Pengembangan Aplikasi Desktop untuk Filtering Sinyal Audio Digital: Studi Kasus Filter Butterworth

*Note: Sub-titles are not captured in Xplore and should not be used

Ahmad Mukhlis Farhan
Informatics Engineering
University Of Darussalam Gontor
Ponorogo, Indonesia
ahmadmukhlisfarhan93@student.cs.unida.gontor.ac.id

Raffa Arvel Nafi'Nadindra
Informatic Engineering
University Of Darussalam Gontor
Ponorogo, Indonesia
raffaarvel@gmail.com

Azhar Zuhro
Informatic Engineering
University Of Darussalam Gontor
Ponorogo, Indonesia
azharzuhro74@gmail.com

Fitroh Dian Nugroho
Informatic Engineering
University Of Darussalam Gontor
Ponorogo, Indonesia
ito.desu12@gmail.com

Abstract*—Dokumen ini menyajikan laporan proyek mengenai pengembangan aplikasi desktop sederhana menggunakan Python yang mampu memproses sinyal audio dengan menerapkan filter digital. Aplikasi ini memungkinkan pengguna untuk mengunggah file audio dalam format WAV, memilih jenis filter (low-pass, high-pass, atau band-pass), dan memvisualisasikan hasilnya dalam domain waktu dan frekuensi. Implementasi proyek memanfaatkan pustaka SciPy untuk fungsi filtering, NumPy untuk komputasi numerik, Matplotlib untuk visualisasi, dan Tkinter untuk membangun antarmuka pengguna grafis (GUI). Hasil dari proyek ini menunjukkan efektivitas filter digital dalam membersihkan noise atau mengisolasi komponen frekuensi tertentu dari sinyal audio, yang diverifikasi melalui analisis spektrum frekuensi.

***Keywords*—filtering audio, low-pass, high-pass, band-pass, Transformasi Fourier Cepat (FFT), Python, Tkinter**

I. *Pendahuluan*

Pengolahan Sinyal Digital (PSD) adalah bidang ilmu teknik yang fundamental dengan berbagai aplikasi, mulai dari telekomunikasi hingga pemrosesan audio. Dalam konteks audio, **filtering** merupakan salah satu teknik PSD yang paling penting, digunakan untuk memodifikasi sinyal dengan menghapus atau menonjolkan rentang frekuensi tertentu. Aplikasi filtering audio meliputi denoising (penghilangan kebisingan), ekualisasi, dan efek suara. Proyek ini bertujuan untuk menyediakan demonstrasi praktis dari konsep filtering digital melalui sebuah aplikasi interaktif.

A. Latar Belakang

Analisis dan manipulasi sinyal audio digital telah menjadi bagian integral dari banyak teknologi modern. Memahami bagaimana komponen frekuensi dalam sinyal audio dapat diisolasi atau dihilangkan adalah keterampilan penting dalam PSD. Filter digital, khususnya low-pass, high-pass, dan band-pass, merupakan alat utama untuk mencapai tujuan ini. Proyek ini akan memvisualisasikan efek dari filter ini pada sinyal audio, memungkinkan pengguna untuk secara langsung mengamati perubahan di domain waktu dan frekuensi.

B. Tujuan Proyek

Tujuan utama dari proyek ini adalah:

1. Mengimplementasikan filter digital jenis **Butterworth** (low-pass, high-pass, dan band-pass) menggunakan pustaka SciPy.
2. Membangun Antarmuka Pengguna Grafis (GUI) berbasis **Tkinter** yang intuitif untuk interaksi pengguna, termasuk unggah data audio, pemilihan jenis filter, dan pengaturan frekuensi cutoff.
3. Memvisualisasikan sinyal audio dalam domain waktu dan spektrum frekuensi (menggunakan FFT) **sebelum dan sesudah** proses filtering.
4. Menganalisis dan mendiskusikan dampak filter yang diterapkan terhadap karakteristik frekuensi sinyal audio.

II. Metode dan Algoritma

A. Akuisisi dan Representasi Data Audio

Data yang digunakan dalam proyek ini adalah rekaman audio digital dalam format `.wav`. Sinyal ini direpresentasikan sebagai deret waktu diskrit x[n], di mana n adalah indeks sampel dan x adalah amplitudo pada waktu tersebut. Proses akuisisi dan pembacaan file audio ditangani oleh pustaka `soundfile`, yang memuat data sebagai array NumPy dan mengidentifikasi *sampling rate* (f_s).

