**T.C.**

**BAHÇEŞEHİR ÜNİVERSİTESİ**

# PYTHON DİLİ İLE SİNYAL İŞLEME MODÜLLERİ

**Yüksek Lisans Tezi**

**YİĞİT CAN AKÇELİK**

**İSTANBUL, 2020**

**T.C.**

**BAHÇEŞEHİR ÜNİVERSİTESİ**

**FEN BİLİMLERİ ENSTİTÜSÜ**

**SES TEKNOLOJİLERİ**

# PYTHON DİLİ İLE SİNYAL İŞLEME MODÜLLERİ

**Yüksek Lisans Tezi**

**YİĞİT CAN AKÇELİK**

**Tez Danışmanı: DR. ÖĞR. ÜYESİ CEMAL BARKIN ENGİN**

**İSTANBUL, 2020**

**T.C.**
**BAHÇEŞEHİR ÜNİVERSİTESİ**

**FEN BİLİMLERİ ENSTİTÜSÜ**
**SES TEKNOLOJİLERİ**

Tezin Adı: Python dili ile sinyal işleme modülleri
Öğrencinin Adı Soyadı: Yiğit Can Akçelik
Tez Savunma Tarihi:

Bu tezin Yüksek Lisans tezi olarak gerekli şartları yerine getirmiş olduğu Fen Bilimleri Enstitüsü tarafından onaylanmıştır.

Doç.Dr. Öğr. Üyesi Burak KÜNTAY
Enstitü Müdürü

Bu Tez tarafımızca okunmuş, nitelik ve içerik açısından bir Yüksek Lisans tezi olarak yeterli görülmüş ve kabul edilmiştir.

| Jüri Üyeleri | İmzalar |
|---|---|
| Tez Danışmanı<br>Dr. Öğr. Üyesi Cemal Barkın ENGİN | ---------- |
| Üye<br>Dr. Öğr. Üyesi Yahya Burak TAMER | ---------- |
| Üye<br>Prof. Dr. Can KARADOĞAN | ---------- |

# TEŞEKKÜR

Yüksek lisansım ve tez çalışmam süresince desteğini esirgemeyen danışmanım Dr. Öğr. Üyesi Cemal Barkın Engin'e, yol göstericiliği ve yardımları için Dr. Öğr. Üyesi Yahya Burak Tamer'e, beni programlama dünyasıyla tanıştıran Alpay Özbay'a, müzikal gelişimimdeki sonsuz katkıları için abim Gökçe Akçelik'e, her zaman yanımda olan annem ve babam Arzu ve Savaş Akçelik'e, pozitif enerjisiyle bu tez çalışmasının gerçekleşmesinde büyük katkısı olan sevgili Hilal Menlioğlu'na teşekkür ederim.

İstanbul, 2020 Yiğit Can AKÇELİK

# ÖZET

PYTHON DİLİ İLE SİNYAL İŞLEME MODÜLLERİ

Yiğit Can AKÇELİK

Yüksek Lisans, Ses Teknolojileri Yüksek Lisans Programı

Tez Danışmanı: Dr. Öğr. Üyesi Cemal Barkın ENGİN

Haziran 2020, 66 sayfa

Bu çalışmada Python programlama dili kullanılarak sinyal işleme modülleri oluşturulması hedeflenmiştir. Bu tasarımlarda esas olarak Phase Vocoder tekniği kullanılmıştır. Tasarlanan modüller aracılığıyla seslerin frekans, genlik, tını ve/veya zaman gibi özellikleri değiştirilebilmektedir. Geliştirilen modüller komut satırından çalıştırılmaktadır. Bu çalışmadaki esas amaç ticari olarak kullanılabilecek bir efekt modülünden ziyade, Python dilinin avantajlarından faydalanarak hızlı ve efektif bir biçimde Phase Vocoder tekniğini uygulamak ve algoritmalar geliştirmektir. Çalışma kapsamında sesin perde değerine müdahale eden bir modül ve sesin süresini uzatan/sıkıştıran bir modül tasarlanmıştır.

**Anahtar Kelimeler:** Python, Dijital Sinyal İşleme, Phase Vocoder

## ABSTRACT

CREATING A DIGITAL SIGNAL PROCESSING MODULE WITH PYTHON

Yiğit Can AKÇELİK

Master's Thesis, Master's Program in Sound Technologies

Supervisor: Dr. Cemal Barkın ENGİN

June 2020, 66 pages

This thesis deals with a simple implementation of the Phase Vocoder technique using Python programming language. Using basic digital signal processing knowledge, sounds' different components such as frequency, phase, amplitude and/or time are manipulated resulting in a pitch shifter module and a time compression/expansion module, which are used through the command line. The aim of this thesis, is not to create a commercially usable plug-in, but rather trying to implement the Phase Vocoder technique with basic coding and signal processing knowledge, while benefiting from the advantages of Python programming language.



Keywords: Digital Signal Processing, Python, Phase Vocoder

# İÇİNDEKİLER

**TABLOLAR** ..... viii
**ŞEKİLLER** ..... ix
**KISALTMALAR** ..... xii
**SEMBOLLER** ..... xiii
**1. GİRİŞ** ..... 1
**2. GENEL BİLGİLER** ..... 4
**2.1 PYTHON** ..... 4
**2.1.1 Python Dilinin Tarihçesi** ..... 4
**2.1.2 Python Dilinin Özellikleri** ..... 5
**2.1.3 Python Dilinin Kullanım Alanları** ..... 6
**2.1.4 Python Kullanılarak Yapılmış Benzer Çalışmalar** ..... 6
**2.2 SESİN PERDE DEĞİŞİMİ** ..... 7
**2.3 PHASE VOCODER** ..... 9
**2.3.1 Fourier Analizi** ..... 9
**2.3.2 Phase Vocoder İşlem Akışı** ..... 10
**2.3.2.1 Pencere fonksiyonu** ..... 12
**2.3.2.2 Sesin FFT ile analizi** ..... 14
**2.3.3.3 Frekans düzlemi işlemleri** ..... 14
**2.3.3.4 Sesin IFFT ile yeniden oluşturulması** ..... 16
**2.3.3.5 Üst üste ekleme işlemi** ..... 17
**3. VERİ VE YÖNTEM** ..... 18
**3.1 MODÜLLERİN İŞLEM AKIŞI** ..... 18
**3.2 ALGORİTMALARIN OLUŞTURULMASI** ..... 19
**3.3 MODÜLLERİN ÇALIŞTIRILMASI** ..... 22
**4. BULGULAR** ..... 24
**4.1 PERDE DEĞİŞTİRİCİ MODÜL TESTLERİ** ..... 24
**4.1.1 Sinyal Testleri** ..... 24
**4.1.1.1 440 Hz'in 1 ve 2 oktav tize taşınması ve Pitch II karşılaştırması** ..... 24
**4.1.1.2 440 Hz'in 1 oktav pese taşınması ve Pitch II karşılaştırması** ..... 28
**4.1.1.3 220 Hz'in 2 oktav tize taşınması ve pencere boyutları karşılaştırması** ..... 30
**4.1.2 İnsan Sesi Testleri** ..... 33
**4.1.2.1 İnsan sesinin farklı pencere fonksiyonu boyutları ile 5 aralık tize taşınması ve Pitch II karşılaştırması** ..... 33
**4.1.2.2 İnsan sesinin farklı pencere fonksiyonu boyutları ile 3 aralık pese taşınması ve Pitch II karşılaştırması** ..... 35
**4.1.2.3 İnsan sesinin 2 aralık tize taşınması ve pencere fonksiyon tipleri karşılaştırması** ..... 40
**4.1.2.4 İnsan sesinin perde değeri değiştirilmeden modülden geçirilmesi ve pencere boyutları karşılaştırması** ..... 45
**4.1.3 Elektrik Gitar Sesi Testleri** ..... 48

**4.1.3.1 Elektrik gitar sesinin farklı pencere fonksiyonu boyutları ile 4 aralık pese taşınması ve Pitch II karşılaştırması ..... 48**
**4.1.3.2 Elektrik gitar sesinin 6 aralık tize taşınması ve pencere boyutları karşılaştırması ..... 53**
**4.1.4 Çok Kanallı Şarkı Testleri ..... 56**
**4.2 SESİ UZATAN/SIKIŞTIRAN MODÜL TESTLERİ ..... 58**
**4.2.1 İnsan Sesinin 4 Katına Uzatılması ve Time Compression/Expansion Karşılaştırması ..... 58**
**4.2.2 İnsan Sesinin 0.25 Katına Sıkıştırılması ve Time Compression/Expansion Karşılaştırması ..... 59**
**4.2.3 İnsan Sesinin Farklı Pencere Fonksiyonu Boyutları ile 10 ve 100 Katına Uzatılması ..... 61**
**5. TARTIŞMA VE SONUÇ ..... 62**
**KAYNAKÇA ..... 67**
**EKLER ..... 70**
**Ek A.1 Perde değiştirici modül kodu ..... 71**
**Ek A.2 Sesi uzatan/sıkıştıran modül kodu ..... 73**
**Ek A.3 Testlerde kullanılan ses dosyaları ..... 75**

# TABLOLAR

Tablo 2.1: Notaların Hz cinsinden frekans değerleri.....7

## ŞEKİLLER

Şekil 1.1: Eventide H910 Harmonizer efekt cihazı .....1
Şekil 1.2: Universal Audio firmasının Manley Massive Passive EQ modellemesi.....2
Şekil 2.1: Dikdörtgen, Bartlett, Hann ve Hamming pencere fonksiyonları.....12
Şekil 2.2: Blackman pencere fonksiyonu.....13
Şekil 2.3: Kaiser pencere fonksiyonu.....13
Şekil 2.4: Phase Vocoder işlem akışı .....16
Şekil 3.1: zero_padding fonksiyonunun işlem akışı.....19
Şekil 3.2: phase_vocoder fonksiyonunun işlem akışı.....20
Şekil 3.3: resample fonksiyonunun işlem akışı.....21
Şekil 3.4: pitch_shift fonksiyonunun çalışma akışı.....22
Şekil 3.5: elastic fonksiyonunun çalışma akışı.....22
Şekil 3.6: pitch_shift modülünün çalışma akışı.....23
Şekil 3.7: elastic modülünün çalışma akışı.....23
Şekil 4.1: Sinyal üretecinin oluşturduğu 880 Hz (üst) ve 1760 Hz (alt) sinyalleri.....25
Şekil 4.2: Modülün (üst) ve Pitch II'nin (alt) oluşturduğu 880 Hz sinyalleri.....26
Şekil 4.3: Modülün (üst) ve Pitch II'nin (alt) oluşturduğu 1760 Hz sinyalleri.....27
Şekil 4.4: Sinyal üretecinin, modülün ve Pitch II'nin oluşturduğu 220 Hz sinyalleri (yukarıdan aşağıya).....29
Şekil 4.5: Modülün 512, 1024 ve 2048 pencere boyutlarında oluşturduğu 880 Hz sinyalleri (yukarıdan aşağıya).....31
Şekil 4.6: Modülün 4096, 8192 ve 16384 pencere boyutlarında oluşturduğu 880 Hz sinyalleri (yukarıdan aşağıya).....32
Şekil 4.7: 1024 pencere boyutunda modülün (üst) ve Pitch II'nin (alt) 5 aralık tize taşıdığı insan sesi kayıtları.....34
Şekil 4.8: 2048 ve 1854 pencere boyutlarında modülün (üst) ve Pitch II'nin (alt) 5 aralık tize taşıdığı insan sesi kayıtları.....35
Şekil 4.9: Modülün 256 (üst) ve 512 (alt) pencere boyutlarında 3 aralık pese taşıdığı insan sesi kayıtları.....36
Şekil 4.10: 1024 pencere boyutunda, modülün (üst) ve Pitch II'nin (alt) 3 aralık pese

taşıdığı insan sesi kayıtları.....37
Şekil 4.11: 2048 ve 1854 pencere boyutlarında modülün (üst) ve Pitch II'nin (alt) 3 aralık pese taşıdığı insan sesi kayıtları.....38
Şekil 4.12: Modülün 4096, 8192 ve 16384 pencere boyutlarında 3 aralık pese taşıdığı insan sesi kayıtları.....39
Şekil 4.13: 512 pencere boyutunda Blackman, Hamming ve Hann pencereleriyle 2 aralık tize taşınan insan sesi kayıtları (yukarıdan aşağıya).....41
Şekil 4.14: 1024 pencere boyutunda Blackman, Hamming ve Hann pencereleriyle 2 aralık tize taşınan insan sesi kayıtları (yukarıdan aşağıya).....42
Şekil 4.15: 2048 pencere boyutunda Blackman, Hamming ve Hann pencereleriyle 2 aralık tize taşınan insan sesi kayıtları (yukarıdan aşağıya.....43
Şekil 4.16: 512 pencere boyutunda Blackman, Hamming ve Hann pencereleriyle 2 aralık tize taşınan insan sesi kayıtları (yukarıdan aşağıya).....44
Şekil 4.17: 32, 64, 128 ve 256 pencere boyutlarında modülden geçirilen insan sesi kayıtları (sol üstten saat yönünde).....45
Şekil 4.18: 512, 1024, 2048 ve 4096 pencere boyutlarında modülden geçirilen insan sesi kayıtları (sol üstten saat yönünde).....46
Şekil 4.19: 8192 (üst) ve 16384 (alt) pencere boyutlarında, modülden geçirilen insan sesi kayıtları.....47
Şekil 4.20: Testte kullanılan insan sesi kaydı.....47
Şekil 4.21: 512 pencere boyutunda modülün (üst) ve Pitch II'nin (alt) 4 aralık pese taşıdığı elektrik gitar kayıtları.....49
Şekil 4.22: 1024 pencere boyutunda modülün (üst) ve Pitch II'nin (alt) 4 aralık pese taşıdığı elektrik gitar kayıtları.....50
Şekil 4.23: 2048 pencere boyutunda modülün (üst) ve Pitch II'nin (alt) 4 aralık pese taşıdığı elektrik gitar kayıtları.....51
Şekil 4.24: 4096 (üst) ve 8192 (alt) pencere boyutlarında modülün 4 aralık pese taşıdığı elektrik gitar kayıtları.....52
Şekil 4.25: 256, 512, 1024 ve 2048 pencere boyutlarında modülün 6 aralık tize taşıdığı elektrik gitar kayıtları (sol üstten saat yönünde).....53
Şekil 4.26: 4096, 8192 ve 16384 pencere boyutlarında modülün 6 aralık tize taşıdığı elektrik gitar kayıtları (yukarıdan aşağıya).....55

Şekil 4.27: Orijinal parça, 1024 ve 2048 pencere boyutlarında değişiklikler ……57
Şekil 4.28: Modülün (üst) ve TCE'nin (alt) 4 katına uzattığı sesler.....59
Şekil 4.29: Modülün (üst) ve TCE'nin (alt) 0.25 katına sıkıştırdığı sesler.....60

## KISALTMALAR

| | | |
|---|---|---|
| DFT | : | Discrete Fourier Transform |
| FFT | : | Fast Fourier Transform |
| IDFT | : | Inverse Discrete Fourier Transform |
| IFFT | : | Inverse Fast Fourier Transform |
| PV | : | Phase Vocoder |
| STFT | : | Short Time Fourier Transform |
| wav | : | Waveform Audio File Format |

## SEMBOLLER

| | | |
|---|---|---|
| Frekans | : | f |
| Hertz | : | Hz |
| Milisaniye | : | ms |
| Periyot | : | T |
| Saniye | : | s |

# 1. GİRİŞ

Teknolojinin gelişimiyle birlikte yıllar içinde dijital ses efektleri de bir hayli gelişmiş ve kullanım alanları artmıştır. Bilgisayarların işlem kapasitesinin artmasıyla 50 yıl önce imkansız gözüken işlemler günümüzde kolaylıkla gerçekleştirilebilmektedir. İlk dijital ses efektleri 1970'li yıllarda üretilmiştir. Amerikan telekomünikasyon firması *Eventide* 0 – 200 milisaniye (ms) aralığında çalışan ilk dijital *delay*, *1745 Digital Delay Line*'ı 1970 yılında ve sesin perde değerini değiştirebilen ilk cihaz olan *H910 Harmonizer*'ı 1975 yılında piyasaya sürmüştür. Sesi bir oktav pese veya tize taşıyabilen H910 aynı zamanda 112.5 ms *delay* özelliğine de sahiptir. İlk etapta televizyon yayınlarında kullanılan H910, müzik prodüksiyonlarında da yoğun olarak kullanılan bir efekt haline gelmiştir. Alman firması *EMT*, 1976 yılında dünyanın ilk dijital *reverb* efekti *EMT 250*'yi üretmiştir. 1978 yılında Amerikalı şirket *Lexicon*'un EMT 250'yi hem işlem kapasitesi hem de parametreleri açısından geliştirerek piyasaya sürdüğü *Lexicon 224* ile birlikte dijital *reverb* efektleri müzik prodüksiyonlarında hızla popüler hale gelmiştir.[1][2]

**Şekil 1.1: Eventide H910 Harmonizer efekt cihazı**




*Kaynak:* https://www.attackmagazine.com/features/long-read/untold-history-eventide-and-the-digital-revolution/

[1] https://www.attackmagazine.com/features/long-read/untold-history-eventide-and-the-digital-revolution/

[2] https://www.vintagedigital.com.au/lexicon-224-digital-reverberator/#

Günümüzde analog efektlerin tamamı dijital olarak kullanılabilmektedir. Buna ek olarak analog düzleminde gerçekleştirilmesi mümkün olmayan efektler de dijital ortamda uygulanabilmektedir. Ayrıca analog ekipmanların dijital modelleri de üretilmektedir. *Equalizer*, kompresör gibi analog ekipmanların devre şemaları esas alınarak dijital ortamda gerçeğine çok yakın modelleri oluşturulabilmektedir. Günümüzde neredeyse tüm efektler dijital ortamda gerçekleştirilmektedir. (McPherson & Reiss 2015)

**Şekil 1.2: *Universal Audio* firmasının *Manley Massive Passive EQ* modellemesi**




*Kaynak:* https://www.uaudio.com/blog/manley-massive-passive-plugins-demo/

Bu tez çalışmasında da bir dijital sinyal işleme modülü tasarlanmıştır. Sesin perde değerini değiştirmeyi sağlayan bu modül, algoritmanın doğası gereği sesin perde değerini değiştirmeden sesin süresini de değiştirebilmektedir. Algoritmanın oluşturulmasında *Phase Vocoder* tekniği esas alınmış ve uygulanmıştır. Bu tasarım açık kaynaklı Python programlama dili vasıtasıyla gerçekleştirilmiştir. Komut satırından çalıştırılmaktadır. Algoritma bilgisayar dizininde belirtilen bir yerdeki ses dosyasını alır, verilen parametrelere göre değişikliği yapar ve yine belirtilen bir dizine dosya adını değiştirerek ses dosyasını kaydeder. Dolayısıyla işlem gerçek zamanlı değildir. Parametre belirlenir ve efektin uygulandığı yeni dosya oluşturulur. 16-Bit, mono ses dosyalarını işleyebilen modül, farklı bir *bit depth* değeri ile karşılaştığında hata verirken, *stereo* bir dosya işlenmek istediğinde dosyayı monoya dönüştürerek istenen değişikliği gerçekleştirir. Ses dosyasının örnek sayısı değeri efektin uygulanmasını etkilememekle beraber, çok yüksek örnek sayılarında işlem bekleneceği üzere daha uzun sürmektedir.

Bu modülün tasarımındaki amaç ticari olarak kullanılacak bir efekt tasarlamak değildir. Amaç, mevcut teknikleri Python dilinin avantajlarından da yararlanarak hızlı ve verimli bir biçimde algoritmaya dönüştürmek ve gerçekleştirilmesi mümkün olan efektler ile denemeler yaparak sonuçlar üzerinden yeni bakış açıları üretmeye çalışmaktır.

Tez çalışmasının ilerleyen bölümleri şu şekilde sıralanmaktadır; Bölüm 2'de çalışma için gerekli literatür taramaları yapılmıştır. Python dili, sesin perde değişimi ve *Phase Vocoder* tekniği hakkında bilgiler verilmiştir. Bölüm 3'te modülün kodu verilmiş ve gerekli açıklamalar yapılmıştır. Bölüm 4'te çeşitli ses dosyalarına modül aracılığıyla değişiklikler uygulanmıştır. Oluşturulan yeni ses dosyalarının spektrogramları çıkartılmıştır ve seslerdeki değişimler yorumlanmıştır.

# 2. GENEL BİLGİLER

## 2.1 PYTHON

Python[3], ileri seviye, yorumlamalı, nesne yönelimli bir programlama dilidir. Ücretsiz ve açık kaynak olan Python, Hollanda'lı yazılımcı Guido van Rossum tarafından geliştirilmiş ve ilk sürümü 1991 yılında yayınlanmıştır. (Tollervey 2015)

### 2.1.1 Python Dilinin Tarihçesi

1980'li yılların sonlarında, Amsterdam, Hollanda'da *Centrum Wiskunde & Informatica* (*National Research Institute for Mathematics and Computer Science*) bünyesinde, daha basit olan ABC programlama dilinin geliştirilmesinde yer alan Guido van Rossum, ABC'nin iyi yönlerinden yola çıkarak, okunması ve yazması mevcut güçlü dillere göre çok daha kolay olan Python üzerinde çalışmaya başladı. 1989 yılının Aralık ayında gerçekleştirimine başlanan Python'ın ilk sürümü Python 0.9.0, 20 Şubat 1991 tarihinde yayınlandı. Yılda 2-3 güncelleme ile sürümleri yenilenen dilin Python 2.0 sürümü 16 Ekim 2000 yılında, çok büyük değişiklikler yapıldığı için geriye dönük uygunluğun olmadığı ve halen kullanılan Python 3.0, 3 Aralık 2008'de yayınlandı. 2.0 sürümleri serisinde yazılan kodlar 3.0 serisinde çalışmadığı için kullanıcılara çeviri modülleri geliştirildi. 2020 yılı ile birlikte 2.0 serisinin son sürümü Python 2.7'nin de kullanımı durdurulacak ve Python 3.0 sürümleri mevcut kalacak. Guido van Rossum 12 Temmuz 2018 yılına kadar Python'ın gelişimini tek başına yönetmiştir. Bu tarihten itibaren dört yazılımcı ile sorumlulukları bölüşme kararı almıştır. (Tollervey 2015)

[3] https://www.python.org/ Python resmi web sitesi.

### 2.1.2 Python Dilinin Özellikleri

Python dilinin en önemli özelliği kolay ve hızlı bir şekilde okunması ve yazılmasıdır. Gelişiminde bu özellik her zaman ön planda tutulmuştur. Bu sayede öğrenmesi en kolay dillerden biri haline gelmiştir. Yorumlanan bir dil olduğu için yazılan kod Python'ın derleyicisinde satır satır çevrilir ve bu sayede hatalar hızlıca bulunabilir. Nesne yönelimli bir dildir, sınıf ve nesneleri destekler. Nesne yönelimli diller, belirli özellik ve fonksiyonlara sahip nesnelerin yaratılabildiği ve birbiriyle etkileşime sokulabildiği programlama dilleridir. Oluşturulacak nesnelerin bulundurabileceği özellikler ve fonksiyonlar, bir sınıf yaratılarak belirlenir. Oluşturulan nesneler bu sınıfın elemanları olur. Python, ücretsiz ve açık kaynaklıdır, dolayısıyla farklı işletim sistemlerinde aynı kod rahatlıkla çalıştırılabilir. Bunların yanında, geniş bir standart kütüphaneye sahip olan Python, çok güçlü bir hesap makinesidir. Bu kütüphaneler sayesinde bir çok matematiksel fonksiyon kullanıma hazırdır. Karmaşık mühendislik hesaplamalarını hızla gerçekleştirebilir. Açık kaynak olması nedeniyle üçüncü parti yazılımcılar ve kullanıcılar kendileri modül ve kütüphane geliştirip paylaşabilmektedir. Kullanıcı ara yüzü tasarımı, multimedya, veri tabanı, otomasyon, dosyalama, bilimsel ve nümerik hesaplama, metin işleme, görsel işleme, ses işleme, veri analitikleri, yapay zeka, makine öğrenmesi gibi bir çok alanda kütüphaneler geliştirilmiştir. Bu bağlamda farklı dallarda özel işlemler yapabilen NumPy, SciPy, Pandas, Matplotlib gibi bir çok kütüphane mevcuttur. Bu tez çalışmasında da NumPy[4] ve SciPy[5] kütüphaneleri kullanılmıştır. C, C# ve C++ dilleri ile birlikte çalışabilen CPython, Cython, IronPython gibi yazılımlar da geliştirilmiştir. Bu sayede farklı diller ile paralel olarak da çalışabilen Python, bir çok farklı alanda kullanılmaya başlamıştır (Phillips 2015, Tollervey 2015).

[4] https://numpy.org/ NumPy kütüphanesinin resmi web sitesi.

[5] https://scipy.org/ SciPy kütüphanesinin resmi web sitesi.

### 2.1.3 Python Dilinin Kullanım Alanları

Python günümüzde farklı dallarda kullanılmaktadır. İşletim sistemleri, internet framework tasarımı ve uygulamaları, nesne ve renk tanıma gibi makina öğrenmesi uygulamaları, büyük veri işlenmesi, bilimsel hesaplamalar ve uygulamalar, yapay zeka geliştirme, programlama dili geliştirme, prototip testleri, görsel işleme uygulamaları bunlara örnek verilebilir. NASA, CERN, Google, Yahoo!, YouTube, Facebook, Amazon, Spotify gibi şirketler Python kullanıcılarından bazılardır. Özellikle günümüzde hızla gelişmekte olan Yapay Zeka ve Makina Öğrenmesi alanlarında büyük verileri hızlı işleyebildiği ve hızlı test yapılabildiği için sıkça kullanılmaktadır. Python, 2019 yılı itibariyle Java ve C'nin ardından en çok kullanılan üçüncü programlama dilidir. Ruby, Swift, Swift Go, Groovy, Julia, Boo, Cobra, ECMAScript gibi bir çok programlama diline ilham kaynağı olmuştur (Tollervey 2015).

### 2.1.4 Python Kullanılarak Yapılmış Benzer Çalışmalar

Python programlama dili kullanılarak yapılmış benzer çalışmalar az da olsa mevcuttur. Sesin süresini değiştirmeden perde değerine müdahale eden, Christopher Woodall tarafından tasarlanan pitch-shifter-py[6] programı vardır. Sesin perde değerine müdahale etmeden süresini değiştiren program olarak, özellikle kısa seslerin uzatılmasında kullanılan, Paul Nasca tarafından C++ diliyle yazılan PaulStretch[7] programının nispeten daha basit olan bir Python implementasyonu[8] mevcuttur. Ses dosyalarına yönelik olan bu programlar, Python ile yazılmış nadir başarılı sinyal işleme programlarındandır. Bu çalışma kapsamında bu programlar da incelenmiş ve sonuç bölümünde çeşitli karşılaştırmalar yapılmıştır.

[6] https://github.com/cwoodall/pitch-shifter-py

[7] http://hypermammut.sourceforge.net/paulstretch/

[8] https://github.com/paulnasca/paulstretch_python

## 2.2 SESİN PERDE DEĞİŞİMİ

İnsan kulağının işitebileceği frekans aralığı 20 Hz – 20.000 Hz olarak kabul edilir (Önen 2017, s.27). Duyduğumuz tüm sesler bu aralıkta yer alır. Örneğin A4 (la) notası saf tonda 440 Hz'e tekabül eder. A5 notası, yani A4'ün bir oktav üstündeki la notası ise 880 Hz'e tekabül eder. A4'ün bir pes oktavı olan A3 ise 220 Hz'dir.

**Tablo 2.1: Notaların Hz cinsinden frekans değerleri**

| | **0** | **1** | **2** | **3** | **4** | **5** | **6** | **7** | **8** |
|---|---|---|---|---|---|---|---|---|---|
| **C** | 16.35 | 32.7 | 65.41 | 130.81 | 261.63 | 523.25 | 1046.5 | 2093 | 4186.01 |
| **C#** | 17.32 | 34.65 | 69.3 | 138.59 | 277.18 | 554.27 | 1108.73 | 2217.46 | 4434.92 |
| **D** | 18.35 | 36.71 | 73.42 | 146.83 | 293.66 | 587.33 | 1174.66 | 2349.32 | 4698.64 |
| **D#** | 19.45 | 38.89 | 77.78 | 155.56 | 311.13 | 622.25 | 1244.51 | 2489.02 | 4978.03 |
| **E** | 20.6 | 41.2 | 82.41 | 164.81 | 329.63 | 659.26 | 1318.51 | 2637.02 | 5274.04 |
| **F** | 21.83 | 43.54 | 87.31 | 174.61 | 349.23 | 698.46 | 1396.91 | 2793.83 | 5587.65 |
| **F#** | 23.12 | 46.25 | 92.5 | 185 | 369.99 | 739.99 | 1479.98 | 2959.96 | 5919.91 |
| **G** | 24.5 | 49 | 98 | 196 | 392 | 783.99 | 1567.98 | 3135.96 | 6271.93 |
| **G#** | 25.96 | 51.91 | 103.83 | 207.65 | 415.3 | 830.61 | 1661.22 | 3322.44 | 6644.88 |
| **A** | 27.7 | 55 | 110 | 220 | 440 | 880 | 1760 | 3520 | 7040 |
| **A#** | 29.14 | 58.27 | 116.54 | 233.08 | 466.16 | 932.33 | 1864.66 | 3729.31 | 7458.62 |
| **B** | 30.87 | 61.74 | 123.47 | 246.94 | 493.88 | 987.77 | 1975.53 | 3951.07 | 7901.13 |

Batı müziğinde genel anlamda 12 ton eşit tamperamanlı sistem kullanılmaktadır. Buna göre bir oktav 12 yarım sesten (aralıktan) oluşmaktadır. Her yarım ses belirli bir notaya tekabül eder. Bir oktav 12 aralıktan meydana geldiğinden ve bir tam oktav değişiminin frekansa etkisi bilindiğinden, frekans değeri ve ses aralığı sayısı arasında aşağıdaki formül elde edilmiştir (Cangal 2008, McPherson & Reiss 2015).

$$f_{son} = 2^{(\frac{s}{12})} * f_{ilk} \qquad \textbf{(2.1)}$$

Burada $f_{ilk}$ mevcut frekans, $f_{son}$ elde edilen frekans ve $s$ ses aralığı sayısını belirtmektedir. Buna göre örneğin 440 Hz değerini sırasıyla 5 ve 12 yarım ses tize ve 6 yarım ses pese çekmek istersek;

$$f_{+5} = 2^{(\frac{5}{12})} * 440 = 587.33\,Hz$$
$$f_{+12} = 2^{(\frac{12}{12})} * 440 = 880\,Hz$$
$$f_{-6} = 2^{(\frac{-6}{12})} * 440 = 311.12\,Hz$$

değerlerini elde ederiz. Bunlar sırasıyla D5 (re), A5 (la) ve D#4 (re diyez) notalarına tekabül etmektedir.

