çerçevenmesi	5
2.1.2. Ses Pencerenmesi	6
2.1.3. Ön İşleme Python Kodu	7
2.2. Ses Etkinliği Algılama	8
2.3. Özellik Çıkarma	8
2.3.1. Gammatone Özelliği	8
2.3.1.1. Gammatone Özelliği Python Kodu	9
2.3.2. MFCC Özelliği	10
2.3.2.1. DCT(Ayrık Kosinüs Dönüşümü)	11
2.3.2.2. MFCC Python Kodu	12
2.3.2.3. Mel Spektrumu	12
2.3.3. Pitch Özelliği	13
2.3.3.1. Pitch Özelliği Python Kodu	14
2.3.4. DFT Katsayıları (Ayrık Fourier Dönüşümü)	14
2.3.4.1. DFT Python Kodu	15
2.3.5. Ortalama Karakök(RMS)	16
2.3.5.1. RSM Python Kodu	16
2.3.6. Kroma Özelliği	17
2.3.6.1. Kroma Python Kodu	17
2.4. Benzerlik Karşılaştırmaları	18
2.4.1. PCCs (Pearson Korelasyon Katsayısı)	18
2.4.1.1. PCC Python Kodu	18
2.4.2. AD (Ortalama Fark)	22
2.5. C4.5 Tabanlı Kararlı Ağacı	22
3. SONUÇLAR	24
3.1. Pitch Özelliği Benzerlik Sonucu	24
3.2. Kroma Özelliği Benzerlik Sonucu	24
3.3. Gammatone Özelliği Benzerlik Sonucu	25
3.4. DFT Özelliği Benzerlik Sonucu	26
3.5. C4.5 Karar Ağacı Sonucu	26
4. ÖNERİLER	28
5. KAYNAKLAR	29
6.EKLER	30
STANDARTLAR ve KISITLAR FORMU	31

ÖZET

Copy-Move yöntemi ses sahteciliğinde kullanılan yaygın yöntemlerden biridir. Ayrıca kolay uygulanması ve gizlilik avantajları nedeniyle tespit edilmesi de bir hayli zordur. Bu projede Copy-Move sahteciliğinin tespiti için çok özellikli bir algoritma üzerinde durulmuştur. Gammatone filtrelenmesi, MFFC, perde dizileri, RMS, Kroma ve DFT (Ayrık Fourier Dönüşümü) olmak üzere altı özellik sese ayrı ayrı uygulanır ve sonuçlar birleştirilir. Sonucun güvenilirliğini arttırmak için Pearson Korelasyon Katsayısı (PCC) ve ortalama fark (AD) yöntemleri kullanılır.

Diğer tek özellikli tespit yöntemleriyle karşılaştırıldığında bu yöntem, Copy-Move tespiti açısından daha güvenilir sonuçlar vermektedir.

1. GENEL BİLGİLER

1.1. Giriş

Ses, canlıların işitme organları tarafından algılanabilen periyodik basınç değişimleridir.

Fiziksel boyutta ses, katı, sıvı veya gaz ortamlarında oluşan basit bir mekanik düzensizliktir. Bir maddedeki moleküllerin titreşmesi sonucunda oluşur. Ses bir enerji türüdür, titreşimle oluşur ve titreşimi enerjiye dönüştürür. Atmosferde titreşen bir nesne, çevresindeki hava moleküllerini hareket ettirir. Hava moleküllerinin dairesel hareketi, kulakların hissedebileceği basınç değişimi dalgaları yaratır. Farklı nesnelerden farklı sesler duymamızın nedeni ise ses dalgalarının frekansındaki farklılıklardır.

Her canlının çıkardığı ses yani frekans değerleri kendine özgüdür. Bulunan ortamların da frekans değerleri vardır ve biz buna dip ses veya gürültü deriz. Canlı olmasalar da bu dip sesler de birbirinden farklıdır. Hatta bir canlı, ortamını değiştirmeden art arda aynı sesleri çıkardığında bunların frekans değerleri çok benzese de birbirinden farklıdır. Kısaca kayıtlardaki frekanslar o ana özgüdür demek daha doğru olur. Ses kayıtlarının incelenmesinin temeli buna dayanır. Tabi ki yapılan kurcalama işleminin anlaşılmasını zorlaştırmak için filtreler uygulanmaktadır. Bunlar sesi dinlediğinizde anlaşılması neredeyse imkansızdır. Bu nedenle sahte sesi tespit etmek için çeşitli yazılımlar geliştirilmektedir.

Adli Konuşmacı tanıma incelemelerinde; asılsız ihbarlar, terör, kaçakçılık, tehdit, dolandırıcılık ve benzeri olaylara konu olmuş ses kayıtları, konuşmacıları tarafından değiştirilmeye çalışılmaları nedeniyle adli ses incelemelerine konu olmaktadır. Suça konu olan ve değiştirilmeye çalışılan konuşma seslerinin asıl sahibinin kim olduğu, itham edilen konuşmacıya ait olup olmadığı veya değiştirilerek ses sahteciliği yöntemleri ile oluşturulup oluşturulmadığı gibi problemler incelemelerin sağlıklı bir şekilde yapılabilmesine etki etmektedir.

Ses kurcalanmasının en çok bilinen ve tercih edilen yöntemi Copy-Move ses sahteciliği olduğu için bu yöntemin tespiti hakkında birçok çalışmalar yapılmış ve algoritmalar geliştirilmiştir. Bu projede en sağlıklı sonuç vermesi açısından çok özellikli bir algoritma üzerinde durulmuştur.

1.2. Ses Sahteciliği

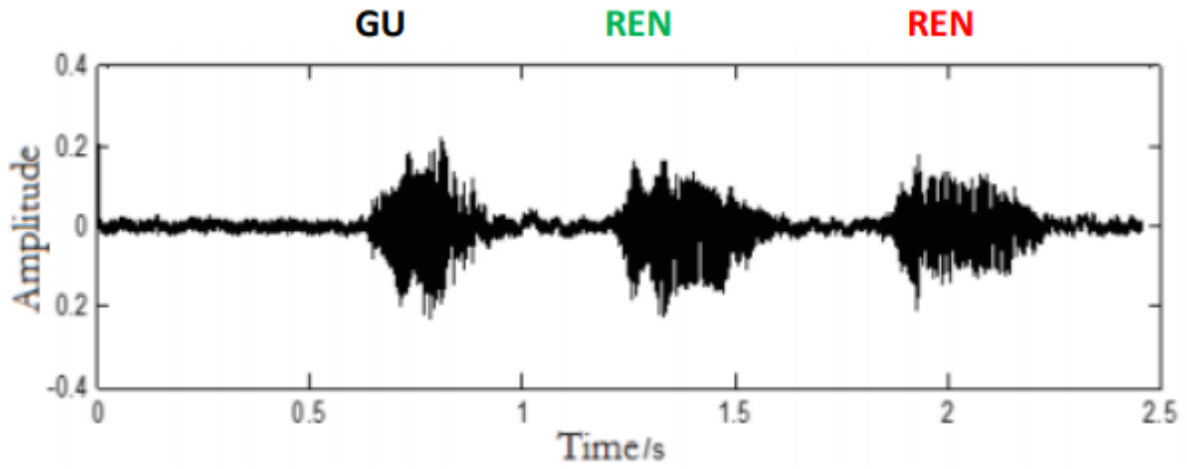
Ses sahteciliği (ses değiştirilmesi) konuşmacı tanıma incelemeleri kapsamında karşılaşılan durumlardan biridir. Konuşmacılar kimliklerini gizlemek amacıyla farklı şekillerdeki ses sahteciliği yöntemlerini kullanmaktadır. Bu yöntemlere örnek vermek gerekirse ses genliğini arttırarak konuşmak, ses tonunu inceltmek ve burnunu kapatarak konuşma gibi yöntemler kullanılmaktadır. Bunların haricinde ise ses kayıtları üzerinde de oynama yapılmaktadır. Örneğin sesin bir bölümünün kopyalanıp aynı ses üzerinde başka bir yere taşınması (Copy-Move), sesin bir bölümünün kopyalanıp başka bir ses kaydı üzerinde bir yere taşınması (Audio Splicing) gibi sıklıkla kullanılan yöntemlere de başvurulabilmektedir.

Konuşmacıların söylemlerini değiştirmeleri, özellikle ses genliği, rengi(tını), tonu, konuşma hızı ve şive gibi birçok parametreleri etkilemektedir.

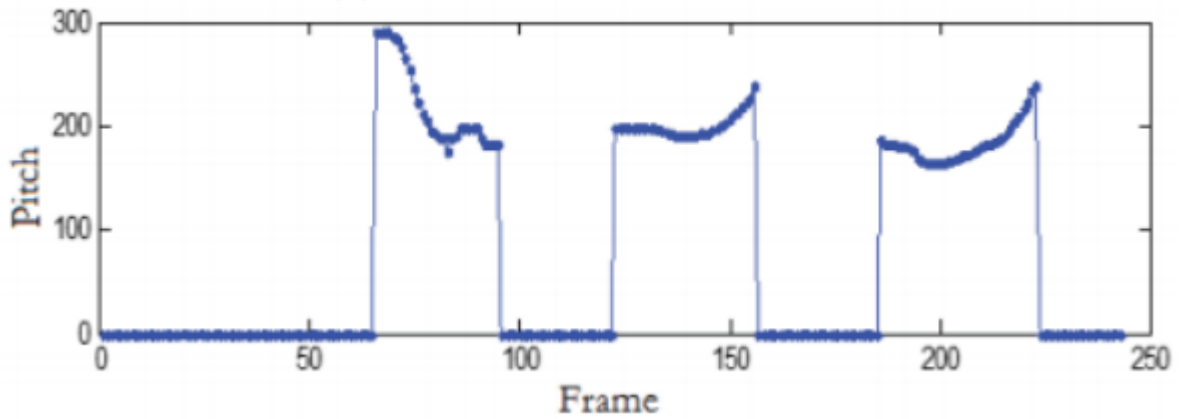
Son birkaç yılda ses sahteciliğini tespit etmek için geliştirilen teknolojiler hızla gelişti. Temel frekans algılama algoritması (PDA), çok özellikli algılama algoritması vb. gibi birçok algoritma geliştirilmiştir.