B. Transformasi Fourier Cepat (FFT)

Untuk menganalisis komposisi frekuensi sinyal audio, **Transformasi Fourier Cepat (FFT)** digunakan. FFT adalah algoritma yang efisien untuk menghitung Transformasi Fourier Diskrit (DFT), yang mengonversi sinyal dari domain waktu ke domain frekuensi. Magnitudo dari hasil FFT (`np.abs(fft_result)`) menunjukkan seberapa kuat setiap komponen frekuensi dalam sinyal. Frekuensi yang sesuai dengan setiap bin FFT dihitung menggunakan `np.fft.fftfreq`.

C. *Desain dan Implementasi Filter Digital*

Filter yang dipilih untuk implementasi adalah **filter Butterworth**. Filter ini dikenal karena respons frekuensinya yang sangat datar di *passband* dan *stopband*, yang meminimalkan distorsi. Pustaka `scipy.signal` menyediakan fungsi-fungsi esensial untuk desain dan aplikasi filter.

- **Desain Filter:** Fungsi `scipy.signal.butter(order, Wn, btype, analog=False)` digunakan untuk mendapatkan koefisien numerator (b) dan denominator (a) dari filter.
    - `order`: Orde filter (misalnya, 5).
    - `Wn`: Frekuensi kritis yang dinormalisasi (cutoff atau band-edge) relatif terhadap frekuensi Nyquist (f_s/2).
    - `btype`: Jenis filter (`low`, `high`, atau `band`).
- **Aplikasi Filter:** Koefisien ini kemudian digunakan oleh `scipy.signal.lfilter(b, a, data)` untuk menerapkan filter pada sinyal audio.

Jenis Filter yang Diimplementasikan:

- **Low-Pass Filter:** Melewatkan frekuensi di bawah frekuensi cutoff dan secara signifikan melemahkan frekuensi di atasnya.
- **High-Pass Filter:** Melewatkan frekuensi di atas frekuensi cutoff dan melemahkan frekuensi di bawahnya.
- **Band-Pass Filter:** Melewatkan frekuensi dalam rentang tertentu antara dua frekuensi cutoff dan melemahkan frekuensi di luar rentang tersebut.

D. *Antarmuka Pengguna Grafis (GUI) dengan Tkinter*

GUI dibangun menggunakan pustaka standar Python, **Tkinter**, yang memungkinkan pembuatan aplikasi desktop. Desain GUI berfokus pada kemudahan penggunaan, dengan elemen-elemen sebagai berikut:

- **Tombol "Unggah File Audio"**: Membuka dialog file untuk memilih file audio `.wav`.
- **Menu Drop-down "Pilih Filter"**: Memungkinkan pengguna untuk memilih antara low-pass, high-pass, atau band-pass.
- **Kolom Input "Frekuensi Cutoff (Hz)"**: Mengizinkan pengguna untuk memasukkan nilai frekuensi cutoff. Ketika `band-pass` dipilih, kedua kolom (`Frekuensi Cutoff Bawah` dan `Frekuensi Cutoff Atas`) akan muncul. Untuk filter lain, hanya satu kolom yang ditampilkan.
- **Tombol "Terapkan Filter"**: Memicu proses filtering dan pembaruan visualisasi.
- **Area Plot Matplotlib**: Terintegrasi ke dalam GUI menggunakan `FigureCanvasTkAgg`, menampilkan plot domain waktu dan frekuensi.

