

Exercise - Audio Processing

IF4021 - Multimedia Information Processing

Nama : Nashwa Putri Laisya

NIM : 122140180

GitHub Repository: <https://github.com/nashwals/IF4021-Multimedia>

In [85]:

```
# Import Library
import numpy as np
import matplotlib
import matplotlib.pyplot as plt
import librosa
import soundfile as sf
from IPython.display import Audio, HTML, display
from scipy.signal import butter, filtfilt
import pyloudnorm as pyln
import os

# Set matplotlib untuk menampilkan plot inline
%matplotlib inline

# Tampilkan versi library untuk dokumentasi
print("Library versions:")
print(f"NumPy: {np.__version__}")
print(f"Matplotlib: {matplotlib.__version__}")
print(f"Librosa: {librosa.__version__}")
```

Library versions:

NumPy: 2.2.6

Matplotlib: 3.10.5

Librosa: 0.11.0

Soal 1: Rekaman dan Analisis Suara Multi-Level

In [86]:

```
# Mengatur path audio dengan os agar bisa dijalankan di mana saja
path_audio = os.path.join(os.getcwd(), 'data', 'soal-1.wav')

# Load file audio
y, sr = librosa.load(path_audio)

# Menampilkan sample rate
print (f"Sample Rate: {sr} Hz")

# Menghitung durasi
duration = librosa.get_duration(y=y, sr=sr)
# Menampilkan durasi
print(f"Durasi: {duration:.2f} detik")
```

```

# Menampilkan jumlah channel
print(f"Jumlah kanal: {'1 -> Mono' if y.ndim == 1 else '2 -> Stereo'")

# Menampilkan jumlah total sampel
print(f"Total Sampel: {len(y)}")

# Menampilkan audio player
display(Audio(y, rate=sr))

```

Sample Rate: 22050 Hz
 Durasi: 25.02 detik
 Jumlah kanal: 1 -> Mono
 Total Sampel: 551750

```

/var/folders/ds/tzk88yl5705g6nytd56kbr10000gn/T/ipykernel_11858/432
042548.py:5: UserWarning: PySoundFile failed. Trying audioread inste
ad.
y, sr = librosa.load(path_audio)
/Users/nashwalaisya/multimedia-uv/lib/python3.13/site-packages/libro
sa/core/audio.py:184: FutureWarning: librosa.core.audio.__audioread_
load
    Deprecated as of librosa version 0.10.0.
    It will be removed in librosa version 1.0.
y, sr_native = __audioread_load(path, offset, duration, dtype)

```

0:00 -0:25

1.1 Waveform Audio

In [87]:

```

# Menampilkan waveform dari file audio

# Membuat array waktu untuk sumbu x dari detik 0 sampai durasi audio
time = np.linspace(0, duration, len(y))

# Menentukan ukuran figure
plt.figure(figsize=(12, 4))

# Membuat plot garis
plt.plot(time, y, color='blue', alpha=0.7)

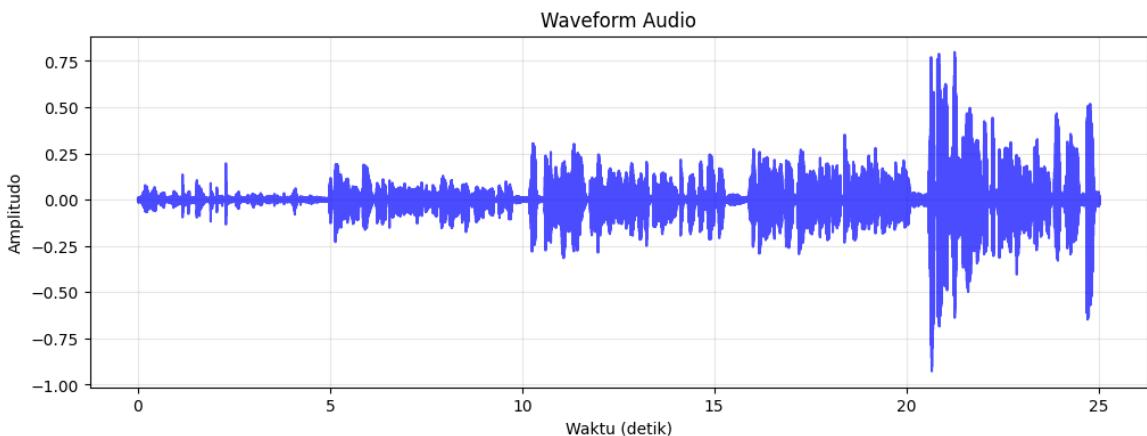
# Menambahkan judul
plt.title('Waveform Audio')

# Menambahkan label sumbu x
plt.xlabel('Waktu (detik)')

# Menambahkan label sumbu y
plt.ylabel('Amplitudo')

# Menambahkan grid
plt.grid(True, alpha=0.3)