1.3. Copy-Move

Hem az maliyetli hem de algılama zorluğu nedeniyle en yaygın olarak kullanılan kurcalama yöntemlerinden birisi olan Copy-Move yöntemi sesin bir bölümünün aynı ses üzerinde başka bir yere kopyalanmasıdır. Bu teknikle yapılan saldırıda homolog veya heterolog olabilir. Temel de ses kaydındaki herhangi bir yerin kopyalanıp başka yerlere eklenmesidir. Örnek vermek gerekirse Şekil 1 de "Gu Ren Ren" olarak söylenip kaydedilmiş ses dosyasını görmekteyiz.

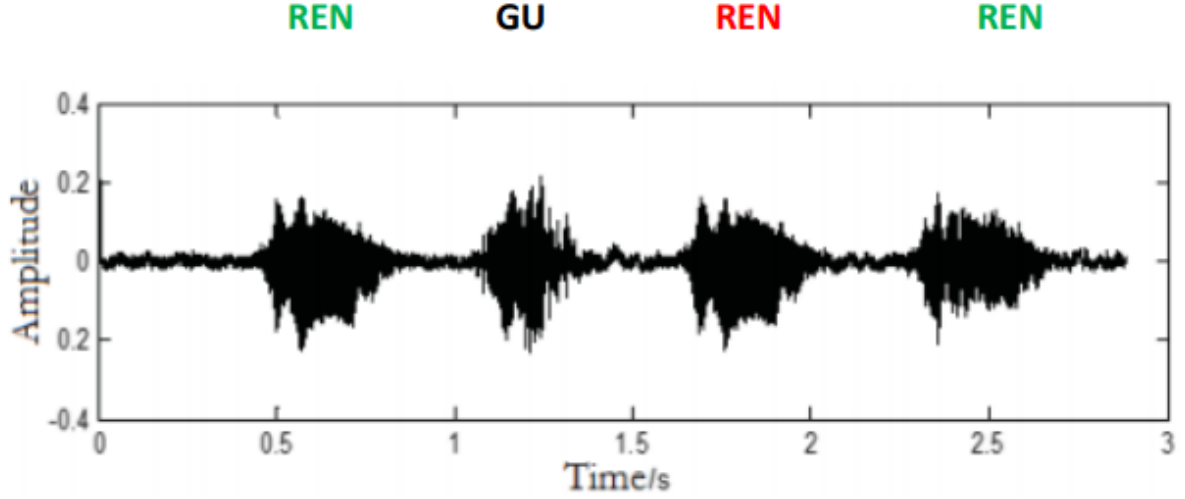


Şekil 1. Orijinal sesin dalga grafiği[5]

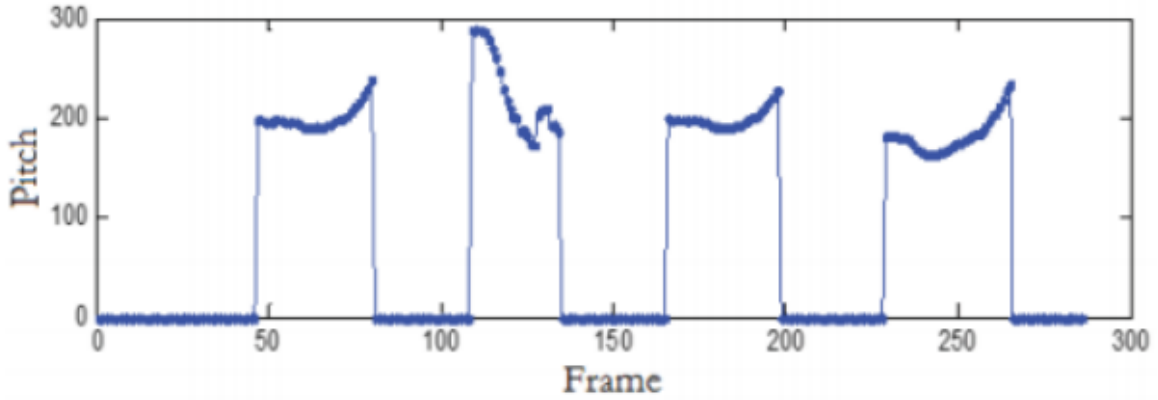


Şekil 2. Orijinal sesin perde dizisi[5]

Burada dikkat etmemiz gereken, söylenen aynı kelimelerin birbirine çok benzese de farklılıklarının olduğunu görmekteyiz. Şimdi bu sesin "REN" hecesini kopyalayıp ses dizisinin başına ekleyelim.



Şekil 3. Copy-Move kullanılarak oluşturulmuş sesin dalga grafiği[5]



Şekil 4. Copy-Move kullanılarak oluşturulan perde (Pitch) dizisi[5]

Görüldüğü üzere sesleri kopyalayıp yapıştırdığımızda bile dip seslerin uyumsuzluğundan dolayı küçük farklar oluşabilir. Bunları oluşturulan perde dizisinde daha rahat görülmektedir.

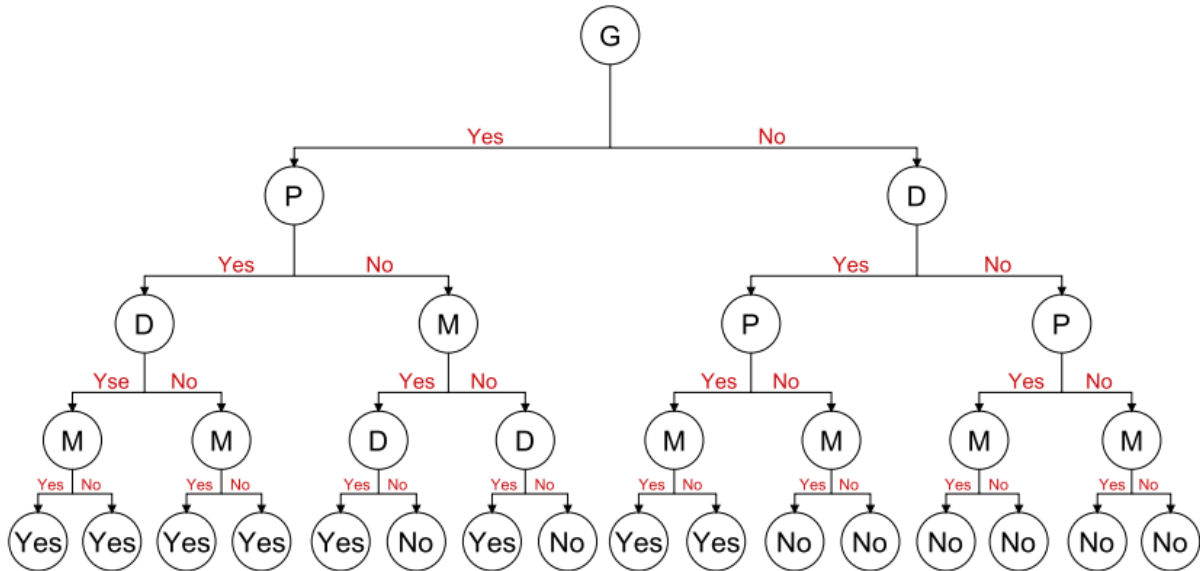
Copy-Move kullanılarak üretilen sahte sesin tespiti için çok özellikli bir algoritma kullanılarak bu özellikleri sonuçları birleştirilerek Copy-Move tespiti yapılabilir. Bu özellikler belirtildiği gibi Gammatone, MCCF, Perde dizileri ve Ayrık Fourier Dönüşümü (DFT) dir.

1.4. Çok Özellikli Tespit Yöntemi

Bu yöntemde ilk olarak giriş sesine ön işlem uygulanmaktadır. Buradaki amaç ses üzerindeki gürültüyü ve pencerelemeyi ortadan kaldırmaktır. Ardından ses etkinliği algılanması yapılır. Algılamanın doğruluğunun Copy-Move tespiti üzerinde doğrudan bir etkisi vardır. Bu nedenle bu algılama en sağlıklı biçimde yapılmalıdır.

İkinci olarak dört tür ana özellik ses üzerinden çıkarılır ve bunlar arasında benzerlik hesaplaması yapılır.

Son olarak ise Şekil 5’te görülen karar ağacı kullanılır. Bu karar tüm yöntemlerle tam anlamıyla uyum sağladığı için Copy-Move tespiti açısından başarılı sonuçlar vermektedir.



Şekil 5. Karar ağacı[1]

2. YAPILAN ÇALIŞMALAR

2.1. Ön İşleme

Ön işleme, orijinal sesin eğilim terimini ortadan kaldırmayı, çerçevelemeyi ve pencerelemeyi içerir. Veri toplama sürecinde, sistem herhangi bir nedenle zaman serilerinde doğrusal veya yavaş değişen bir eğilim üretecektir. Bu yöntemde oluşan sorunu ortadan kaldırmak için en küçük kareler metodu kullanılır.

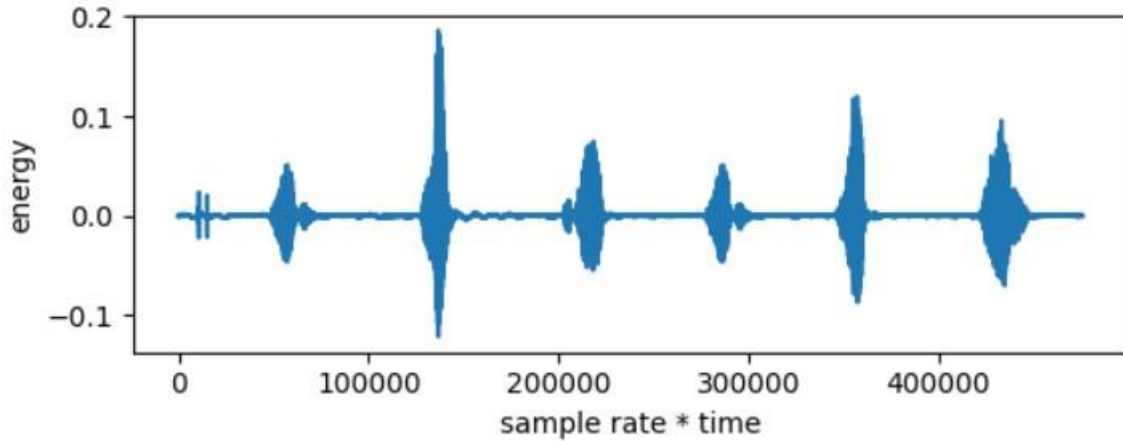
Bir ses sinyalinin örnek verilerinin $\{x_k\}$ ($k=1, 2, 3, n$) olduğu varsayalım. Burada n örnek boyutudur çünkü örnek veriler birer sabit süredir. Daha sonra (1) kullanılarak sesteki eğilim bulunur.

$$\hat{x}_k = a_0 + a_1 k + a_2 k^2 + \dots + a_m k^m = \sum_{j=0}^m a_j k^j \quad (1)$$

k : Örnek nokta sayısı

m : Sesteki eğilim terimi

Çerçeveleme ve pencereleme ses işleminin ortak süreçleridir. Yani bütün algoritmalarda bu iki özellik kullanılmaktadır.