Bu bilgilerin ışığında, mevcut bir sesin perdesini bir oktav tize taşımak için sesin içindeki frekansların iki katına çıkması gerekmektedir. Ses dosyası iki kat hızlı çalınarak frekanslar iki katına çıkartılabilir, ancak bu şekilde ses dosyasının süresi de yarı yarıya azalacaktır. O halde önce sesin perde değerlerine etki etmeden uzunluğunu iki katına çıkarıp ardından iki kat hızlı çalarak, başlangıçtaki ses dosyasının frekans değerlerini iki katına çıkarmış oluruz. Bu sayede elimizdeki ses dosyasının uzunluğu aynı kalırken perdesini bir oktav tize taşımayı başarabiliriz.

Bu mantıktan yola çıkarak, bir sesin perdesini tize taşımak için, öncelikle değiştirilecek aralık sayısına bağlı olan katsayı hesaplanmalıdır. Ses dosyası bu katsayı oranında uzatılmalı, ardından yine katsayıyla orantılı hızda çalınmalıdır. Örneğin perdeyi 2 aralık tize taşımak için gereken katsayı; $2^{(\frac{2}{12})} \cong 1.12246$'dır. Buna göre ses dosyasının uzunluğu perde değişmeden 1.12246 katına çıkarılmalı, ardından dosya yine 1.12246 katı hızda çalınmalıdır. Benzer şekilde, bir sesin perdesini pese taşımak için, öncelikle değiştirilecek aralık sayısına bağlı katsayı hesaplanmalıdır. Ardından ses dosyası bu katsayı oranında sıkıştırılmalı ve katsayıyla orantılı olarak yavaşlatılarak çalınmalıdır. Örneğin perdeyi 1 aralık pese taşımak için gereken katsayı; $2^{(\frac{-1}{12})} \cong 0.94387$'dir. Buna göre ses dosyasının uzunluğu perde değişmeden 0.94387 oranında sıkıştırılmalı, ardından 0.94387 oranında yavaşlatılarak çalınmalıdır. Özetle, bir sesin perdesini tize taşımak için ilk olarak sesi uzatmak, pese taşımak için ise sıkıştırmak gerekmektedir.

Sesin yavaşlatılması veya hızlandırılması ise dijital ortamda sesin örneklenme hızı üzerinden gerçekleştirilebilir. Bu işlemlerin gerçekleştirilmesinde yaygın olarak kullanılan yöntem Phase Vocoder tekniğidir. İlk olarak 1966 yılında Flanagan ve Golden tarafından geliştirilmeye başlanan Phase Vocoder, yıllar içerisinde Zaman Genişletme Sıkıştırma, Perde Değiştirici, *Auto-Tune* gibi bir çok ses efektinin geliştirilmesinde kullanılmıştır (McPherson & Reiss 2015, Roads 2001).

## 2.3 PHASE VOCODER

Phase Vocoder (PV), faz bilgileri kullanılarak sesin zaman ve frekans düzlemlerinde işlenmesi tekniğidir. Ses dijital olarak analiz edilir ve yeniden oluşturulur. Çoğu ses efekti (filtre, kompresör, harmonik efektler, *delay* vb.) sinyale zaman düzleminde uygulanır. PV efektleri ise frekans düzleminde çalışır. Perde değiştirme, zaman genişletme, *auto-tune*, insan sesine uygulanan robotlaştırma, fısıltıya dönüştürme gibi efektler frekans düzleminde yapılan işlemlerdir. PV'da işlenecek olan ses dosyası eşit parçalara bölünür, frekans düzleminde bu parçaların frekans, genlik ve faz değerleri istenen işleme göre değiştirilir ve zaman düzlemine geri dönülerek parçalar tekrar birleştirilir ve ses dosyası oluşturulur. Zaman düzlemi ile frekans düzlemi arasındaki bu geçişler Fourier Transformları sayesinde olur (McPherson & Reiss 2015, Roads 2001).

### 2.3.1 Fourier Analizi

*Jean-Baptiste Joseph, Baron de Fourier*, 1768 – 1830 yılları arasında yaşamış Fransız bir mühendis ve aristokrattır. Fourier, periyodik sinyallerin, onları oluşturan farklı genlik, frekans ve fazlara sahip basit sinüs dalgalarına ayrıştırılabileceğini ortaya koyan bir teori geliştirmiştir. Ancak ciddi hesaplama gerektirdiği için bilgisayarların gelişimine kadar kullanımı çok güç olmuştur. 1960'lı yıllarda James Cooley ve John Tukey, Fourier transformlarını dijital ortamda geliştirerek *fast Fourier Transform* (FFT) algoritmalarını oluşturmuşlardır. FFT algoritmaları sayesinde hesaplamalar ciddi şekilde kolaylaşmıştır. Fourier Serileri ve Fourier Transformu periyodik sinyalleri, onları oluşturan sinüs dalgalarına ayırmak için gerekli algoritmaları sağlamaktadır.

Ancak gerçek hayatta karşılaştığımız ve işlemek istediğimiz sesler periyodik değil, komplekstir. Periyodik sesler, T = (1/f) sürede bir kez tekrar eden, sinüs, kare, üçgen veya testere dalga formunda olan basit seslerdir (Roads 2001, s. 245). Gündelik hayatta, kulağımızla işittiğimiz sesler ise bir çok farklı periyodik sesten oluşan, karmaşık ve farklı spektral bilgiler içeren seslerdir. Fourier Transformu zamanda değişken sesler için uyarlanmış ve *Short-Time Fourier Transform* (STFT) geliştirilmiştir. STFT tekniğinde, sinyalden belirli bir örnek sayısında küçük bir parça alınır ve bu parçaya FFT uygulanır. Bu FFT işleminin sonucunda sinyalin *discrete Fourier Transformu* (DFT) elde edilmiş olur. DFT, FFT'si alınan sinyal parçasının içindeki her bir sinüs dalgasının genlik ve faz değerlerini bulundurur. Bu sayede sinyalin içindeki frekansların zamanla nasıl değiştiği de gözlenebilir. Bu şekilde zaman düzleminde gerçekleştirilmesi çok daha zor ya da imkansız olacak işlemleri yapmak mümkün hale gelmiştir. Efektlerin yanı sıra spektogramlar da STFT'nun en yaygın kullanım alanlarından bir tanesidir. FFT ve DFT tersine de çalışabilen fonksiyonlardır. Frekans düzleminden zaman düzlemine dönmek için *inverse Fast Fourier Transform* (IFFT) / *inverse discrete Fourier Transform* (IDFT) kullanılır. FFT alınarak frekans düzlemine alınan ses parçacıkları, IFFT kullanılarak tekrar zaman düzlemine alınır (McPherson & Reiss 2015, Puckette 2007, Smith 2007, Roads 2001).

### 2.3.2 Phase Vocoder İşlem Akışı

PV işleminde ilk olarak ses dosyasının bölüneceği parçacıkların uzunluğu (örnek sayısı) *N* seçilir. Seçilen örnek sayısına pencere boyutu (*window* size) adı verilir. Pencere boyutu tek seferde analiz edilecek ve işlenecek örnek sayısıdır. Sinyal yine *N* uzunluğunda bir pencere fonksiyonu (*window function*) ile çarpılır. Pencere fonksiyonu artarda *N* sayısında örnek için sıfır hariç bir değerken, diğer tüm noktalarda sıfırdır. Bu şekilde sadece seçilen penceredeki *N* örnek kalır ve diğerleri sıfır olur. Sinyal pencereden geçtikten sonra, FFT uygulanır. *N* adet örnek içerdiği için FFT algoritmasından geçecek örnek sayısı (FFT boyutu) *M*, *N*'den büyük veya ona eşit olmalıdır. Hesaplamaların verimliliği açısından FFT boyutunun 2'nin kuvveti olarak seçilmesi sağlıklı olacaktır (Roads 2001, s. 255). Sinyale pencere fonksiyonu ve FFT uygulanarak STFT tamamlanmış olur. FFT sonucunda, frekans düzleminde genlik ve

faz değerleri bilinen *M* adet nokta elde edilmiş olur. FFT çıkış olarak, her frekans kutucuğu için, genlik değerini belirten bir tam sayı ve faz açısını belirten karmaşık bir sayıdan oluşan ikili elemanlar verir. Gerçekleştirilmek istenen efekte göre gerekli değişiklikler bu değerler üzerinden yapılır. İşlenen ses penceresine IFFT uygulanır ve zaman düzleminde *M* adet örnek elde edilir. Elde edilen örnekler, işlenmiş final ses dosyasını hafızada tutan *output buffer*'a eklenir ve ilk pencerenin işlemi tamamlanmış olur. Sıradaki pencereye ilerlemek için, *N*'in 2'nin katlarından birine bölünmesiyle elde edilen adım sayısı *H* seçilir ve hem ses dosyasında hem de *output buffer*'da *H* kadar ilerlenir. Sıradaki pencereye aynı işlemler uygulanır, *output buffer*'a eklenir ve tekrar adım sayısı kadar ilerlenir. Bu döngü ses dosyasının tamamı işlenene kadar devam eder. Adım sayısı nadiren eşit olmakla birlikte, çoğu zaman pencere boyutundan küçük seçilir. Yalnızca zaman genişletme sıkıştırma işlemlerinde adım sayısı, analiz ve sentez adımlarında aynı değildir. Zaman efektinin doğası gereği giriş sinyalinde adım sayısı *H* kadar ilerlenirken, *output buffer*'ın oluşturulması esnasında adım sayısı *H* genişletme/sıkıştırma oranıyla çarpılır ve elde edilen değer ses yeniden sentezlenirken adım sayısı olarak kullanılır. Bu üst üste ekleme işlemine *overlap - add* metodu adı verilir. Kesişen pencereler işlem öncesinde ve sonrasında analiz edilerek, zamanı uygun final ses dosyası elde edilir. Pencere boyutu ile adım sayısındaki oran, pencerelerin kesişim miktarını verir.

$$kesişim\ oranı = 1 - \frac{adım\ sayısı}{pencere\ boyutu} \qquad \textbf{(2.2)}$$

Örneğin 1024 örneklik bir pencerenin 128 adım sayısıyla analizinde kesişim oranı;

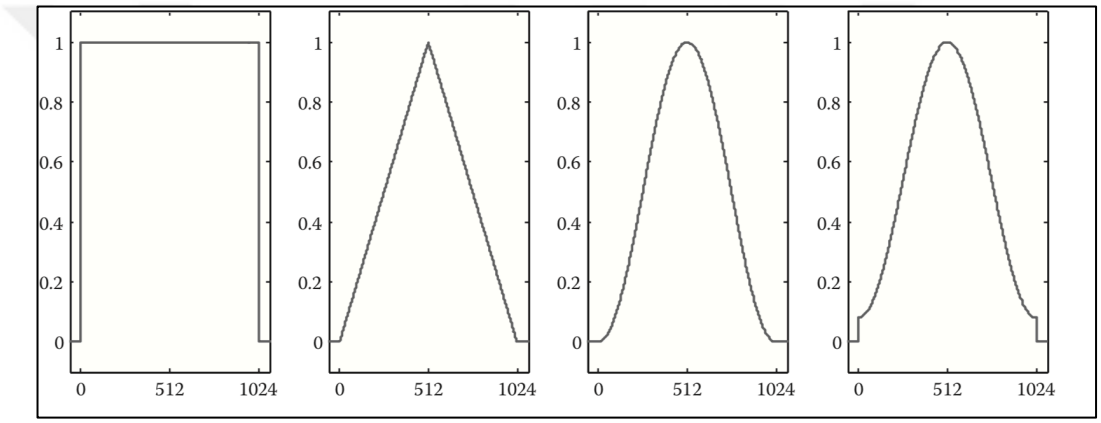
$$kesişim\ oranı = 1 - \frac{128}{1024} = 0.875$$ 'tir.

Yani ardışık pencereler birbirleriyle yüzde 87.5 oranında kesişmektedirler. PV tekniğinde verimli analiz ve sentez için kesişim oranı en az yüzde 50 olmalıdır. Oran yükseldikçe işlemin kalitesi artar ancak işlem süresi de kayda değer biçimde uzar (McPherson & Reiss 2015, Puckette 2007, Smith 2007, Roads 2001).

### 2.3.2.1 Pencere fonksiyonu

PV tekniğinde ses, *N* boyutunda pencereler halinde işleme sokulur ve her frekansın zamanda nerede olduğu bilinebilir. Ancak bu işlemin bir yan etkisi de vardır. Pencereye alınan sesin baş ve son kısımlarında frekansların genlik değerleri birbirine karışır ve kenarlarda birikmeler meydana gelir. Bu birikmelere spektral sızıntı denir. Bu oluşumları azaltmak için pencere fonksiyonları kullanılır. Çan eğrisi şeklinde pencere fonksiyonları genelde en iyi çözümü sağlarlar (Roads, 2001).

**Şekil 2.1: Dikdörtgen, Bartlett, Hann ve Hamming pencere fonksiyonları**

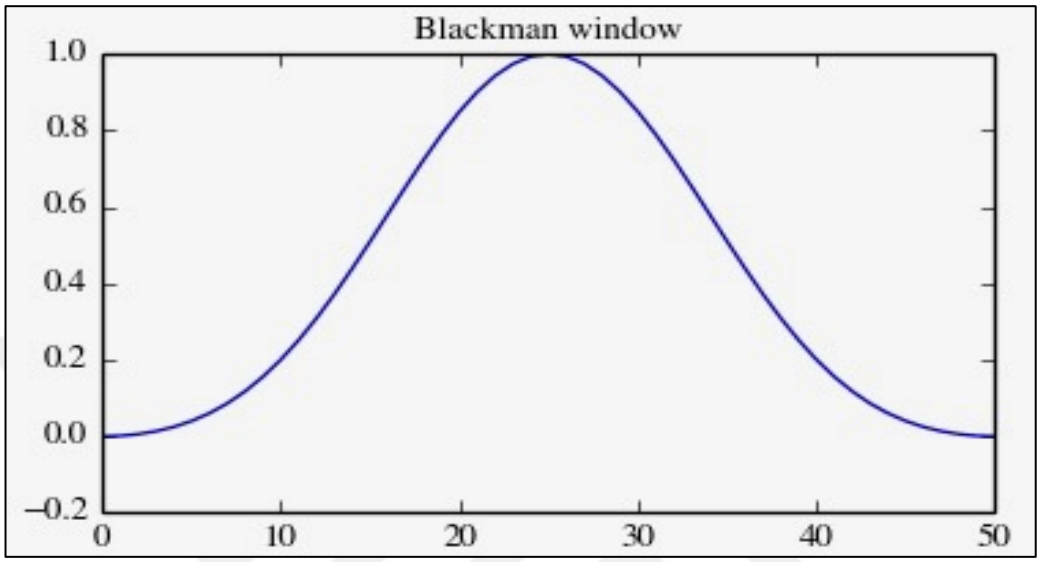



*Kaynak:* Reiss, J. M. & McPherson A. P., Audio Effects, 2015, UK: CRC Press

PV işleminde genellikle kullanılan pencere fonksiyonları Hann penceresi (Julius von Hann), Hamming penceresi (Richard W. Hamming), Blackman penceresi (Ralph B. Blackman) ve Kaiser penceresidir (James Kaiser). Bu pencere fonksiyonlarının tamamı çan eğrisi şekline sahiptir ve birikmeler azdır. Oluşturulmak istenen efekte ve değişiklik yapılacak sesin özelliklerine göre bunlardan bir tanesi en uygun sonucu verecektir. Hann penceresi nispeten dar bir eğriye sahiptir ve frekans cevabı merkezde düzenli yükselmektedir. Hamming penceresi ise daha geniş bir alanı kaplar ve frekans cevabı dış kısımlarda dengeliyken merkeze gelindiğinde ani yükselir. Hann ve Hamming pencerelerine göre daha dar bir frekans cevabına sahip olan Blackman penceresi, Hamming penceresi kadar olmasa da merkezde ani bir yükselişe sahiptir. Fazla birikmeye sebep oldukları için dikdörtgen pencere fonksiyonlarından kaçınılmalıdır.

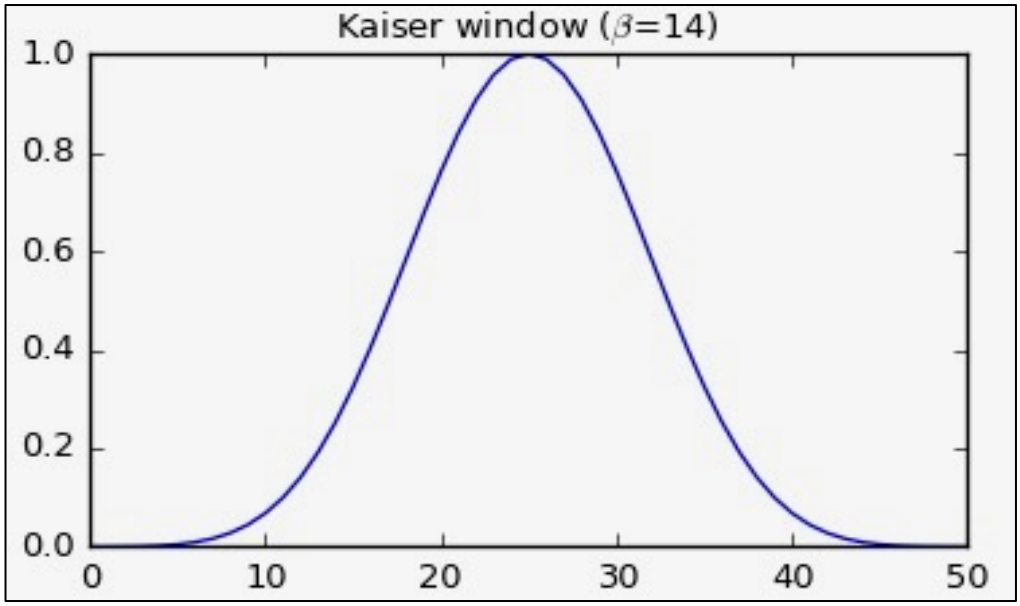
Pencere fonksiyonu sinyalle çarpılarak uygulanır. Sinyalin pencerenin dışına denk gelen kısımları böylece sıfıra eşitlenir (McPherson & Reiss 2015, Roads 2001).

**Şekil 2.2: Blackman pencere fonksiyonu**




*Kaynak:* https://docs.scipy.org/doc/scipy-0.14.0/reference/generated/scipy.signal.blackman.html

**Şekil 2.3: Kaiser pencere fonksiyonu**




*Kaynak:* https://docs.scipy.org/doc/scipy-0.18.1/reference/generated/scipy.signal.kaiser.html

#### 2.3.2.2 Sesin FFT ile analizi

FFT, sonuç olarak DFT veren bir algoritmadır. Zamanda *M* adet noktayı alır, frekans düzleminde eşit aralıklarla bölünmüş *M* adet frekans kutucuğu verir. Zaman düzleminden frekans düzlemine yapılan bu geçiş PV'ın analiz adımıdır. FFT boyutu pencere boyutundan asla küçük olmamalıdır. Pencere boyutunun iki katına yakın olacak şekilde 2'nin kuvvetlerinden birinin seçilmesi sağlıklıdır. Örneğin pencere boyutu 256 örnek büyüklüğünde seçilirse, FFT boyutunun 512 olması sağlıklı olacaktır. FFT' deki diğer 256 örnek sıfıra eşittir. Seçilen *M* değerine göre *M*-noktalı FFT uygulanır. FFT boyutu *M* ve pencere boyutu *N* seçimleri işlemin çözünürlüğünü belirler. Pencere boyutu küçüldükçe sesin analiz edileceği zaman aralıkları kısalacağı için zaman çözünürlüğü artar ve daha kısa sürelerdeki değişimler gözlenebilir. Ancak buna bağlı olarak FFT sonucunda M adet eşit aralıklı frekans kutucuğu elde edilir ve frekans çözünürlüğü düşük olur. Daha büyük bir pencere seçilirse frekansların zamandaki değişimi daha uzun sürede gözlenebilir ancak bu sayede frekansların değerlerine dair daha sağlıklı bilgi elde edilir. Özetle pencere boyutu ve FFT boyutunun arasındaki ters orantı işlemin çözünürlüğünü belirler ve erişilmek istenen ses efektine göre seçilmelidirler (McPherson & Reiss 2015, Roads 2001).

#### 2.3.3.3 Frekans düzlemi işlemleri

Zaman düzleminden frekans düzlemine geçildiğinde uygulanabilir hale gelen PV tekniğini bir çok farklı şekilde değerlendirmek mümkündür. Makine dinlemesi ve müzik enformatiği alanlarında sinyal içeriği analizi, perde belirlenmesi (*pitch detection*) ve *transient* seslerin ayrıştırılması gibi fonksiyonları vardır. Genellikle insan sesi efektlerinde kullanılan PV tekniği ile gerçekleştirilebilecek ses efektlerinden bazıları ise zaman genişletme sıkıştırma, perde değiştirme, insan sesine yapılan robotlaştırma/sentetikleştirme ve fısıltıya dönüştürmedir. Frekans düzleminde, uygulanacak efekte göre frekansların genlik ve faz bilgilerine müdahale edilir. Her işlem için farklı denklem ve algoritmalar oluşturulur. Robotlaştırma/sentetikleştirme efektlerinde, genlik değerlerinde değişiklik yapılmadan yalnızca faz açısı değerini bulunduran kompleks sayılar sıfırlanır ve faz bilgisi tamamen ortadan kaldırılır. Bu

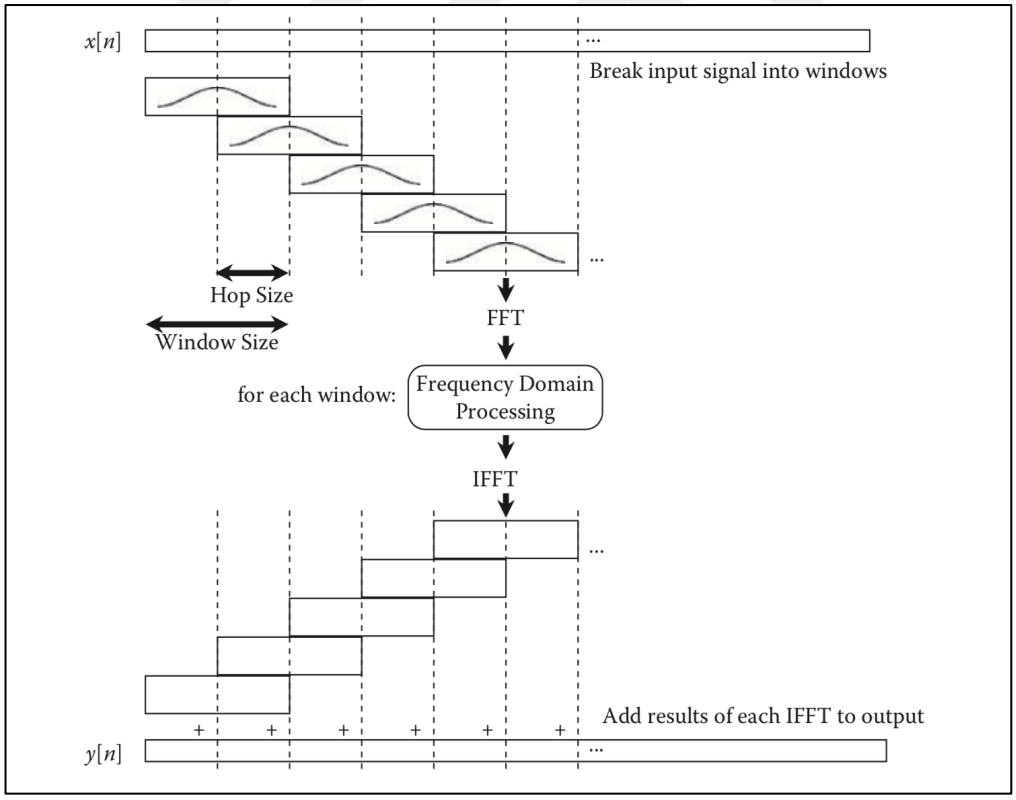
şekilde sesin baskın frekanslarına müdahale edilmez ancak zaman bilgisi olmadığı için devamlılığı yok olur ve oluşan fazlalık sesler vasıtasıyla efekt elde edilir. Fısıltı efektlerinde, faz açısı değerlerine rastgele değerler atanır. Bu sayede yine genlik değerleri değişmeden baskın frekanslar korunur ancak seslerin perde algıları kaybolur ve fısıltıya benzer efektler elde edilir. Faz değerlerine bu şekilde değişiklikler yapmak seste *reverb* efektini andıran (*phasiness*) bir etki bırakır. Bunun sebeplerinden bir tanesi ardışık pencerelerde bulunan aynı frekans kutucuklarının faz değerlerinin birbirleriyle tutarlı olmamasıdır. Özellikle ani (*transient*) ve yüksek enerjili seslerde, faz değerlerinin tutarlı olmaması sebebiyle frekans kutucukları arasında genlik geçişleri olur ve *transient* sesler bulanıklaşarak fazlalık sesler oluşturur. Frekans düzleminde, genlik değeri en yüksek olan frekans kutucukları takip edilerek bu değerlerin yayılması önlenebilir. Perde değiştirme efektlerinde ise ardışık pencerelerin faz bilgileri korunarak birbirine bağlanır. Bir penceredeki frekans kutucuklarının faz değerleri, bir önceki pencerenin frekans kutucuklarının bıraktığı yerden devam eder. Bu sayede süreklilik kaybolmadan ve fazlalık seslerin oluşumu azaltılarak sesin perdesi değiştirilir. Ancak bu işleminde bir handikabı vardır. Bir önceki penceredeki frekans kutucuklarına göre senkronize edilen faz değerleri, aynı pencerede bulunan frekans kutucukları arasında farklılıklar gösterebilir. Hem bir önceki penceredeki faz bilgilerini, hem de mevcut penceredeki faz değerlerini kusursuz olarak eşitlemek mümkün değildir (Smith, 2007). Küçük farklılıklar olması kaçınılmazdır. Perde değiştirme işlemlerinde yapılan bu faz takipleri, gerçek zamanlı çalışan perde değiştiricilerin neden gecikmeli çalıştığını da ortaya koymaktadır. Sesten alınan bir pencerenin faz değerleri, bir önceki pencereye göre senkronize edildiğinden dolayı, perde değiştiriciler en az adım sayısı kadar önden çalışmalıdır. Bu tez çalışmasında çevrimdışı çalışan modüller geliştirildiğinden, sesin tamamını işleme alabilmek için, değişiklik yapılacak ses dosyasının başına ve sonuna sıfır değerinde boş örnekler eklenmiştir. Bu işleme *zero-padding* adı verilir. Eklenen bu örnekler bilgi ve dolayısıyla ses içermez. Bu sayede istenen faz bağlantıları sesin tamamı işlenerek sağlanabilir. Ayrıca, faz takibi yapılan işlemler, faz değerlerinin sıfırlandığı ya da rastgele atandığı işlemlerden daha çok işlem gücü gerektirir. Faz değerlerinin, bir önceki pencerenin bıraktığı yerden devam etmesini mümkün kılmak için gereken hesaplamalar, faz bilgilerinin sıfırlanması ya da rastgele atanması için gereken hesaplamalardan daha fazladır. Dolayısıyla faz takibi yapılan işlemler daha

uzun sürer. Genel olarak PV efektleri ele alındığında, bazı işlemlerde faz ve genlik bilgilerinde yapılan değişikliklerin oluşturduğu sesler fazlalık ve istenmeyen ses olarak nitelendirilebilirken, başka işlemlerde erişilmek istenen efektin yapıtaşı olduğu söylenebilir. Ayrıca her efektin maksimum verimle çalışabileceği pencere boyutları farklıdır. Uygun değerlerin sağlanması için deneme – yanılma şeklinde ilerlemek sağlıklı olacaktır (McPherson & Reiss 2015, Smith 2007, Roads 2001).

### 2.3.3.4 Sesin IFFT ile yeniden oluşturulması

Frekans düzleminde yapılan değişikliklerin ardından IFFT kullanılarak zaman düzlemine geri dönülür. Bu PV tekniğinin yeniden oluşturma adımıdır. Zaman düzleminde M adet nokta elde edilir ve *output buffer*'a eklenir.