```



Penjelasan:

Dari visualisasi dapat dilihat bahwa di tiap 5 detik suara menghasilkan amplitudo yang berbeda-beda. 5 detik pertama adalah suara berbisik, visualisasi menggambarkan amplitudo yang kecil, artinya suara pelan. 5 detik selanjutnya (detik 5 s.d 10) amplitudo lebih besar dari sebelumnya, artinya suara lebih keras dari berbisik alias normal. Mulai detik 10, amplitudo lebih besar dibandingkan 5 detik sebelumnya, artinya suara lebih keras lagi dari suara normal. Di detik 15, amplitudo hampir sama dengan 5 detik sebelumnya tetapi lebih rapat yang berarti suara hampir sama seperti sebelumnya. Dan mulai detik 20 sampai 25, amplitudo sangat besar yang menandakan suara yang dihasilkan sangat keras.

Kesimpulannya, waveform yang ditampilkan merepresentasikan keras-pelannya audio. Sesuai dengan suara yang direkam, mulai dari berbisik, suara normal, keras, cempreng, hingga sangat keras, dapat dilihat dengan waveform.

1.2 Spectrogram Audio

In [88]:

```
# Menghitung STFT
D = librosa.stft(y, n_fft=1024, hop_length=256)

# Mengubah ke skala log-dB (amplitudo ke dB)
DB = librosa.amplitude_to_db(abs(D), ref=np.max)

# Menentukan ukuran figure
plt.figure(figsize=(12, 6))

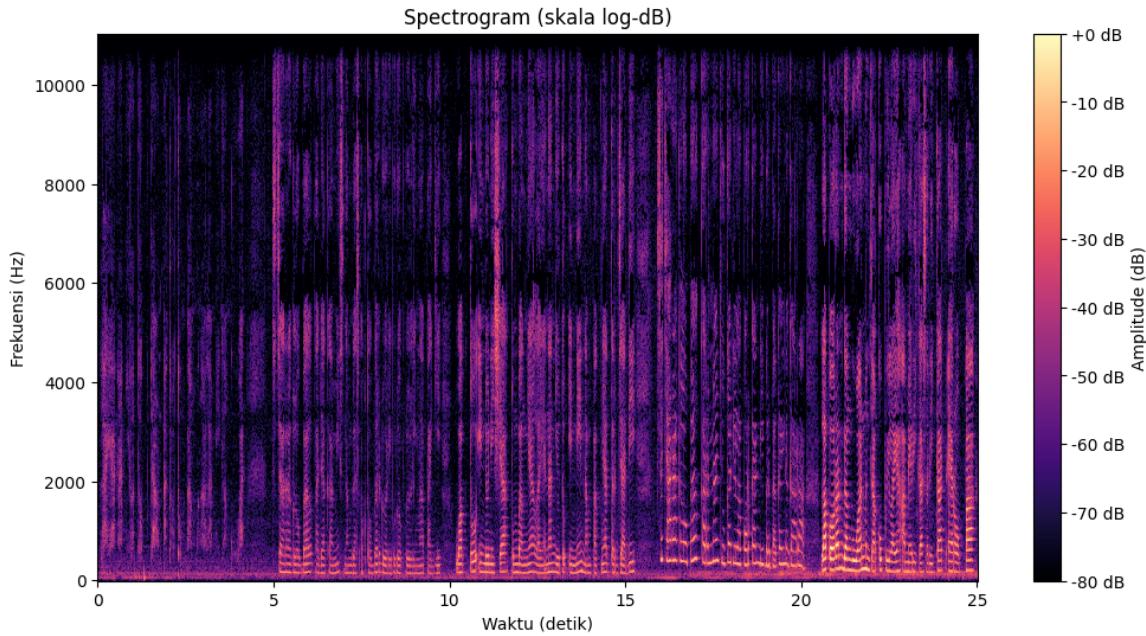
# Menampilkan spectrogram dalam skala log-dB
img = librosa.display.specshow(DB, sr=sr, hop_length=256, x_axis='t'

# Menambahkan colorbar
plt.colorbar(img, format='%+2.0f dB', label='Amplitude (dB)')

# Menambahkan label dan judul
plt.title('Spectrogram (skala log-dB)')
plt.xlabel('Waktu (detik)')
```

```
plt.ylabel('Frekuensi (Hz)')
```

Out[88]: Text(0, 0.5, 'Frekuensi (Hz)')



Penjelasan:

Dari spectrogram dapat dilihat bahwa di tiap 5 detik suara menghasilkan frekuensi yang berbeda-beda. Mulai dari detik 0 sampai detik 5, frekuensi konsisten dengan energi yang tidak begitu besar. Mulai dari detik 5 sampai detik 10, energi/intensitas lebih tinggi dari sebelumnya dan intensitas yang konsisten tinggi berada di frekuensi rendah sekitar 0-500 Hz. Kemudian di detik 10 sampai detik 15, pada frekuensi yang sama, yaitu sekitar 0-500 Hz (rendah), intensitas yang dihasilkan lebih tinggi dari sebelumnya (warna lebih cerah). Dari detik 15 sampai detik 20, intensitas yang tinggi berada pada frekuensi yang lebih tinggi dari sebelumnya. Dan di detik 20 sampai 25, frekuensi lebih tinggi dan intensitas yang tinggi.