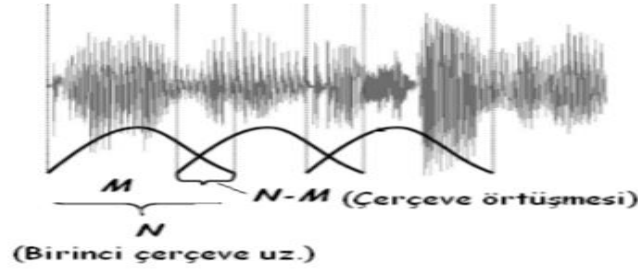


Şekil 7. Kullanılan sesin enerji spektrumu

2.1.1. Ses Çerçevenmesi

Ses üretim organlarının sözcüklere bağlı olarak yer değiştirmesinden dolayı konuşma işareti de sürekli olarak değişir. Konuşma işareti, parametrelerin sabit kaldığı kabul edildiği çerçeve olarak adlandırılan küçük parçalara ayrılmalıdır. Çünkü tüm işaret boyunca FFT hesaplanırsa, farklı fonemlere ait spektral bilgilerin tutulmasında kayıplar oluşur. Tüm işaretin FFT'sini almak yerine çerçevenin FFT'si hesaplanır. Çerçeve uzunluğu 10-30 ms arasında değişir. Her bir çerçeveye örtüşme uygulanır.

Konuşma örneğinden ortalaması çıkartıldıktan sonra, konuşma değişimlerine karşı sabit kabul edilebilecek parçalar şu şekilde ifade edilir. Konuşma işareti N örnek uzunluğunda konuşma parçalarına bölünür. İlk çerçeve N örnekten oluşurken sonraki çerçeve ilk çerçeveden M örnek sonra başlar ve böylece $N-M$ örnek örtüşür. Şekil 8 'de bir ses çerçeveleme işlemi gösterilmiştir.

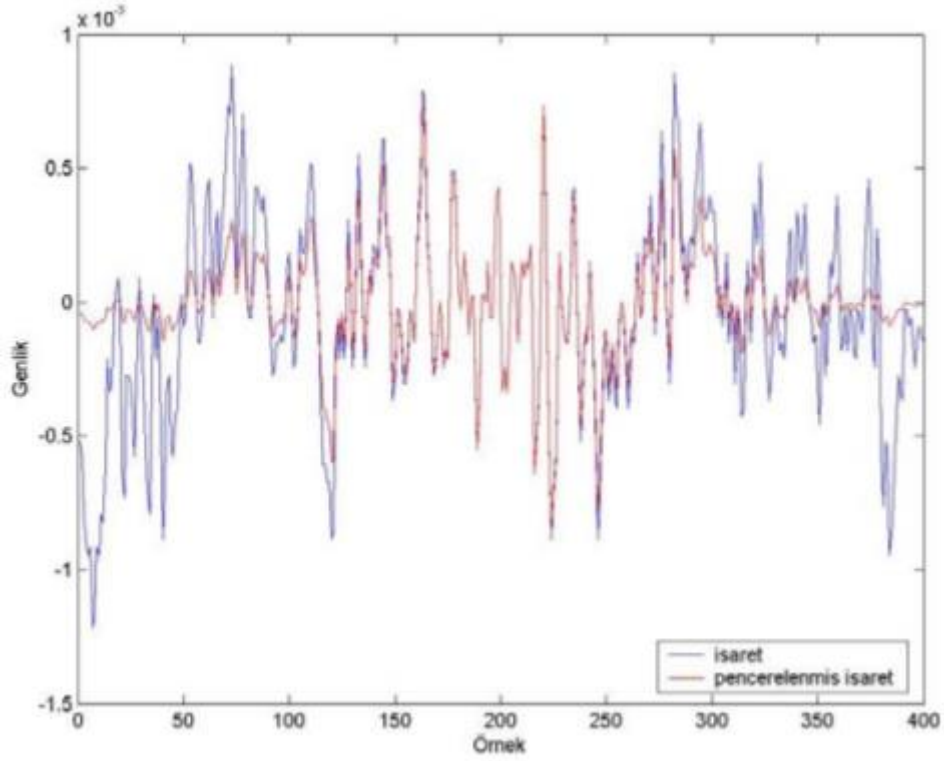


Şekil 8. Ses çerçevenmesi[3]

2.1.2. Ses Pencerenmesi

MFCC katsayılarını elde etmek için ikinci yapılan işlem pencerelemedir. Pencerelemenin amacı çerçeveleme işlemi sonucunda oluşan spektral etkilerin azaltılmasıdır. Pencereleme ile çerçevelerde süresizliğin önüne geçilir. Bu sayede sesin orta bölgeleri güçlendirilirken kenar bölgeleri zayıflatılır. Hamming pencereleme fonksiyonu (2) sıklıkla kullanılan bir ses pencereleme yöntemidir.

Şekil 9'de 25 ms'lik konuşma çerçevesi ve bu konuşma çerçevesinin Hamming pencereleme uygulandıktan sonraki durumu görülmektedir. Şekil 9'dan görüleceği üzere Hamming pencerelenen bir çerçevelik konuşma parçası, sıfıra yakın bir değer ile başlayıp çerçeve süresinin yaklaşık 1/3'ünden itibaren çerçevelenen işaretin değerlerini takip etmekte ve sıfıra yakın bir değer ile sonlanmaktadır. Bu şekilde çerçevelerin sonunda oluşacak ani değişimlerin önüne geçilir.



Şekil 9. Çerçeveleme ve Hamming penceresi uygulanmış ses[3]

$$w[k+1] = 0.54 - 0.46 \cos\left(2\pi \frac{k}{N-1}\right) \quad k = 0, \dots, N-1 \quad (2)$$

2.1.3. Ön İşleme Python Kodu

```
def enframe(sinyal, nw, inc):
    sinyal_length=len(sinyal)
    if sinyal_length<=nw
        nf=1
    else:
        nf=int(np.ceil((1.0*sinyal_length-nw+inc)/inc))
        pad_length=int((nf-1)inc+nw)
        zeros=np.zeros((pad_length-sinyal_length,))
        pad_signal=np.concatenate((sinyal,zeros))

    indices=np.tile(np.arange(0,nw),(nf,1))+np.tile(np.arange(0,nfinc,inc),(nw,1)).T
    indices=np.array(indices,dtype=np.int32)
    frames=pad_signal[indices]

    window= signal.windows.hann(nw+2)[1:-1]

    return frames*window
```

2.2. Ses Etkinliđi Algılama

Ses ön işlemede ses etkinliđi algılama çok önemlidir ses işlemenin doğruluđu açısından doğrudan bir etkiye sahiptir. Gürültüsüz sesi kullandığımızda sesin etkinliđini algılamada kısa süreli ortalama enerji kullanabiliriz. Ancak genellikle sesi gürültülü olarak algılarız. Bu nedenle ses etkinliđi algılamadaki temel sorun sesin gürültüden nasıl ayrıştırılacağıdır. Bu projede de bu sorunu ortadan kaldırmak için çift eşikli ses etkinliđi algılama yöntemi kullanılmıştır. Sesin spektrumu gürültünün spektrumundan farklıdır. Sesin enerjisi frekans üzerinde dağılırken büyük bir deđişim gösterir ancak gürültünün enerjisi frekans üzerinde eşit olarak dağılır. Bu yüzden ses etkinliđi algılama da ve sesin gürültüden ayırt edilmesinde bu özellik kullanılabilir.

Çift eşikli ses algılama yöntemi ses segmentlerini algılamada iyi bir yoldur. Bu yöntemde ilk olarak algılamayı yapmak için daha yüksek bir eşik deđeri seçmektir (T2). Sesin deđeri T2'den büyükse segmentin sese ait olduđu sonucuna varılır. İkinci olarak daha düşük bir eşik deđeri seçmemiz gerekir (T1). İlk adımda bulunan segmentler sayesinde ortadan hem sol hem de sağa doğru T1'den daha düşük olan noktalar bulunur.

T1 ve T2'yi bulmak için ses frekansının varyansı hesaplanır. Sesin $X(n)$ olduđu varsayılırsa çerçeveleme ve pencerelemeden sonra her çerçeveye ayrık fourier dönüşümü (DFT) uygulanır. Daha sonra X_i işleme sokulur, burada i çerçeve sayısıdır. i . çerçevenin kısa süreli enerjisi E_i ve varyansı D_i (3) ve (4) ile hesaplanır.

$$E_i = \frac{1}{N} \sum_{k=0}^{N-1} |X_i(k)| \quad (3)$$

$$D_i = \frac{1}{N-1} \sum_{k=0}^{N-1} [|X_i(k)| - E_i]^2 \quad (4)$$

N : Çerçeve boyutu

2.3. Özellik Çıkarma

Özellik çıkarma Copy-Move tespitinde hayati bir süreçtir. Başarılı bir özellik çıkarma yüksek doğrulukla sonuçlanır. Bu bölümde Copy-Move algılamada kullanılan dört özellik ele alınmıştır.

