Penerapan Interpolasi dalam Mengubah Kecepatan Audio Tanpa Mengubah *Pitch*

Muhammad Izzat Jundy 13523092[1]
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jl. Ganesha 10 Bandung 40132, Indonesia
[1]13523092@std.stei.itb.ac.id, [1]izzatjundy04@gmail.com

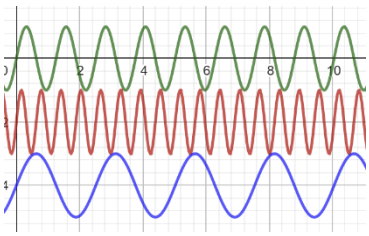
***Abstrak*— Pengubahan kecepatan video sering dilakukan untuk menambah dinamika dalam pengeditan video. Namun, perubahan ini biasanya turut mengubah *pitch* audio, yang lebih rendah ketika diperlambat dan lebih tinggi ketika dipercepat, terutama pada perangkat lunak pengedit video yang kurang canggih. Penelitian ini bertujuan untuk menyelidiki konsep dasar yang membedakan algoritma canggih tersebut dengan perangkat lunak pengedit video sederhana. Dengan memahami konsep dasar ini, diharapkan ditemukan cara yang lebih efektif untuk mengubah kecepatan audio tanpa mengubah *pitch*, sehingga dapat mengurangi gangguan pada kualitas suara dalam pengeditan video. Dilakukan implementasi dengan memanfaatkan *short-time fourier transform* dan interpolasi. Diperoleh hasil kasar yang dapat memberikan gambaran metode yang dapat dikembangkan untuk digunakan dalam mengubah kecepatan audio tanpa mengubah *pitch*-nya.**

***Kata kunci*—*pitch*, audio, interpolasi, *short-time fourier transform*.**

I. PENDAHULUAN

Dalam video editing, pengubahan kecepatan video kerap kali dilakukan untuk menambah dinamika dari video yang diedit. Pengubahan kecepatan ini tidak hanya mengubah kecepatan dari gambar, melainkan juga suara yang membersamainya. Efek samping dari perubahan kecepatan suara yang seringkali dialami seorang editor video adalah berubahnya *pitch* dari suara yang diubah kecepatannya. *Pitch* dari suara akan lebih rendah ketika diperlambat, dan akan lebih tinggi ketika dipercepat. Hal ini sering terjadi pada editor video yang menggunakan perangkat lunak pengedit video yang kurang canggih. Ketika kecepatan audio diubah, perangkat lunak pengedit video yang sederhana sering kali mengandalkan teknik dasar yang hanya mengubah durasi audio tanpa memisahkan elemen-elemen frekuensi yang membentuk suara tersebut. Akibatnya, perubahan kecepatan audio langsung berpengaruh pada frekuensinya, yang secara otomatis mengubah *pitch* suara yang dihasilkan. *Pitch* yang lebih tinggi saat mempercepat audio dan *pitch* yang lebih rendah saat memperlambat audio menjadi efek yang tidak dapat dihindari dalam pengeditan video menggunakan perangkat lunak tersebut.

Efek samping ini terjadi ketika audio langsung dimampatkan saat dipercepat atau langsung diregangkan saat diperlambat berdasarkan domain waktu. Peregangan atau pemadatan langsung tersebut akan menyebabkan frekuensi menjadi lebih tinggi saat audio dimampatkan (lebih banyak gelombang dalam satuan waktu) dan menjadi lebih rendah saat audio diregangkan (lebih sedikit gelombang dalam satuan waktu). Perubahan frekuensi inilah yang menyebabkan terjadinya perubahan *pitch*, yakni menjadi lebih tinggi saat audio dipercepat dan menjadi lebih rendah saat audio diperlambat. Perubahan ini, meskipun tidak dapat dihindari pada perangkat lunak pengedit video sederhana, sering kali mengganggu kualitas keseluruhan dari video yang diedit. Visualisasi gelombang yang menunjukkan perubahan ini dapat dilihat pada gambar 1, di mana gelombang hijau menggambarkan gelombang suara asli, gelombang merah menggambarkan gelombang yang dipadatkan saat dipercepat, dan gelombang biru menggambarkan gelombang yang diregangkan saat diperlambat.