Kesimpulannya, spectrogram yang ditampilkan merepresentasikan frekuensi yang dihasilkan audio. Sesuai dengan suara yang direkam, mulai dari berbisik, suara normal, keras, cempreng, hingga sangat keras, spectrogram memperlihatkan bagaimana perubahan frekuensinya.

1.3 Resampling Audio

In [89]: # Resampling audio ke 8000 Hz

```
y_resampled = librosa.resample(y, orig_sr=sr, target_sr=8000)  
sr_resampled = 8000
```

```
print(f"Sample Rate setelah Resampling: {sr_resampled} Hz")  
print(f"Total Sampel setelah Resampling: {len(y_resampled)}")
```

```
# Menampilkan durasi setelah resampling
```

```
duration_resampled = librosa.get_duration(y=y_resampled, sr=sr_resampled)
```

```

print(f"Durasi setelah Resampling: {duration_resampled:.2f} detik")

sf.write('data/soal-1-resampled1.wav', y_resampled, sr_resampled)

# Menampilkan audio player untuk audio yang sudah di-resample
display(Audio(y_resampled, rate=sr_resampled))

```

Sample Rate setelah Resampling: 8000 Hz
 Total Sampel setelah Resampling: 200182
 Durasi setelah Resampling: 25.02 detik

0:00 -0:25

In [90]:

```

# Resampling audio ke 32000 Hz
y_resampled_2 = librosa.resample(y, orig_sr=sr, target_sr=32000)
sr_resampled_2 = 32000

print(f"Sample Rate setelah Resampling: {sr_resampled_2} Hz")
print(f"Total Sampel setelah Resampling: {len(y_resampled_2)}")

# Menampilkan durasi setelah resampling
duration_resampled_2 = librosa.get_duration(y=y_resampled_2, sr=sr_
print(f"Durasi setelah Resampling: {duration_resampled_2:.2f} detik

sf.write('data/soal-1-resampled2.wav', y_resampled_2, sr_resampled_2)

# Menampilkan audio player untuk audio yang sudah di-resample
display(Audio(y_resampled_2, rate=sr_resampled_2))

```

Sample Rate setelah Resampling: 32000 Hz
 Total Sampel setelah Resampling: 800726
 Durasi setelah Resampling: 25.02 detik

0:00 -0:25

Perbandingan

Setelah re-sampling ke sample rate yang lebih rendah (8000 Hz) maupun ke sample rate yang lebih tinggi (32000 Hz), durasi audio tidak berubah, tetapi kualitas suara yang dihasilkan berbeda. Suara yang dihasilkan dengan resampling ke 8000 Hz menjadi kurang jelas/jernih sedangkan kualitas suara yang dihasilkan dengan resampling ke 32000 Hz menjadi lebih jelas/jernih didengar.

Soal 2: Noise Reduction dengan Filtering

In [91]:

```

# Mengatur path audio dengan os agar bisa dijalankan di mana saja
path_audio = os.path.join(os.getcwd(), 'data', 'soal-2.wav')

# Load file audio

```

```

y, sr = librosa.load(path_audio)

# Menampilkan sample rate
print(f"Sample Rate: {sr} Hz")

# Menghitung durasi
duration = librosa.get_duration(y=y, sr=sr)
# Menampilkan durasi
print(f"Durasi: {duration:.2f} detik")

# Menampilkan jumlah channel
print(f"Jumlah kanal: {'1 -> Mono' if y.ndim == 1 else '2 -> Stereo'")

# Menampilkan jumlah total sampel
print(f"Total Sampel: {len(y)}")

# Menampilkan audio player
display(Audio(y, rate=sr))

```

Sample Rate: 22050 Hz

Durasi: 10.13 detik

Jumlah kanal: 1 -> Mono

Total Sampel: 223411

```
/var/folders/ds/tzk88yl5705g6nytd56kbr10000gn/T/ipykernel_11858/278
207745.py:5: UserWarning: PySoundFile failed. Trying audioread instead.
```

```
y, sr = librosa.load(path_audio)
```