2.3.1. Gammatone Özelliđi

Bir gammatone filtresi, bir gama dağılımının ve sinüzoidal tonun ürünü olan bir darbe tepkisi ile tanımlanan doğrusal bir filtredir. İşitme sisteminde yaygın olarak kullanılan bir işitsel filtre modelidir. Gammatone filtresi (5)'le hesaplanır.

$$g(t) = at^{n-1}e^{-2\pi bt} \cos(2\pi ft + \phi) \quad (5)$$

Gammatone özelliğinin hesaplama süreci, MFCC'lerin çıkarma şemasına benzerdir. Aslında gammatone özelliği (6)'da gösterilen gammatone filtre bankası $H(f)$ 'ye dayanmaktadır.

$$H(f) = \frac{c}{2}(n-1)!(2\pi b)^{-n}[P(f) + P^*(f)] \quad (6)$$

Burada c sabit orantı katsayısıdır, b bant genişliği katsayısıdır ve (7) ile hesaplanır.

$$b = \frac{[(n-1)!]^2}{(2n-2)!2^{2-2n}\pi}ERB \quad (7)$$

$$ERB = 24.7 \times (0.00437f + 1) \quad (8)$$

2.3.1.1. Gammatone Özelliği Python Kodu

```
gfcc=Gtgram(signal.data,signal.fs,0.030,0.015,channels=12,f_min=150)

gfcc=gfcc*1000

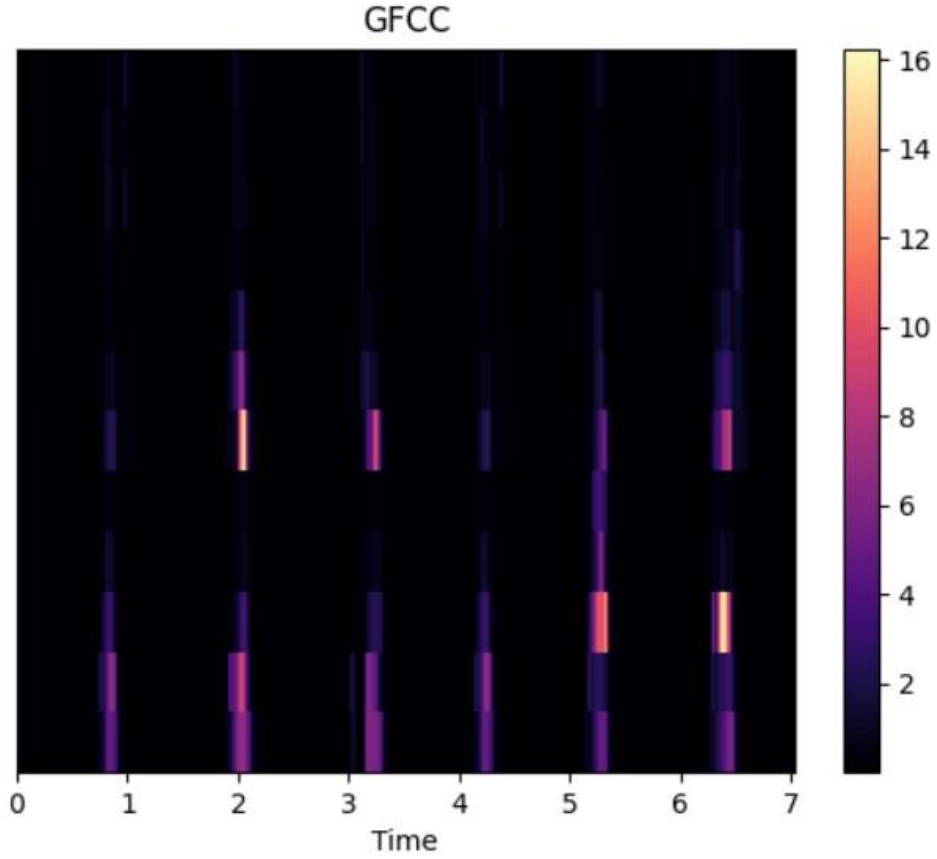
gfccDeger=np.sum(gfcc,axis=0)

gfccKontrol=np.zeros(len(gfccDeger))

for i in range(len(gfccDeger)):

    if (gfccDeger[i]>1):
        gfccKontrol[i]=gfccDeger[i]
    else :
        gfccKontrol[i]=0

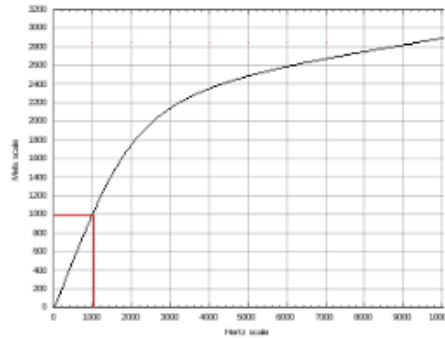
fig, ax = plt.subplots()
img=librosa.display.specshow(gfcc, fmin=150,sr=signal.fs, fmax=8000,
ax=ax,x_axis='time')
fig.colorbar(img, ax=ax)
ax.set(title='GFCC')
plt.show()
```

Şekil 10. Gammatone kod çıktısı

2.3.2. MFCC Özelliği

Konuşma tanıma ve konuşmacı tanıma açısından MFCC olarak da adlandırılan Mel-frekans kepsral katsayıları önemli bir rol oynamaktadır. Mel frekansı kepsral katsayıları, konuşma tanıma sistemlerinde en çok kullanılan özelliklerden biri haline gelmiştir. Kepsral katsayılar doğrusal ölçekli olmakla beraber insan kulağı 1kHz'in altındaki frekansları doğrusal ölçekli, 1kHz'in üstündeki frekansları logaritmik ölçekli olarak duymaktadır. MFCC'nin kullanım amacı kepsral katsayıların, insan işitme sistemiyle uyumlandırılmasıdır. Şekil 11'de, 1.000Hz'in altında doğrusal aralıklı, 1.000Hz'in üstünde logaritmik aralıklı frekans bantlarından oluşan Mel ölçeği yer almaktadır.

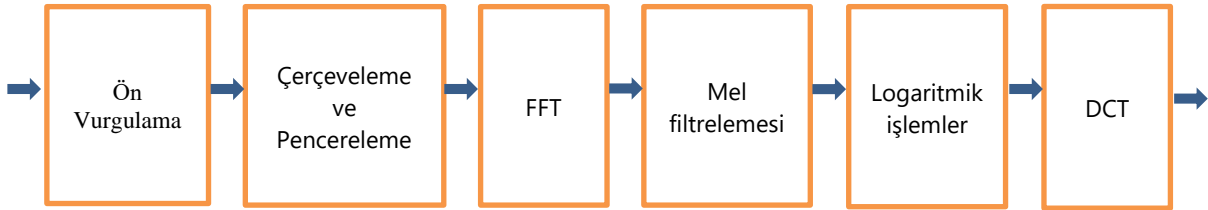


Şekil 11. MFCC frekansı grafiği[3]

Referans noktası olarak seçilen 1kHz'lık ses 1.000 mel olarak ifade edilir. Hz cinsinden verilen frekansı mel olarak ifade etmek için (9) formülü kullanılır.

$$mel(f) = 2595 \cdot \log_{10} \left(1 + \frac{f}{700} \right) \quad (9)$$

Mel spektrumunu ifade etmek için kullanılan yaklaşımlardan biri, her bir mel frekans bileşeni için bir filtre kullanmaktır. Bu filtre bankası üçgen bant geçiren frekans karakteristiğine sahiptir ve bant genişliği sabit mel frekans aralıklarıyla belirlenir. İç içe geçmiş üçgen filtrelerden her birinin kesim frekansı, komşu iki filtre tarafından belirlenir. Filtre bankası 13 doğrusal aralıklı (133Hz ile 1kHz arası), 27 logaritmik aralıklı (frekanslar 1.0711703 çarpanı ile aralıklandırılır) filtreden oluşur. Kulağın duyma hassasiyetini modellemek amacıyla bu yapı kullanılmıştır. Mel filtre bankasında istenilen frekans aralığındaki (örnekleme frekansına bağlıdır) frekans değerlerine karşılık gelen ağırlık değerlerinin hesaplanmasında üçgen benzerliğinden yararlanılır.



Şekil 12. MFCC özellik çıkarma süreci

Şekil 12, MFCC'lerin özellik çıkarma sürecini göstermektedir. İlk olarak sinyalin spektrumunu düz hale getirmek için ön vurgulama işlemi yapılır. Ardından daha önce bahsedilen çerçeveleme ve pencereleme işlemleri uygulanır. FFT (Hızlı Fourier Dönüşümü) yapılır ve Mel filtre bankası kullanılır (9). Son olarak gerekli logaritma işlemleri ve DCT uygulanır.

2.3.2.1. DCT (Ayrık Kosinüs Dönüşümü)

Mel Frekanslı Sepstral Katsayıları (MFCC) bulunması işleminin son aşamasında, mel spektrumunun logaritması zaman bölgesine çevrilir. Bu zaman bölgesine çevirme işlemi ayrık kosinüs dönüşümü (DCT) yardımıyla yapılmaktadır (10).