**Şekil 2.4: PV işlem akışı**




*Kaynak:* Reiss, J. M. & McPherson A. P., Audio Effects, 2015, UK: CRC Press

#### 2.3.3.5 Üst üste ekleme işlemi

PV tekniği ile her pencerenin frekans içeriği incelenir, değiştirilir ve zaman düzleminde tekrar oluşturulur. Pencerelerin ilerleyişi bu işlemin gerçekleşmesinin önemli bir bölümüdür. Bir pencere analiz edilip yeniden oluşturulduğunda, sinyalde adım sayısı *H* kadar ilerlenir. Buna göre bir penceredeki ses yeniden oluşturulduğunda sıradaki pencereye geçmek için ses dosyasında *H* örnek ilerlenir ve yeni pencere bu örnekten başlar. Adım sayısı asla pencere boyutundan büyük olmamalıdır. Aksi takdirde sinyalin işleme girmeyen kısımları olacaktır. PV efektlerinde adım sayısı küçüldükçe işlemin çözünürlüğü artar, ancak bu daha fazla işlem gücü gerektirir. Adım sayısı genellikle pencere boyutunun yarısı, çeyreği veya çeyreğinin yarısı olarak seçilir. Bu değerlerde kesişim oranları sırasıyla yüzde 50, 75 ve 87.5 olur. Yüzde 87.5 kesişim oranı işlem kalitesi ve süresi bakımından verimli bir seçenek olacaktır. Sinyalin pencereden geçen bölümleri *output buffer*'a kesişen kısımları üst üste gelecek şekilde eklenerek ses dosyasının tamamına efekt uygulanmış olur. Özetle, PV tekniğinde adım sayısı ve pencere boyutu seçimleri çok önemlidir ve uygulanmak istenen efekte göre değişik seçimler gerekir. Yanlış seçimler zaten *artifact* oluşumlarını engellemek için uygulanan pencere fonksiyonunun *artifact* oluşturmasına yol açacaktır. Ayrıca analiz bölümünde kesişim miktarını mümkün olduğunca arttırmak, istenmeyen seslerin engellenmesine yardımcı olacaktır. Yani analiz bölümünde, pencere fonksiyonuna etkileri de göz önüne alınarak mümkün olan en küçük adım sayısı seçilmelidir (McPherson & Reiss 2015, Smith 2007, Roads 2001).

# 3. VERİ VE YÖNTEM

Bu bölümde modüllerin çalışma prensipleri ile akışı açıklanmıştır ve kodu paylaşılmıştır.

## 3.1 MODÜLLERİN İŞLEM AKIŞI

2. bölümde açıklandığı gibi, perde değerinde değişiklik yapılmak istenen ses, öncelikle istenen değişim miktarına bağlı olan bir faktöre ($f = 2^{n/12}$) göre PV tekniği kullanılarak uzatılır ya da sıkıştırılır, ardından yine aynı faktöre bağlı olarak hızlandırılır veya yavaşlatılır. Bu bağlamda programın çalışma akışı şu şekilde olmalıdır;

i. Faktör hesaplanır, pencere boyutu ve adım sayısı belirlenir. *Zero-padding* uygulanarak sesin başına adım sayısı kadar, sonuna ise adım sayısı ve pencere boyutunun toplamı kadar boş örnek eklenir.
ii. Sesten, bir tanesi adım sayısı kadar ileriden başlamak üzere, pencere boyutu kadar iki adet örnek alınır.
iii. Alınan örnekler bir pencere fonksiyonu ile çarpılır ve FFT gerçekleştirilir.
iv. Oluşacak faz problemlerini engellemek için frekans düzleminde, adım sayısı kadar ileride olan pencerenin faz açısı bilgileri, geride olan pencerenin bıraktığı yerden başlayacak şekilde düzenlenir ve kümülatif faz bilgisine işlenir.
v. Önde olan pencereye IFFT uygulanır ve zaman düzlemine geri dönülür. Elde edilen ses *output buffer*'a eklenir.
vi. Seste adım sayısı kadar ilerlenir ve ii. maddeye dönülür. Seste pencere boyutu ve adım sayısının toplamından daha az örnek kaldığı takdirde vii. maddeye geçilir.
vii. Elde edilen ses örnekleri faktöre göre yeniden düzenlenir ve değişiklik tamamlanır.

Buna göre öncelikle değişiklik yapılacak ses dosyasını ve değişiklik miktarını giriş olarak alacak bir *pitch_shift* programı yazılacaktır. Bu program, sesin kaydırılacağı

aralık miktarı n'i kullanarak faktörü hesaplayacak ve bu bilgileri öncelikle sesin başına ve sonuna boş örnek ekleyecek olan *zero_padding* fonksiyonuna, ardından sesin STFT'sinin hesaplanacağı *phase_vocoder* fonksiyonuna ve son olarak da örnekleri yeniden düzenleyecek olan *resample* fonksiyonuna göndererek istenen değişikliği gerçekleştirecektir.

Modüllerin tasarımlarında NumPy ve SciPy kütüphaneleri kullanılmıştır. NumPy hesaplamalar ve matematiksel fonksiyonlar için, SciPy ise wav formatındaki dosyaları okuyabilmek ve wav formatında dosya oluşturabilmek için kullanılmıştır. NumPy kütüphanesinde, PV tekniğinde sıkça kullanılan, frekans karışmalarını ve birikmelerini en aza indirgeyen ve en verimli pencere fonksiyonlarından olan Hann, Hamming ve Blackman pencere fonksiyonları mevcuttur. Modüllerde ilk etapta, kenarlarında frekans cevabı nispeten daha dengeli sıfırlanan Hann pencere fonksiyonu kullanılacaktır. Modüllerin işleyişinde, çıkış dosyasında yüzde 87.5 kesişim oranı gözetilmiştir. Bu nedenle adım sayısı, her zaman pencere boyutunun sekizde biri olacak şekilde belirlenmiştir. Kodda değişiklik yaparak bu oranı değiştirmek mümkündür. Bölüm 4'te farklı pencere fonksiyon tipleri ve pencere boyutlarıyla gerçekleştirilen değişikliklerin sonuçları paylaşılmıştır.

## 3.2 ALGORİTMALARIN OLUŞTURULMASI

İlk olarak, değişiklik yapılacak ses dosyasının tamamının işlenebilmesi için, ses dosyasının başına ve sonuna boş örnekler ekleyecek olan *zero_padding* fonksiyonu yazılacaktır. Bu fonksiyon giriş olarak iki veri alacaktır. Bunlar değişiklik yapılacak olan ses dosyası ve pencere boyutudur. Fonksiyonun akışı Şekil 3.1'de incelenebilir.

**Şekil 3.1: *zero_padding* fonksiyonunun işlem akışı**

```
def zero_padding(wave, window_size):
    sound = wave[1]
    window_size = int(window_size)
    hop_size = int(window_size / 8)
    sound = np.append(sound, np.zeros(window_size + hop_size))
    sound = np.append(np.zeros(hop_size), sound)
    return sound
```

Ses dosyasında bulunan örnekler okunur. Adım sayısı, pencere boyutunun sekizde biri olacak şekilde belirlenir. Ses dosyasının başına adım sayısı kadar, sonuna ise adım sayısı ile pencere boyutunun toplamı kadar, sıfır değerinde boş örnekler eklenir. Boş örnekler eklenen ses dosyası bir sonraki adıma hazırdır. (Downey 2015, 2014)

İkinci olarak, sese STFT uygulayarak uzatacak/sıkıştıracak olan *phase_vocoder* fonksiyonu yazılacaktır. Bu fonksiyon giriş olarak üç veri alacaktır. Bunlar sırasıyla; değişiklik yapılacak olan ses dosyası, kaydırılacak aralık sayısı n'e bağlı faktör ve pencere boyutudur. Fonksiyonun işlem akışı Şekil 3.2'de görülmektedir.

**Şekil 3.2: *phase_vocoder* fonksiyonunun işlem akışı**

```
def phase_vocoder(sound, factor, window_size):
    window = np.hanning(window_size)
    factor = 1.0/factor
    sound *= (1.0/32768.0)
    hop_size = int(window_size/8)
    sound_len = int(len(sound))
    phases = np.zeros(window_size)
    output = np.zeros(int(sound_len / factor + window_size + hop_size))

    for i in np.arange(0, sound_len - window_size - hop_size, hop_size*factor):
        i = int(i)
        ws1 = sound[i: i + window_size]
        ws2 = sound[i + hop_size: i + hop_size + window_size]
        ws1_ = np.fft.fft(window * ws1)
        ws2_ = np.fft.fft(window * ws2)
        amp = abs(ws2_)
        phases = (phases + np.angle(ws2_/ws1_)) % (2*np.pi)
        new = amp * np.exp(1j*phases)
        wsx = np.fft.ifft(new) * window
        i2 = int(i / factor)
        output[i2: i2 + window_size] += wsx.real

    output[output > 1.0] = 1.0
    output[output < -1.0] = -1.0
    output *= 32767.0

    return output.astype('int16')
```

NumPy kütüphanesinde mevcut olan Hann penceresi, pencere boyutunda oluşturulur. Pencere boyutunun sekizde biri olacak şekilde adım sayısı belirlenir. Frekans düzleminde yapılacak olan faz bilgileri değişiklikleri için kullanılacak olan, pencere boyutu adedinde sıfırdan oluşan dizi oluşturulur. İşlemin sonunda oluşturulacak olan ses dosyası uzunluğunda, tüm değerleri sıfır olan bir dizi oluşturulur. Ses dosyasından, biri adım sayısı kadar ileriden başlayacak şekilde, pencere boyutu kadar iki ayrı örnek grubu

alınır. Alınan iki pencere, pencere fonksiyonu ile çarpılır ve FFT'leri alınır. Adım sayısı kadar ileride olan pencerenin genlik değerleri alınır ve faz bilgileri sıfırlanır. Önde olan pencerenin frekans kutucuklarının faz açısı değerleri, gerideki pencerenin bıraktığı yerden devam edecek şekilde düzenlenir. Elde edilen yeni faz değerleri, daha önce faz bilgileri sıfırlanan genlik değerleriyle çarpılır. Bu sayede, aynı genlik değerlerinde, önceki penceredekilerle tutarlı faz açılarına sahip frekans kutucukları elde edilir.[9] Önde olan pencerenin IFFT'si alınır ve tekrar pencere fonksiyonu ile çarpılır. Elde edilen pencere boyutu adet örnek yaratılan sıfırlar dizisine eklenir. Seste adım sayısı kadar ilerlenir ve aynı işlem tekrar edilir. Giriş sesinin tamamı işlendiğinde çıkış dosyası 16-Bit çözünürlüğünde normalize edilir ve işlem tamamlanmış olur. (Downey 2015, 2014)

Üçüncü olarak, örnekleri yeniden düzenleyecek olan *resample* fonksiyonu yazılacaktır. Uzatılan/sıkıştırılan sesi hızlandırmak/yavaşlatmak için ses örnekleri yeniden düzenlenecektir. Bu işlem için bir boyutlu doğrusal interpolasyon kullanılacaktır. Fonksiyonun işlem akışı Şekil 3.3'de görülebilir. Fonksiyon giriş olarak düzenlenecek ses dosyasını ve faktörü alacaktır. Giriş ve çıkış ses dosyalarının örnek cinsinden uzunlukları belirlenir. Giriş dosyasına, çıkış ses dosyasının boyutuna göre doğrusal interpolasyon uygulanır ve çıkış dosyasına yazılır (Downey 2015, McPherson & Reiss 2015, Roads 2001).

**Şekil 3.3: *resample* fonksiyonunun işlem akışı**

```
def resample(sound, factor):
    sound_len = len(sound)
    output_len = int(sound_len/factor)
    interpol = interp1d(np.arange(0, sound_len), sound)
    resampled = np.linspace(0, sound_len - 1, output_len)
    return interpol(resampled).astype('int16')
```

Son olarak ana komutları alarak *zero_padding*, *phase_vocoder* ve *resample* fonksiyonlarını yönetecek *pitch_shift* fonksiyonu yazılacaktır. Giriş olarak üç veri alacaktır. Bunlar sırasıyla perde değerinde değişiklik yapılacak olan ses dosyası,

[9] https://cycling74.com/tutorials/the-phase-vocoder-%E2%80%93-part-i

değişikliğin aralık cinsinden değeri ve pencere boyutudur. Fonksiyonun işlem akışı Şekil 3.4'de görülebilir. Adım sayısı, pencere boyutunun sekizde biri olacak şekilde belirlenir. Aralık sayısından faktör hesaplanır. Ardından sırasıyla *zero_padding, phase_vocoder* ve *resample* fonksiyonları çalıştırılarak çıkış dosyası tamamlanır.

**Şekil 3.4: *pitch_shift* fonksiyonunun işlem akışı**

```
def pitch_shift(wave, n, window_size=2048):
    hop_size = int(window_size / 8)
    factor = 2**(1.0 * n / 12.0)
    sound_zero = zero_padding(wave, window_size)
    sound_new = phase_vocoder(sound_zero, factor, window_size)
    return resample(sound_new, factor)
```

*Phase_vocoder* ve *zero_padding* fonksiyonlarını, oluşturulacak bir *elastic* fonksiyonu ile çalıştırıp, sesi uzatmak ya da sıkıştırmak ve bu haliyle bir dosya oluşturmak da haliyle mümkündür. Bu fonksiyon giriş verisi olarak ses dosyasını, faktörü ve pencere boyutunu alacaktır. Fonksiyona giriş verisi olarak verilecek olan faktör değeri, ses dosyasının ne kadar uzatılacağı ya da sıkıştırılacağının oranıdır. Örneğin ses dosyası iki katına çıkarılacaksa faktör 2, yarıya indirilecekse 0.5'tir. Fonksiyonun işlem akışı Şekil 3.5'de görülebilir. Fonksiyona alınan ses önce *zero_padding,* ardından *phase_vocoder* fonksiyonlarından geçirilir. Bu sayede sesin perde değerlerine müdahale edilmeden süresinde değişiklik gerçekleştirilmiş olur.

**Şekil 3.5: *elastic* fonksiyonunun işlem akışı**

```
def elastic(wave, factor, window_size):
    sound_zero = zero_padding(wave, window_size)
    return phase_vocoder(sound_zero, factor, window_size)
```

## 3.3 MODÜLLERİN ÇALIŞTIRILMASI

Oluşturulan *pitch_shift* modülünün çalışma akışı, Şekil 3.6'dan da görülebileceği üzere şu şekildedir; ilk olarak NumPy ve SciPy kütüphaneleri çağırılır. Seçilen ses dosyası okunur ve her ses örneğinin bir sayıyla gösterildiği diziye dönüştürülür. Sesin

kaydırılacağı aralık sayısı ile pencere boyutu belirlenir ve *pitch_shift* fonksiyonu çalıştırılır. Fonksiyonun çıktı verisi olan dizi, örnekleme hızı ve oluşturulacak olan dosyanın adı verileri girilerek 16-Bit wav dosyasına yazılır. Ses dosyasının perde değerleri, belirtilen aralık sayısı kadar kaydırılmış olur.

**Şekil 3.6: *pitch_shift* modülünün çalışma akışı**

```
import scipy.io.wavfile
import numpy as np
from scipy.interpolate import interp1d
from scipy.io.wavfile import write
from scipy.io.wavfile import read

sound = scipy.io.wavfile.read('test.wav')
sound_shift = pitch_shift(sound, 12, 1024)
scipy.io.wavfile.write('test_shift.wav', 44100, sound_shift)
```

Oluşturulan *elastic* modülünün çalışma akışı Şekil 3.7'den de görülebileceği üzere şu şekildedir; ilk olarak yine NumPy ve SciPy kütüphaneleri çağırılır. Seçilen ses dosyası okunur ve her ses örneğinin bir sayıyla gösterildiği diziye dönüştürülür. Faktör ve pencere boyutu belirlenir ve *elastic* fonksiyonu çalıştırılır. Fonksiyonun çıktı verisi olan dizi, örnekleme hızı ve oluşturulacak olan dosyanın adı verileri girilerek 16-Bit wav dosyasına yazılır. Bu şekilde ses dosyası istenen miktarda uzatılmış veya sıkıştırılmış olur.

**Şekil 3.7: *elastic* modülünün çalışma akışı**

```
import scipy.io.wavfile
import numpy as np
from scipy.io.wavfile import write
from scipy.io.wavfile import read

sound = scipy.io.wavfile.read('test.wav')
sound_new = elastic(sound, 10, 1024)
scipy.io.wavfile.write('test_elastic.wav', 44100, sound_new)
```

# 4. BULGULAR

Bu bölümde çeşitli ses dosyalarına modüller aracılığıyla değişiklikler yapılmış, sonuçlar yorumlanmış ve karşılaştırılmıştır. Farklı ses kaynakları, farklı parametrelerle benzer işlemlerden geçirilmiştir.

## 4.1 PERDE DEĞİŞTİRİCİ MODÜL TESTLERİ

Sinüs sinyalleri, stüdyo ortamında kaydedilmiş insan sesi, elektrik gitar ve perküsyon kayıtları gibi çeşitli ses kaynaklarına modüller aracılığıyla değişiklikler yapılmıştır. Bu değişikliklerin aynıları Pro Tools'da bulunan Pitch II plug-iniyle de gerçekleştirilmiş ve sonuçları karşılaştırılmıştır. Ayrıca aynı ses kaynağına aynı işlem uygulanarak, farklı pencere boyutlarının ve pencere fonksiyonu tiplerinin etkileri incelenmiştir.

### 4.1.1 Sinyal Testleri

#### 4.1.1.1 440 Hz'in 1 ve 2 oktav tize taşınması ve Pitch II karşılaştırması

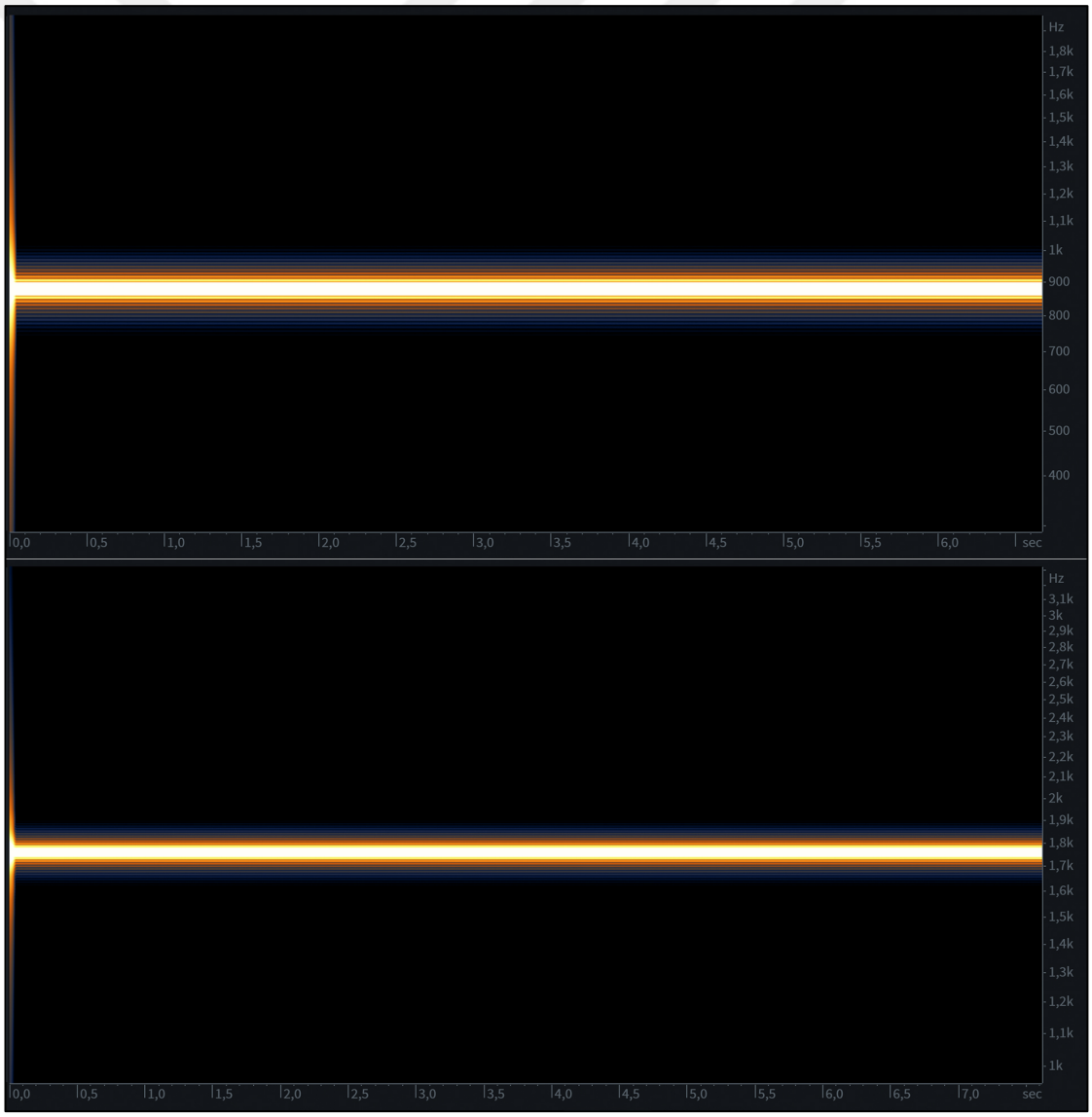
Bu testler için Pro Tools'da bulunan Signal Generator plug-ini kullanılarak mono, 16-Bit, örnekleme hızı 44100 örnek / saniye olan 440 Hz sinüs dalgası oluşturulmuştur. 440 Hz tonu, hem Pitch II ile hem de perde değiştirici modül ile 12 aralık (1 tam oktav) ve 24 aralık (2 tam oktav) tize taşınmıştır. Ayrıca karşılaştırma için 880 Hz ve 1760 Hz sinüs dalgaları sinyal üreteciyle de oluşturulmuştur. Pitch II'de keskin *transient* seslerin yayılımını engellemekte kullanılan *threshold* seçeneği kapatılmış ve pencere boyutu mümkün olan en yüksek değer olan 42.0 ms seçilmiştir. Basit bir matematikle ms cinsinden olan pencere boyutu değeri, örnek sayısına çevrilebilir;

$$pencere\ boyutu\ (\text{örn}) = pencere\ boyutu\ (ms) * \text{örnekleme hızı}\ (\frac{\text{örn}}{s}) \quad \textbf{(4.1)}$$

$$pencere\ boyutu = \frac{42}{1000} * 44100 = 1852.2\ örnek$$

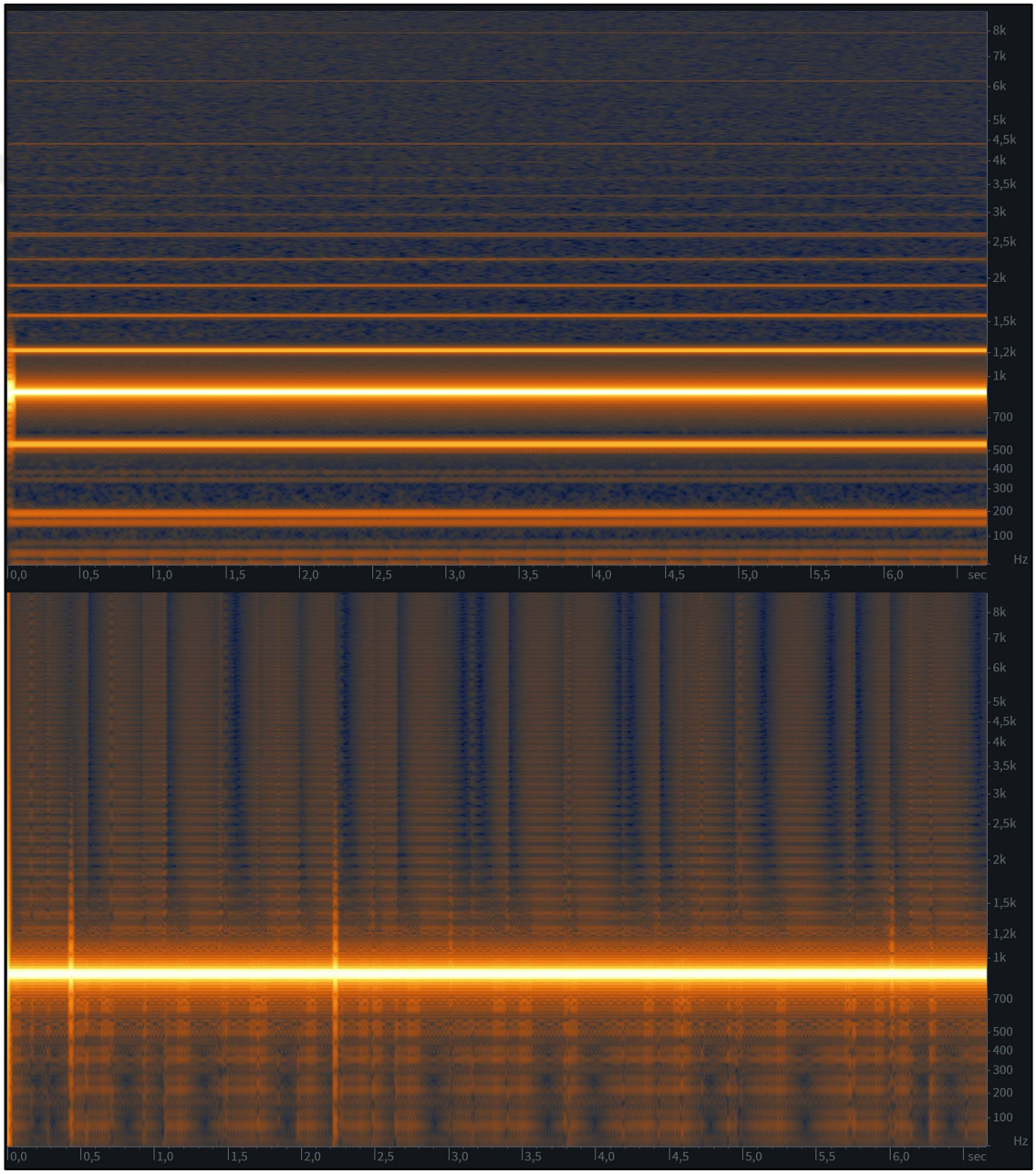
Örnek sayısı değeri küsuratlı olamayacağından 1853 olarak kabul edilebilir. Pencere boyutu değerlerinin yakın olması açısından Perde Değiştirici Modülün pencere boyutu 2048 olarak belirlenmiştir. Pencere fonksiyonu olarak Hann pencere fonksiyonu seçilmiştir. Bu bağlamda sinyal üretecinin ürettiği sesler, Pitch II plug-ininin oluşturduğu sesler ve perde değiştirici modülün oluşturduğu seslerin spektrogramları çıkartılmıştır (Şekil 4.1).

**Şekil 4.1: Sinyal üretecinin oluşturduğu 880 Hz (üst) ve 1760 Hz (alt) sinyalleri**

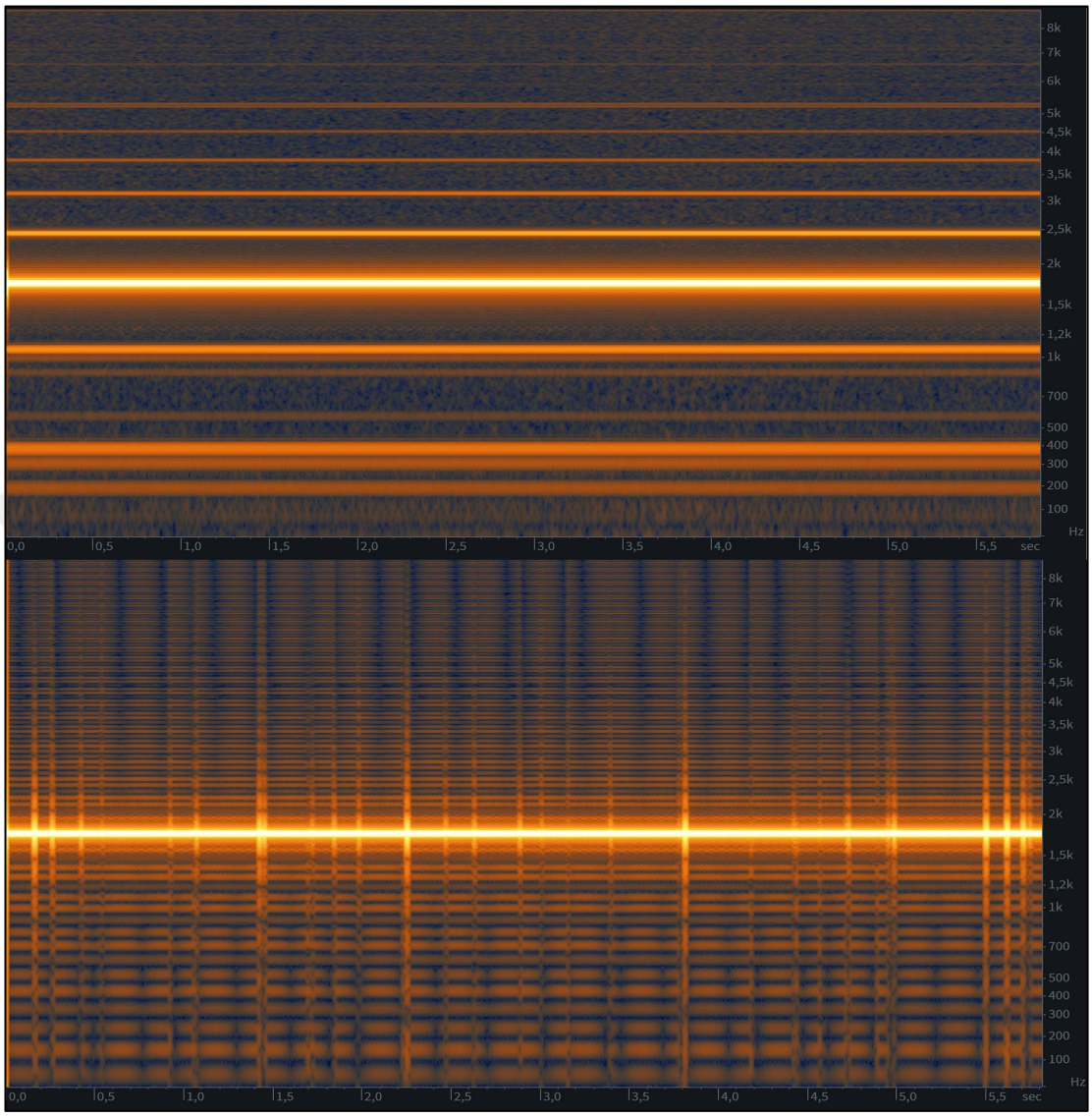


Yapılan karşılaştırmalarda perde değiştirici modülün, Pitch II gibi sesleri doğru yerlerine taşıdığı görünmektedir, ancak FFT sırasında frekans kutucukları arasında sızıntılar meydana gelmiştir. 880 Hz ve 1760 Hz etrafındaki fazlalıklar Şekil 4.2 ve 4.3'te görülmektedir. Bu sızıntılar, sinyale eklenen çeşitli harmonikler gibi duyulmaktadır.