Gambar 1. Gelombang awal (hijau), gelombang setelah dimampatkan secara langsung (merah), dan gelombang setelah diregangkan secara langsung berdasarkan domain waktu

Namun, meskipun pengubahan *pitch* ini menjadi masalah yang umum dalam pengeditan video, terdapat metode-metode yang lebih canggih untuk mengubah kecepatan audio tanpa mengubah *pitch*-nya. Salah satu contohnya adalah teknologi yang digunakan oleh platform seperti YouTube. YouTube, yang merupakan salah satu platform video terbesar dan paling banyak digunakan, memiliki fitur yang memungkinkan pemutaran video pada kecepatan yang lebih cepat atau lebih lambat tanpa mengubah *pitch* audio yang diputar. Fitur ini sangat berguna dalam berbagai konteks, seperti untuk mempercepat durasi video tutorial tanpa mengorbankan

kejelasan suara, atau memperlambat video tanpa membuat suara terdengar aneh atau tidak wajar. Dengan menggunakan algoritma yang canggih, YouTube berhasil mengubah kecepatan pemutaran video sambil menjaga agar suara tetap terdengar alami dan sesuai dengan konteks.

Penelitian ini bertujuan untuk menyelidiki secara kasar metode yang digunakan oleh YouTube dan platform serupa dalam fitur pengubahan kecepatan pemutaran video. Walaupun penelitian ini tidak akan membahas algoritma-algoritma yang digunakan YouTube secara mendalam, mengingat teknologi yang digunakan oleh platform besar seperti YouTube diduga sangat kompleks, tujuan utama dari penelitian ini adalah untuk mengulik konsep-konsep dasar yang membedakan algoritma canggih yang digunakan oleh YouTube dengan perangkat lunak pengedit video yang lebih sederhana dan kurang canggih. Dengan pemahaman lebih dalam tentang konsep dasar ini, diharapkan dapat ditemukan cara-cara yang lebih efektif untuk menangani pengeditan audio dan video tanpa mengorbankan kualitas suara secara signifikan.

II. Dasar Teori

A. Frekuensi

Frekuensi adalah salah satu konsep dasar dalam fisika dan teknik yang mengacu pada jumlah getaran, osilasi, atau siklus yang terjadi dalam waktu tertentu. Dalam konteks gelombang, frekuensi adalah banyak gelombang tiap satuan waktu. Satuan untuk frekuensi adalah Hertz (Hz), dengan 1 Hz sama dengan satu siklus gelombang (satu bukit dan satu lembah) per detik. Frekuensi berbanding terbalik dengan waktu yang diperlukan untuk membentuk satu bukit dan satu lembah. Hubungan perbandingan tersebut dijabarkan dalam (1).

$$f = \frac{1}{T} \tag{1}$$

Dengan f adalah frekuensi dan T adalah waktu yang diperlukan untuk membentuk satu bukit dan satu lembah.

B. Pitch dan Hubungannya dengan Frekuensi

Pitch adalah sebuah persepsi subjektif yang mereprentasikan tinggi rendahnya suara. *Pitch* berhubungan langsung dengan frekuensi. Frekuensi adalah banyak gelombang dalam satuan waktu. Semakin tinggi frekuensi sebuah suara, maka akan semakin tinggi *pitch*-nya. Sebaliknya, semakin rendah frekuensi sebuah suara, maka akan semakin rendah *pitch*-nya.

Pitch adalah sebuah persepsi subjektif yang merepresentasikan tinggi rendahnya suara yang kita dengar. Sebagai contoh, suara sebuah piano yang memainkan nada tinggi seperti C4 memiliki *pitch* yang lebih tinggi dibandingkan dengan suara sebuah drum yang menghasilkan *pitch* rendah. Meskipun *pitch* adalah persepsi manusia terhadap suara, *pitch* sangat bergantung pada parameter fisik yang dapat diukur, yaitu frekuensi. Semakin tinggi frekuensi suatu suara, maka semakin banyak siklus gelombang yang terjadi dalam satu detik, dan hal ini menyebabkan kita merasakan suara tersebut sebagai *pitch* yang lebih tinggi.