$$c_y(n) = u_n \sum_{k=0}^{K-1} (\log Y_k) \cos\left(\frac{(2k+1)n\pi}{2K}\right) \quad (10)$$

n = mel frekans kepsral katsayısı indeksi

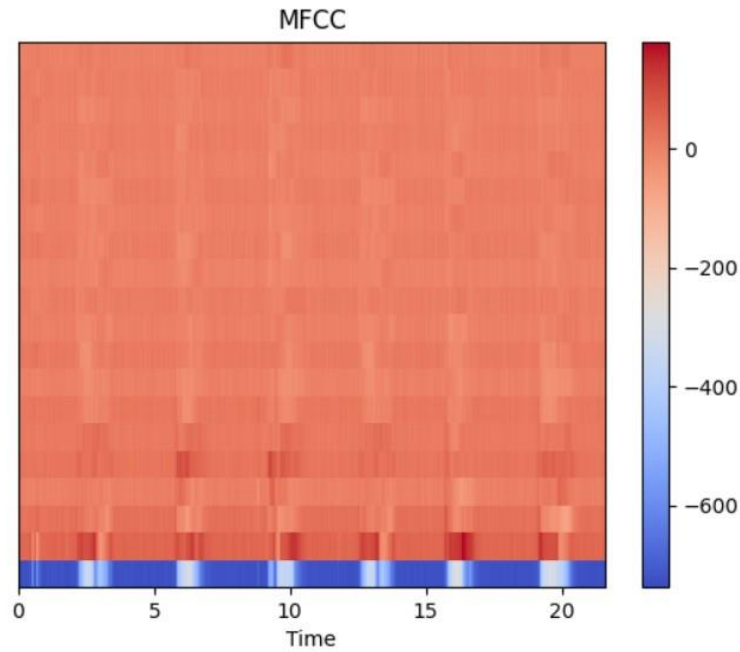
k = mel filtresi indeksi

K = toplam mel filtresi sayısı

2.3.2.2. MFCC Python Kodu

```
lmfcc=librosa.feature.mfcc(signal.data,signal.fs);

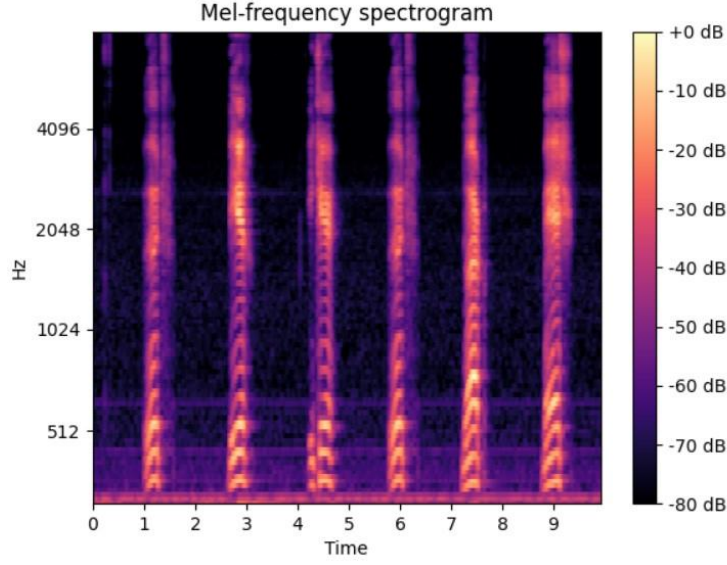
fig, ax = plt.subplots()
img = librosa.display.specshow(lmfcc, x_axis='time', ax=ax,fmin=150)
fig.colorbar(img, ax=ax)
ax.set(title='MFCC')
plt.show()
```



Şekil 13. MFCC kod çıktısı

2.3.2.3. Mel Spektrumu

```
S = librosa.feature.melspectrogram(y=signal.data, sr=signal.fs,n_fft=8192, n_mels=128,
fmax=8000)
fig, ax = plt.subplots()
S_dB = librosa.power_to_db(S, ref=np.max)
img = librosa.display.specshow(S_dB, x_axis='time',
                                y_axis='mel', sr=signal.fs,
                                fmax=8000, ax=ax,fmin=150)
fig.colorbar(img, ax=ax, format='%+2.0f dB')
ax.set(title='Mel-frequency spectrogram')
plt.show()
```



Şekil 14. Mel spektrum kod çıktısı

2.3.3. Pitch Özelliği

Boğazda bulunan ses telleri, periyodik darbeler oluşturur ve bu darbelerin frekanslarına temel frekans adı verilir. Algılanan temel frekans perde frekansı olarak adlandırılmaktadır. Temel frekansın daha iyi anlaşılabilmesi için insan ses üretim mekanizmasının bilinmesi gerekmektedir. Ses üretiminde akciğerler, hava üreten bir enerji kaynağı gibi davranır. Kişinin konuşması ile birlikte hava akciğerlerden ses yolundaki boğaza doğru hareket eder. Konuşma sesi üretmek için ses telleri ve ses yolu belirli bir yapı alır. İnsan ses sisteminin en önemli parçası, ses tellerini içeren boğazdır. Ses tellerinin aktivitesi üretilen sesin ötümlü veya ötümsüz olacağını belirler. Ötümlü sesler için ses telleri hızlıca açılıp kapanarak havayı modüle eder.

Bu kısımda normalleştirilmiş otokorelasyon fonksiyonu kullanılarak sesin perde periyotları çıkarılır. Sesin zaman serisinin $X(n)$ olduğu varsayılın, çerçeveleme ve pencereleme işlemlerinden sonra sesin i . çerçevesi N çerçeve boyutu $X_i(m)$ olsun. Otokorelasyon fonksiyonu (11) ile tanımlanır ve normalleştirilmiş otokorelasyon fonksiyonu (12) ile bulunur.

$$R_i(k) = \sum_{m=1}^{N-m} x_i(m)x_i(m+k) \quad (11)$$

$$r_i(k) = R_i(k)/R_i(0) \quad (12)$$

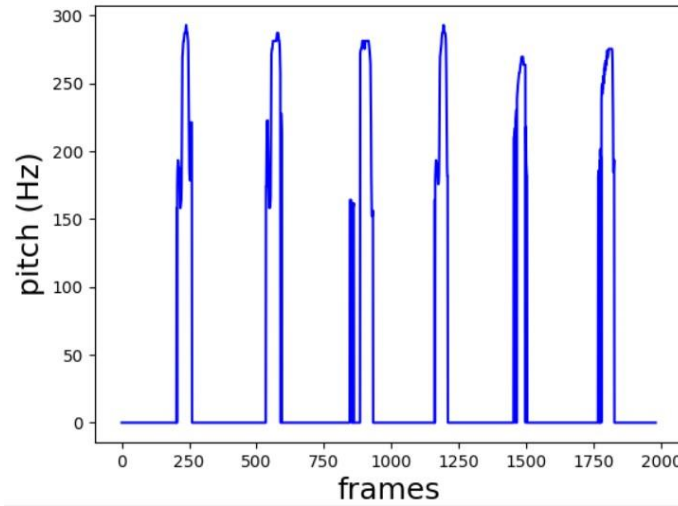
Perde periyodu, orijinal sinyal ile gecikmeli sinyali arasındaki benzerlik karşılaştırılarak belirlenir. Aralık periyodunu tahmin etmek için normalleştirilmiş otokorelasyon fonksiyonunun iki maksimum nokta arasındaki mesafesi bulunur. Perde özelliği genellikle frekans olarak ölçülür ve temel frekansı ifade eder. Farklı hecelerden

çıkarılan perde dizileri genellikle birbirinden farklıdır. Bu nedenle copy-move algılaması için bu özellik kullanılabilir.

2.3.3.1. Pitch Özelliği Python Kodu

```
pitch = pYAAPT.yaapt(signal, **{'f0_min' : 150.0, 'frame_length' : 15.0,
'frame_space' : 5.0})
pitch_interp=np.copy(pitch.samp_interp)
pitch_values=np.copy(pitch.samp_values)

plt.plot(pitch.samp_values, label='samp_values', color='blue')
plt.xlabel('frames', fontsize=18)
plt.ylabel('pitch (Hz)', fontsize=18)
plt.show()
```



Şekil 15. Pitch özelliği kod çıktısı

2.3.4. DFT Katsayıları (Ayrık Fourier Dönüşümü)

Ayrık fourier dönüşümü, sinyal işlemenin önemli bir yoludur. Hareket algılama sürecinde sinyal ve gürültüyü bağımsız olarak analiz edebileceğimiz doğrusal bir özelliğe sahiptir. Sesin zaman serisinin $x(n)$ olduğunu varsayalım, DFT (13) ile hesaplanabilir.

$$X(k) = DFT[x(n)] = \sum_{n=0}^{N-1} x(n)e^{-j\frac{2\pi}{N}kn} \quad (13)$$

Bilindiği üzere DFT temel bir dijital sinyal işleme yöntemidir ve ses alanında yaygın olarak kullanılmaktadır. Ayrıca copy-move tespitinde de önemli bir rol oynamaktadır. Zaman alanı karşılaştırmasını daha uygun ve etkili olan frekans alanına dönüştürür bu nedenle algılama yöntemlerinden biri olarak DFT tercih edilir.

2.3.4.1. DFT Python Kodu

```

frame_size = int(np.fix(15 * signal.fs / 1000))
frame_jump = int(np.fix(5 * signal.fs / 1000))

fir_filter = pYAAPT.BandpassFilter(signal.fs,parametre)
signal.filtered_version(fir_filter)

nframe=Fonk.enframe(signal.filtered,frame_size,frame_jump)

specData=np.fft.fft(nframe,8192)

frame_energy = np.abs(specData[:, int(60 - 1):int(400)]).sum(axis=1)

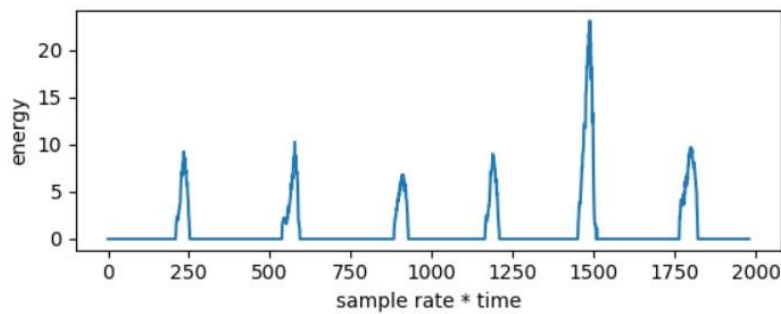
frame_mean=np.mean(frame_energy)
frame_energy=frame_energy/frame_mean

frame_energyKontrol=np.zeros(len(frame_energy))

for i in range(len(frame_energy)):
    if (frame_energy[i]>1):
        frame_energyKontrol[i]=frame_energy[i]
    else :
        frame_energyKontrol[i]=0

plot_a = plt.subplot(211)
plot_a.plot(frame_energyKontrol)
plot_a.set_xlabel('sample rate * time')
plot_a.set_ylabel('energy')
plt.show()