0:00 -0:10

In [92]:

```

# Menampilkan spectrogram dari file audio
# Menghitung STFT
D = librosa.stft(y, n_fft=1024, hop_length=256)

# Mengubah ke skala log-dB (amplitudo ke dB)
DB = librosa.amplitude_to_db(abs(D), ref=np.max)

# Menentukan ukuran figure
plt.figure(figsize=(12, 6))

# Menampilkan spectrogram dalam skala log-dB
img = librosa.display.specshow(DB, sr=sr, hop_length=256, x_axis='time')

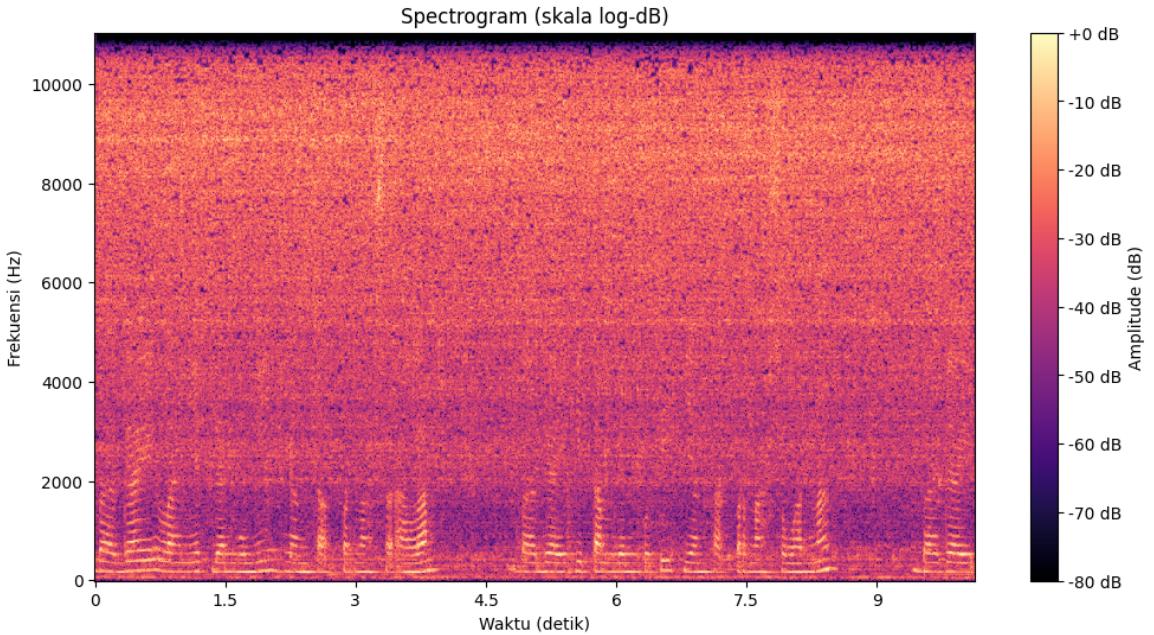
# Menambahkan colorbar
plt.colorbar(img, format='%.2f dB', label='Amplitude (dB)')

# Menambahkan label dan judul
plt.title('Spectrogram (skala log-dB)')
plt.xlabel('Waktu (detik)')
plt.ylabel('Frekuensi (Hz)')

```

Out[92]:

Text(0, 0.5, 'Frekuensi (Hz)')



Note: Spectrogram ini saya tampilkan agar menjadi acuan saya untuk menggunakan filter equalizer

```
In [93]: def butter_filter(data, cutoff, sr, filter_type='low', order=5):
    nyquist = 0.5 * sr
    if filter_type == 'band':
        normal_cutoff = [c / nyquist for c in cutoff]
    else:
        normal_cutoff = cutoff / nyquist

    b, a = butter(order, normal_cutoff, btype=filter_type, analog=False)
    filtered = filtfilt(b, a, data)
    return filtered
```