**Şekil 4.2: Modülün (üst) ve Pitch II'nin (alt) oluşturduğu 880 Hz sinyalleri**



**Şekil 4.3: Modülün (üst) ve Pitch II'nin (alt) oluşturduğu 1760 Hz sinyalleri**



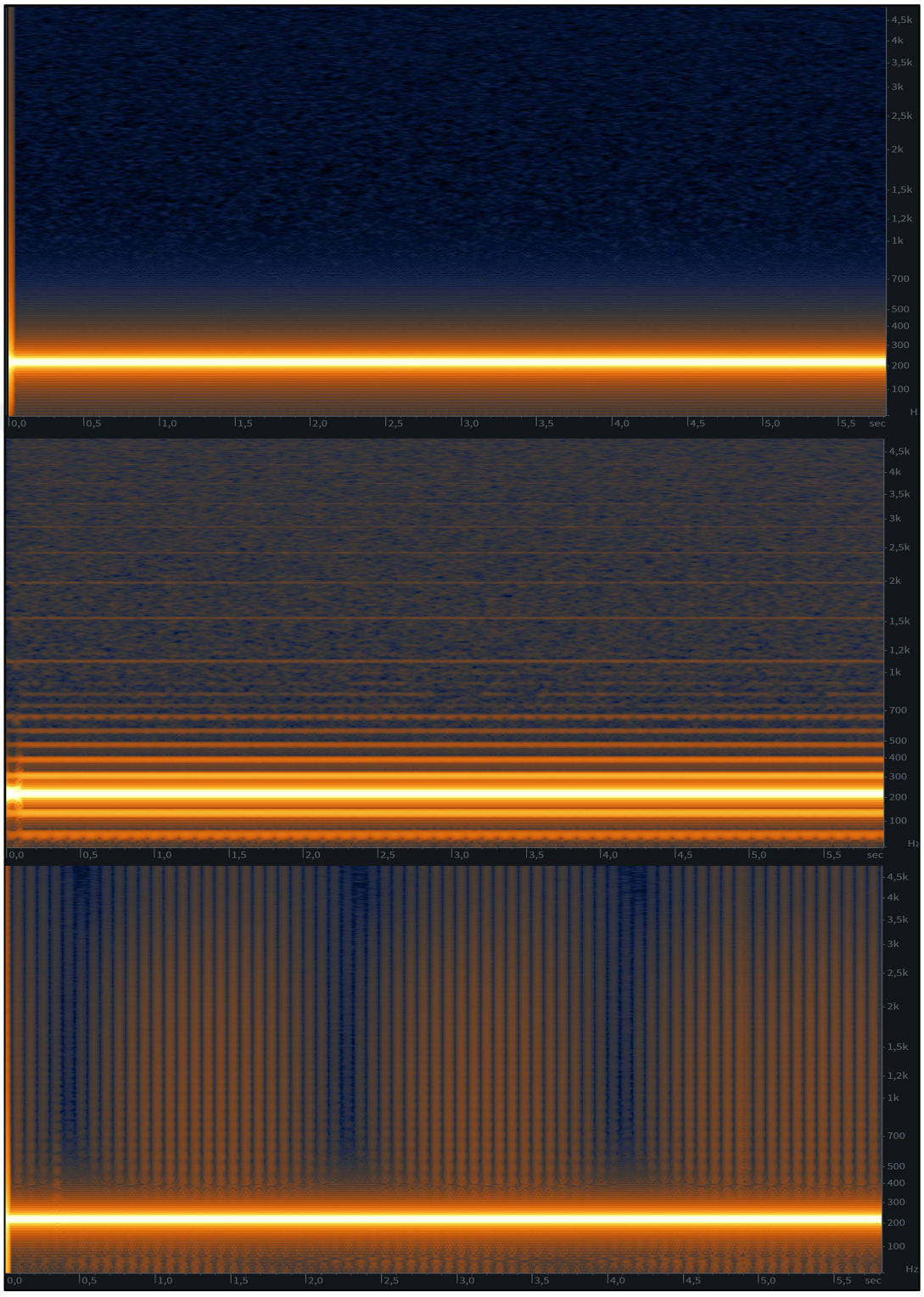

Ayrıca modülün oluşturduğu sesin seviyesi 10 dB civarında daha düşüktür. Bunun sebebinin, frekans düzleminde oluşturulan faz bilgilerinin genlik değerlerini de etkilemesi olduğu söylenebilir. Spektrogramlardan sağlıklı karşılaştırmalar yapabilmek adına, modülün oluşturduğu seslerin seviyeleri Izotope RX7 programından 10 dB arttırılmıştır. Pitch II'nin oluşturduğu sesler incelendiğinde, çok az sızıntı olduğu görülebilir ancak genel duyuma fazla etki etmediği söylenebilir. Ancak yer yer sesin pencereler halinde işleme alınıyor olmasından kaynaklanabilecek bir çıtırtı sesi duyulmaktadır. Bu çıtırtı sesinin spektrumun her yerinde olduğu spektrogramlardan görülebilir. Bu durumun gerçekleştiği noktaların bazılarında anlık frekans kaymalarının

ve/veya kesintilerinin duyulduğu söylenebilir. Modülün yaptığı değişikliklerde böyle bir ses yoktur. Daha önce bahsedildiği gibi, modülün yaptığı değişikliklerde hedef frekans kutucuğundan etrafındaki kutucuklara, uzaklaştıkça azalan genlik geçişleri olmuştur. Bu fazlalıklar, oluşturulan 880 Hz sesinde özellikle 535 Hz ve 1225 Hz civarında, 1760 Hz sesindeyse özellikle 1175 Hz ve 2450 Hz civarında yoğunluk göstermektedir. Bu genlik geçişlerinin bir sebebinin de frekans düzleminde yapılan faz işlemleri olduğu söylenebilir. Faz değişiklikleri tutarlı olsa da kusursuz bir sonuç vermemektedir.

#### 4.1.1.2 440 Hz'in 1 oktav pese taşınması ve Pitch II karşılaştırması

Bu testler için daha önce oluşturulmuş olan 16-Bit, 44100 örnekleme hızında 440 Hz sinüs dalgası kullanılmıştır. 440 Hz tonu, hem Pitch II ile hem de perde değiştirici modül ile 12 aralık yani 1 tam oktav pese taşınmıştır. Ayrıca karşılaştırma için 220 Hz sinüs dalgası sinyal üreteciyle de oluşturulmuştur. Pitch II'de pencere boyutu 42.0 milisaniye (~1853 örnek) olarak seçilmiştir *threshold* fonksiyonu kapalı tutulmuştur. Perde değiştirici modülde pencere boyutu 2048 örnek olarak belirlenmiştir. Pencere fonksiyonu olarak Hann pencere fonksiyonu seçilmiştir. Bu bağlamda sinyal üretecinin ürettiği 220 Hz sesi, Pitch II plug-ininin oluşturduğu 220 Hz sesi ve perde değiştirici modülün oluşturduğu 220 Hz sesinin spektrogramları çıkartılmıştır. Yapılan karşılaştırmalarda perde değiştirme işleminin başarılı olduğu gözlenmiştir. Ancak yine çeşitli sızıntılar meydana gelmiştir. Ayrıca modülün oluşturduğu sesin seviyesi yine 10 dB civarında daha düşüktür. Spektrogramlardan sağlıklı karşılaştırmalar yapabilmek adına, modülün oluşturduğu seslerin seviyeleri Izotope RX7'den 10 dB arttırılmıştır. Gerçekleşen sızıntıların sesin dokusuna yaptığı değişiklikler duyulmaktadır. Sinyalin baskın frekansı net bir şekilde duyulmaktadır, ancak yine baskın frekanstan uzaklaştıkça azalan harmonik oluşumları gözlenmektedir. Özellikle üst frekanslarda dip gürültüsünü andıran sesler oluşmuştur. Bunun haricinde, sesin dokusunun, sinyal üretecinin ürettiği 220 Hz sinyaline bir hayli yakın olduğu söylenebilir. Pitch II'nin değişiklik yaptığı seslerde yine bir çıtırtı sesi duyulmaktadır. Bu çıtırtı sesinin spektrumun üst kısımlarına da yansıdığı spektrogramlardan görülebilir. Modülün değişiklik yaptığı seslerde böyle bir durum yoktur.

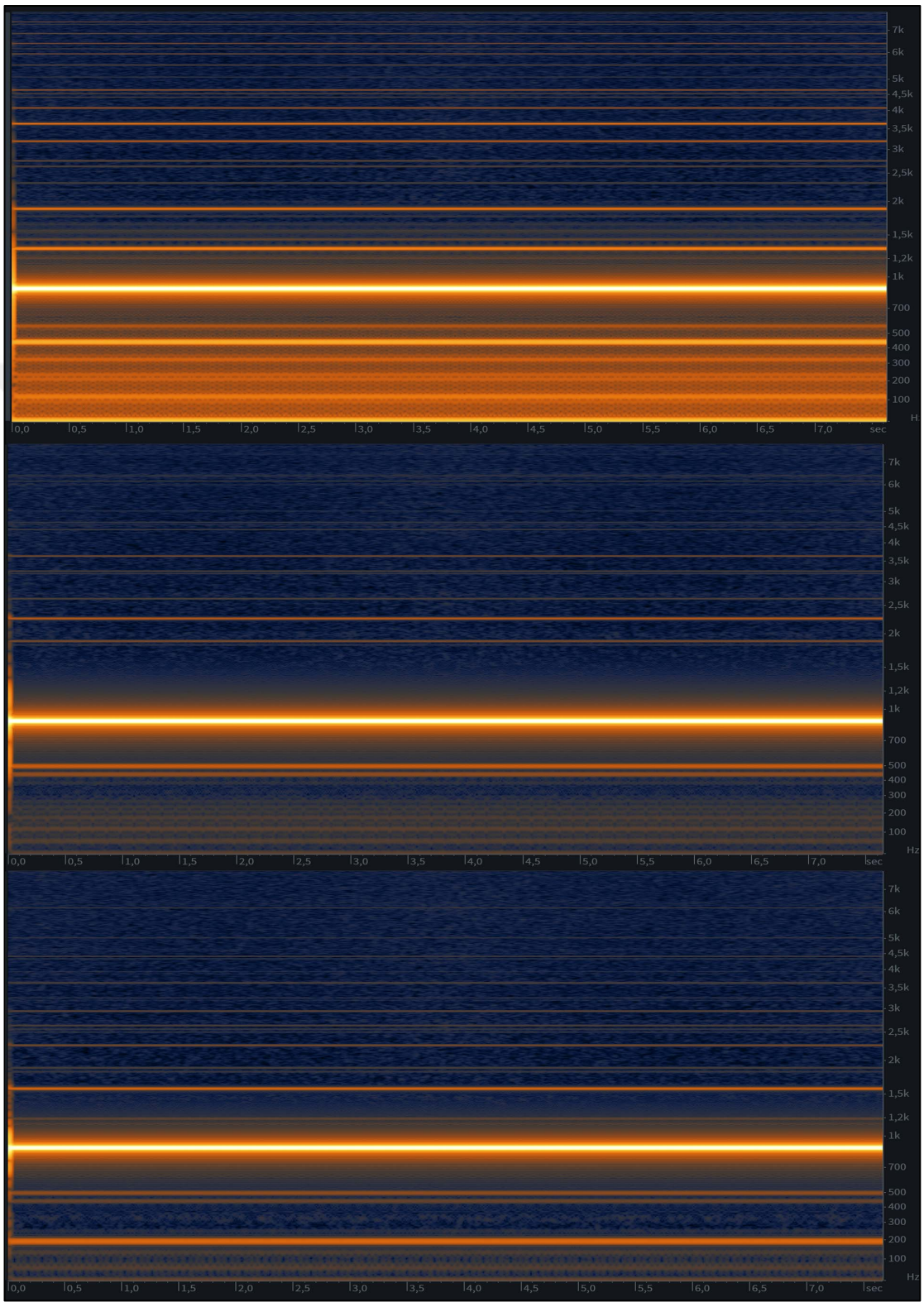
**Şekil 4.4: Sinyal üretecinin, modülün ve Pitch II'nin oluşturduğu 220 Hz sinyalleri (yukarıdan aşağıya)**



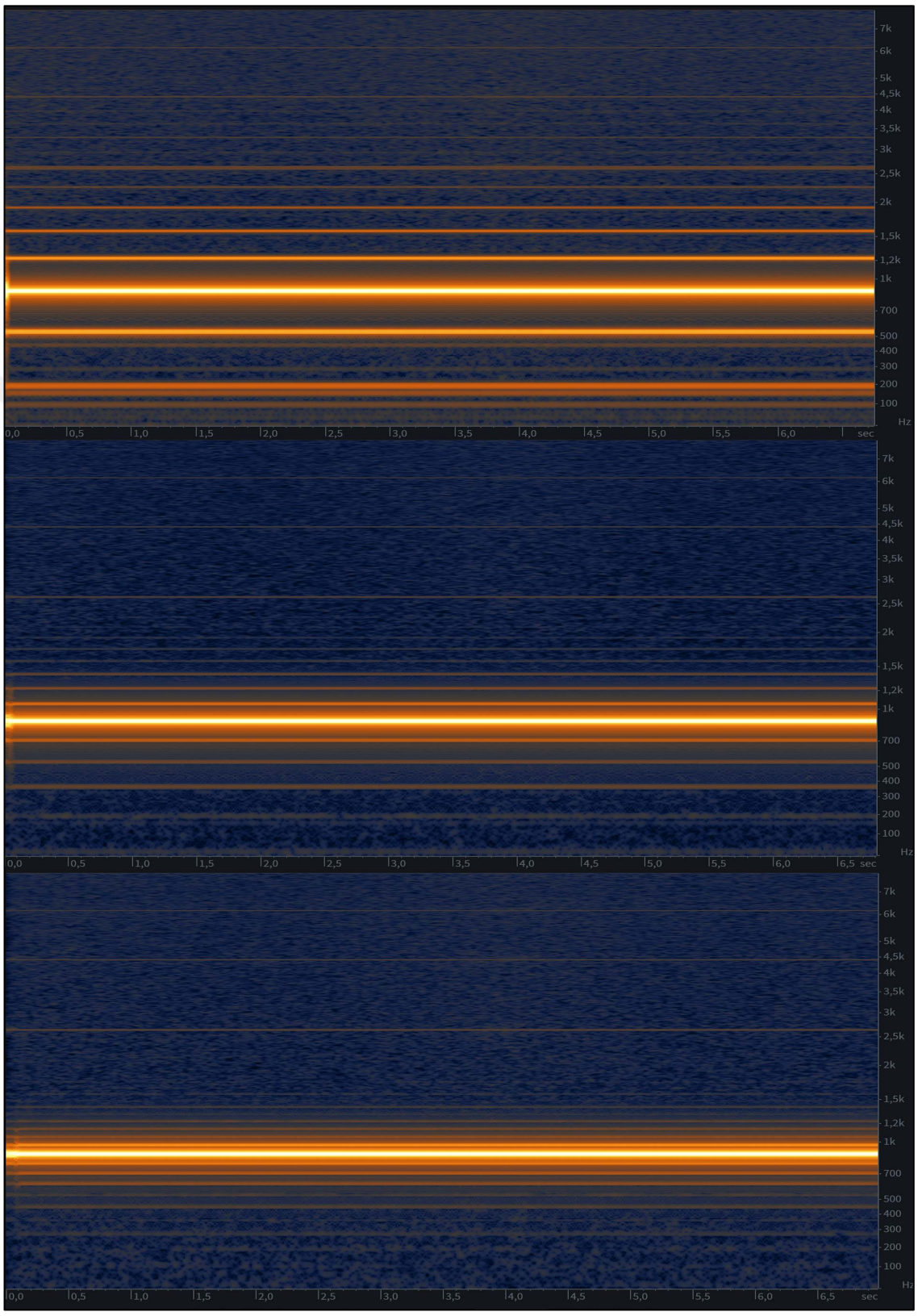
#### 4.1.1.3 220 Hz'in 2 oktav tize taşınması ve pencere boyutları karşılaştırması

Bu testler için daha önce oluşturulmuş olan 16-Bit, 44100 örnekleme hızında 220 Hz sinüs dalgası, perde değiştirici modül ile, farklı pencere boyutu değerleri kullanılarak 2 oktav tize taşınmıştır. İşlemlerde Hann pencere fonksiyonu kullanılmıştır. Testler 512, 1024, 2048, 4096, 8192 ve 16384 pencere fonksiyonu boyutlarında gerçekleştirilmiştir. Perdeleri değiştirilen seslerin spektrogramları çıkartılmış ve perde değişikliklerin başarılı olduğu ancak her pencere boyutunda farklı miktarlarda birikmeler olduğu gözlenmiştir. Ayrıca her pencere boyutunda oluşturulan ses dosyasının genel ses seviyesinin de değiştiği gözlenmektedir. Pencere boyutu küçüldükçe, FFT işlemiyle elde edilen frekans kutucuklarının sayısı azalmakta ve çözünürlük düşmektedir. Bu nedenle pencere boyutu küçüldükçe oluşan harmoniklerin sayısı artmaktadır. Bu sebepten ötürü 512, 1024, 2048 ve 4096 pencere boyutundaki örneklerin her birinde, baskın olan 880 Hz frekansıyla birlikte farklı harmonik oluşumlar vardır. 8192 ve 16384 pencere boyutlarında frekans çözünürlüğünün artışıyla birlikte bu harmonikler çok daha az görülmektedir. Bu pencere boyutlarında oluşturulan seslerin, sinyal üretecinin oluşturduğu 880 Hz sinyalinin dokusuna çok yakın olduğu söylenebilir.

**Şekil 4.5: Modülün 512, 1024 ve 2048 pencere boyutlarında oluşturduğu 880 Hz sinyalleri (yukarıdan aşağıya)**



**Şekil 4.6: Modülün 4096, 8192 ve 16384 pencere boyutlarında oluşturduğu 880 Hz sinyalleri (yukarıdan aşağıya)**



### 4.1.2 İnsan Sesi Testleri

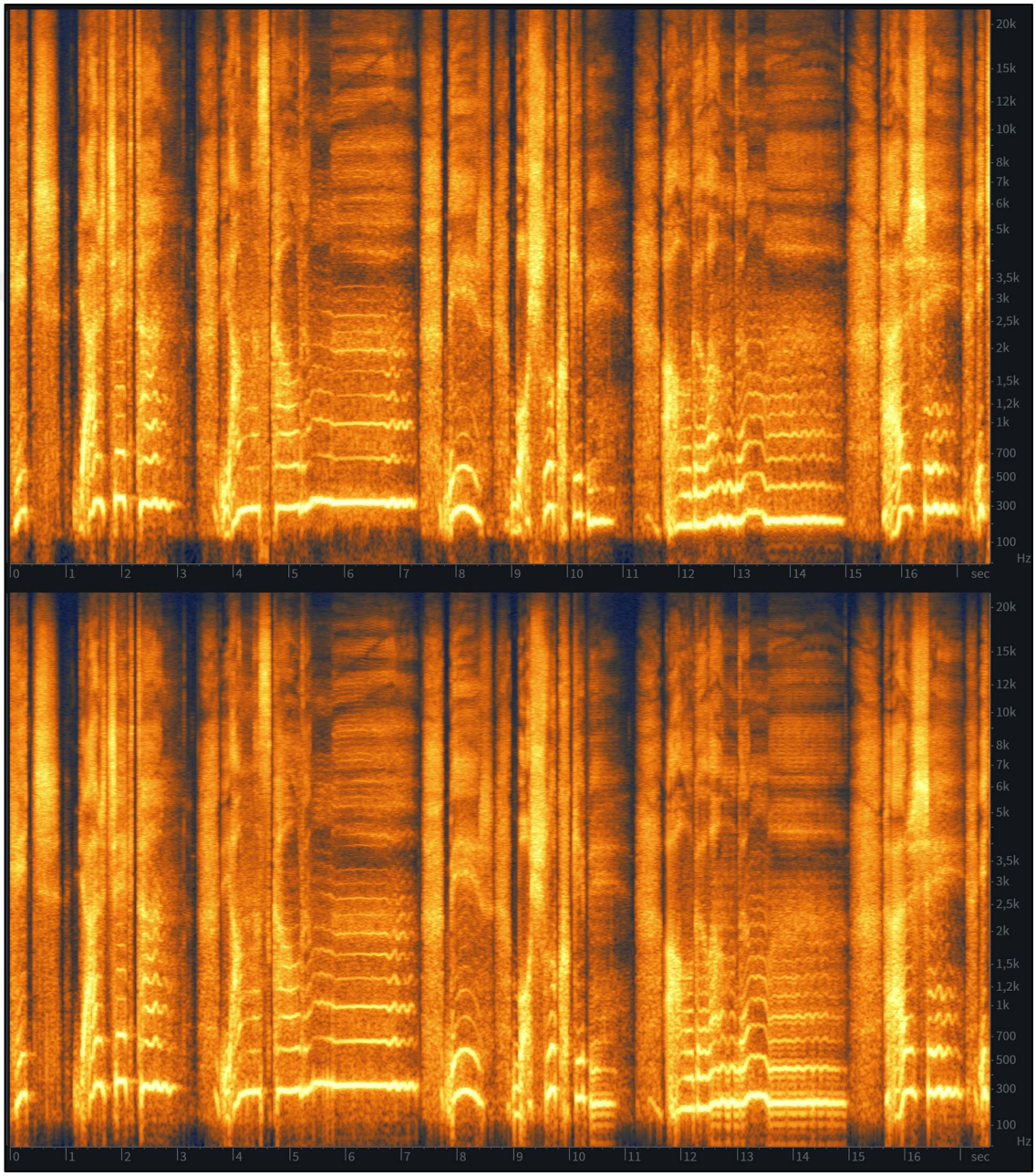
Bu testlerde Bahçeşehir Üniversitesi Ses Kayıt Stüdyolarında yapılan vokal kayıtları kullanılmıştır. Kayıtlar Neumann U87 mikrofon ve Rupert Neve 5012 Dual Mic Pro Analog Pre-amp kullanılarak yapılmıştır. Kullanılan seslerin tamamı mono, 16 – Bit, 44100 örnek/saniye örnekleme hızındadır. Kayıtlar, 100 Hz frekansından, 12 dB/oktav *low cut* filtreden geçirilerek işlemden geçirilmişlerdir.

#### 4.1.2.1 İnsan sesinin farklı pencere fonksiyonu boyutları ile 5 aralık tize taşınması ve Pitch II karşılaştırması

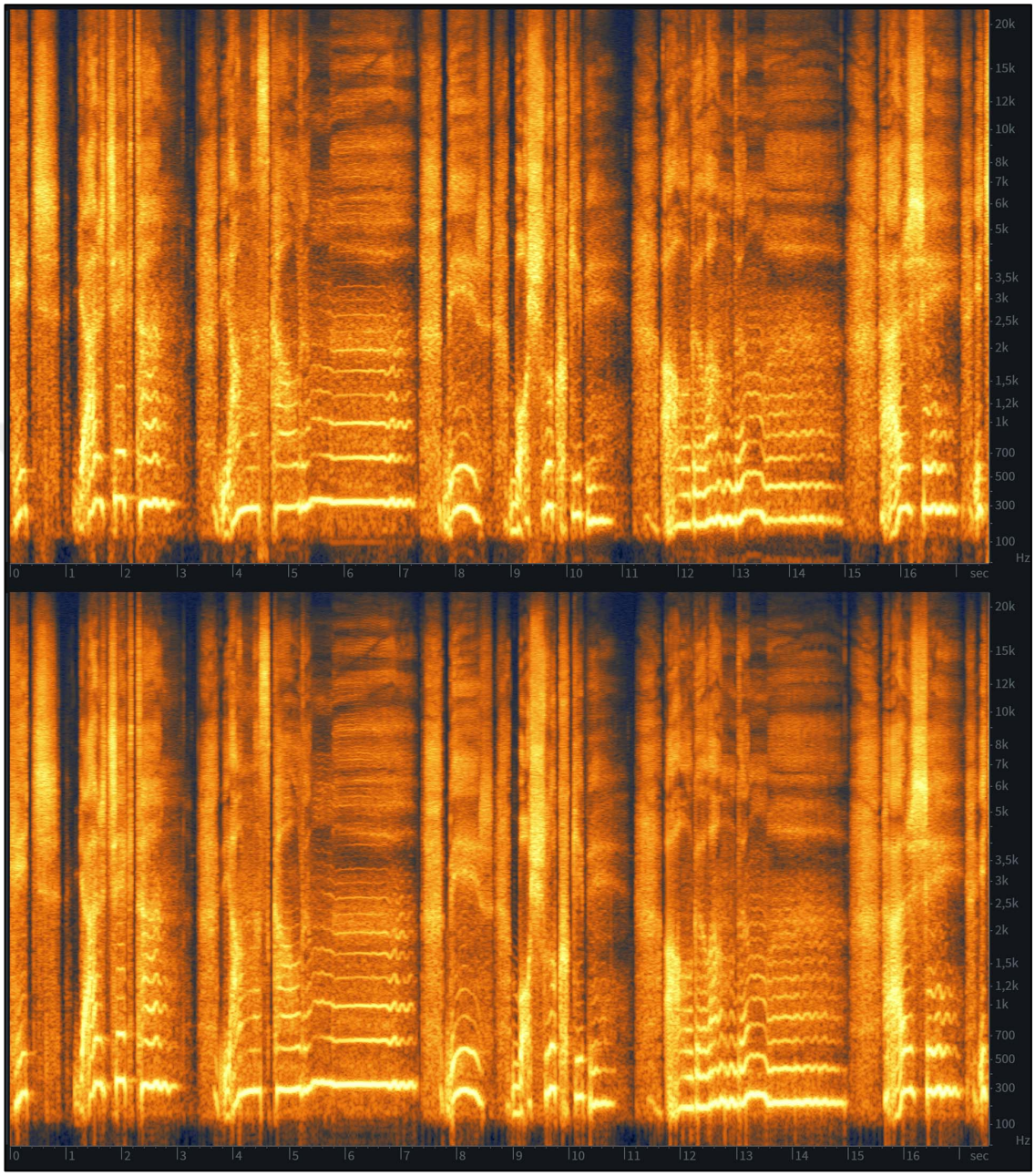
Bu testlerde vokal kaydı, farklı pencere fonksiyonu boyutlarında, perde değiştirici modül ve Pitch II ile 5 aralık tize taşınmıştır. Modülün işleyişinde Hann pencere fonksiyonu kullanılmıştır. Kayıtlar modülden 1024 ve 2048, Pitch II'den ise 23.2 ms (~1024 örnek) ve 42.0 ms (~1853 örnek) pencere boyutlarıyla, *threshold* seçeneği kapalı olarak işleme konmuşlardır. Oluşturulan seslerin spektrogramları çıkartılmış ve perde değişimi işlemlerinin başarılı olduğu gözlenmiştir. Ancak modülün oluşturduğu seslerin seviyelerinin, Pitch II'nin oluşturduğu seslerin seviyesinden 5 dB civarında daha yüksek olduğu gözlenmiştir. Spektrogramlardan sağlıklı karşılaştırma yapabilmek adına Pitch II'nin oluşturduğu seslerin seviyeleri Izotope RX7 programında 5 dB arttırılmıştır. Spektrogramlar incelendiğinde testlerin yakın sonuçlar verdiği görülmektedir. Modülün oluşturduğu seslerde, *reverb* efektini andıran oluşumlar gözlenmektedir. 1024 pencere boyutundaki örnekte küçük bir oda gibi duyulan düşük bir efekt duyulmaktadır. 2048 pencere boyutundaki örnekte ise, orta boyutlarda bir oda *reverb* efektini andıran sesler duyulmaktadır. Pencere boyutunun artmasıyla hem algılanan *reverb* efektinin büyüdüğü, hem de seviyesinin yükseldiği söylenebilir. Bu etkilerin sebeplerinin sesi pencereler halinde FFT işleminden geçmesi ve yeni oluşturulan faz değerleri olduğu söylenebilir. Pitch II'nin işlediği seste böyle bir durum olmamakla beraber, özellikle vokal performansının daha bağlı (*legato*) kısımlarında, yer yer çok kısa (bir kaç örneklik) kesintiler ve/veya kaymalar olduğu söylenebilir. Modülün işlediği seslerde böyle bir durum söz konusu değildir. Ayrıca, yine Pitch II'nin oluşturduğu 1024 pencere boyutundaki seste, baskın frekanslardan yakın çevrelerine sızıntılar olduğu

spektrogramlardan görülmektedir. Bu sızıntıların genel duyumu etkilediği söylenebilir. Modülün oluşturduğu seslerde böyle bir durum yoktur.