```



Şekil 16. DFT kod çıktısı

2.3.5. Ortalama Karekök (RMS)

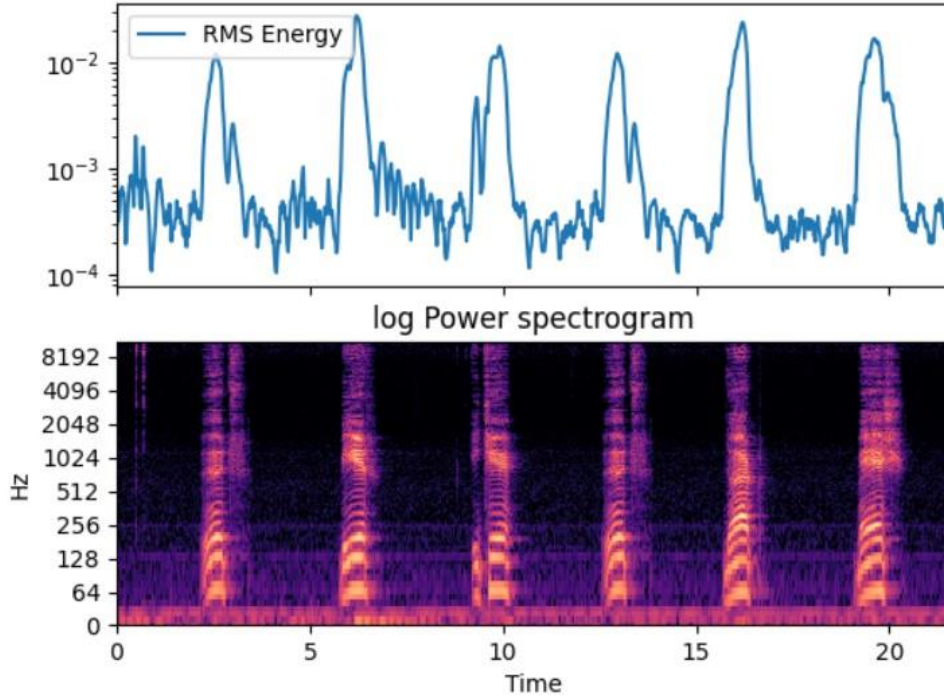
Karekök ortalama; matematikte *root mean square* (kısaltması RMS ya da rms) ayrıca kuadratik ortalama olarak da bilinir. Değişen miktarların büyüklüğünün ölçülmesinde kullanılan istatistik bir ölçüttür. Değişimin artı ve eksi yönde olduğu dalgalarda özellikle çok faydalıdır.

Sürekli olarak değişen bir fonksiyonun sürekli olmayan değer serisi için hesaplanabilir. Karekök ortalama ismi karelerin ortalamasının karekökünün alınmasından gelir.

2.3.5.1. RMS Python Kodu

```
S, phase = librosa.magphase(librosa.stft(signal.data))
rms = librosa.feature.rms(S=S)

fig, ax = plt.subplots(nrows=2, sharex=True)
times = librosa.times_like(rms)
ax[0].semilogy(times, rms[0], label='RMS Energy')
ax[0].set(xticks=[])
ax[0].legend()
ax[0].label_outer()
librosa.display.specshow(librosa.amplitude_to_db(S, ref=np.max),
                          y_axis='log', x_axis='time', ax=ax[1])
ax[1].set(title='log Power spectrogram')
plt.show()
```



Şekil 17. RMS kod çıktısı

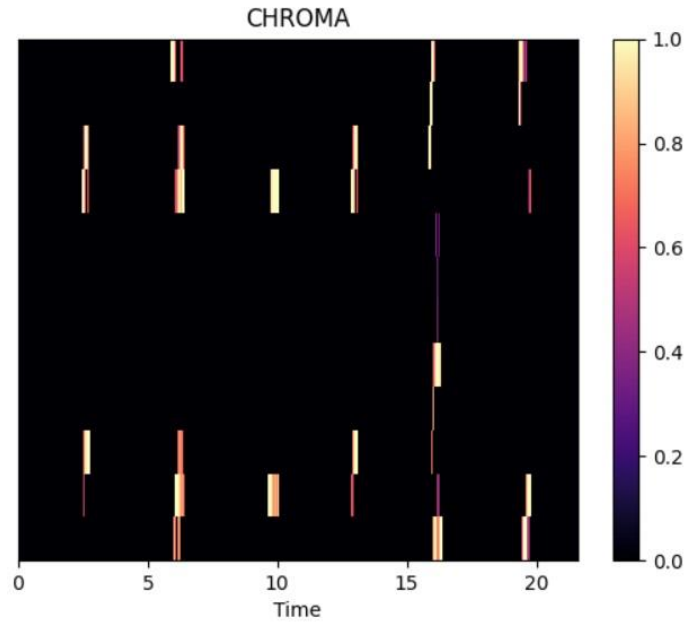
2.3.6. Kroma Özelliği

Kroma özelliği veya kromagram terimi, on iki farklı perde sınıfıyla yakından ilgilidir. "Perde sınıfı profilleri" olarak da adlandırılan kroma tabanlı özellikler, perdeleri anlamlı bir şekilde kategorize edilebilen (genellikle on iki kategoriye) ve akordu eşit ölçülü ölçeğe yaklaşan müziği analiz etmek için güçlü bir araçtır. Kroma özelliklerinin ana özelliklerinden biri, tını ve enstrümantasyondaki değişikliklere karşı sağlam olurken müziğin armonik ve melodik özelliklerini yakalamalarıdır.

2.3.6.1. Kroma Python Kodu

```
chroma=librosa.feature.chroma_cqt(y=signal.data,
sr=signal.fs,fmin=150,threshold=0.75)

fig, ax = plt.subplots()
img = librosa.display.specshow(chroma, x_axis='time', ax=ax)
fig.colorbar(img, ax=ax)
ax.set(title='CHROMA')
plt.show()
```



Şekil 18. Kroma özelliği kod çıktısı

2.4. Benzerlik Karşılaştırmaları

Benzerlik karşılaştırması Bölüm 2.3'te bahsedilen altı özelliği kullanarak ses segmentlerini karşılaştırmaktır. Bu kısımda benzerlik hesaplaması yapmak için iki yol kullanılır.

2.4.1. PCCs (Pearson Korelasyon Katsayısı)

Pearson korelasyon katsayısı, iki değişkenin doğrusal korelasyon derecesinin bir ölçüsüdür. İki segmentin pearson korelasyon katsayısı 1'e yakınsa, tespit edilen sesin copy-move sesi olduğu anlaşılır. Pearson korelasyon katsayısı (14) ile tanımlanır.

$$r = \frac{\sum_{i=1}^n (x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_{i=1}^n (x_i - \bar{x})^2} \sqrt{\sum_{i=1}^n (y_i - \bar{y})^2}} \quad (14)$$

Burada \bar{x} x dizisinin ortalamasıdır. \bar{y} y dizisinin ortalamasıdır. Değerlerin çözüm kümesi [-1,1] aralığındadır, x ve y benzerse PCC'leri 1'e yakındır. Burada gammatone özelliği ve DFT katsayılarında PCC özelliği kullanılır. Gammatone özelliği ve DFT katsayılarının boyutlarının, MFCC'lerin özelliği ve perde özelliğinden daha büyük olduğu görülebilir. Gammatone özelliği 64 kanala sahiptir ve DFT katsayıları ses segmentleri ile aynı boyuttadır.

2.4.1.1. PCC Python Kodu

```
def karsilastirma(kard, karb, deger2x):

    minx=min(kard[1],karb[1])
    r=-999999999

    x=deger2x[int(kard[0]):int(kard[0]+kard[1])]
    y=deger2x[int(karb[0]):int(karb[0]+karb[1])]

    sum1 = np.sum(x)
    sum2 = np.sum(y)

    if (sum1 == 0 and sum2 == 0):
        return 1
    elif (sum1 == 0 and sum2 != 0):
        return 0
    elif (sum1 != 0 and sum2 == 0):
        return 0

    for i in range(int(abs(kard[1]-karb[1]))):

        y=deger2x[int(karb[0]+i):int(karb[0]+kard[1]+i)]
        r=max(r, scipy.stats.pearsonr(x,y)[0])

    r = max(r, scipy.stats.pearsonr(x, y)[0])
    return r
```

```

def gelismisBenzerlikBulma(r_matrix ,segmentler,sayac,temp):

    r_matrix0=np.zeros((sayac,sayac))
    r_matrix1=np.zeros((sayac,sayac))
    r_matrix2=np.zeros((sayac,sayac))
    r_matrix3=np.zeros((sayac,sayac))
    r_matrix4=np.zeros((sayac,sayac))
    r_matrix5=np.zeros((sayac,sayac))
    r_matrix6=np.zeros((sayac,sayac))
    r_matrix7=np.zeros((sayac,sayac))
    r_matrix8=np.zeros((sayac,sayac))
    r_matrix9=np.zeros((sayac,sayac))
    r_matrix10=np.zeros((sayac,sayac))
    r_matrix11=np.zeros((sayac,sayac))

    for k in range(12):

        deger=temp[k]

        for i in range(sayac):

            for j in range(sayac):

                if segmentler[i][1]>=segmentler[j][1]:

                    if(k==0):

r_matrix0[i][j]=karsilastirma(segmentler[j],segmentler[i],deger)
                    elif(k==1):

r_matrix1[i][j]=karsilastirma(segmentler[j],segmentler[i],deger)
                    elif(k==2):
                        r_matrix2[i][j]          =          karsilastirma(segmentler[j],
segmentler[i], deger)
                    elif(k==3):
                        r_matrix3[i][j]          =          karsilastirma(segmentler[j],
segmentler[i], deger)
                    elif (k == 4):
                        r_matrix4[i][j]          =          karsilastirma(segmentler[j],
segmentler[i], deger)
                    elif (k == 5):
                        r_matrix5[i][j]          =          karsilastirma(segmentler[j],
segmentler[i], deger)
                    elif (k == 6):
                        r_matrix6[i][j]          =          karsilastirma(segmentler[j],
segmentler[i], deger)
                    elif (k == 7):
                        r_matrix7[i][j]          =          karsilastirma(segmentler[j],
segmentler[i], deger)
                    elif (k == 8):
                        r_matrix8[i][j]          =          karsilastirma(segmentler[j],
segmentler[i], deger)
                    elif (k == 9):
                        r_matrix9[i][j]          =          karsilastirma(segmentler[j],
segmentler[i], deger)
                    elif (k == 10):