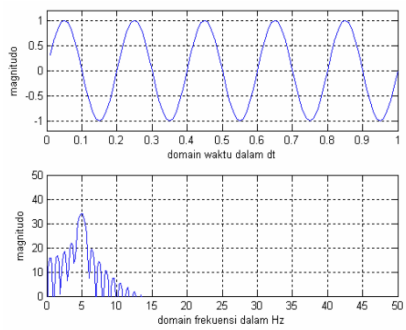
Sebaliknya, ketika frekuensi suara menurun, artinya gelombang suara tersebut memiliki lebih sedikit siklus dalam satu detik, yang menyebabkan *pitch* yang terdengar menjadi lebih rendah. Misalnya, suara dengan frekuensi 20 Hz akan terdengar jauh lebih rendah dibandingkan dengan suara dengan frekuensi 1000 Hz. Dengan kata lain, *pitch* yang kita dengar tidak terlepas dari jumlah gelombang yang melewati suatu titik dalam waktu tertentu. Frekuensi suara yang lebih tinggi cenderung menghasilkan suara yang tajam, seperti suara fluit atau biola, sementara frekuensi suara yang lebih rendah menghasilkan suara yang dalam dan berat, seperti suara bass pada gitar atau suara gendang.

Penting untuk dicatat bahwa hubungan antara *pitch* dan frekuensi ini bersifat linier dalam rentang tertentu. Namun, persepsi manusia terhadap perubahan *pitch* tidak selalu bersifat linier. Misalnya, perubahan frekuensi dari 200 Hz menjadi 400 Hz mungkin dirasakan sebagai peningkatan *pitch* yang lebih besar dibandingkan perubahan dari 1000 Hz menjadi 1200 Hz. Ini disebabkan oleh cara otak manusia memproses perubahan *pitch*, yang lebih sensitif terhadap perubahan pada frekuensi rendah dibandingkan dengan frekuensi tinggi. Oleh karena itu, walaupun secara fisik frekuensi dan *pitch* berhubungan langsung, persepsi *pitch* sering kali bergantung pada konteks dan rentang frekuensi yang lebih luas.

Dalam konteks pengeditan video dan audio, pemahaman yang baik tentang hubungan antara *pitch* dan frekuensi sangat penting, terutama saat melakukan perubahan kecepatan pada audio. Perubahan frekuensi yang tidak diinginkan dapat menyebabkan perubahan *pitch* yang tidak diinginkan pula, yang dapat mengganggu pengalaman audio. Jadi, diperlukan metode untuk dapat mengubah kecepatan audio tanpa mengubah *pitch* yang dibahas pada penelitian ini.

C. Short-Time Fourier Transform

Transformasi Fourier adalah sebuah metode untuk menganalisis domain waktu dari suatu sinyal dan mengubahnya ke domain frekuensi. Dengan melakukan hal tersebut, kita dapat mengetahui secara lebih jelas tentang komponen frekuensi yang menyusun sebuah sinyal. Transformasi Fourier memiliki kekurangan berupa ketidakjelasan domain waktu. Hal ini menyebabkan Transformasi Fourier hanya berfungsi dengan baik pada sinyal stasioner. Pada transformasi sinyal non-stasioner menggunakan Transformasi Fourier, informasi perubahan frekuensi terhadap satuan waktu tidak akan tertangkap. Oleh sebab itu, diperlukan *Short-Time Fourier Transform*.



Gambar 2. Sinyal dalam domain waktu (atas) dan sinyal dalam domain frekuensi (bawah)
Sumber: https://tribudi.lecturer.pens.ac.id/LN_Sinyal_sistem_Prak/prak_SinyalSistem_6.pdf

Short-Time Fourier Transform melakukan transformasi dari domain waktu ke domain frekuensi-waktu dengan menggunakan partisi, sehingga informasi waktu dapat tetap terjaga. Partisi dibuat dengan sesuatu yang dinamakan jendela. Jendela ini "mengambil" bagian sinyal sedemikian sehingga dapat dianggap sebagai sinyal stasioner, yang kemudian ditransformasi menggunakan Transformasi Fourier. Proses *Short-Time Fourier Transform* menghasilkan keluaran berupa matriks kompleks yang merepresentasikan magnitudo dan fase dari frekuensi pada setiap segmen waktu (menjadi spektogram).