2.1 Low-pass

```
In [94]: # Low-pass (High-cut di 2000 Hz)
low_filtered = butter_filter(y, cutoff=2000, sr=sr, filter_type='low')

print("Audio setelah Low-pass Filter di 2000 Hz:")
display(Audio(low_filtered, rate=sr))

# Menampilkan spectrogram dari audio yang sudah di-low-pass filter
# Menghitung STFT
D = librosa.stft(low_filtered, n_fft=1024, hop_length=256)

# Mengubah ke skala log-dB (amplitudo ke dB)
DB = librosa.amplitude_to_db(abs(D), ref=np.max)

# Menentukan ukuran figure
plt.figure(figsize=(12, 6))

# Menampilkan spectrogram dalam skala log-dB
img = librosa.display.specshow(DB, sr=sr, hop_length=256, x_axis='t'

# Menambahkan colorbar
```

```

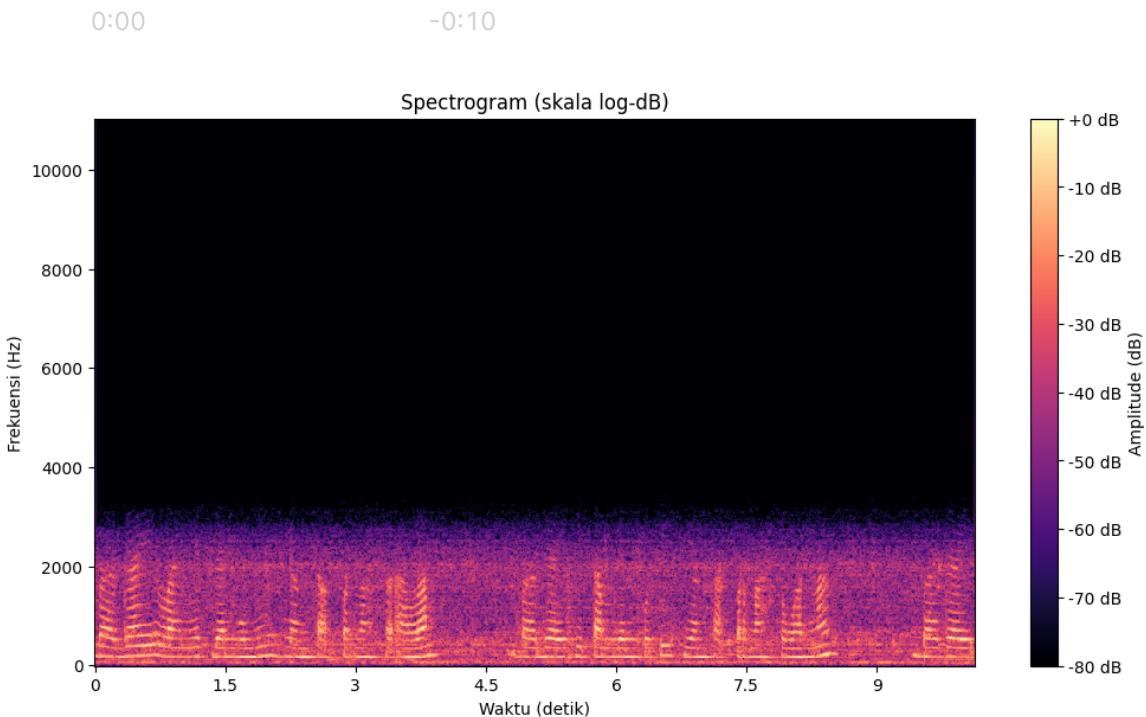
plt.colorbar(img, format='%+2.0f dB', label='Amplitude (dB)')

# Menambahkan label dan judul
plt.title('Spectrogram (skala log-dB)')
plt.xlabel('Waktu (detik)')
plt.ylabel('Frekuensi (Hz)')

sf.write('data/soal-2-lowpass.wav', low_filtered, sr)

```

Audio setelah Low-pass Filter di 2000 Hz:



2.2 High-pass

```

In [95]: # High-pass (Low-cut di 200 Hz)
high_filtered = butter_filter(y, cutoff=200, sr=sr, filter_type='hi'
print("Audio setelah High-pass Filter di 200 Hz:")
display(Audio(high_filtered, rate=sr))

# Menampilkan spectrogram dari audio yang sudah di-high-pass filter
# Menghitung STFT
D = librosa.stft(high_filtered, n_fft=1024, hop_length=256)

# Mengubah ke skala log-dB (amplitudo ke dB)
DB = librosa.amplitude_to_db(abs(D), ref=np.max)

# Menentukan ukuran figure
plt.figure(figsize=(12, 6))

# Menampilkan spectrogram dalam skala log-dB
img = librosa.display.specshow(DB, sr=sr, hop_length=256, x_axis='t'

# Menambahkan colorbar
plt.colorbar(img, format='%+2.0f dB', label='Amplitude (dB)')