```

```

        r_matrix10[i][j] =
karsilaştirma(segmentler[j],segmentler[i], deger)
        elif (k == 11):
            r_matrix11[i][j] =
karsilaştirma(segmentler[j],segmentler[i], deger)

    else:
        if(k==0):
            r_matrix0[i][j] = karsilaştirma(segmentler[i], segmentler[j],
deger)
        elif(k==1):
            r_matrix1[i][j] = karsilaştirma(segmentler[i],
segmentler[j], deger)
        elif(k==2):
            r_matrix2[i][j] = karsilaştirma(segmentler[i],
segmentler[j], deger)
        elif(k==3):
            r_matrix3[i][j] = karsilaştirma(segmentler[i],
segmentler[j], deger)
        elif (k == 4):
            r_matrix4[i][j] = karsilaştirma(segmentler[i],
segmentler[j], deger)
        elif (k == 5):
            r_matrix5[i][j] = karsilaştirma(segmentler[i],
segmentler[j], deger)
        elif (k == 6):
            r_matrix6[i][j] = karsilaştirma(segmentler[i],
segmentler[j], deger)
        elif (k == 7):
            r_matrix7[i][j] = karsilaştirma(segmentler[i],
segmentler[j], deger)
        elif (k == 8):
            r_matrix8[i][j] = karsilaştirma(segmentler[i],
segmentler[j], deger)
        elif (k == 9):
            r_matrix9[i][j] =
karsilaştirma(segmentler[i],segmentler[j], deger)
        elif (k == 10):
            r_matrix10[i][j] = karsilaştirma(segmentler[i],
segmentler[j], deger)
        elif (k == 11):
            r_matrix11[i][j] = karsilaştirma(segmentler[i],
segmentler[j], deger)

r_matrix=r_matrix0+r_matrix1+r_matrix2+r_matrix3+r_matrix4+r_matrix5+r_m
atrix7+r_matrix8
    r_matrix=r_matrix+r_matrix9+r_matrix10+r_matrix11

    r_matrix = r_matrix/12.0

    return r_matrix

```

```

def benzerlikBulma(r_matrix,segmentler,sayac,deger):
    for i in range(sayac):

        for j in range(sayac):

            if segmentler[i][1] >= segmentler[j][1]:
                r_matrix[i][j] = karsilastirma(segmentler[j], segmentler[i],
deger)

            else:
                r_matrix[i][j] = karsilastirma(segmentler[i], segmentler[j],
deger)


def segmentBulma(segmentler,sayac, kontrol, tak):

    for i in range(len(kontrol)):

        if i > 0:
            tak = 1
        if kontrol[i - 1] == 0 and kontrol[i] != 0 and tak == 1:
            j = i
            while (kontrol[j] != 0 and j < len(kontrol)):
                j += 1

            if j - i >= 10:
                segmentler[sayac][0] = i + 3
                segmentler[sayac][1] = j - i - 4

                sayac += 1
                i = i + j
    return sayac

```

2.4.2. AD (Ortalama Fark)

Ortalama fark, iki değerin farklılığının somutlaşmış halidir. Ortalama farkın değeri 0'a yakınsa kopyalanan frameleri bulmak için ortalama fark kullanılır. Ortalama fark (15) ile bulunur.

$$AD = \frac{1}{N_{\min}} \sum_{i=1}^{N_{\min}} |x_i - y_i| \quad (15)$$

Burada N_{\min} dizinin minimum uzunluğudur. İki dizi ne kadar benzerse AD değeri o kadar küçüktür.

2.5. C4.5 Tabanlı Karar Ağacı

Karar ağacı, sınıflandırma ve tahmin problemlerinde kullanılan bir veri madenciliği tekniğidir. Karar ağacı bir kök düğümden başlar ve çatal düğümlerden geçerek yaprak düğüme ulaşır.

Bu projede, bahsedilen dört özelliği kullanarak copy-move algılamasının sonuçlarının nihai kararını vermek için C4.5 karar ağacı kullanılmıştır. Öncelikle her özelliğin kazanç oranı hesaplanır (16) (17) ardından ağacın dallanması için en büyük kazanç oranına sahip özellik seçilir.

$$split_info_A(D) = - \sum_{j=1}^v \frac{|D_j|}{|D|} \log_2 \left(\frac{|D_j|}{|D|} \right) \quad (16)$$

$$gain_ratio(A) = \frac{gain(A)}{split_info(A)} \quad (17)$$

Burada D sınıf, A özelliklerden biridir ve kazanç A (18) ile hesaplanır.

$$gain(A) = info(D) - info_A(D) \quad (18)$$

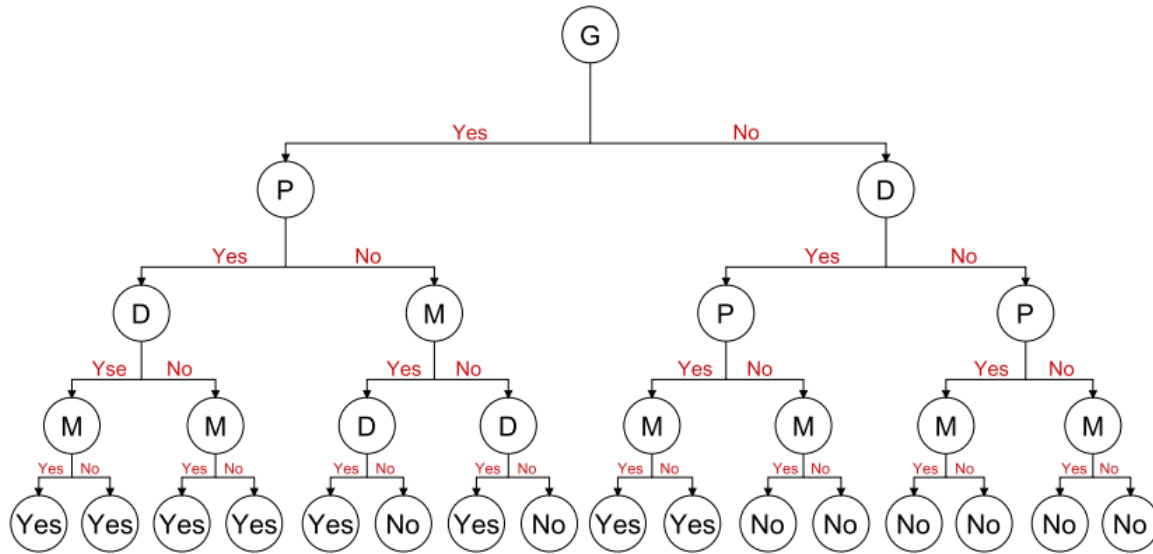
(19) D'nin entropisidir. P_i , bir örneğin i. sınıfa ait olma olasılığıdır. (20) A'nın D'yi sınıflandırması için gereken bilgidir.

$$info(D) = - \sum_{i=1}^m p_i \log_2(p_i) \quad (19)$$

$$info_A(D) = \sum_{j=1}^v \frac{|D_j|}{|D|} info(D_j) \quad (20)$$

C4.5 karar ağacı kullanılarak gürültüsüz sesin copy-move tespiti yanı sıra gürültülü sesin copy-move tespiti de yapılabilir. Çünkü algılamayı yapmak için dört özellik vardır ve bu özelliklerin algılama sonucunun birçok kombinasyonu vardır.

Şekil 19'da G'nin gammaton özelliğini, P'nin perde özelliğini, D'nin DFT katsayılarını ve M'nin MFCCs özelliğini temsil ettiği, C4.5 karar ağacı ile dört özelliğin kombinasyonları gösterilmiştir. Yaprak düğümler copy-move uygulanmış sesin karar sonucudur, yani sonuç EVET ise sahte ses belirlenmiş olur. Yukarda belirtildiği gibi C4.5 karar ağacı öncelikle bu dört özelliğin bilgi kazancını hesaplar ve en büyüğünü kök olarak seçer. Şekil 13'te gammatone özelliği kök düğüm olarak seçilmiştir. Gammatone özelliğinin tespit sonucuna göre karar ağacı ikiye ayrılır. Daha sonra diğer üç özelliğin bilgi kazancı hesaplanmaya devam edilir. C4.5 karar ağacı tüm yaprak düğümlere ulaşana kadar bu işlem devam eder.



Şekil 19. C4.5 Karar Ağacı

3. SONUÇLAR

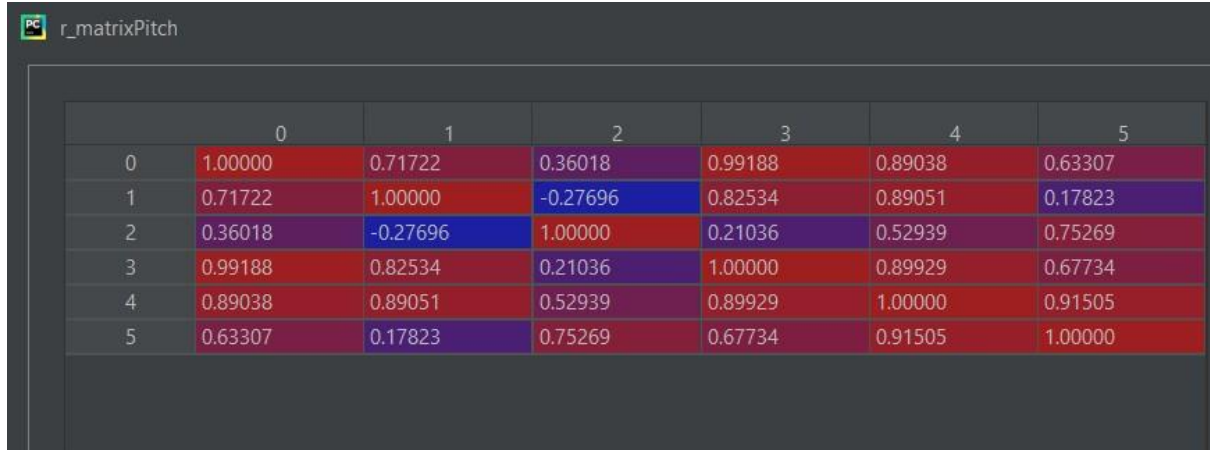
3.1. Pitch Özelliği Benzerlik Sonucu

```
segmentler=np.zeros((30,2))
sayac=0
tak=0 ## dizinin sıfırdan büyüklüğünü test eder

sayac = Fonk.segmentBulma(segmentler,sayac,pitch_values,tak)
segmentler= segmentler[0:sayac]

r_matrixPitch=np.zeros((sayac,sayac))
Fonk.benzerlikBulma(r_matrixPitch,segmentler,sayac,pitch_interp)

print("\n")
print("pitch benzerlik oranı: \n")
print(r_matrixPitch)
```