**Şekil 4.7: 1024 pencere boyutunda modülün (üst) ve Pitch II'nin (alt) 5 aralık tize taşıdığı insan sesi kayıtları**



**Şekil 4.8: 2048 ve 1854 pencere boyutlarında, modülün (üst) ve Pitch II'nin (alt) 5 aralık tize taşıdığı insan sesi kayıtları**

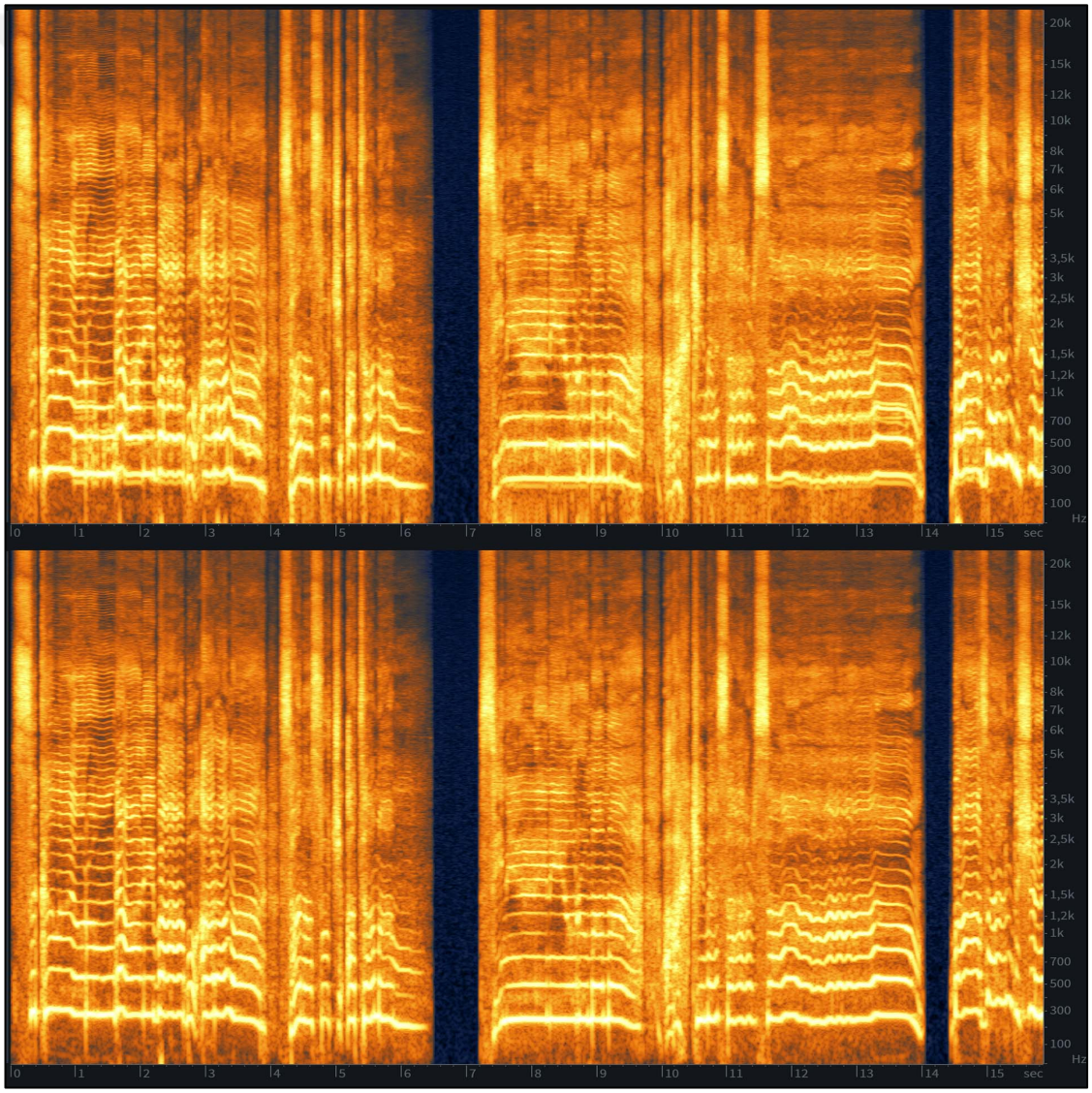



### 4.1.2.2 İnsan sesinin farklı pencere fonksiyonu boyutları ile 3 aralık pese taşınması ve Pitch II karşılaştırması

Bu testlerde vokal kaydı, farklı pencere fonksiyonu boyutlarında, perde değiştirici modül ve Pitch II ile 3 aralık pese taşınmıştır. Kayıtlar modülden 256, 512, 1024, 2048,

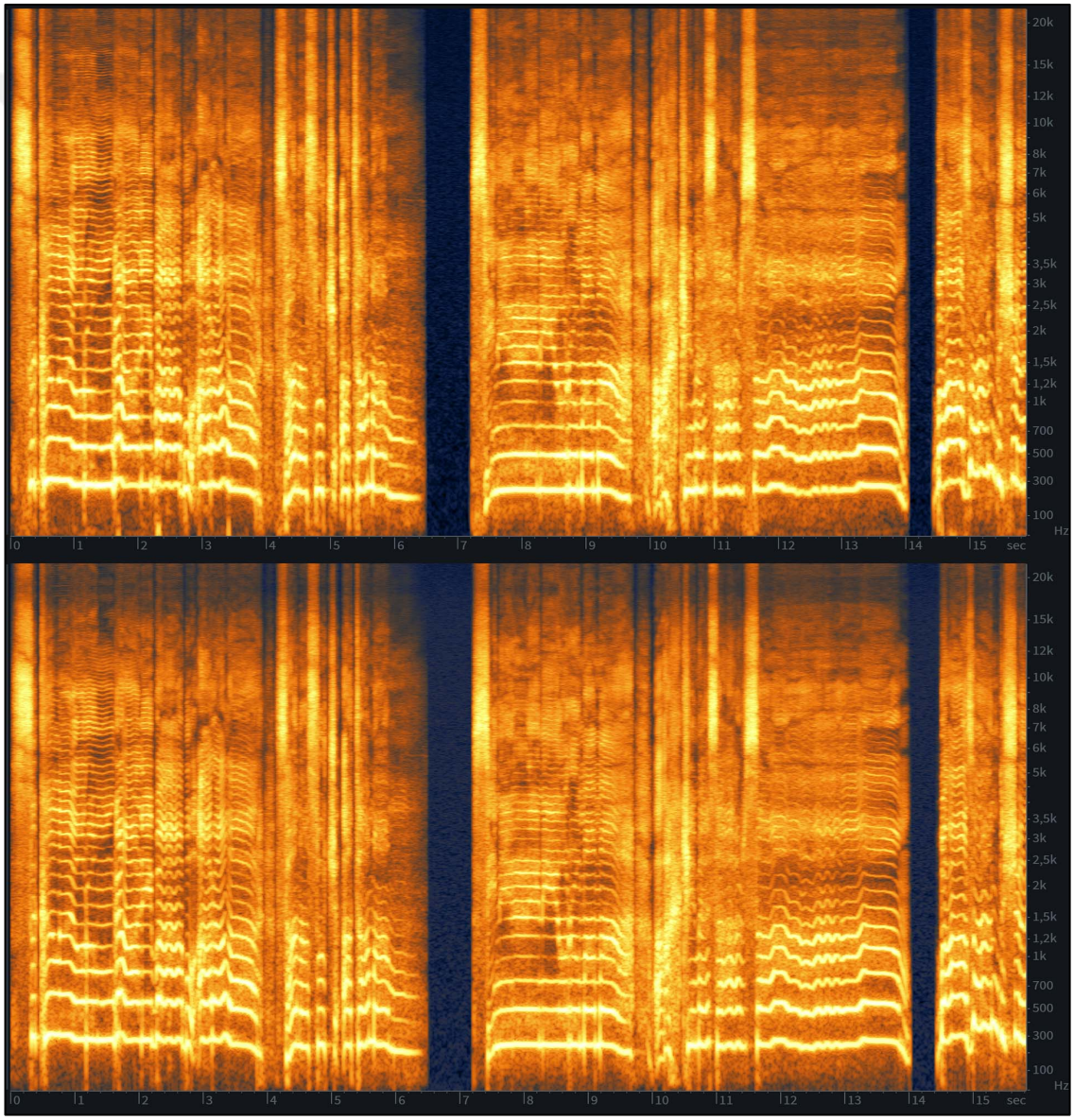
4096, 8192 ve 16384, Pitch II programından ise 23.2 ms (~1024 örnek) ve 42.0 ms (~1853 örnek) pencere boyutlarıyla, *threshold* seçeneği kapalı olarak işleme konmuşlardır. Oluşturulan seslerin spektrogramları çıkartılmış ve perde değişimi işlemlerinin başarılı olduğu gözlenmiştir. Modülün oluşturduğu seslerin seviyelerinin, yine Pitch II'nin oluşturduğu seslerin seviyesinden 5 dB civarında daha yüksek olduğu gözlenmiştir. Spektrogramlardan sağlıklı karşılaştırma yapabilmek adına Pitch II'nin oluşturduğu seslerin seviyeleri Izotope RX7 programında 5 dB arttırılmıştır.

**Şekil 4.9: Modülün 256 (üst) ve 512 (alt) pencere boyutlarında 3 aralık pese taşıdığı insan sesi kayıtları**



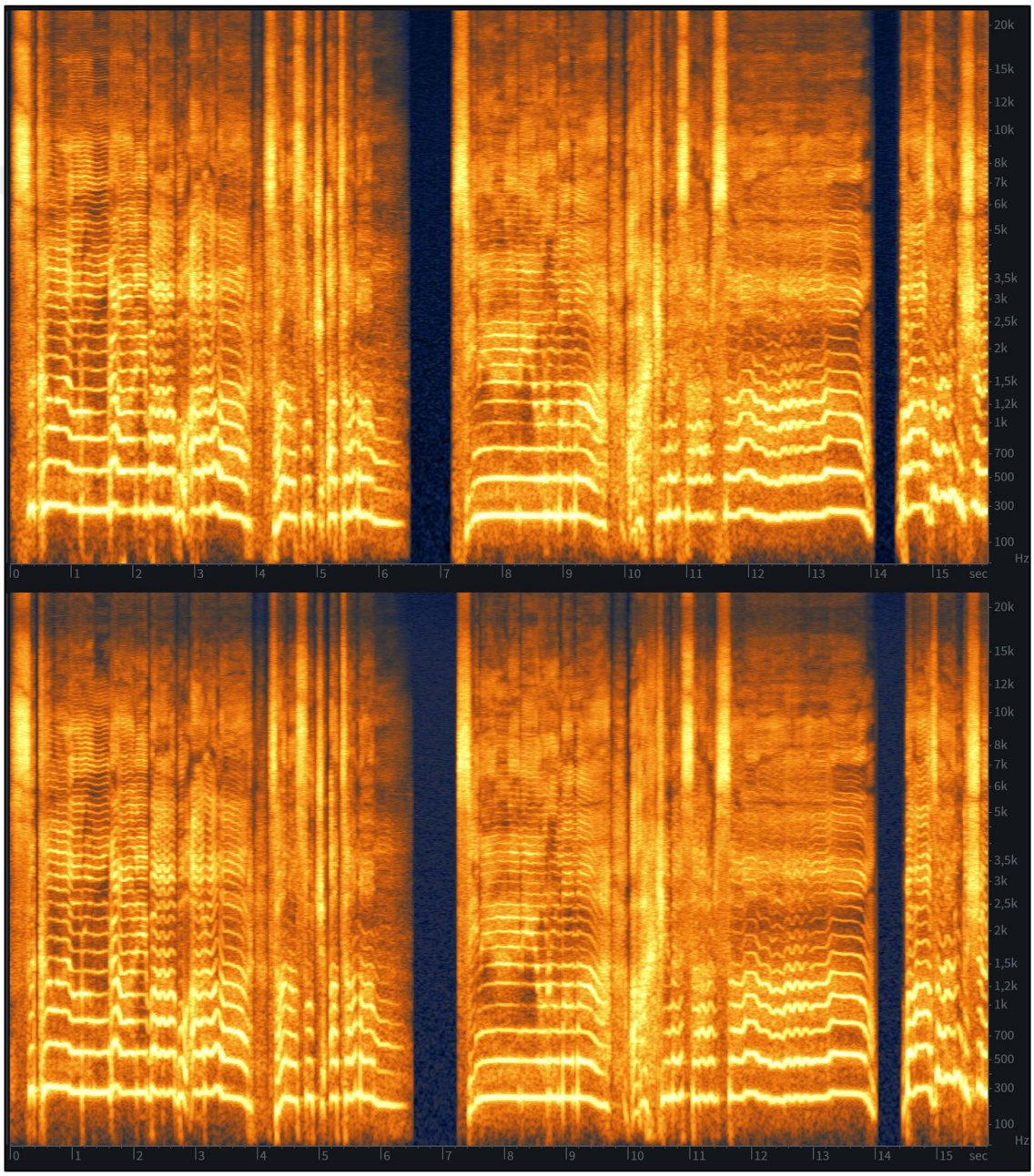
Modülün çalıştırıldığı farklı pencere boyutları göz önüne alındığında, 1024 pencere boyutundaki örnekten itibaren yine pencere boyutu arttıkça büyüyen bir *reverb* efektinin duyulduğu söylenebilir. 1024 pencere boyutunda seviyesi ve büyüklüğü en düşük olan bu efekt, 2048'de küçük bir stüdyo odasını, 4096'da orta boylarda bir odayı, 8192 pencere boyutunda 30 ms civarında *pre-delay'e* sahip büyük bir odayı, 16384 pencere boyutunda ise 50 ms civarında *pre-delay'e* sahip daha büyük bir alanı andırmaktadır.

**Şekil 4.10: 1024 pencere boyutunda modülün (üst) ve Pitch II'nin (alt) 3 aralık pese taşıdığı insan sesi kayıtları**

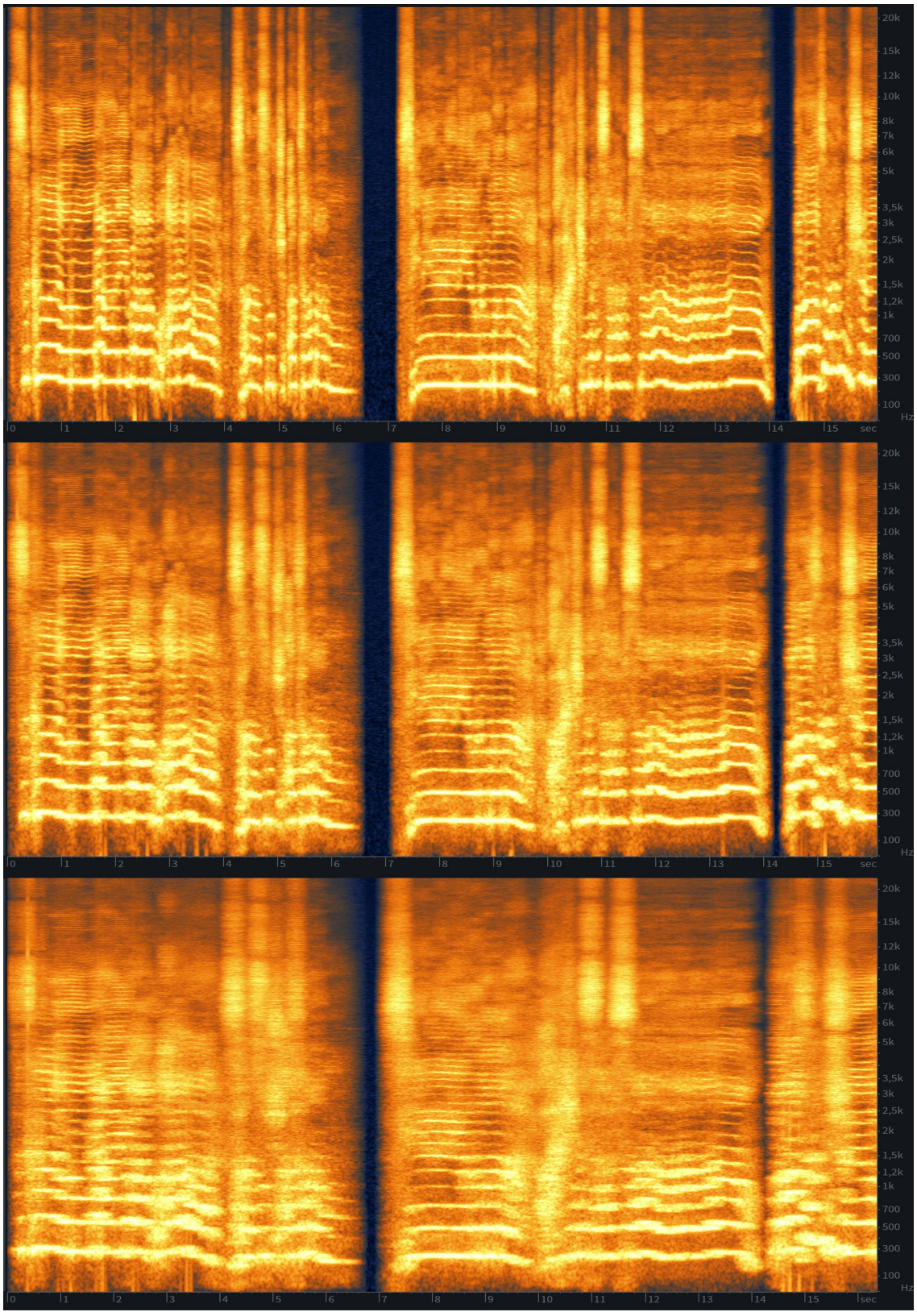


8192 ve 16384 örneklerinde, pencere boyutunun büyüklüğünden ötürü sesler örnekler arasında yayılmaya ve bulanıklaşmaya başlamıştır. Bu durum özellikle keskin ve yüksek genlik seviyesine sahip seslerde duyulmaktadır. 16384 pencere boyutundaki örnekte, bu durumun anlaşılırlığı bir hayli azalttığı söylenebilir.

**Şekil 4.11: 2048 ve 1854 pencere boyutlarında, modülün (üst) ve Pitch II'nin (alt) 3 aralık pese taşıdığı insan sesi kayıtları**



**Şekil 4.12: Modülün 4096, 8192 ve 16384 pencere boyutlarında 3 aralık pese taşıdığı insan sesi kayıtları (yukarıdan aşağıya)**



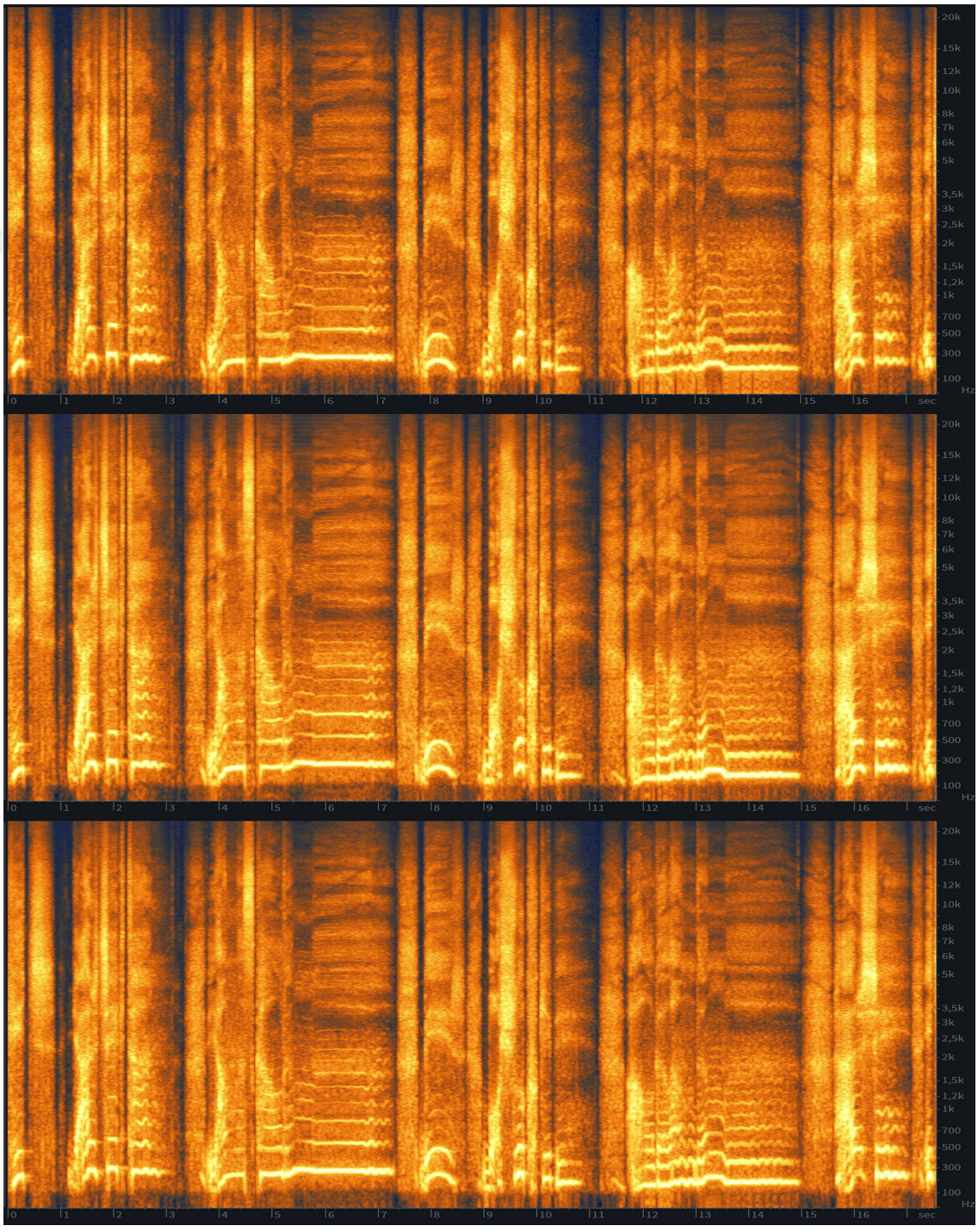
256 pencere boyutundaki örnekte, pencere boyutunun çok küçük olmasından dolayı frekans çözünürlüğünün düşmesiyle, *detune* efektini andıran çeşitli harmonik oluşumları gözlenmektedir. Bu durum baskın frekansların, yakınlarındaki frekans kutucuklarına sızmasıyla gerçekleşmektedir. 512 pencere boyutundaki örnekte böyle bir durum yoktur. Sesin dokusu 1024 pencere boyutundaki örneğe yakındır. Tek farkın 1024 örneğindeki hafif *reverb* efekti olduğu söylenebilir. 1024 ve 2048 örneklerinin, Pitch II ile oluşturulan seslerle karşılaştırıldığında, bahsedilen efekt haricinde yakın bir dokuya sahip oldukları söylenebilir.

#### 4.1.2.3 İnsan sesinin 2 aralık tize taşınması ve pencere fonksiyon tipleri karşılaştırması

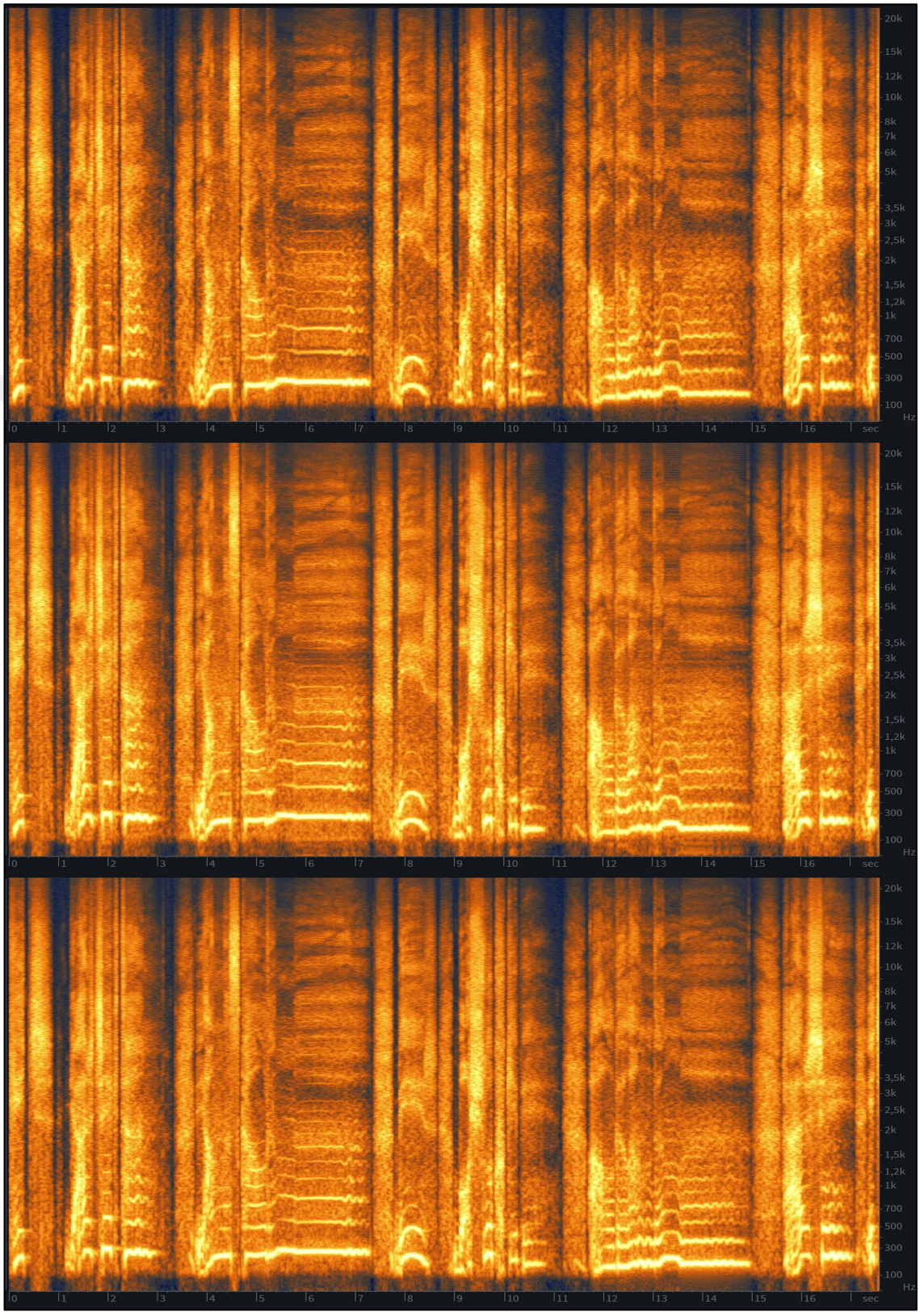
Bu testlerde vokal kaydı, perde değiştirici modül aracılığıyla 2 aralık tize taşınmıştır. Aynı değişiklik yüzde 87.5 kesişim oranı gözetilerek farklı pencere fonksiyon tipleri ve farklı pencere boyutları ile gerçekleştirilmiştir. Hann, Hamming ve Blackman pencere fonksiyonları 512, 1024, 2048 ve 4096 pencere boyutlarında test edilmiştir. Vokal kaydının ve işlem sonucu elde edilen kayıtların spektrogramları çıkartılmış ve perde değişimlerinin başarılı olduğu gözlenmiştir. Frekans düzleminde yapılan faz değeri değişikliklerinin, pencere fonksiyon tiplerinin ve seçilen pencere boyutu değerlerinin her kaydı farklı etkilediği görülmüştür. Blackman pencere fonksiyonuyla oluşturulan seslerin seviyelerinin, diğer tiplere göre, pencere boyutu fark etmeksizin yaklaşık 3 dB daha düşük olduğu söylenebilir. 512 pencere boyutundaki örneklerde, pencere tipi fark etmeksizin, özellikle vokal partisyonunun legato kısımlarında yine bir *detune* efektini andıran harmonik oluşumlar gözlenmektedir. Bu oluşumların Blackman pencere fonksiyonuyla oluşturulan seslerde diğer tiplere kıyasla biraz daha fazla olduğu söylenebilir. 1024 pencere boyutunda, tüm örneklerde küçük bir *reverb* efekti duyulmaktadır. Hamming pencere fonksiyonunun oluşturduğu 1024 pencere boyutundaki seste, özellikle alt frekanslarda diğer tiplere göre daha fazla birikmeler gözlenmektedir. 2048 pencere boyutunda, tüm örneklerdeki *reverb* efekti biraz daha büyük duyulmaktadır. Hamming pencere fonksiyonunun oluşturduğu seslerde yine diğer tiplere göre daha fazla birikmeler gözlenmektedir. 4096 pencere boyutunda, daha önce görülen *reverb* efektleri daha da büyümüştür. Yine önceki örneklerde olduğu gibi

Hamming pencere fonksiyonuyla oluşturulan seste özellikle alt frekanslarda fazlalıklar gözlenmektedir. Örneklerin tamamı göz önüne alındığında, Hamming pencere fonksiyonunun, diğer tiplere kıyasla, daha fazla birikmelere sebep olduğu söylenebilir.