Magnitudo yang dimaksud adalah amplitudo atau energi dalam domain frekuensi-waktu, didapatkan dari nilai absolut dari elemen-elemen matriks kompleks. Sementara fase adalah sudut dari elemen kompleks. Hubungan antar ketiganya dijabarkan sebagai (2).

$$S(f,t) = |S(f,t)|e^{j\theta(f,t)} \tag{2}$$

Dengan f adalah frekuensi, t adalah waktu, $S(f,t)$ adalah elemen-elemen matriks kompleks, $|S(f,t)|$ adalah magnitudo, dan $\theta(f,t)$ adalah fase komponen frekuensi pada waktu t. Audio dapat dipercepat atau diperlambat tanpa mengubah *pitch* jika hanya magnitudonya yang diubah, sementara fasenya dibiarkan. Sebaliknya, kita dapat mengubah *pitch* tanpa mengubah kecepatan audio jika hanya fasenya yang diubah, sementara magnitudonya dibiarkan.

Audio yang diubah kecepatannya tanpa mengubah *pitch* perlu mempertahankan kaitan antartitik pada fase agar tetap sesuai dengan yang asli. Oleh sebab itu, prediksi titik yang kosong menjadi sangat penting untuk dapat mempertahankan konsistensi *pitch* asli. Prediksi ini dapat dicari dengan metode interpolasi. Semakin canggih algoritma interpolasi yang digunakan, akan semakin akurat dan presisi prediksi titiknya, dan akan semakin tidak berubah *pitch*-nya ketika diubah kecepatannya.

D. Interpolasi

Interpolasi adalah sebuah metode untuk memprediksi letak suatu titik di antara dua titik yang diketahui. Interpolasi sering digunakan untuk mengisi kebolongan data agar lebih halus. Terdapat beberapa jenis interpolasi, yaitu sebagai berikut.

D.1. Interpolasi Linier

Interpolasi Linier adalah metode interpolasi yang menghubungkan langsung dua titik yang diketahui secara linier. Interpolasi jenis ini merupakan jenis interpolasi yang paling sering digunakan karena kemudahan implementasinya. Persamaan interpolasi linier dijabarkan sebagai (3).

$$y = y_1 + \frac{(x - x_1)(y_2 - y_1)}{x_2 - x_1} \tag{3}$$

Dengan x_1, x_2, y_1, dan y_2 adalah titik-titik yang diketahui, x adalah absis dari ordinat yang ingin dicari, serta y adalah ordinat yang ingin dicari menggunakan interpolasi linier.

D.2 Interpolasi Polinomial

Interpolasi Polinomial adalah metode interpolasi yang menghubungkan dua titik dengan polinomial derajat tinggi. Interpolasi ini dilakukan dengan mencari polinomial yang melalui serangkaian titik data tertentu. Dalam konteks ini, interpolasi bertujuan untuk menemukan sebuah polinomial yang "melalui" titik-titik yang telah diberikan, yaitu untuk menentukan polinomial yang nilai-nilainya di titik tersebut sesuai dengan data yang tersedia. Bentuk umum interpolasi polinomial dijabarkan sebagai (4).

$$P(x) = a_n x^n + a_{n-1}x^{n-1} + \cdots + a_1 x + a_0 \tag{4}$$

Dengan $a_n, \ldots, a_0$ adalah koefisien-koefisien yang ditentukan berdasarkan titik-titik yang diketahui dan n derajat tertinggi polinom.