```

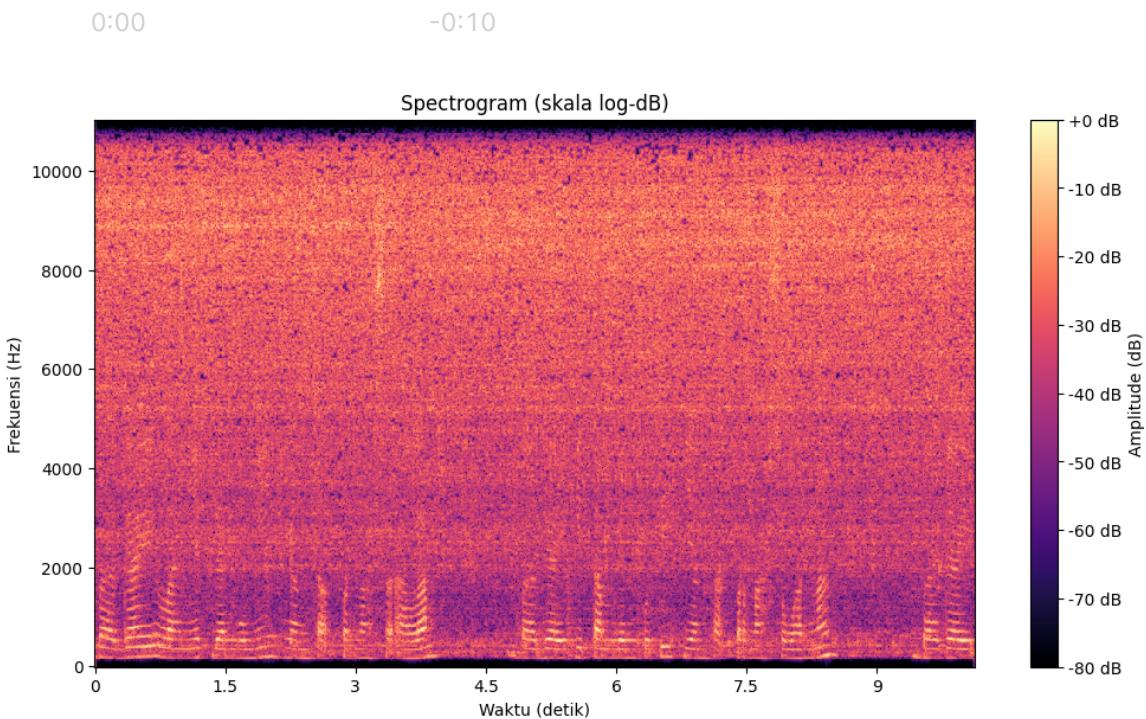
```

# Menambahkan label dan judul
plt.title('Spectrogram (skala log-dB)')
plt.xlabel('Waktu (detik)')
plt.ylabel('Frekuensi (Hz)')

sf.write('data/soal-2-highpass.wav', high_filtered, sr)

```

Audio setelah High-pass Filter di 200 Hz:



2.3 Band-pass

```

In [96]: # Band-pass (frekuensi 200 Hz – 2000 Hz)
band_filtered = butter_filter(y, cutoff=[200, 2000], sr=sr, filter_
print("Audio setelah Band-pass Filter:")
display(Audio(band_filtered, rate=sr))

# Menampilkan spectrogram dari audio yang sudah di-band-pass filter
# Menghitung STFT
D = librosa.stft(band_filtered, n_fft=1024, hop_length=256)

# Mengubah ke skala log-dB (amplitudo ke dB)
DB = librosa.amplitude_to_db(abs(D), ref=np.max)

# Menentukan ukuran figure
plt.figure(figsize=(12, 6))

# Menampilkan spectrogram dalam skala log-dB
img = librosa.display.specshow(DB, sr=sr, hop_length=256, x_axis='t'

# Menambahkan colorbar
plt.colorbar(img, format='%.2f dB', label='Amplitude (dB)')

# Menambahkan label dan judul
plt.title('Spectrogram (skala log-dB)')

```

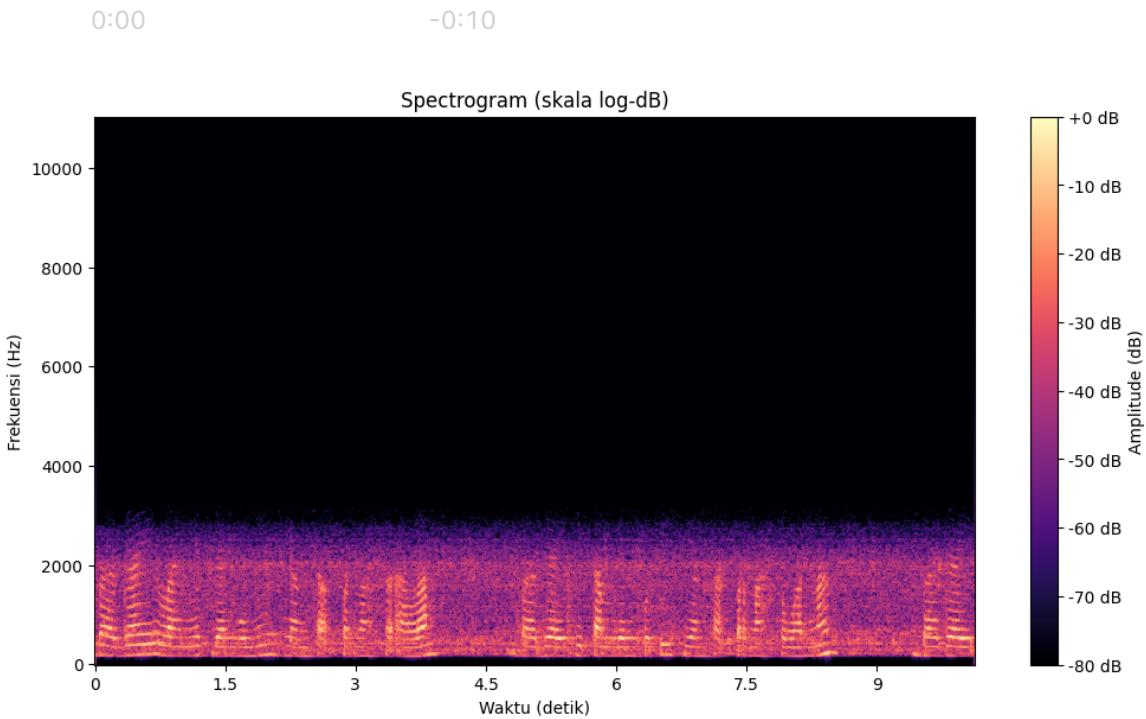
```

plt.xlabel('Waktu (detik)')
plt.ylabel('Frekuensi (Hz)')

sf.write('data/soal-2-bandpass.wav', band_filtered, sr)