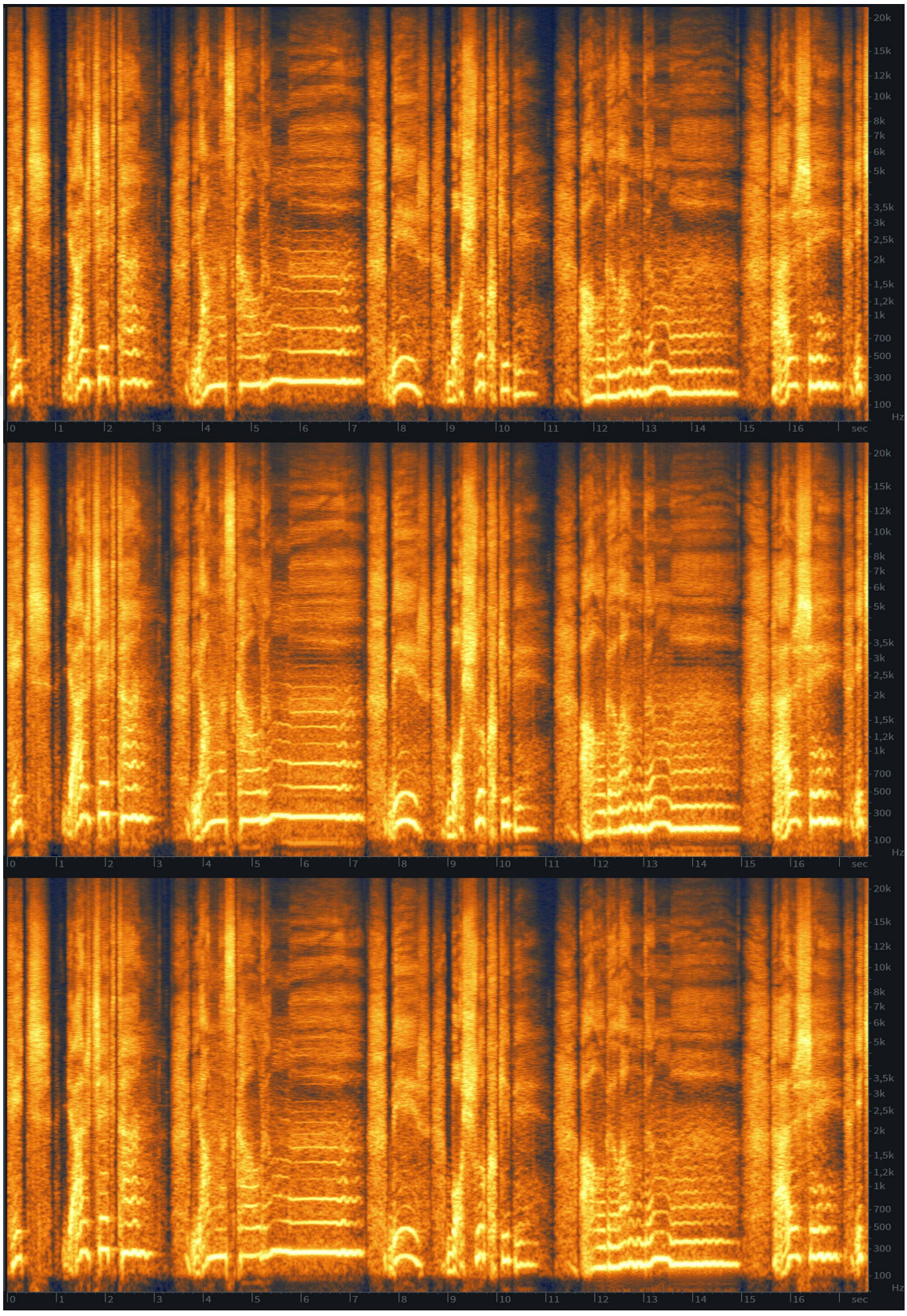
**Şekil 4.13: 512 pencere boyutunda Blackman, Hamming ve Hann pencereleriyle 2 aralık tize taşınan insan sesi kayıtları (yukarıdan aşağıya)**



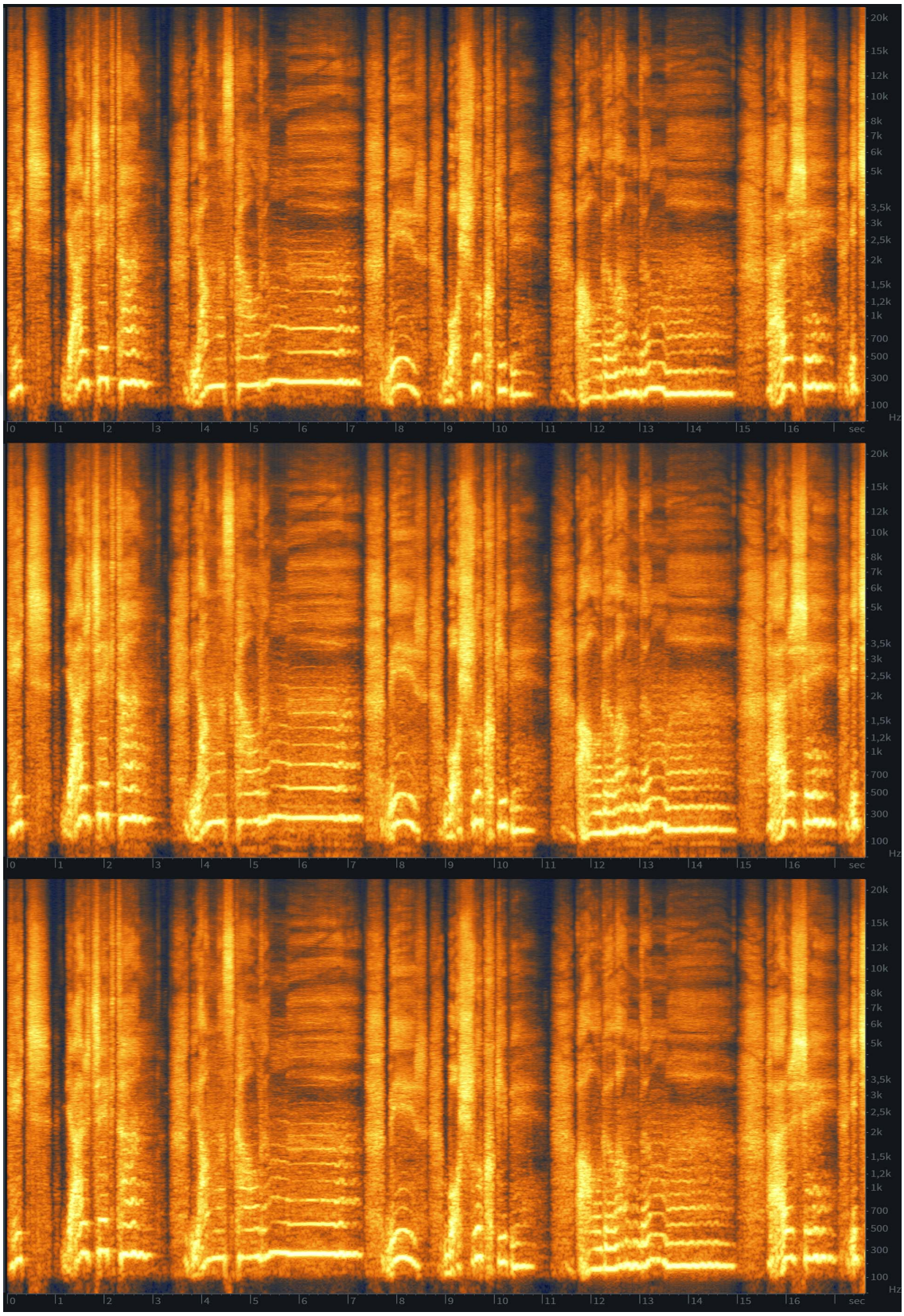
**Şekil 4.14: 1024 pencere boyutunda Blackman, Hamming ve Hann pencereleriyle 2 aralık tize taşınan insan sesi kayıtları (yukarıdan aşağıya)**



**Şekil 4.15: 2048 pencere boyutunda Blackman, Hamming ve Hann pencereleriyle 2 aralık tize taşınan insan sesi kayıtları (yukarıdan aşağıya)**



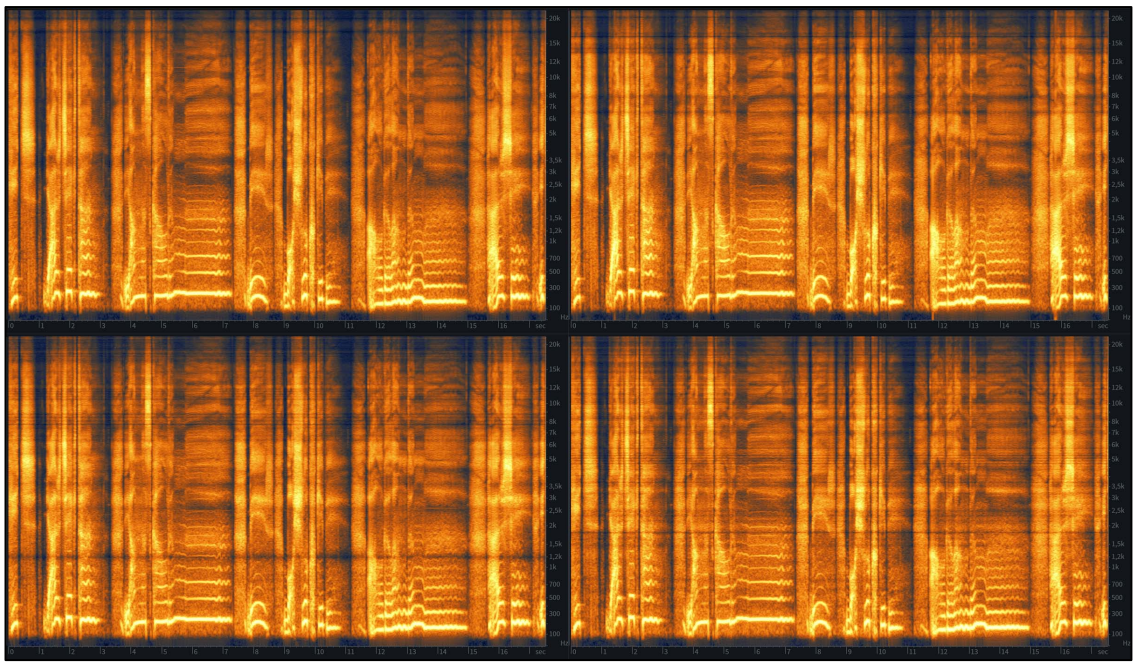
**Şekil 4.16: 4096 pencere boyutunda Blackman, Hamming ve Hann pencereleriyle 2 aralık tize taşınan insan sesi kayıtları (yukarıdan aşağıya)**



#### 4.1.2.4 İnsan sesinin perde değeri değiştirilmeden modülden geçirilmesi ve pencere boyutları karşılaştırması

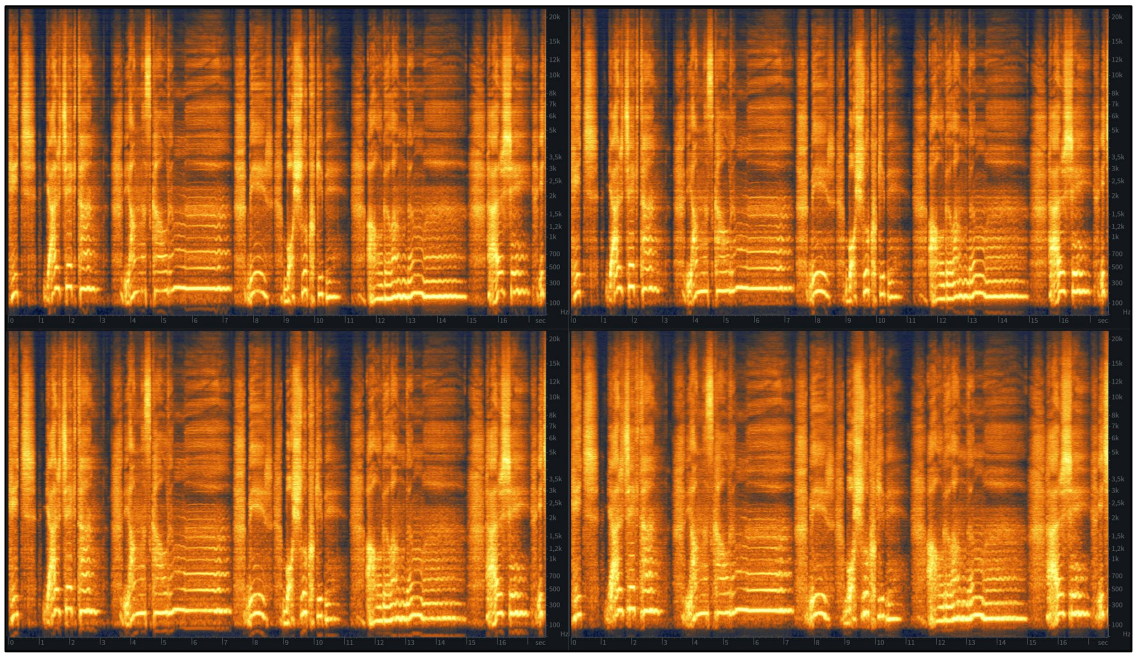
Bu testler, modülün işlem sırasında oluşturduğu fazlalık seslerin, perde değerlerinde değişiklik yapılmadığı durumlarda nasıl sonuçlara yol açtığını incelemek amacıyla yapılmıştır. Testlerde insan sesleri kullanılarak, çeşitli vokal efektlerinin oluşumu hedeflenmiştir. Vokal kayıtları perde değerleri değiştirilmeden, yani aralık sayısı n programa 0 olarak girilerek 32, 64, 128, 256, 512, 1024, 2048, 4096, 8192 ve 16384 pencere boyutlarında analiz edilip yeniden oluşturulmuştur. Hann pencere fonksiyonu kullanılmıştır. Yeniden oluşturulan seslerin ve değişiklik yapılan orijinal kaydın spektrogramları çıkartılmış ve karşılaştırmalar yapılmıştır. Yeniden oluşturulan seslerin seviyelerinin, anlık küçük farklılıklar olmakla birlikte, genel olarak birbirlerine yakın oldukları söylenebilir. Orijinal kaydın seviyesi ise yeniden oluşturulan seslerden yaklaşık 3 dB daha düşüktür. Her pencere boyutunda farklı miktarlarda ve frekanslarda sızıntılar ve oluşumlar görülmekle birlikte, elde edilen dokuların, genel olarak orijinal kayda yakın bir tınısı olduğu söylenebilir.

**Şekil 4.17: 32, 64, 128 ve 256 pencere boyutlarında modülden geçirilen insan sesi kayıtları (sol üstten saat yönünde)**

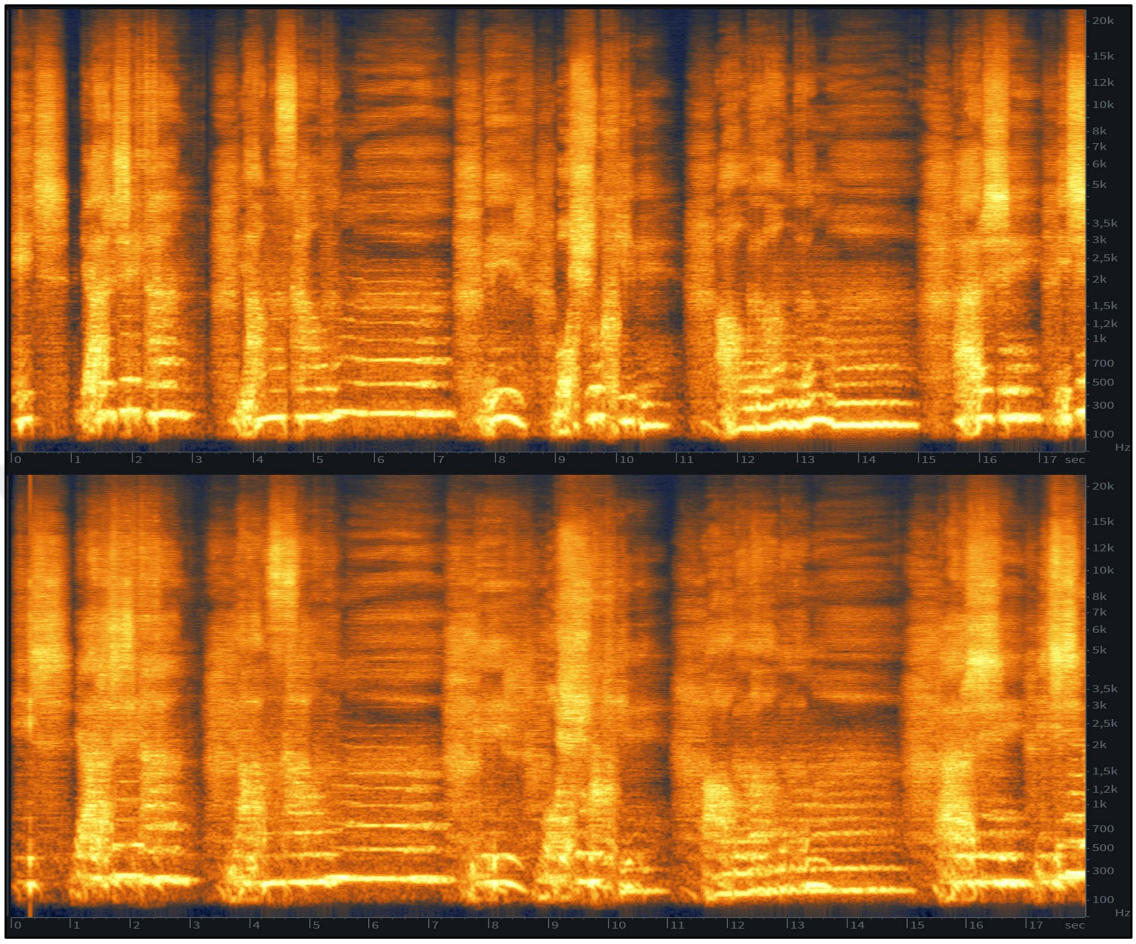


32 ve 64 pencere boyutundaki örneklerde bas ve *low mid* frekansların biraz daha arka plana gittiği söylenebilir. 128 ve 256 pencere boyutundaki örneklerdeyse bu frekansların orijinal kayda göre biraz daha baskın olduğu söylenebilir. Bu değişikliklerin sebebi, farklı pencere boyutlarında farklı sızıntılar meydana gelmesidir. 512 pencere boyutundan itibaren, yeniden oluşturulan seslerde, pencere boyutu arttıkça seviyesi ve büyüklüğü de artan bir *reverb* efektinden bahsetmek mümkündür. 512 pencere boyutunda küçük bir odayı andıran bir *reverb* duyulduğu söylenebilir. Bu örnekte bas frekanslar orijinal kayda göre biraz daha ön plandadır. 1024 pencere boyutundaki örnek, bir önceki örnekten biraz daha büyük bir oda gibi duyulmaktadır. Bas frekanslar bu örnekte nispeten daha düşüktür. 2048 pencere boyutundaki örnek büyük bir oda *reverb* efekti gibi duyulmaktadır. Bas frekanslar bu örnekte yine biraz daha ön plana gelmiştir. 4096 pencere boyutundaki örnekte 20 ms civarında *pre-delay*'e sahip büyük bir alanı andıran bir *reverb* efekti olduğu söylenebilir. 8192 pencere boyutundaki örnekte duyulan sesler, *overdrive* efektinden geçirilmiş bir *delay* efektini andırmaktadır. 16384 pencere boyutundaki örnekte ise, yine büyük bir alan gibi duyulmakla beraber, pencere boyutunun çok büyük olmasından ötürü süreksizlik başlamıştır. Vokal kaydının anlaşılırlığı azalmaktadır.

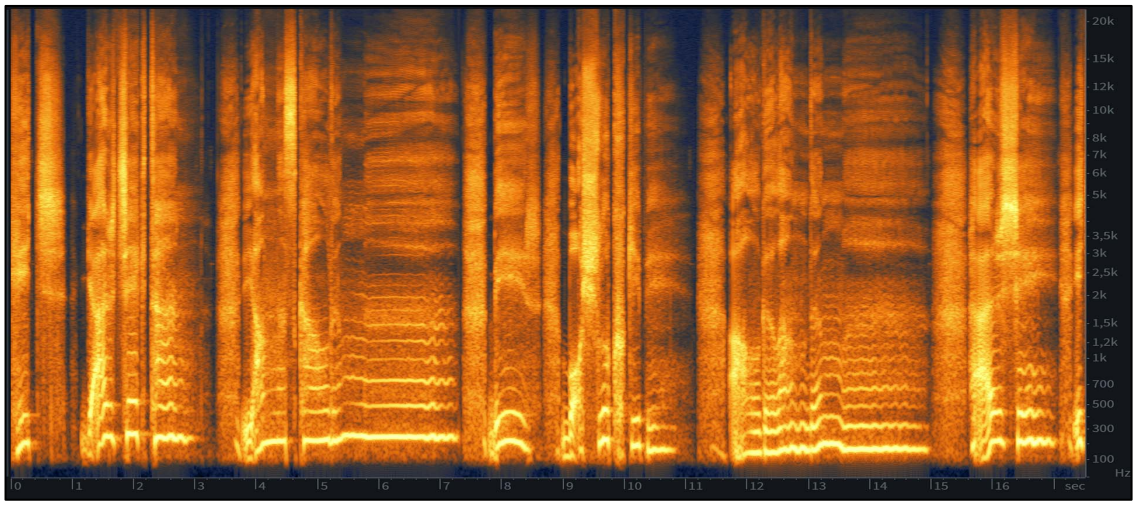
**Şekil 4.18: 512, 1024, 2048 ve 4096 pencere boyutlarında modülden geçirilen insan sesi kayıtları (sol üstten saat yönünde)**



**Şekil 4.19: 8192 (üst) ve 16384 (alt) pencere boyutlarında, modülün geçirilen insan sesi kayıtları**




**Şekil 4.20: Testte kullanılan insan sesi kaydı**



Özetlemek gerekirse tüm pencere boyutlarında farklı spektral durumlar mevcuttur. 512 pencere boyutundaki örnekten itibaren, pencere boyutu arttıkça seviyesi ve büyüklüğü artan eko oluşumları gözlenmiştir. Bu *reverb* efektinin haricinde, yeniden oluşturulan seslerin sahip olduğu dokuların, orijinal kayda genel olarak yakın olduğunu söylemek mümkündür.

### 4.1.3 Elektrik Gitar Sesi Testleri

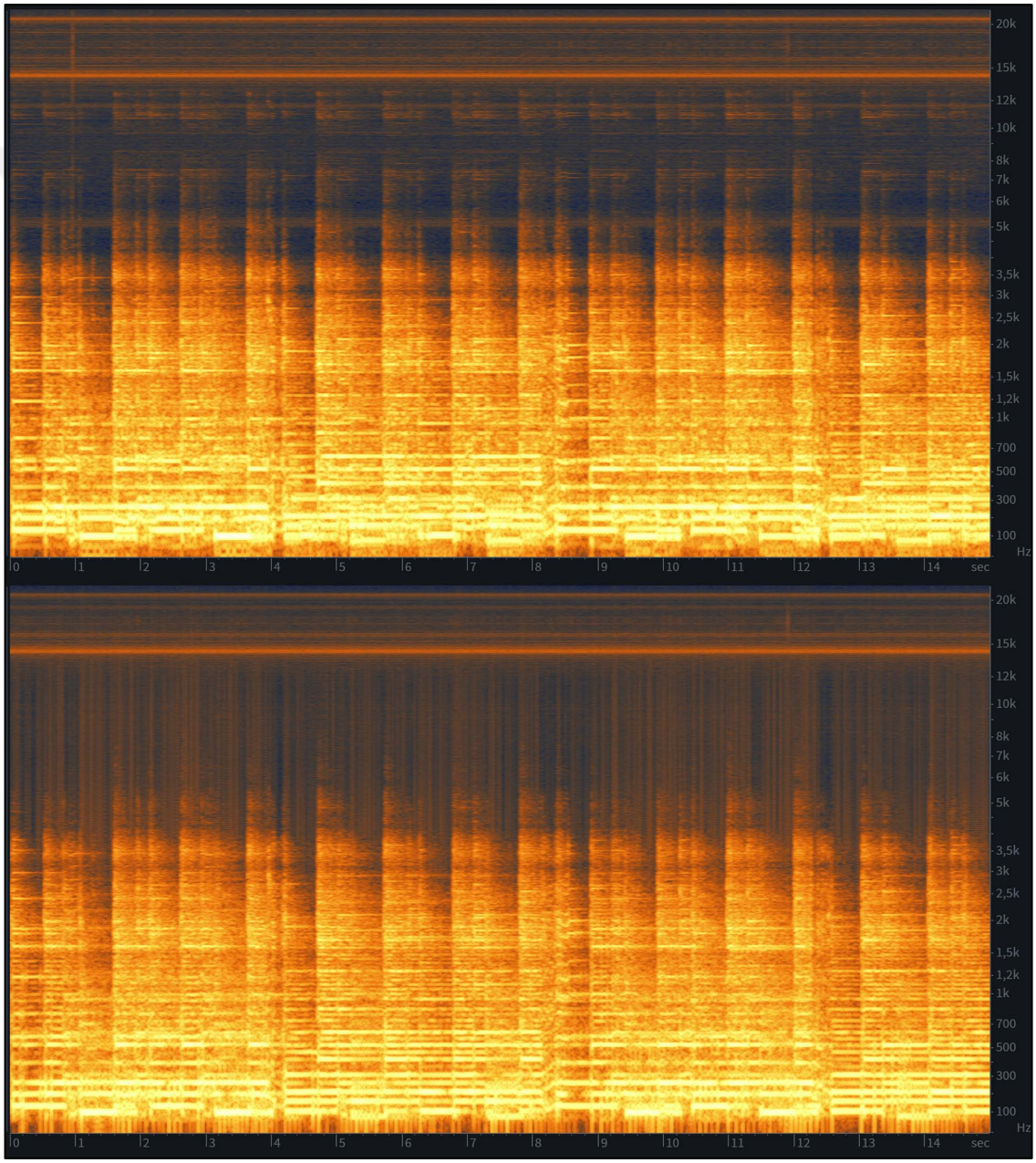
Bu testlerde Bahçeşehir Üniversitesi Ses Kayıt Stüdyolarında yapılan elektrik gitar kayıtları kullanılmıştır. Kayıtların tamamında Marshall JVM205H 50 Watt Tube Head ve Marshall 1960A Guitar Speaker Cabinet kullanılmıştır. Kayıtlar ElectroVoice RE20 ve Shure SM57 mikrofonlar ile Rupert Neve 5012 Dual Mic Pro Analog Pre-amp kullanılarak yapılmıştır. Mikrofonlar arasında RE20, 10 dB daha yüksek olacak şekilde seviye dengesi kurulmuş, fazları eşitlenmiş ve değişikliğe uğrayacak ses dosyaları 90 Hz. frekansından 12 dB/oktav *low cut* filtreden geçirilmiştir. Testlerde kullanılacak seslerin tamamı mono, 16 – Bit, 44100 örnek/saniye örnekleme hızındadır.

#### 4.1.3.1 Elektrik gitar sesinin farklı pencere fonksiyonu boyutları ile 4 aralık pese taşınması ve Pitch II karşılaştırması

Bu testlerde, Fender Stratocaster gitarla yapılmış olan ve akor ile arpejlerden oluşan elektrik gitar kayıtları, farklı pencere fonksiyonu boyutlarında, perde değiştirici modül ve Pitch II ile 4 aralık pese taşınmıştır. Modülün işleyişinde Hann pencere fonksiyonu kullanılmıştır. Kayıtlar modülden 512, 1024, 2048, 4096 ve 8192, Pitch II'den ise 11.6 ms (~512 örnek), 23.2 ms (~1024 örnek) ve 42.0 ms (~1853 örnek) pencere boyutlarıyla, *threshold* seçeneği kapalı olarak işleme konmuşlardır. Oluşturulan seslerin spektrogramları çıkartılmış ve perde değişimi işlemlerinin başarılı olduğu gözlenmiştir. Modülün oluşturduğu seslerin seviyelerinin, Pitch II'nin değişiklik yaptığı seslerin seviyelerinden yaklaşık olarak 5 dB daha yüksek olduğu görülmüştür. Spektrogramlarda sağlıklı karşılaştırma yapabilmek adına Pitch II'nin kullanıldığı seslerin seviyeleri Izotope RX7 programından 5 dB arttırılmıştır. Pitch II'nin değişiklik yaptığı seslerde,

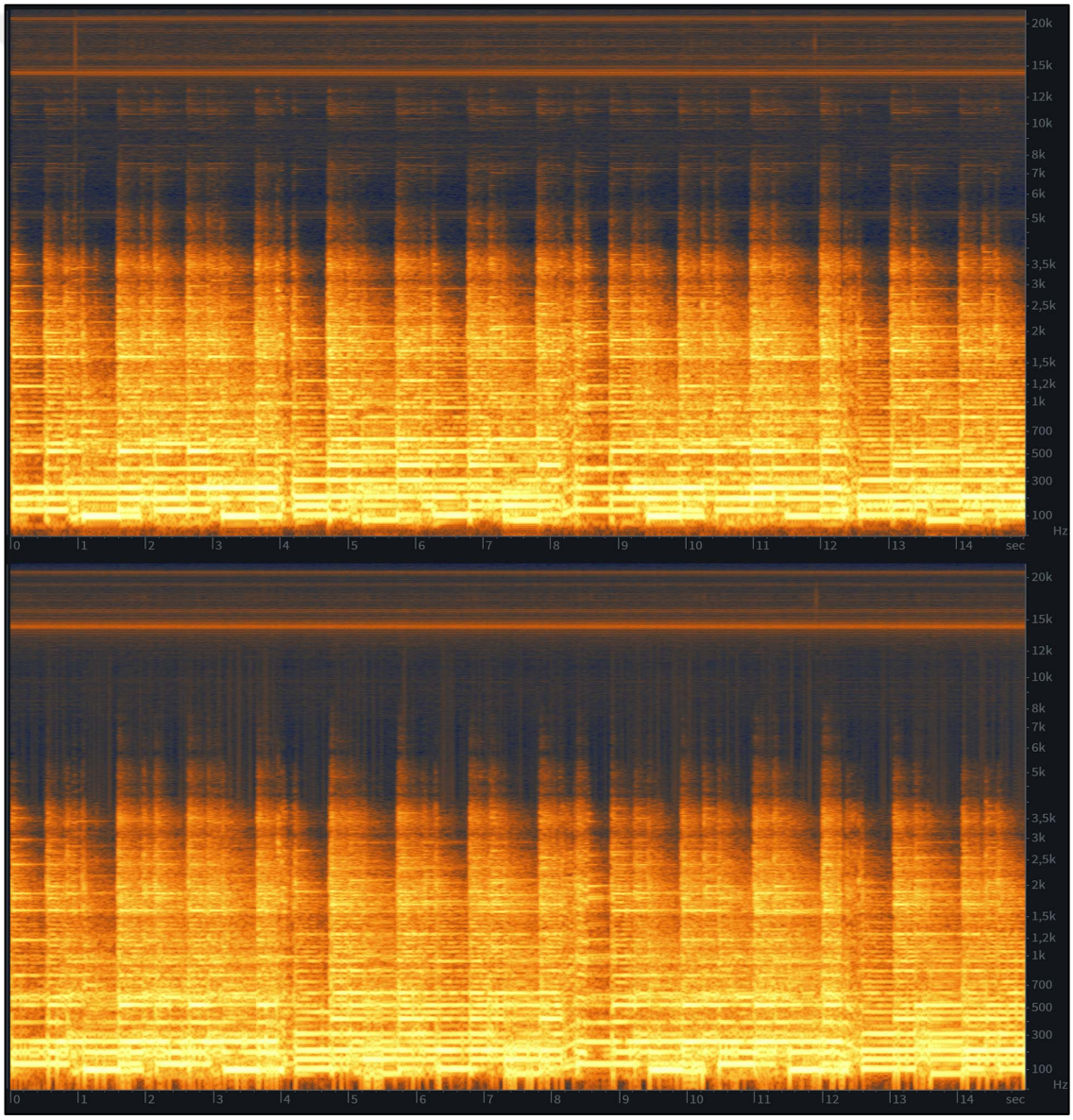
orijinal kayıtta olan dip gürültüsü form değiştirerek ön plana gelmiştir. Pencere boyutu küçüldükçe artan bu plak çıtırtısı benzeri ses modülün oluşturduğu seslerde yoktur. Bu ses haricinde Pitch II'nin seslerin perdelerini başarılı bir şekilde değiştirdiği söylenebilir.