D.3 Interpolasi Spline

Interpolasi spline adalah metode interpolasi yang mendekati fungsi dengan menggunakan polinomial yang lebih rendah derajatnya, yang dikenal sebagai spline. Spline yang umum digunakan dalam interpolasi adalah spline kubik, yang membagi data menjadi segmen-segmen kecil, dan pada setiap segmen, fungsi interpolasinya adalah polinomial kubik. Tujuan utama dari interpolasi spline adalah untuk menghasilkan sebuah kurva yang halus, yang melewati semua titik data yang diberikan, tetapi tetap memiliki kekontinuan tinggi, terutama pada turunan pertama dan kedua.

Spline kubik biasanya digunakan untuk mengatasi masalah yang timbul pada interpolasi polinomial sederhana. Interpolasi polinomial dapat menghasilkan osilasi atau bahkan perilaku yang tidak diinginkan, terutama pada data yang memiliki banyak titik. Spline kubik menyelesaikan masalah ini dengan memberikan sebuah pendekatan yang lebih stabil, yaitu setiap segmen interpolasi tidak hanya bergantung pada dua titik data (seperti dalam interpolasi linier), tetapi pada empat titik data (dua titik sebelum dan dua titik setelah suatu titik

data).

Salah satu karakteristik penting dari spline kubik adalah kehalusan. Fungsi spline kubik tidak hanya meminimalkan kesalahan antara data dan fungsi interpolasi, tetapi juga memastikan bahwa fungsi dan turunan pertama serta kedua dari spline tersebut bersifat kontinu di seluruh segmen. Dengan cara ini, interpolasi spline menghasilkan kurva yang halus dan lebih cocok untuk aplikasi yang membutuhkan ketelitian tinggi.

D.4 Interpolasi Bilinier dan Bikubik

Interpolasi bilinier adalah metode interpolasi dua dimensi yang digunakan untuk memperkirakan nilai dari suatu titik yang terletak di dalam grid atau matriks data yang diketahui. Konsep dasar dari interpolasi bilinier adalah melakukan interpolasi linear secara berturut-turut terlebih dahulu dalam satu arah (misalnya, arah horizontal) dan kemudian dalam arah yang lainnya (misalnya, arah vertikal).

Sementara interpolasi bikubik adalah pengembangan dari interpolasi bilinier yang menggunakan polinomial kubik untuk memperkirakan nilai dalam dua dimensi. Dibandingkan dengan interpolasi bilinier yang hanya menggunakan informasi dari empat titik di sekitar titik yang akan diinterpolasi, interpolasi bikubik menggunakan informasi dari 16 titik sekitar untuk menghasilkan perkiraan yang lebih halus dan lebih akurat.

D.5 Interpolasi Fourier

Interpolasi Fourier adalah metode interpolasi yang menggunakan Transformasi Fourier untuk merepresentasikan data dalam bentuk deret atau transformasi frekuensi. Metode ini sangat efektif dalam menginterpolasi sinyal atau data yang periodik atau berbentuk gelombang. Konsep dasar dari interpolasi Fourier adalah mengubah data dari domain waktu ke domain frekuensi, kemudian mengubahnya kembali ke domain waktu setelah melakukan interpolasi pada domain frekuensi.

Pada interpolasi Fourier, sinyal atau data yang ingin diinterpolasi dianggap sebagai kombinasi dari sinyal sinusoidal (frekuensi-frekuensi tertentu). Dengan menggunakan Transformasi Fourier, kita dapat mendekomposisi sinyal menjadi deret Fourier yang terdiri dari komponen-komponen frekuensi tertentu. Setelah deret Fourier diperoleh, interpolasi dilakukan dengan memanipulasi koefisien-koefisien Fourier tersebut. Hal ini memungkinkan kita untuk memperbesar atau memperkecil frekuensi dalam data tanpa menghasilkan distorsi yang signifikan.

Keunggulan dari interpolasi Fourier adalah kemampuannya untuk menangani data dengan sifat periodik atau data yang mengandung frekuensi tinggi, serta untuk menghasilkan hasil yang halus tanpa adanya distorsi atau hal lain yang tidak diinginkan, seperti yang sering terjadi pada interpolasi polinomial.