III. Implementasi

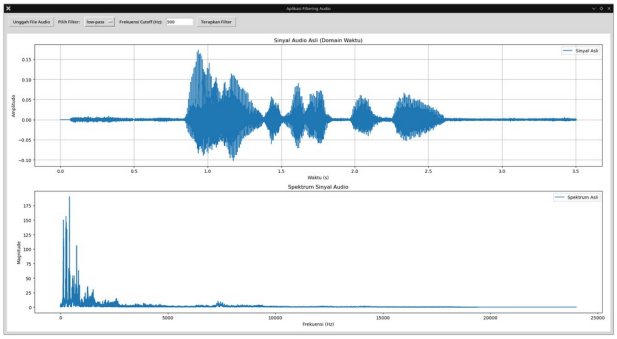
Pemrosesan sinyal adalah inti dari aplikasi ini, diimplementasikan menggunakan pustaka **SciPy** dan **NumPy**. Fungsi `butter_filter` adalah komponen kunci yang menerapkan filter Butterworth. Fungsi ini mampu menerima **nilai tunggal** (`float`) untuk filter `low-pass` dan `high-pass` atau **list/array** (`[frekuensi_bawah, frekuensi_atas]`) untuk filter `band-pass`. Fleksibilitas ini memastikan filter bekerja efektif tanpa kesalahan tipe data. **NumPy** digunakan untuk komputasi numerik, seperti mengubah data audio ke array yang dapat diproses dan melakukan Transformasi Fourier Cepat (FFT) untuk analisis spektrum.

Kode sumber lengkap dan yang telah diuji dapat ditemukan di repositori GitHub berikut untuk referensi dan kolaborasi lebih lanjut:

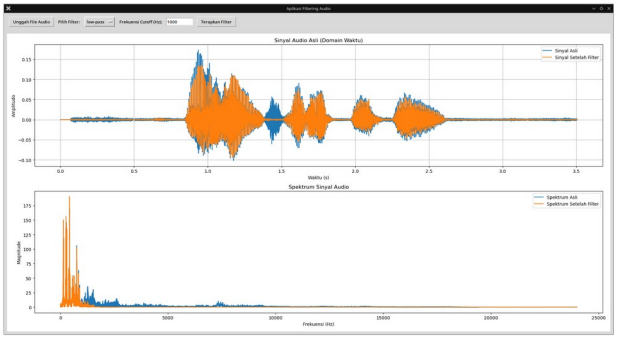
https://github.com/mukhlisfarhan/Audio-Filtering-Analysis.git

IV. Visualisasi Hasil

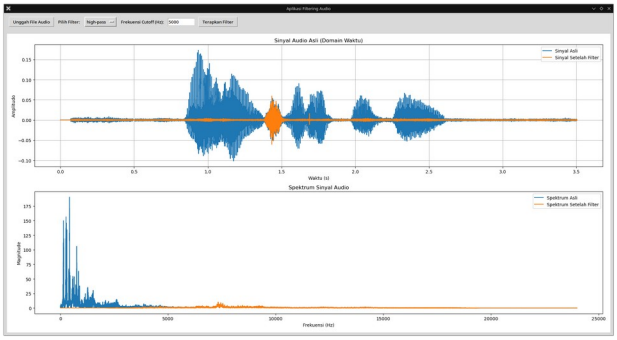
Bagian ini menampilkan tangkapan layar (screenshot) dari aplikasi yang sedang berjalan dan hasil visualisasi filtering.



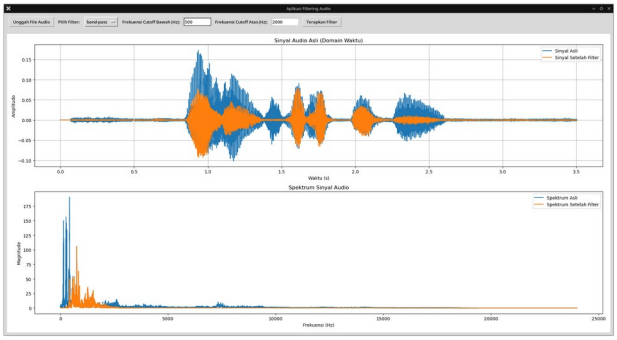
Gambar di atas menunjukkan antarmuka pengguna aplikasi dengan tombol unggah, pilihan filter, input frekuensi cutoff, dan area plot untuk menampilkan sinyal audio.



Pada gambar di atas, plot domain waktu (atas) menunjukkan sinyal audio asli dan yang sudah difilter low-pass. Perubahan yang lebih signifikan terlihat pada plot domain frekuensi (bawah), di mana komponen frekuensi di atas 1000 Hz secara drastis dilemahkan oleh filter.



Gambar ini mengilustrasikan efek high-pass filter. Frekuensi di bawah 5000 Hz dilemahkan, menonjolkan komponen frekuensi tinggi.



Gambar ini menunjukkan band-pass filter yang hanya mempertahankan frekuensi antara 500 Hz dan 2000 Hz, melemahkan frekuensi di luar rentang tersebut.