**Şekil 4.21: 512 pencere boyutunda modülün (üst) ve Pitch II'nin (alt) 4 aralık pese taşıdığı elektrik gitar kayıtları**



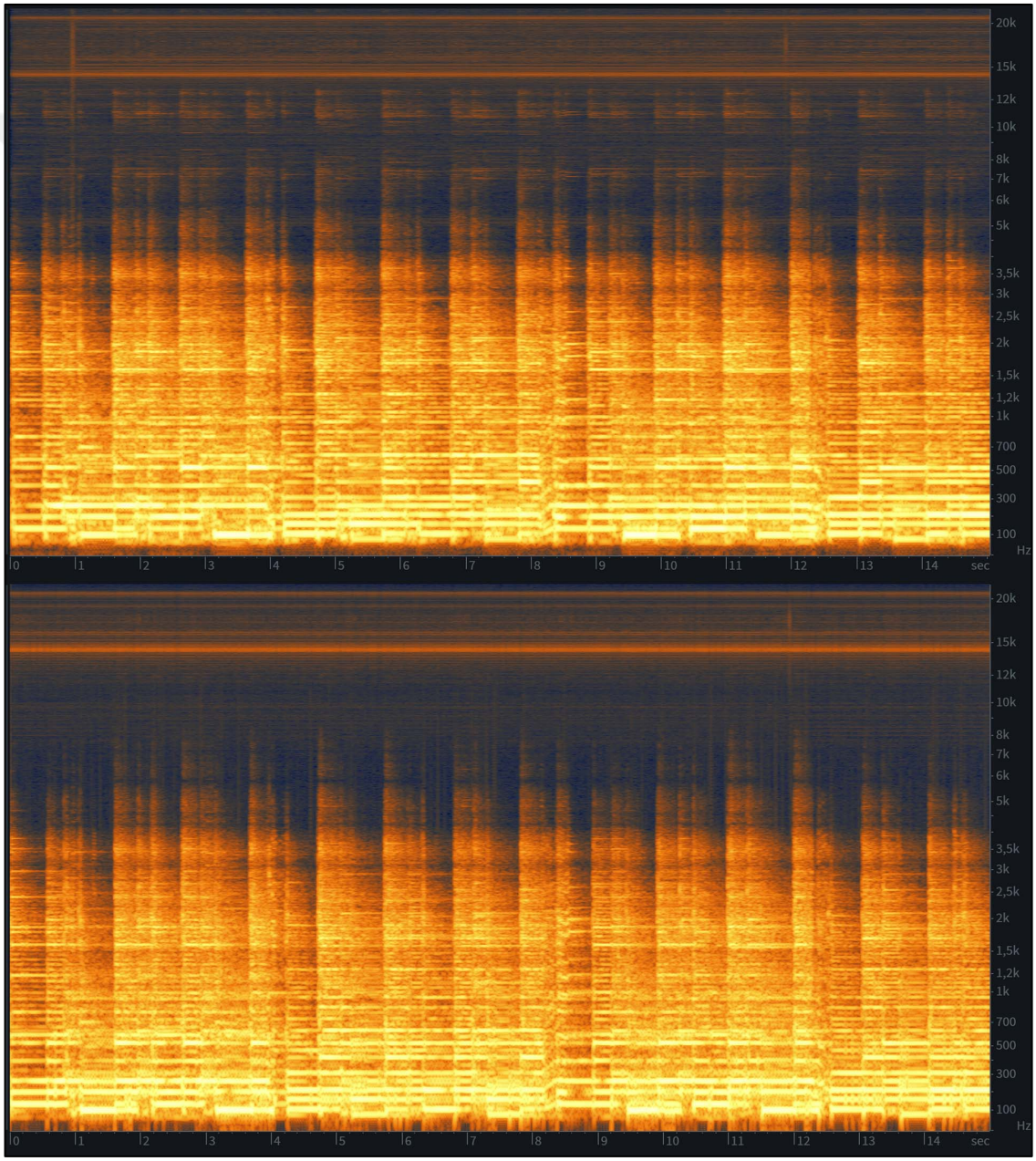
Modülün oluşturduğu 512 ve 1024 pencere boyutundaki örneklerde, özellikle partisyonun akor kısımlarında, *detune* ya da hız parametresi çok yüksek olan bir *phaser* efektini andıran oluşumlar gözlenmektedir. Bu oluşumlar 512 pencere boyutundaki örnekte nispeten daha fazladır. Bu oluşumların sebebi, pencere boyutunun küçülmesiyle, frekans çözünürlüğünün düşmesi ve dolayısıyla frekans kutucukları arasındaki sızıntılardır. Pencere boyutu büyüdükçe bu oluşumların yok olduğu görülmektedir.

**Şekil 4.22: 1024 pencere boyutunda modülün (üst) ve Pitch II'nin (alt) 4 aralık pese taşıdığı elektrik gitar kayıtları**



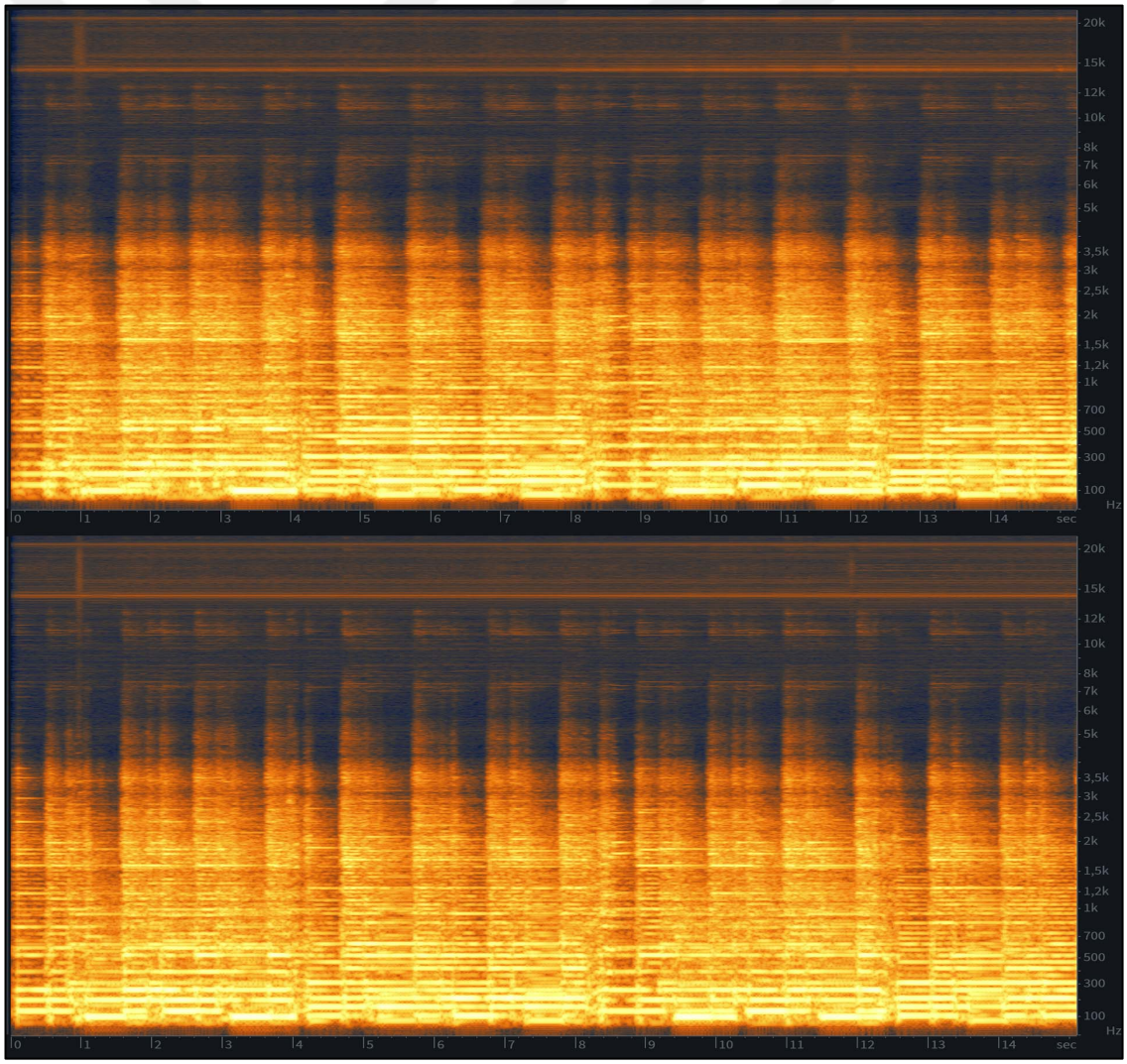
2048 ve 4096 pencere boyutundaki örnekler orijinal kaydın dokusuna yakın olmakla beraber bu örneklerde de *reverb* benzeri oluşumlardan bahsetmek mümkündür. 2048 pencere boyutundaki örnekte, küçük bir oda, 4096 pencere boyutundaki örnekteyse orta büyüklükte bir *reverb* efekti duyulmaktadır. 8192 pencere boyutunda bu reverb efektinin daha da büyüdüğü ve dolayısıyla partisyonda süreksizlik olduğu söylenebilir.

**Şekil 4.23: 2048 pencere boyutunda modülün (üst) ve Pitch II'nin (alt) 4 aralık pese taşıdığı elektrik gitar kayıtları**



Ayrıca orijinal kayıtta da bulunan *jitter* sesinin pencere boyutu büyüdükçe yayıldığı ve form değiştirdiği spektrogramlardan görülebilir. Pencere boyutu büyüdükçe, jitter sesinin git gide *hi-hat* sesini andırdığı söylenebilir. Bu ses özellikle 8192 pencere boyutundaki örnekte 200 ms'ye kadar uzamaktadır. Özetlemek gerekirse, düşük pencere boyutlarında, özellikle karmaşık akorlarda hedef frekansların etrafında, perde algısını bozan oluşumlar gözlenmiştir. Pencere boyutu büyüdükçe *reverb* efekti oluşumu gerçekleşmiştir. Ancak bu *reverb* efektlerine rağmen orijinal kaydın dokusuna yakın sesler elde edilmiştir.

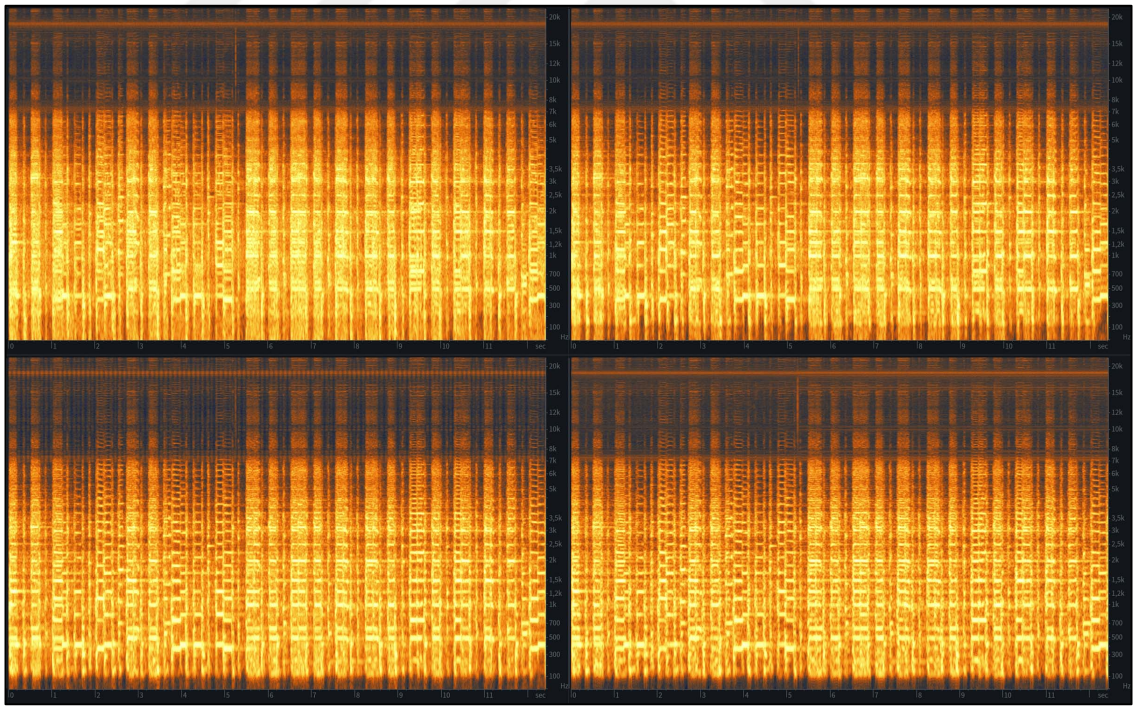
**Şekil 4.24: 4096 (üst) ve 8192 (alt) pencere boyutlarında modülün 4 aralık pese taşıdığı elektrik gitar kayıtları**



#### 4.1.3.2 Elektrik gitar sesinin 6 aralık tize taşınması ve pencere boyutları karşılaştırması

Bu testlerde, Gibson Les Paul Standard model gitarla yapılan ve kesik kesik çalınan (*staccato*) partisyonlar içeren elektrik gitar kayıtları, farklı pencere fonksiyonu boyutlarında, perde değiştirici modül ile 6 aralık tize taşınmıştır. Modülün işleyişinde yüzde 87.5 kesişim oranı korunarak Hann pencere fonksiyonu kullanılmıştır. Testler 256, 512, 1024, 2048, 4096, 8192 ve 16384 pencere boyutlarında gerçekleştirilmiştir. Değişiklik yapılan seslerin spektrogramları çıkartılmış ve perde değişikliklerinin başarılı olduğu görülmüştür. Oluşturulan seslerin seviyesinin, orijinal kayda göre 5 dB daha yüksek olduğu gözlenmiştir. Her pencere boyutunda farklı sızıntılar ve dolayısıyla farklı sesler oluşmaktadır.

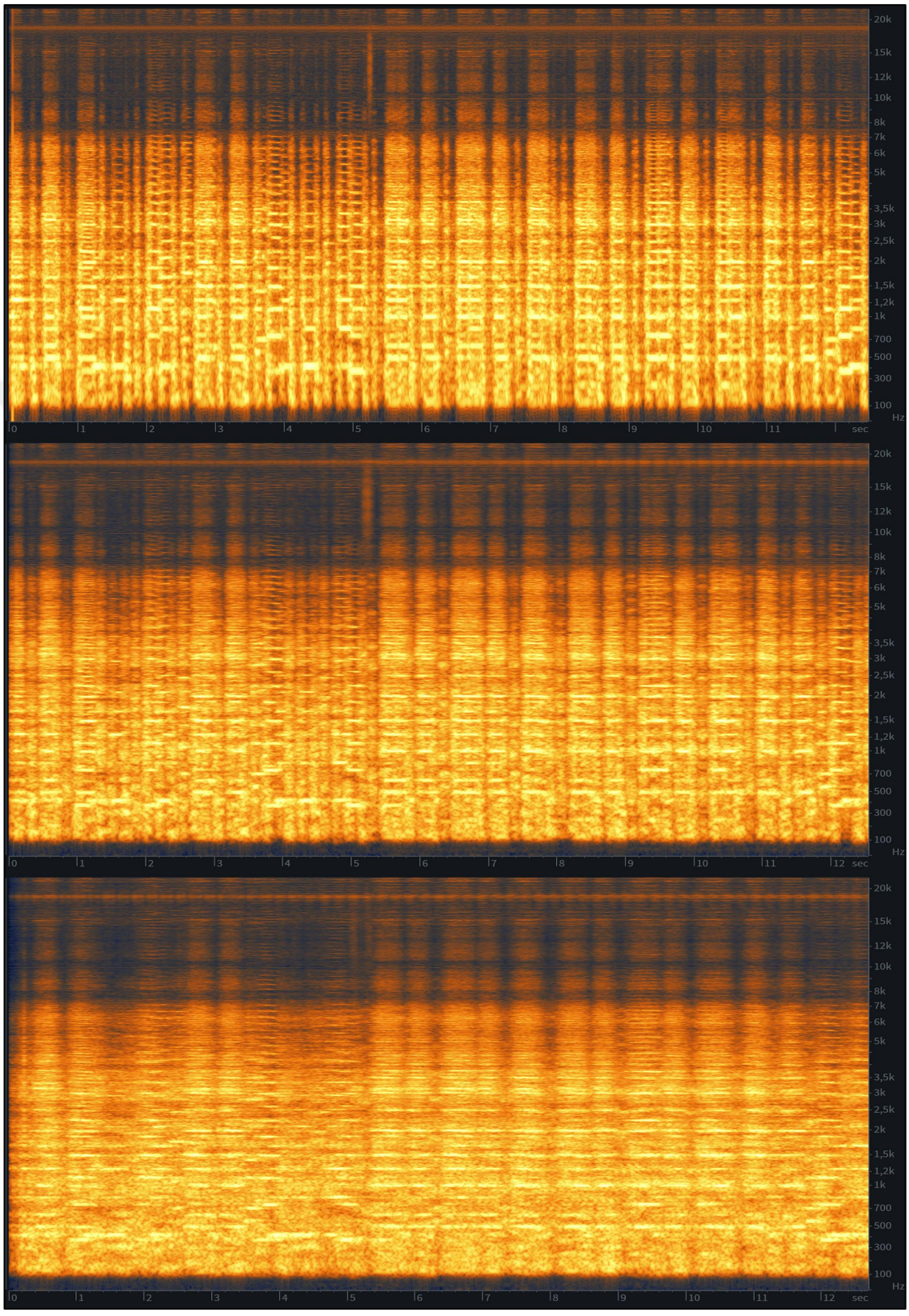
**Şekil 4.25: 256, 512, 1024 ve 2048 pencere boyutlarında modülün 6 aralık tize taşıdığı elektrik gitar kayıtları (sol üstten saat yönünde)**



Önceki testlerde olduğu gibi, pencere boyutu arttıkça seviyesi artan ve büyüyen bir *reverb* efektinden bahsetmek mümkündür. 256 pencere boyutunda meydana gelen

sızıntılar yine hız parametresi çok yüksek bir *phaser* efektini andırmaktadır. Bu örnekte yeniden oluşturulan sesin orijinal kayıttaki sesin dokusundan uzaklaştığı söylenebilir.
1024, 2048 ve 4096 pencere boyutlarındaki örneklerde, gittikçe artan bir *reverb* efekti gözlenmektedir. 8192 ve 16384 pencere boyutundaki örneklerde, yine nispeten büyük ve uzak duyulan bir *reverb* efekti mevcuttur. Ancak pencere boyutunun çok büyük olmasından dolayı üst üste gelen pencerelerin artmasıyla, özellikle 16384 örneğinde ses bulanıklaşmakta ve partisyondaki kesik kesik tavırlar yok olmaktadır. Bu testte, en büyük ve en küçük pencere boyutundaki örnekler haricindeki seslerin, genel olarak orijinal kayda yakın olduğu söylenebilir.

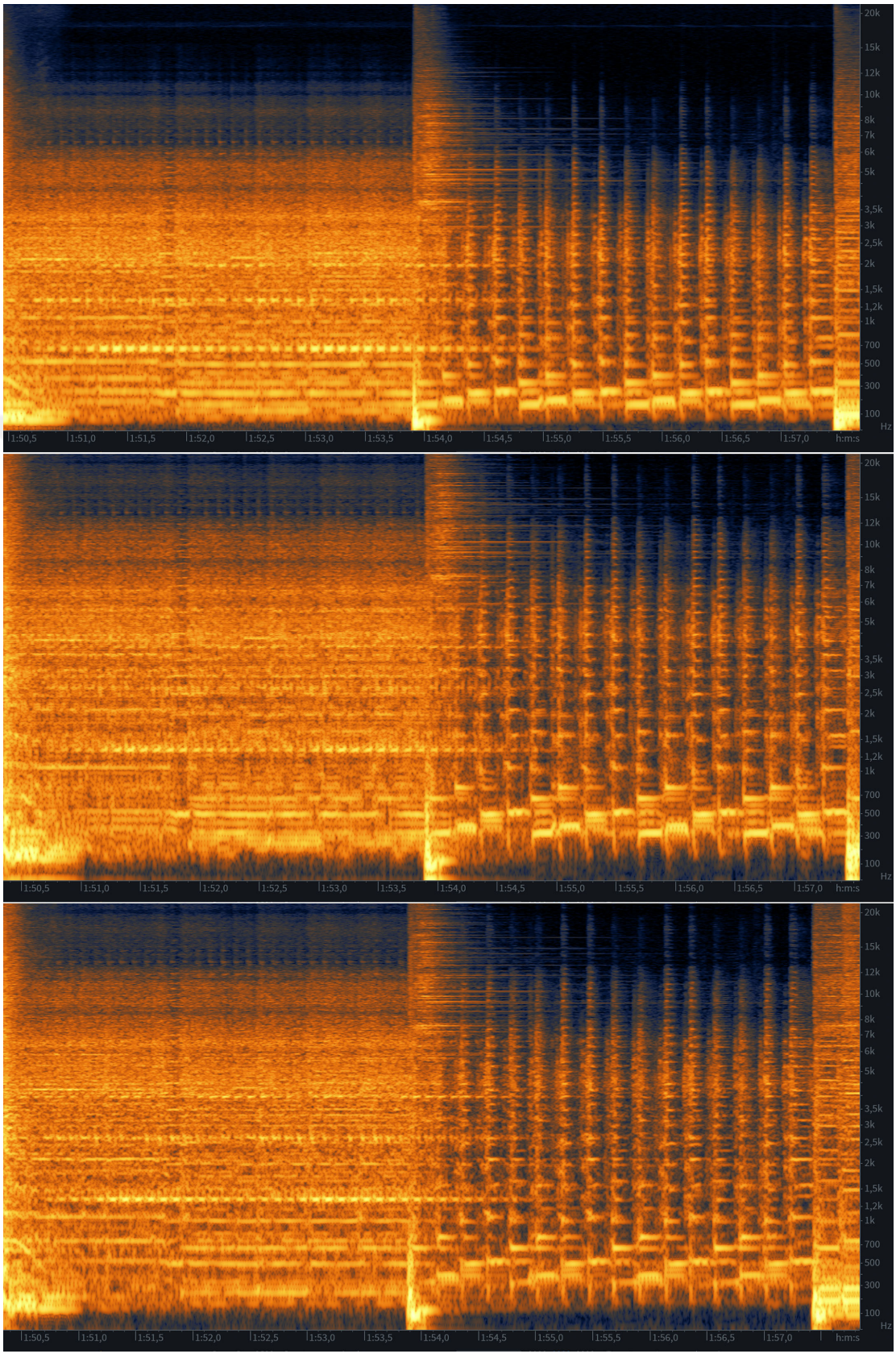
**Şekil 4.26: 4096, 8192 ve 16384 pencere boyutlarında modülün 6 aralık tize taşıdığı elektrik gitar kayıtları (yukarıdan aşağıya)**



### 4.1.4 Çok Kanallı Şarkı Testleri

Bu testlerde, mono, 16-bit, 44100 örnek/saniye örnekleme hızında, çok kanallı bitmiş bir müzik parçası, perde değiştirici modül aracılığıyla bir oktav tize taşınmıştır. Modülün işleyişinde Hann pencere fonksiyonu kullanılmıştır. Testler 1024 ve 2048 pencere boyutlarında gerçekleştirilmiştir. Parçanın orijinal halinin ve yeni oluşturulan seslerin spektrogramları çıkartılmış ve perde değişikliği işlemlerinin başarılı olduğu gözlenmiştir. Modülün oluşturduğu seslerin seviyelerinin, parçanın orijinal haline göre yaklaşık 5 dB daha yüksek olduğu söylenebilir. Her iki örnekte de farklı oluşumlar ve duysal dokular gözlenmektedir. 2048 pencere boyutundaki örnekte, spektral çözünürlüğü daha yüksek olduğundan ötürü, oluşan fazlalıklar 1024 pencere boyutundaki örneğe kıyasla daha azdır. Her iki örnekte de süreklilikte bir kayıp söz konusu değildir.

**Şekil 4.27: Orijinal parça, 1024 ve 2048 pencere boyutlarında değişiklikler**



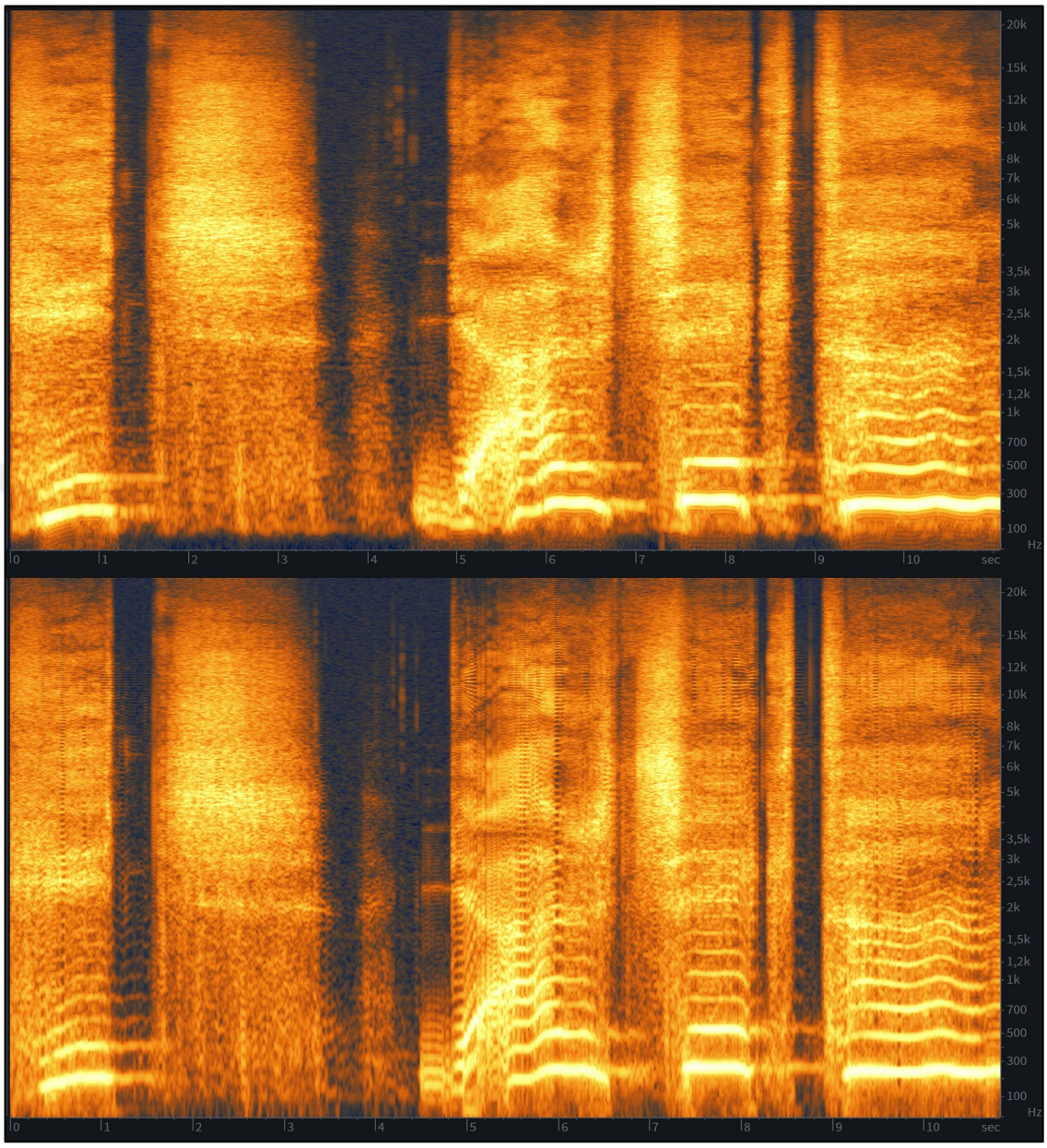
## 4.2 SESİ UZATAN/SIKIŞTIRAN MODÜL TESTLERİ

Bu testlerde, Bahçeşehir Üniversitesi Ses Kayıt Stüdyolarında yapılmış olan çeşitli vokal kayıtlarına, sesi uzatan/sıkıştıran modül aracılığıyla değişiklikler yapılmıştır. Bu değişikliklerin aynıları Pro Tools'da bulunan Time Compression/Expansion (TCE) plug-iniyle de gerçekleştirilmiş ve sonuçları karşılaştırılmıştır. TCE plug-ininde pencere boyutu parametresi yoktur ve sesin süresi en fazla dört katına çıkartılabilmekte ve çeyreğine sıkıştırılabilmektedir. Testlerde ayrıca modül aracılığıyla aynı ses kaynağına aynı işlem uygulanarak, farklı pencere boyutlarının ve pencere fonksiyonu tiplerinin etkileri incelenmiştir. Modül işleyişinde yüzde 87.5 kesişim oranı gözetilmiş ve Hann pencere fonksiyonu kullanılmıştır. Testlerde kullanılan kayıtlar Neumann U87 mikrofon ve Rupert Neve 5012 Dual Mic Pro Analog Preamfi kullanılarak yapılmıştır. Seslerin tamamı mono, 16 – Bit, 44100 örnek/saniye örnekleme hızındadır. Sesler, 100 Hz frekansından, 12 dB/oktav *low cut* filtreden geçirilerek işlemden geçirilmişlerdir.

### 4.2.1 İnsan Sesinin 4 Katına Uzatılması ve Time Compression/Expansion Karşılaştırması

Bu testlerde vokal kaydı, modül ve TCE plug-ini vasıtasıyla 4 katına uzatılmıştır. Modülde pencere boyutu 2048 seçilmiştir. Oluşturulan yeni seslerin spektrogramları çıkartılmıştır. TCE'nin oluşturduğu sesin ve orijinal kaydın seviyeleri birbirine yakınken, modülün oluşturduğu sesin seviyesi yaklaşık 6 dB daha yüksektir. Modülün işlediği seste frekans sızıntıları ve karışmaları mevcuttur. İki işlemde de süre dört katına çıkmıştır, ancak elde edilen seslerin dokuları arasında farklılıklar mevcuttur. Modülün işlediği kayıtta, FFT adımında gerçekleşen frekans karışmalarının ve üst üste ekleme işleminin sonucu olan fazlalık sesler mevcuttur. Küçük – orta boy bir oda *reverb* efekti duyulmaktadır. Ayrıca *doubler* veya *chorus* benzeri bir efekt duyulduğu da söylenebilir. Bu sebeplerden ötürü modülün oluşturduğu sesin diğerine kıyasla daha bulanık ve kirli olduğu söylenebilir. TCE'nin işlediği seste böyle fazlalıklar yoktur. Ancak bu seste de, modülün işlediği seste olmayan farklı bir durum vardır. Bazen çıtırtı sesini andıran, *click*/*jitter* vari sesler duyulmaktadır.