III. IMPLEMENTASI

A. Tujuan Implementasi dan Pemilihan Bahasa Pemrograman

Implementasi ini bertujuan untuk menunjukkan bahwa audio dapat diubah kecepatannya tanpa mengubah *pitch*-nya menggunakan *Short-Time Fourier Transform* dan interpolasi. Hasil dari implementasi ini hanya akan memberikan gambaran, tanpa benar-benar tidak mengubah *pitch* sama sekali. Hal ini disebabkan karena terbatasnya algoritma interpolasi dan pengetahuan penulis untuk memprediksi titik-titik pada fase setelah diregangkan atau dimampatkan.

Pemilihan bahasa Python didasari oleh lengkapnya library yang dimiliki sesuai dengan kebutuhan pada penelitian ini. Library yang digunakan pada penelitian ini adalah Numpy, Librosa, dan Soundfile. Numpy digunakan untuk melakukan interpolasi dan operasi matematika umum lainnya, Librosa digunakan untuk melakukan *Short-Time Fourier Transform*, dan SoundFile digunakan untuk mengonversi hasil olahan audio menjadi file audio, dalam kasus ini dalam format .wav.

B. Memuat File Audio yang Akan Dieksekusi

File Audio disimpan pada variabel “audio” menggunakan fungsi “load” dari library Librosa, dengan parameter input berupa nama file (variabel “input_file”).

```
# Memuat file audio
audio, sr = librosa.load(input_file, sr=None)
```

Gambar 3. Memuat file audio sesuai dengan alamat value dari variabel “input_file”

C. Mentransformasi Audio dari Domain Waktu ke Domain Frekuensi-Waktu

Untuk dapat mengubah kecepatan audio tanpa mengubah *pitch*, kita perlu mengubah magnitudo dari audio dalam domain frekuensi-waktu. Untuk bisa melakukannya, kita perlu mentransformasi audio dari yang awalnya pada domain waktu menjadi domain frekuensi-waktu. Hal ini dapat dilakukan dengan menggunakan metode *Short-Time Fourier Transform*. Pada implementasi ini, pengimplementasian metode *Short-Time Fourier Transform* dilakukan menggunakan library Librosa, tepatnya menggunakan fungsi “stft” untuk mengubah file audio menjadi spektogram. Kemudian, spektogram dipisah menjadi magnitudo dan fase menggunakan fungsi “magphase”.

```
# Melakukan STFT untuk mendapatkan spektrogram
stft = librosa.stft(audio)

# Memisah spektogram menjadi magnitudo dan fase
magnitude, phase = librosa.magphase(stft)
```

Gambar 4. Implementasi STFT dan pemisahan spektogram menjadi magnitudo dan fase

D. Meregangkan atau Memampatkan Magnitudo

Perubahan kecepatan audio dilakukan pada tahap ini, yaitu meregangkan atau memampatkan magnitudo. Implementasi peregangan atau pemampatan magnitudo dilakukan dengan membuat sebuah array kosong baru yang nantinya diisi dengan magnitudo yang sudah diregangkan atau dimampatkan, lengkap dengan hasil interpolasinya.

```
# Jumlah frame waktu (kolom)
n_frames = magnitude.shape[1]

# Jumlah frame target setelah perlambatan
target_frames = int(n_frames * (1 / speed_factor))

# Membuat array untuk magnitude yang telah diperpanjang
magnitude_stretched = np.zeros((magnitude.shape[0], target_frames))
```

Gambar 5. Implementasi persiapan wadah kosong untuk diisi dengan magnitude yang telah diregangkan atau dimampatkan

Berikutnya, magnitudo yang diregangkan atau dimampatkan dimasukkan ke dalam array kosong tersebut. Peregangan memanfaatkan interpolasi pada titik-titik yang kosong.

```
# Melakukan interpolasi per kolom (untuk setiap frekuensi)
for i in range(magnitude.shape[0]):  # Loop untuk setiap frekuensi
    magnitude_stretched[i, :] = np.interp(
                                    np.linspace(0, n_frames, target_frames),
                                    np.arange(n_frames),
                                    magnitude[i, :]
                                )
```

Gambar 6. Implementasi memasukkan magnitudo yang telah diregangkan dan diisi hasil interpolasi ke dalam array kosong