	0	1	2	3	4	5
0	1.00000	0.71722	0.36018	0.99188	0.89038	0.63307
1	0.71722	1.00000	-0.27696	0.82534	0.89051	0.17823
2	0.36018	-0.27696	1.00000	0.21036	0.52939	0.75269
3	0.99188	0.82534	0.21036	1.00000	0.89929	0.67734
4	0.89038	0.89051	0.52939	0.89929	1.00000	0.91505
5	0.63307	0.17823	0.75269	0.67734	0.91505	1.00000

Şekil 20. Pitch kod çıktısı

3.2. Kroma Özelliği Benzerlik Sonucu

```
segmentler=np.zeros((30,2))
sayac=0
tak=0

sayac=Fonk.segmentBulma(segmentler,sayac,chromaDeger,tak)
segmentler= segmentler[0:sayac]
r_matrixChromaG=np.zeros((sayac,sayac))

r_matrixChromaG=Fonk.gelismisBenzerlikBulma(r_matrixChromaG,segmentler,sayac,chroma)

print("\n")
print("chroma Gelismis benzerlik oranı: \n")
print(r_matrixChromaG)
```

PC r_matrixChromaG

	0	1	2	3	4	5
0	1.00000	0.63918	0.69987	0.99761	0.15903	0.40310
1	0.63918	1.00000	0.61327	0.67306	-0.08011	0.52755
2	0.69987	0.61327	1.00000	0.69908	-0.02958	0.54548
3	0.99761	0.67306	0.69908	1.00000	0.11488	0.40625
4	0.15903	-0.08011	-0.02958	0.11488	1.00000	0.18748
5	0.40310	0.52755	0.54548	0.40625	0.18748	1.00000

Şekil 21. Kroma kod çıktısı

3.3. Gammatone Özelliği Benzerlik Sonucu

```
segmentler=np.zeros((30,2))
sayac=0
tak=0

sayac=Fonk.segmentBulma(segmentler,sayac,gfccKontrol,tak)
segmentler= segmentler[0:sayac]

r_matrixGfcc=np.zeros((sayac,sayac))

r_matrixGfcc=Fonk.gelismisBenzerlikBulma(r_matrixGfcc,segmentler,sayac,gfcc)

print("\n")
print("GFCC benzerlik oranı: \n")
print(r_matrixGfcc)
```

PC r_matrixGfcc

	0	1	2	3	4	5
0	1.00000	0.54746	0.56822	0.96487	0.56910	0.54162
1	0.54746	1.00000	0.71266	0.46130	0.80481	0.61868
2	0.56822	0.71266	1.00000	0.51339	0.58047	0.65396
3	0.96487	0.46130	0.51339	1.00000	0.46398	0.57861
4	0.56910	0.80481	0.58047	0.46398	1.00000	0.67624
5	0.54162	0.61868	0.65396	0.57861	0.67624	1.00000

Şekil 22. Gammatone kod çıktısı

3.4. DFT Özelliği Benzerlik Sonucu

```
segmentler=np.zeros((30,2))
sayac=0
tak=0

sayac=Fonk.segmentBulma(segmentler,sayac, frame_energyKontrol,tak)
segmentler= segmentler[0:sayac]

r_matrixDft=np.zeros((sayac,sayac))

Fonk.benzerlikBulma(r_matrixDft,segmentler,sayac,frame_energyKontrol)

print("\n")
print("DFTbenzerlik oranı: \n")
print(r_matrixDft)
```

	0	1	2	3	4	5
0	1.00000	0.91154	0.95194	0.99344	0.94512	0.96674
1	0.91154	1.00000	0.88781	0.86256	0.81978	0.90396
2	0.95194	0.88781	1.00000	0.94536	0.95652	0.94551
3	0.99344	0.86256	0.94536	1.00000	0.94819	0.95475
4	0.94512	0.81978	0.95652	0.94819	1.00000	0.87097
5	0.96674	0.90396	0.94551	0.95475	0.87097	1.00000

Şekil 23. DFT kod çıktısı

3.5. C4.5 Karar Ağacı Sonucu

```
r_matrix4G=r_matrixPitch+r_matrixChromaG+r_matrixGfcc+r_matrixDft
r_matrix4G=r_matrix4G/4.0

print("\n")
print("c4.5 based detection benzerlik oranı: \n")
print(r_matrix4G)
print("\n")

i=sayac-1
while(i>=0):

    for j in range(i):

        if (r_matrix4G[i][j]>=0.98 and i!=j):

            print(i,".segment ile ",j,"segment birbirinden kopyalanmistir.\n")

    i=i-1
```

PC r_matrix4G

	0	1	2	3	4	5
0	1.00000	0.70385	0.64505	0.98695	0.64091	0.63613
1	0.70385	1.00000	0.48419	0.70556	0.60875	0.55710
2	0.64505	0.48419	1.00000	0.59205	0.50920	0.72441
3	0.98695	0.70556	0.59205	1.00000	0.60659	0.65424
4	0.64091	0.60875	0.50920	0.60659	1.00000	0.66243
5	0.63613	0.55710	0.72441	0.65424	0.66243	1.00000

Şekil 24. C4.5 kod çıktısı

4. ÖNERİLER

Copy-Move ses sahteciliğinin tespiti yapılırken bir önceki çalışmada kullanılan PDA algoritması başarılı bir algoritma olmakla beraber %80 oranında bir başarı göstermiştir. Bu çalışmamızda çok özellikli tespit yönteminde ise bu oran %94'e kadar çıkmaktadır. Dolayısıyla kullanılan bu yöntem bize daha sağlıklı sonuçlar verdiği için copy-move tespitinde sağlıklı bir yöntem diyebiliriz.

5. KAYNAKLAR

- [1]<https://www.sciencedirect.com/science/article/abs/pii/S2214212617304404>
- [2]https://en.wikipedia.org/wiki/Gammatone_filter
- [3]<https://dergipark.org.tr/tr/download/article-file/202719>
- [4]https://web.itu.edu.tr/~kartalya/proje1/proje_sunum.pdf
- [5]<https://ieeexplore.ieee.org/abstract/document/8947002/>
- [4]<https://www.technopat.net/2018/01/29/bitrate-ve-sample-rate-nedir-dijital-sesin-temelkavramlari/>
- [5]<https://asa.scitation.org/doi/abs/10.1121/1.2916590>
- [6]<https://ieeexplore.ieee.org/abstract/document/8629277>
- [7]<https://ieeexplore.ieee.org/abstract/document/8005541>
- [8]<https://ieeexplore.ieee.org/abstract/document/8599318>
- [9]<https://ieeexplore.ieee.org/abstract/document/5743729>
- [10]<https://ieeexplore.ieee.org/abstract/document/9163568>
- [11]<https://ieeexplore.ieee.org/abstract/document/7178277>
- [12]<https://dergipark.org.tr/tr/pub/ubgmd/issue/18272/192662>
- [13]<https://ieeexplore.ieee.org/document/7954589>
- [14]<https://www.scipy.org/>
- [15]<https://numpy.org/> [14]<https://matplotlib.org/>
- [16]<https://www.semanticscholar.org/>
- [17]https://wikimili.com/en/Pitch_detection_algorithm
- [18]<https://github.com/>
- [19]<https://www.quora.com>
- [20]http://bjbschmitt.github.io/AMFM_decompy/pYAAPT.html
- [21]<https://librosa.org/doc/main/generated/librosa.feature.mfcc.html>
- [22]<https://github.com/detly/gammatone>

6. EKLER

Disiplinler arası çalışma yapıldı ve Elektrik Elektronik Mühendisliği bölümüyle ortak çalışmaya katıldık. Bu çalıştay da bitirme tezine yönelik çalışmalar yaptık.

STANDARTLAR ve KISITLAR FORMU

Projenin hazırlanmasında uyulan standart ve kısıtlarla ilgili olarak, aşağıdaki soruları cevaplayınız.

1. Projenizin tasarım boyutu nedir? (Yeni bir proje midir? Var olan bir projenin tekrarı mıdır? Bir projenin parçası mıdır? Sizin tasarımınız proje toplamının yüzde olarak ne kadarını oluşturmaktadır?)

Copy-Move kullanılarak yapılan ses sahteciliğinin tespit yöntemi olarak yayınlanmış bir bildirinin gerçekleştirilmesidir.

2. Projenizde bir mühendislik problemini kendiniz formüle edip, çözdünüz mü? Açıklayınız.

Çözmedik.

3. Önceki derslerde edindiğiniz hangi bilgi ve becerileri kullandınız?

Python programlama dili, Algoritma kurma ve çözme işlemleri

4. Kullandığınız veya dikkate aldığınız mühendislik standartları nelerdir? (Proje konunuzla ilgili olarak kullandığınız ve kullanılması gereken standartları burada kod ve isimleri ile sıralayınız).

5. Kullandığınız veya dikkate aldığınız gerçekçi kısıtlar nelerdir? Lütfen boşlukları uygun yanıtlarla doldurunuz.

- a) Ekonomi

Ekonomik bir bağımlılığımız bulunmamaktadır.

- b) Çevre sorunları:

Herhangi bir çevre sorunu teşkil etmemektedir.

c) Sürdürülebilirlik:

Durumu göre kodu güncelleyerek sürdürülebilirlik kazandırılabilir.

d) Üretilirlik:

e) Etik:

Proje etik açıdan bir sorun teşkil etmemektedir.

f) Sağlık:

Sağlık açısından bir sorun teşkil etmemektedir

g) Güvenlik:

Kodun herhangi bir saldırıya açık olması durumunda güvenlik sorun teşkil etmektedir.

h) Sosyal ve politik sorunlar:

Sosyal ve Politik olarak ses sahteciliğinin delillerinin doğruluğunu kontrol eder.