V. Analisis dan Diskusi

Dari hasil visualisasi yang ditampilkan, dapat disimpulkan bahwa aplikasi filtering audio ini berfungsi dengan baik dalam menerapkan filter digital pada sinyal audio.

- **Low-Pass Filter:** Efektif dalam menghilangkan noise frekuensi tinggi atau komponen suara yang tajam, menghasilkan sinyal yang terdengar lebih halus. Ini tercermin dari spektrum frekuensi yang menunjukkan pelemahan signifikan pada frekuensi di atas titik cutoff.
- **High-Pass Filter:** Berguna untuk menghilangkan *hum* frekuensi rendah atau komponen suara dasar yang tidak diinginkan, menonjolkan aspek suara yang lebih "treble" atau detail. Spektrum frekuensi menunjukkan pelemahan pada frekuensi di bawah titik cutoff.
- **Band-Pass Filter:** Ideal untuk mengisolasi pita frekuensi tertentu, misalnya untuk menonjolkan vokal atau instrumen tertentu dalam sebuah rekaman. Spektrum frekuensi menunjukkan bahwa hanya komponen dalam rentang frekuensi yang ditentukan yang dipertahankan.

Pemilihan **orde filter** (misalnya, `order=5` dalam implementasi) memengaruhi ketajaman transisi antara *passband* dan *stopband*. Orde yang lebih tinggi menghasilkan transisi yang lebih curam tetapi dapat meningkatkan kompleksitas komputasi dan memperkenalkan *ringing artifacts*. **Frekuensi cutoff** adalah parameter krusial yang harus dipilih dengan hati-hati untuk menghindari hilangnya informasi penting atau distorsi yang tidak diinginkan.

VI. Kesimpulan dan Potensi Pengembangan

Proyek ini telah berhasil mengimplementasikan sebuah aplikasi filtering audio berbasis Python dengan GUI Tkinter yang fungsional. Aplikasi ini secara efektif mendemonstrasikan prinsip-prinsip dasar filter digital dan dampaknya terhadap sinyal audio di domain waktu dan frekuensi. Integrasi pustaka seperti SciPy, NumPy, dan Matplotlib memungkinkan pemrosesan sinyal yang kuat dan visualisasi yang jelas.

Potensi Pengembangan Lebih Lanjut:

1. **Fitur Denoising Lanjut:** Mengintegrasikan algoritma denoising yang lebih canggih, seperti filter adaptif atau metode berbasis *spectral subtraction*.
2. **Filter Real-time:** Menambahkan kemampuan untuk menangkap audio dari mikrofon secara *real-time* dan menerapkan filter secara langsung.
3. **Tipe Filter Tambahan:** Mengimplementasikan jenis filter lain seperti *notch filter* (untuk menghilangkan frekuensi tunggal yang sangat spesifik, misalnya *hum* listrik 50/60 Hz) atau *equalizer* grafis.
4. **Kontrol Audio:** Menambahkan fungsionalitas untuk memutar sinyal audio asli dan yang sudah difilter langsung dari aplikasi.
5. **Ekspor Hasil:** Menyediakan opsi untuk menyimpan sinyal audio yang telah difilter ke dalam file baru.
6. **Antarmuka Lebih Lanjut:** Mengembangkan GUI yang lebih interaktif, misalnya dengan *slider* untuk kontrol frekuensi cutoff secara dinamis atau tampilan spektrogram.

VII. Tautan Video Presentasi

https://youtu.be/nWdsMCBLE2Q

References

[1] Oppenheim, A. V., & Schafer, R. W. (2009). Discrete-time signal processing. Pearson Education.

[2] Lyons, R. G. (2011). Understanding digital signal processing. Prentice Hall.

[3] SciPy Project. (n.d.). SciPy Reference Guide. Retrieved from https://docs.scipy.org/doc/scipy/reference/index.html

[4] Hunter, J. D. (2007). Matplotlib: A 2D graphics environment. Computing in Science & Engineering, 9(3), 90-95.

[5] Oliphant, T. E. (2006). A guide to NumPy. Trelgol Publishing.

[6] Python Software Foundation. (n.d.). Tkinter. Retrieved from https://docs.python.org/3/library/tkinter.html