E. Menyesuaikan Ukuran Fase dengan Ukuran Magnitudo

Tahap ini merupakan salah satu tahap yang menentukan seberapa tidak berubahnya *pitch* dari audio yang diregangkan atau dimampatkan. Tepatnya, terletak pada algoritma interpolasi pada saat meregangkan. Pada implementasi ini, digunakan algoritma interpolasi bawaan dari numpy, yaitu interpolasi linier pada fungsi "interp" seperti pada magnitudo.

```
# Lakukan interpolasi untuk phase agar sesuai dengan dimensi baru
phase_stretched = np.zeros_like(magnitude_stretched)  # Array fase kosong baru
for i in range(phase.shape[0]):  # Loop untuk setiap frekuensi
    phase_stretched[i, :] = np.interp(
                                np.linspace(0, n_frames, target_frames),
                                np.arange(n_frames),
                                phase[i, :]
                            )
```

Gambar 7. Implementasi penyesuaian fase dengan dimensi baru

F. Penyatuan Magnitudo dengan Fase, Invers Short-Time Fourier Transform, dan Penulisan File Audio

Pengolahan pada domain frekuensi-waktu telah selesai. Langkah-langkah terakhir yang masih perlu dilakukan adalah mengembalikan magnitudo dan fase menjadi satu kesatuan spektogram menggunakan (2), lalu mengubah spektogram kembali menjadi file audio (domain waktu) menggunakan invers dari *Short-Time Fourier Transform* dengan fungsi "istft" dari library Librosa, dan diakhiri dengan menulis file audio ke sebuah direktori file output menggunakan fungsi "write" dari library SoundFile.

```
# Menggabungkan magnitude dengan fase
stretched_stft = magnitude_stretched * np.exp(1j * phase_stretched)

# Menggunakan Invers STFT untuk mengembalikan ke domain waktu
audio_slow = librosa.istft(stretched_stft)

# Menyimpan audio yang sudah diolah ke file output
sf.write(output_file, audio_slow, sr)
print(f"Audio berhasil disimpan ke {output_file}")
```

Gambar 8. Implementasi penggabungan magnitudo dengan fase, invers STFT, dan penulisan file audio

IV. KESIMPULAN

Berdasarkan penelitian yang telah dilakukan, metode *Short-Time Fourier Transform* (STFT) dan interpolasi terbukti dapat digunakan untuk mempercepat atau memperlambat audio tanpa mengubah *pitch*-nya. Implementasi yang dilakukan memungkinkan manipulasi sinyal audio dengan tetap mempertahankan karakteristik frekuensi aslinya, sehingga menghasilkan output yang lebih alami dan tidak terdistorsi.

Dalam implementasi ini, penggunaan interpolasi linier menunjukkan bahwa metode sederhana dapat memberikan hasil yang memadai untuk tugas-tugas dasar, seperti peregangan atau pemampatan magnitudo pada domain frekuensi-waktu. Interpolasi linier efektif dalam mengisi titik-titik kosong yang muncul akibat perubahan skala waktu. Namun, interpolasi linier memiliki keterbatasan dalam menangani pola perubahan nonlinier atau kompleks. Oleh karena itu, penggunaan algoritma interpolasi yang lebih canggih, seperti interpolasi spline atau interpolasi Fourier, dapat meningkatkan akurasi dan kualitas hasil akhir.

Proses penyesuaian fase menjadi salah satu hal yang paling berpengaruh dalam menjaga *pitch* agar tidak berubah. Dengan mempertahankan hubungan antartitik pada fase, perubahan kecepatan audio tidak mengakibatkan pergeseran frekuensi, sehingga *pitch* tetap konsisten. Namun, keberhasilan metode ini sangat bergantung pada algoritma interpolasi yang digunakan, yang masih dapat ditingkatkan.

Penelitian ini juga menunjukkan bahwa pemilihan algoritma yang sesuai dalam setiap tahap pada implementasi. Meskipun hasil implementasi ini belum sepenuhnya optimal, implementasi yang dilakukan memberikan gambaran yang cukup untuk bereksplorasi dalam pemrosesan audio. Potensi integrasi implementasi ini mencakup berbagai bidang, mulai dari perangkat lunak untuk mengedit video, sampai dalam platform streaming sebagai salah satu fitur.