**Şekil 4.28: Modülün (üst) ve TCE'nin (alt) 4 katına uzattığı sesler**

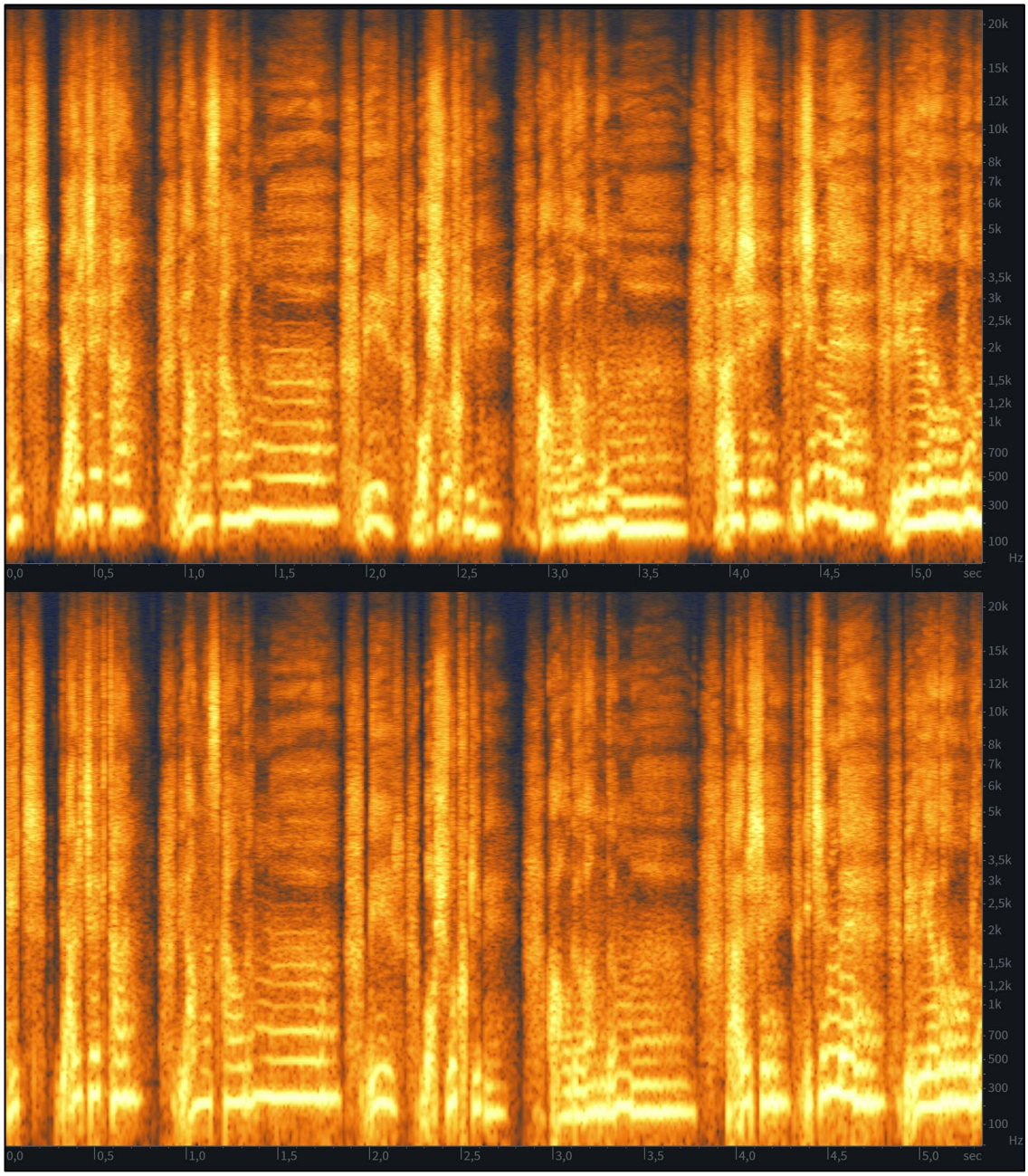



### 4.2.2 İnsan Sesinin 0.25 Katına Sıkıştırılması ve Time Compression/Expansion Karşılaştırması

Bu testlerde vokal kaydı, modül ve TCE plug-ini ile 0.25 katına (çeyreğine) sıkıştırılmıştır. Modülde pencere boyutu 2048 seçilmiştir. Oluşturulan yeni seslerin spektrogramları çıkartılmıştır. Modülün işlediği seste frekans sızıntıları ve karışmaları mevcuttur. Değişiklik yapılan sesler arasında farklılıklar vardır. Modülün işlediği

kayıtta yine fazla sesler mevcuttur. Küçük bir odayı andıran bir eko ve *chorus* benzeri bir efekt duyulmaktadır. TCE'nin işlediği seste böyle bir durum yoktur. Özetle modül, aynı işlemi, sese fazlalıklar ekleyerek ve sesin dokusunu değiştirerek gerçekleştirmiştir.

**Şekil 4.29: Modülün (üst) ve TCE'nin (alt) 0.25 katına sıkıştırdığı sesler**



### 4.2.3 İnsan Sesinin Farklı Pencere Fonksiyonu Boyutları ile 10 ve 100 Katına Uzatılması

Bu testlerde vokal kaydı, farklı pencere boyutları kullanılarak 10 ve 100 katına uzatılmıştır. Testler 512, 1024, 2048, 4096 ve 8192 pencere boyutlarında gerçekleştirilmiştir. Süresi 10 katına çıkarılan sesler incelendiğinde, her pencere boyutunda birbirinden farklı dokular oluştuğu görülmektedir. Baskın frekanslardaki sızıntılar, her örnekte farklı etkiler yaratmıştır. Küçük pencere boyutlarında bu sızıntıların daha fazla olduğu söylenebilir. 2048 örneğinde küçük bir oda, 4096'da ise daha büyük bir alan gibi duyulan bir *reverb* efekti duyulmaktadır. 8192 pencere boyutundaki örnekte, bu *reverb* efektinin biraz olduğu söylenebilir. Ancak üst üste gelen kısımların sürelerinin artmasıyla daha ziyade bir *delay* efekti gibi duyulmaktadır. Süresi 100 katına çıkarılan sesler incelendiğinde, bir önceki testte olduğu gibi, baskın frekansların sızıntılarından dolayı gerçekleşen ve pencere boyutu küçüldükçe artan sızıntılar görülmektedir. Her pencere boyutunda farklı bir dokunun oluştuğu söylenebilir. Oluşturulan seslerin insan sesinden çok uzaklaşmasından dolayı eko efektinden söz etmek mümkün değildir.

# 5. TARTIŞMA VE SONUÇ

Bu tez çalışmasında, Python programlama dilinin çeşitli avantajlarından faydalanmak suretiyle, basit ve hızlı çalışan bir perde değiştirici modül tasarlanmıştır. Modülün işleyişinde, seslerin pencereler halinde analiz edilip ardından yine pencereler halinde yeniden oluşturulduğu Phase Vocoder tekniği esas alınmıştır. Perde değerinde değişiklik yapılmak istenen ses, PV tekniği uygulanarak değişiklik yapılacak aralık miktarına bağlı bir faktöre göre uzatılır ya da kısaltılır. Ardından oluşturulan yeni ses örnekleri, doğrusal interpolasyon vasıtasıyla düzenlenerek baştaki ses dosyasının uzunluğuna getirilir. Bu sayede orijinal sesin süresine müdahale edilmeden perde değeri değiştirilmiş olur. Ayrıca, yeniden düzenleme işlemini gerçekleştirmeden, yalnızca belli bir faktöre bağlı olarak sesi uzatan ya da sıkıştıran bir modül de tasarlanmıştır. Python dili için geliştirilmiş olan, ücretsiz ve açık kaynaklı NumPy ve SciPy kütüphaneleri sayesinde bilimsel hesaplamalar kolaylaşmış ve modüllerin kodu bir hayli kısalmıştır. Modüllerin algoritmalarının oluşturulmasında NumPy kütüphanesinde bulunan dizi obje tipleri, pencere fonksiyonları, doğrusal interpolasyon fonksiyonu ve SciPy kütüphanesinde bulunan *wave* objesi kullanılmıştır. İşlemler, modüllerin daha hızlı çalışması için çevrimdışı gerçekleşmektedir. Yani gerçek zamanlı dinleme mümkün değildir. Modüller 16 – Bit, 44100 Hz çözünürlüğünde mono ses dosyalarını kabul etmektedir. Çalışma kapsamında, farklı ses kaynaklarına modüller aracılığıyla çeşitli değişiklikler yapılmış ve etkileri incelenmiştir. Karşılaştırmalar için bir referans olması açısından Pro Tools'da bulunan Pitch II ve Time Compression/Expansion programları kullanılmıştır. Ses kayıtlarına, parametreler değiştirilerek, çeşitli işlemler uygulanmış ve sonuçları karşılaştırılmıştır. Yapılacak değişikliğin miktarı, pencere fonksiyonu boyutu ve pencere fonksiyon tipi gibi değişkenlerin, sinüs sinyalleri, farklı yapı ve tavırdaki (*staccato*, legato vs.) insan sesi, elektrik gitar sesi gibi çeşitli ses kaynaklarını, her işlemde farklı etkilediği görülmüştür. Ayrıca perde değiştirici modül aracılığıyla, sesi, perde değerinde değişiklik yapmadan sinyal akışından geçirip farklı dokuda sesler üretmek mümkün olmuştur. Bu bölümde, yapılan tüm bu testlerin sonuçları irdelenmiş ve genel olarak yorumlanarak, mevcut işleyişin nasıl daha iyiye götürülebileceği tartışılmıştır.

Perde değiştirici modül ile gerçekleştirilen testlerde, perde değiştirme işleminin matematiksel olarak başarılı olduğu ve sesleri istenen yerlere taşıyabildiği görülmektedir. Ancak orijinal kayıtlarda görülmeyen, çeşitli frekanslarda fazlalıklar oluşmuştur. Bu oluşumların başlıca sebebi, FFT sırasında frekans kutucukları arasında gerçekleşen sızıntılardır. Frekans düzleminde işlem yapılırken, ardışık frekans noktaları arasında istenmeyen enerji geçişleri olmaktadır. Zaman düzlemine geri dönülerek üst üste ekleme işleminin gerçekleşmesiyle bu fazlalık sesler yayılmakta ve genlikleri artmaktadır. Bu fazlalık sesler, her pencere fonksiyonu tipinde ve boyutunda farklı miktarda oluşmaktadır. Test sonuçları incelendiğinde, yine farklı ses kaynaklarında biraz farklı sonuçlar olmakla birlikte, Hann pencere fonksiyonunun genel olarak bu sesleri en az üreten tip olduğu söylenebilir. Hamming pencere fonksiyonu, Hann pencere fonksiyonuna kıyasla daha kötü bir sonuç vermektedir. Alt frekanslarda istenmeyen geçişler diğer tiplere göre daha fazladır. Blackman pencere fonksiyonunun kullanıldığı örneklerde ise fazlalıklar baskın frekansların etrafında daha fazladır ve yer yer çeşitli efektleri andırmaktadır. Bu noktadan sonra yapılacak olan değerlendirmelerin tamamı Hann pencere fonksiyonu kullanılan testlerle alakalıdır. Tüm yorumlar bu durum göz önüne alınarak yapılmıştır.

Modülün oluşturduğu fazlalık seslerin başlıca sebebi FFT esnasında gerçekleşen sızıntılar olmakla birlikte, bu oluşumları etkileyen başka faktörler de mevcuttur. Değişiklik yapılacak ses kaynağının özellikleri, yapılacak değişikliğin miktarı, sesin pencereler halinde analiz edilmesi, pencere fonksiyonunun tipi, frekans düzleminde yapılan ardışık pencerelerin faz farklarına müdahale edilmesi ve pencere fonksiyonunun boyutu bu etkenlerdendir. Pencere boyutunun işlemlerdeki etkisi detaylıca incelenmiştir. Pencere boyutu küçüldükçe, FFT adımındaki frekans kutucuklarının sayısı da azaldığı için spektral çözünürlük düşmekte ve özellikle baskın frekansların etrafında sesin orijinal dokusunu bozan enerjiler birikmektedir. Bu enerjiler daha büyük pencere boyutlarında daha küçük alanlara yayılma eğilimindedirler. Her ses kaynağının kendine has özelliklerine göre farklı sonuçların görüldüğü işlemlerde, belirli bir pencere boyutuna kadar (genellikle 512 veya 1024), yer yer *phaser* ve *chorus* efektlerini andıran oluşumlar görülmüştür. Pencere boyutunun belirli bir eşiği geçmesinin ardından, bu efektler yerlerini ekoyu andıran bir efekte bırakma eğilimindelerdir. Pencere boyutunun

büyümesiyle, üst üste gelen alanların artmasının bu noktada doğrudan etkisi olduğu söylenebilir. Pencere boyutu büyüdükçe, daha büyük duyulan bir *reverb* efektini andıran oluşumlardan söz etmek mümkündür. Pencere boyutu, yine bir eşik değeri geçtiğinde (4096 veya 8192), kesişen sürelerin iyiden iyiye artmasıyla, bu yankı efekti bulanmakta ve orijinal kayıttaki devamlılık ve anlaşılırlık azalmaktadır. Pencere boyutunda yapılan değişiklikler, kullanılan sesin özelliklerine göre farklı efektlere ve spektral durumlara sebep olmaktadır. Bunların yanında yapılacak değişikliğin miktarı da işlemin kalitesini doğrudan etkilemektedir. Bunlara ek olarak, perde değiştirici modül, değişiklik yapılacak aralık sayısı 0 olarak girilip de çalıştırılabilmektedir. Bu sayede sesin perde değerlerinde değişiklik yapılmaksızın, pencereler halinde analiz edilip yeniden oluşturulabilmekte ve orijinal kayda farklı dokular kazandırılabilmektedir. Pencere boyutu seçimleri bu noktada yine önem teşkil etmektedir. Farklı pencere boyutlarında, farklı seviye ve büyüklüklerde *reverb* efektleri gözlenmiştir. Genel olarak pencere boyutunu büyüdükçe, hem efektin seviyesinin arttığı hem de büyüdüğü söylenebilir. Sinüs sinyali testlerinde, her pencere boyutunda farklı miktar ve lokasyonlarda gerçekleşen harmonik oluşumları gözlenmiştir. Daha karmaşık seslerle kıyaslandıklarında, sinüs sinyalleri tek bir frekans değeri içerdiği için, bu testlerde daha belirgin ve yüksek genlik değerlerinde geçişler gözlenmiştir. Pencere boyutu büyüdükçe FFT işlemindeki frekans çözünürlüğü de arttığından, küçük pencere boyutlarında, büyüklere nazaran bu geçişler daha belirgindir.

Sesi uzatan/sıkıştıran modül ile yapılan testlerde de seslerin süreleri başarıyla değiştirilmiştir. Ancak yine perde değiştirici modüldekine benzer ekstra seslerin oluşumları gözlenmektedir. Bu fazlalıklar, özellikle düşük pencere boyutlarında, baskın frekansların etrafında görülmektedir. Ayrıca, sesin uzatıldığı örneklerde pencere boyutu arttıkça büyüyen bir *reverb* efekti ve sesin sıkıştırıldığı örneklerde daha çok *chorus* ya da *phaser* efektlerini andıran dokular duymak mümkündür. 8192 gibi büyük pencere boyutlarında üst üste gelen örnek sayılarının artmasıyla *reverb* efektinin azaldığı ve yerini daha bulanık bir dokuya bıraktığı gözlenmiştir. Bunlara ek olarak, ses 100 katı gibi aşırı değerlere uzatıldığında da başarılı sonuçlar elde edildiği söylenebilir. Özellikle yüksek pencere boyutlarında, beklenenin ötesinde bir gürültü oluşumu yoktur.

Python dili ile yapılmış benzer çalışmalarla karşılaştırmalar yapıldığında, pitch-shifter-py programının algoritmasının, perde değiştirici modüle kıyasla daha sofistike ve detaylı olmasından ötürü daha iyi sonuçlar verdiği görülmüştür. Frekans düzleminde faz ve genlik değerlerine daha sağlıklı müdahaleler vardır. PV algoritmasının daha gelişmiş ve kontrollü olduğu, dolayısıyla FFT esnasında fazlalık oluşumlarınınn daha az olduğu söylenebilir. Kısa sesleri çok daha uzun sürelere çıkarmak amacıyla tasarlanmış olan PaulStretch programıyla karşılaştırıldığında, sesi uzatan/sıkıştıran modülün daha farklı dokular oluşturduğu söylenebilir. Bu noktada farkı yaratan durumun frekans düzleminde yapılan faz işlemleri olduğu söylenebilir. Modülde ardışık pencerelerin faz farkları takip edilerek kümülatif bir faz bilgisi oluşturulurken, PaulStretch programında her pencereye rastgele faz değerleri atanmaktadır.

Bu tez kapsamında, basit kodlama ve dijital sinyal işleme bilgileri kullanılarak, iki adet sinyal işleme modülü tasarlanmıştır. Modüllerin tasarımında, benzer işlevleri gören programlarla karşılaştırıldıklarında da görülebileceği üzere, kusur ve hatalar vardır. Bu bağlamda, mevcut algoritmaları iyileştirmek için yapılabilecekler irdelenmiştir. Öncelikle dijital sinyal işleme alanında daha fazla çalışma yapılarak, işlemlerin istenmeyen etkileri azaltılabilir ve kontrol alanları arttırılabilir. Fourier transformları, frekans düzleminde yapılan işlemler ve bunların etkileri üstüne çalışılarak faz bilgileri ve genel spektral durum açısından daha sağlıklı tasarımlar yapılabilir. Yapılan çalışmada, frekans düzleminde yapılan genlik ve faz değerleri değişikliklerinin, oluşacak olan efekti doğrudan etkilediği görülmüştür. Bu değişikliklerin, PV tekniğinin en önemli adımı olduğunu söylemek mümkündür. Frekans düzleminde farklı faz bilgileri değişiklikleri gerçekleştirilerek değişik ses efektleri üstüne çalışılabilir. Ayrıca, frekans düzleminde faz değerleriyle birlikte genlik değerlerine de müdahale edilerek yüksek genlikli frekansların sızıntıları önlenebilir. Ayrıca yine genlik değerlerine değişiklikler yapılarak oluşan harmoniklerin seviyeleri ve frekans değerlerine de müdahale etmek mümkün olabilir. Bu şekilde farklı ses efektleri elde edilebilir. Bunlara ek olarak, modülün işleyişinde değiştirilecek perde değeri ondalık sayı olarak da girilebileceğinden, mikrotonal olarak da perde değiştirmek mümkündür. Bu sayede, 12 ton eşit tamperamanlı batı müziği sistemine kısıtlı kalmadan, farklı duysal ve müzikal yapılar oluşturulabilir. Çeşitli enstrümanlarla yapılan kayıtlar, mikrotonal perde

değişiminden geçirilip, enstrümanların doğal halleriyle elde edilmesi mümkün olmayan, yarı tondan daha küçük aralıklarda sesler elde edilebilir. Öte yandan kodlama alanında çalışmalar yapılarak, modüller C/C++ diline çevrilebilir. Bu şekilde, kullanılan çeşitli obje ve fonksiyonlar sıfırdan yazılarak işlemlere özel değişiklikler yapılabilir. Bu şekilde tüm işlem süreçlerinde daha fazla kontrol sahibi olunabilir. Gerçek zamanlı işlem ve dinleme de bu şekilde mümkün olabilir.

# KAYNAKÇA

***Kitaplar***


Boulanger, R., & Lazzarini V., (Ed.). 2011. *The Audio Programming Book.* Massachusetts: The MIT Press.

Cangal, N., 2008. *Armoni*. 4. basım. Ankara: Arkadaş Yayınevi.

Downey, A. B., 2015. *Think Python.* 2. basım. Massachusetts: Green Tea Press.

Downey, A. B., 2014. *Think DSP.* Massachusetts: Green Tea Press.

McPherson, A.P. & Reiss J.D., 2015. *Audio effects theory, implementation and application.* Londra: CRC Press.

Miranda, E. R., 2002. *Computer sound design synthesis techniques and programming.* 2. basım. Oxford: Focal Press.

Önen, U., 2017. *Ses kayıt ve müzik teknolojileri.* 2. edisyon, 10. basım. İstanbul: Çitlembik Yayınları.

Phillips, D., 2015. *Python 3 object-oriented programming.* 2. basım. Birmingham: Packt.

Puckette, M., 2007. *The theory and technique of electronic music.* Singapur: World Scientific Publishing.

Roads, C., 2001. *Microsound*. Massachusetts: The MIT Press.

Smith, J.O., 2007. Mathematics of the Discrete Fourier Transform with Audio Applications. 2. basım. USA: W3K.

Tollervey, N. H., 2015. *Python in education.* California: O'Reilly Media.

Zölzer, U., 2008. *Digital audio signal processing.* 2. basım. UK: Wiley.

***Diğer Yayınlar***


Dudas, R. & Lippe, C. *The Phase Vocoder Part I.* https://cycling74.com/tutorials/the- phase-vocoder-%E2%80%93-part-i [erişim 25 Nisan 2020]

*History of Python.* https://www.geeksforgeeks.org/history-of-python/ [erişim 20 Aralık 2019]

NumPy resmi dokümantasyonu. https://www.numpy.org/ [erişim 19 Eylül 2018]

Python resmi dokümantasyonu. https://www.python.org/ [erişim 16 Eylül 2018]

SciPy resmi dokümantasyonu. https://www.scipy.org/ [erişim 19 Eylül 2018]

Tiwari, S. *Python for Engineers Audio and Digital Signal Processing in Python.* https://www.pythonforengineers.com/audio-and-digital-signal-processingdsp-in-python/ [erişim 8 Kasım 2018]

Woodall, C., 2018. *Python Pitch Shifter.* https://github.com/cwoodall/pitch-shifter-py [erişim 27 Mayıs 2019]

Nasca, P., 2011. *Paul's Extreme Sound Stretch.* http://hypermammut.sourceforge.net/paulstretch/ [erişim 10 Aralık 2018]

# EKLER

## Ek A.1 Perde değiştirici modül kodu

```
import scipy.io.wavfile
import numpy as np
from scipy.interpolate import interp1d
from scipy.io.wavfile import write
from scipy.io.wavfile import read

def read_wave(filename):
    wave = scipy.io.wavfile.read(filename)
    return wave

def zero_padding(wave, window_size):
    sound = wave[1]
    window_size = int(window_size)
    hop_size = int(window_size / 8)
    sound = np.append(sound, np.zeros(window_size + hop_size))
    sound = np.append(np.zeros(hop_size), sound)
    return sound

def phase_vocoder(sound, factor, window_size):
    window = np.hanning(window_size)
    factor = 1.0/factor
    sound *= (1.0/32768.0)
    hop_size = int(window_size/8)
    sound_len = int(len(sound))
    phases = np.zeros(window_size)
    output = np.zeros(int(sound_len / factor + window_size + hop_size))

    for i in np.arange(0, sound_len - window_size - hop_size, hop_size*factor):
        i = int(i)
        ws1 = sound[i: i + window_size]
        ws2 = sound[i + hop_size: i + hop_size + window_size]
        ws1_ = np.fft.fft(window * ws1)
        ws2_ = np.fft.fft(window * ws2)
        amp = abs(ws2_)
        phases = (phases + np.angle(ws2_/ws1_)) % (2*np.pi)
```

```
            new = amp * np.exp(1j*phases)
            wsx = np.fft.ifft(new) * window
            i2 = int(i / factor)
            output[i2: i2 + window_size] += wsx.real

    output[output > 1.0] = 1.0
    output[output < -1.0] = -1.0
    output *= 32767.0

    return output.astype('int16')

def resample(sound, factor):
    sound_len = len(sound)
    output_len = int(sound_len/factor)
    interpol = interp1d(np.arange(0, sound_len), sound)
    resampled = np.linspace(0, sound_len - 1, output_len)
    return interpol(resampled).astype('int16')

def pitch_shift(wave, n, window_size=2048):
    hop_size = int(window_size / 8)
    factor = 2**(1.0 * n / 12.0)
    sound_zero = zero_padding(wave, window_size)
    sound_new = phase_vocoder(sound_zero, factor, window_size)
    return resample(sound_new, factor)

scipy.io.wavfile.write('filename', 44100, sound)

###########

sound = read_wave('test.wav')

pitch_shift(sound, 12)

```

## Ek A.2 Sesi uzatan/sıkıştıran modül kodu

```
import scipy.io.wavfile
import numpy as np
from scipy.io.wavfile import write
from scipy.io.wavfile import read

def read_wave(filename):
    wave = scipy.io.wavfile.read(filename)
    return wave

def zero_padding(wave, window_size):
    sound = wave[1]
    window_size = int(window_size)
    hop_size = int(window_size / 8)
    sound = np.append(sound, np.zeros(window_size + hop_size))
    sound = np.append(np.zeros(hop_size), sound)
    return sound

def phase_vocoder(sound, factor, window_size):
    window = np.hanning(window_size)
    factor = 1.0/factor
    sound *= (1.0/32768.0)
    hop_size = int(window_size/8)
    sound_len = int(len(sound))
    phases = np.zeros(window_size)
    output = np.zeros(int(sound_len / factor + window_size + hop_size))

    for i in np.arange(0, sound_len - window_size - hop_size, hop_size*factor):
        i = int(i)
        ws1 = sound[i: i + window_size]
        ws2 = sound[i + hop_size: i + hop_size + window_size]
        ws1_ = np.fft.fft(window * ws1)
        ws2_ = np.fft.fft(window * ws2)
        amp = abs(ws2_)
        phases = (phases + np.angle(ws2_/ws1_)) % (2*np.pi)
        new = amp * np.exp(1j*phases)
```

```
36          wsx = np.fft.ifft(new) * window
37          i2 = int(i / factor)
38          output[i2: i2 + window_size] += wsx.real
39
40      output[output > 1.0] = 1.0
41      output[output < -1.0] = -1.0
42      output *= 32767.0
43
44      return output.astype('int16')
45
46  def elastic(wave, factor, window_size):
47      sound_zero = zero_padding(wave, window_size)
48      return phase_vocoder(sound_zero, factor, window_size)
49
50  scipy.io.wavfile.write('filename', 44100, sound)
51
52  ###########
53
54  sound = read_wave('test.wav')
55
56  elastic(sound, 2, 2048)
57
```

**Ek A.3 Testlerde kullanılan ses dosyaları**

Perde Değiştirici Modül Testleri

Sinyal testleri

440-880Hz_M.wav
440-880Hz_P2.wav
440-1760Hz_M.wav
440-1760Hz_P2.wav
440Hz_SG.wav
880Hz_SG.wav
1760Hz_SG.wav
220Hz_SG.wav
440-220Hz_M.wav
440-220Hz_P2.wav
220-880Hz_512.wav
220-880Hz_1024.wav
220-880Hz_2048.wav
220-880Hz_4096.wav
220-880Hz_8192.wav
220-880Hz_16384.wav

İnsan sesi testleri

vox_+5_1024_P2.wav
vox_+5_1854_P2.wav
vox_+5_M_1024.wav
vox_+5_M_2048.wav
vox_1_orj.wav
vox_-3_1024_P2.wav
vox_-3_1854_P2.wav

vox_-3_M_256.wav
vox_-3_M_512.wav
vox_-3_M_1024.wav
vox_-3_M_2048.wav
vox_-3_M_4096.wav
vox_-3_M_8192.wav
vox_-3_M_16384.wav
vox_2_orj.wav
vox_+2_M_512_BLACKMAN.wav
vox_+2_M_512_HAMMING.wav
vox_+2_M_512_HANN.wav
vox_+2_M_1024_BLACKMAN.wav
vox_+2_M_1024_HAMMING.wav
vox_+2_M_1024_HANN.wav
vox_+2_M_2048_BLACKMAN.wav
vox_+2_M_2048_HAMMING.wav
vox_+2_M_2048_HANN.wav
vox_+2_M_4096_BLACKMAN.wav
vox_+2_M_4096_HAMMING.wav
vox_+2_M_4096_HANN.wav
vox_3_orj.wav
vox_4_orj.wav
vox_32.wav
vox_64.wav
vox_128.wav
vox_256.wav
vox_512.wav
vox_1024.wav
vox_2048.wav
vox_4096.wav
vox_8192.wav
vox_16384.wav

Elektrik gitar sesi testleri

STRAT_-4_M_512.wav
STRAT_-4_M_1024.wav
STRAT_-4_M_2048.wav
STRAT_-4_M_4096.wav
STRAT_-4_M_8192.wav
STRAT_-4_P2_512.wav
STRAT_-4_P2_1024.wav
STRAT_-4_P2_1854.wav
STRAT.wav
Gibson_6_M_256.wav
Gibson_6_M_512.wav
Gibson_6_M_1024.wav
Gibson_6_M_2048.wav
Gibson_6_M_4096.wav
Gibson_6_M_8192.wav
Gibson_6_M_16384.wav
Gibson.wav

Çok kanallı şarkı testleri

dipsiz_+1_1024.wav
dipsiz_+1_2048.wav
dipsiz_1644_mono.wav

Sesi uzatan/sıkıştıran modül testleri

vox_4_M.wav
vox_4_TCE.wav
vox_forelastic_4.wav

vox_0.25_M.wav

vox_0.25_TCE.wav

vox_forelastic_0.25.wav

vox_10_512.wav

vox_10_1024.wav

vox_10_2048.wav

vox_10_4096.wav

vox_10_8192.wav

vox_100_512.wav

vox_100_1024.wav

vox_100_2048.wav

vox_100_4096.wav

vox_100_8192.wav