```

Audio setelah Band-pass Filter:



```

In [97]: # Menampilkan seluruh spectrogram dalam satu plot untuk perbandingan
plt.figure(figsize=(14, 8))

# Terapkan filter
unfiltered = y
low_filtered = butter_filter(y, cutoff=2000, sr=sr, filter_type='low')
high_filtered = butter_filter(y, cutoff=200, sr=sr, filter_type='high')
band_filtered = butter_filter(y, cutoff=[200, 2000], sr=sr, filter_type='band')

# --- Hitung STFT untuk semua hasil filter ---
D = librosa.stft(unfiltered, n_fft=1024, hop_length=256)
DB = librosa.amplitude_to_db(abs(D), ref=np.max)

D_low = librosa.stft(low_filtered, n_fft=1024, hop_length=256)
DB_low = librosa.amplitude_to_db(abs(D_low), ref=np.max)

D_high = librosa.stft(high_filtered, n_fft=1024, hop_length=256)
DB_high = librosa.amplitude_to_db(abs(D_high), ref=np.max)

D_band = librosa.stft(band_filtered, n_fft=1024, hop_length=256)
DB_band = librosa.amplitude_to_db(abs(D_band), ref=np.max)

# --- Subplot 1: Unfiltered ---
plt.subplot(2, 2, 1)
img0 = librosa.display.specshow(DB, sr=sr, hop_length=256, x_axis='time')
plt.title('Audio Asli')
plt.xlabel('Waktu (detik)')
plt.ylabel('Frekuensi (Hz)')

```

```

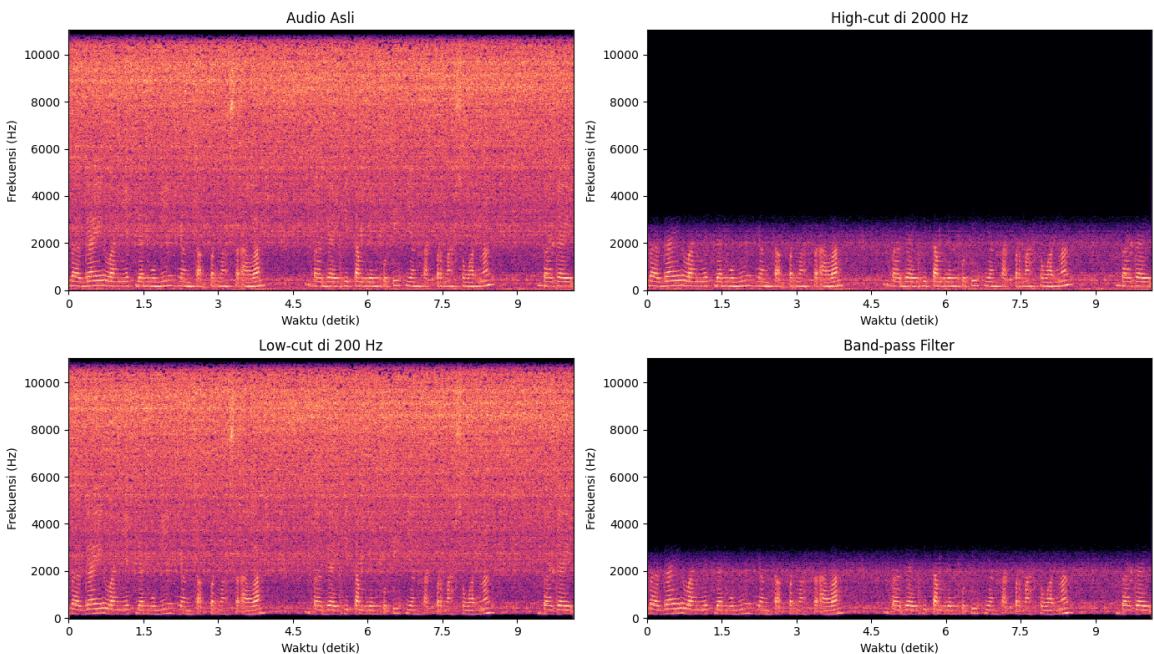
# --- Subplot 2: Low-pass ---
plt.subplot(2, 2, 2)
img1 = librosa.display.specshow(DB_low, sr=sr, hop_length=256, x_ax
plt.title('High-cut di 2000 Hz')
plt.xlabel('Waktu (detik)')
plt.ylabel('Frekuensi (Hz)')

# --- Subplot 3: High-pass ---
plt.subplot(2, 2, 3)
img2 = librosa.display.specshow(DB_high, sr=sr, hop_length=256, x_a
plt.title('Low-cut di 200 Hz')
plt.xlabel('Waktu (detik)')
plt.ylabel('Frekuensi (Hz)')

# --- Subplot 3: Band-pass ---
plt.subplot(2, 2, 4)
img3 = librosa.display.specshow(DB_band, sr=sr, hop_length=256, x_a
plt.title('Band-pass Filter')
plt.xlabel('Waktu (detik)')
plt.ylabel('Frekuensi (Hz)')

plt.tight_layout()
plt.show()