Secara keseluruhan, penelitian ini tidak hanya membuktikan bahwa kecepatan audio dapat diubah tanpa mengubah *pitch*-nya, tetapi juga membuka peluang untuk pengembangan lebih lanjut di masa depan. Dengan peningkatan teknologi dan pengetahuan, hasil pengolahan

audio akan lebih akurat dan mendekati kualitas suara asli. Dengan demikian, industri kreatif dan ilmiah pun juga akan berkembang.

V. SARAN

Dari penelitian ini, ditemukan bahwa metode *Short-Time Fourier Transform* dan interpolasi dapat digunakan dalam mengubah kecepatan audio tanpa mengubah *pitch*. Namun, hasil dari implementasinya masih kurang maksimal dan merupakan pendekatan kasar saja. Oleh sebab itu, untuk dapat meningkatkan kualitas hasil implementasi, diberikan saran sebagai berikut.

A. Peningkatan Akurasi Interpolasi

Penelitian ini menggunakan algoritma interpolasi yang sederhana. Disarankan untuk menggunakan algoritma interpolasi yang lebih kompleks seperti interpolasi spline atau interpolasi Fourier, agar prediksi titik dapat lebih akurat dan presisi.

B. Penggunaan Algoritma Transformasi dan Invers yang Lebih Canggih

Penelitian ini menggunakan implementasi yang mengubah kualitas dan sedikit *pitch* dari audio bahkan ketika tidak dilakukan pengubahan kecepatan audio. Agar kualitas audio tetap terjaga dan *pitch* tetap konsisten, perlu digunakan algoritma yang lebih canggih dalam proses transformasi dan juga invers.

C. Integrasi dengan Pemrosesan Real-Time

Implementasi penelitian ini dilakukan secara luring. Akan sangat menarik dan membantu apabila implementasi ini diintegrasikan pada sesuatu seperti perangkat lunak untuk mengedit video. Integrasi akan lebih optimal jika implementasi dapat digunakan secara real-time pada saat proses mengedit video.

REFERENSI

[1] Atkinson, K. E. (2008). *An Introduction to Numerical Analysis* (2nd ed.). John Wiley & Sons.

[2] Berg, R. E., & Stork, D. G. (2005). *The Physics of Sound* (2nd ed.). Pearson Education.

[3] Burden, R. L., & Faires, J. D. (2010). *Numerical Analysis* (9th ed.). Brooks/Cole.

[4] Fletcher, N. H., & Rossing, T. D. (2010). *The Physics of Musical Instruments*. Springer.

[5] Gerald, C. F., & Wheatley, P. O. (2004). *Applied Numerical Analysis* (7th ed.). Addison Wesley.

[6] Lathi, B. P. (2010). *Modern Digital and Analog Communication Systems*. Oxford University Press.

[7] Mitra, S. K. (2001). *Digital Signal Processing: A Computer-Based Approach*. McGraw-Hill.

[8] Oppenheim, A. V., & Schafer, R. W. (2009). *Discrete-Time Signal Processing*. Pearson.

[9] Pierce, A. D. (1989). *Acoustics: An Introduction to Its Physical Principles and Applications*. Acoustical Society of America.

[10] Pierce, J. R. (1992). *The Science of Musical Sound*. Freeman.

[11] Press, W. H., Teukolsky, S. A., Vetterling, W. T., & Flannery, B. P. (1992). *Numerical Recipes in C: The Art of Scientific Computing*. Cambridge University Press.

[12] Smith, S. W. (1997). *The Scientist and Engineer's Guide to Digital Signal Processing*. California Technical Publishing.

PERNYATAAN

Dengan ini saya menyatakan bahwa makalah yang saya tulis ini adalah tulisan saya sendiri, bukan saduran, atau terjemahan dari makalah orang lain, dan bukan plagiasi.

Bandung, 2 Januari 2025

Muhammad Izzat Jundy (13523092)