```



Perbandingan

Dari spectrogram, dapat dibandingkan frekuensi audio sebelum difilter dan setelah difilter.

- Setelah difilter Low-pass atau High-cut di 2000 Hz, dapat dilihat kalau intensitas suara pada frekuensi di atas 2000 Hz tidak ada lagi (warna gelap).
- Setelah difilter High-pass atau Low-cut di 200 Hz, dapat dilihat kalau intensitas suara pada frekuensi di bawah 200 Hz tidak ada lagi dan frekuensi di atas 200 Hz dibiarkan, terlihat dengan warna hijau kuning.

- Setelah difilter Band-pass di 200 - 2000 Hz, dapat dilihat bahwa bagian bawah dan atas berwarna gelap. Frekuensi di bawah 200 Hz dan di atas 2000 Hz berwarna gelap (hilang).

Penjelasan

1. Noise yang muncul pada rekaman saya adalah suara keran yang mengeluarkan air secara deras. Menurut saya, noise tersebut berfrekuensi di atas 2000 Hz karena pada spectrogram, intensitas/warna yang bersifat konstan dan tidak ada perubahan berada di frekuensi 2000 ke atas, sedangkan pada frekuensi 0 sampai dengan 2000 intensitasnya berubah-ubah (warna berubah-ubah).
 2. Filter yang paling efektif untuk mengurangi noise tersebut menurut saya adalah high-cut atau low-pass karena noise berada di frekuensi 2000 Hz ke atas. Mengapa tidak band-pass? Karena menurut saya dengan filter band-pass, ucapan saya (suara selain noise) yang dihasilkan kurang jelas karena frekuensi rendahnya dihilangkan.
 3. Nilai cutoff yang memberikan hasil terbaik adalah 2000 Hz
 4. Kualitas suara setelah filtering lumayan baik, setidaknya suara keran berkurang meskipun ucapan tidak menjadi lebih jelas.
-

Soal 3: Pitch Shifting dan Audio Manipulation

```
In [98]: # Mengatur path audio dengan os agar bisa dijalankan di mana saja
path_audio = os.path.join(os.getcwd(), 'data', 'soal-1.wav')

# Load file audio
y, sr = librosa.load(path_audio)

# Menampilkan sample rate
print (f"Sample Rate: {sr} Hz")

# Menghitung durasi
duration = librosa.get_duration(y=y, sr=sr)
# Menampilkan durasi
print(f"Durasi: {duration:.2f} detik")

# Menampilkan jumlah channel
print(f"Jumlah kanal: {'1 -> Mono' if y.ndim == 1 else '2 -> Stereo'")

# Menampilkan jumlah total sampel
print(f"Total Sampel: {len(y)}")

# Menampilkan audio player
display(Audio(y, rate=sr))
```

Sample Rate: 22050 Hz
Durasi: 25.02 detik
Jumlah kanal: 1 → Mono
Total Sampel: 551750

```
/var/folders/ds/tzk88yl5705g6nytd56kbr10000gn/T/ipykernel_11858/432  
042548.py:5: UserWarning: PySoundFile failed. Trying audioread instead.
```

```
y, sr = librosa.load(path_audio)
```

0:00 -0:25

```
In [99]: # Pitch shifting naik 7 semitone  
y_pitch_up_7 = librosa.effects.pitch_shift(y, sr=sr, n_steps=7)  
  
print("Audio setelah Pitch Shifting Naik 7 Semitone:")  
display(Audio(y_pitch_up_7, rate=sr))  
  
# Pitch shidting naik 12 semitone  
y_pitch_up_12 = librosa.effects.pitch_shift(y, sr=sr, n_steps=12)  
  
print("Audio setelah Pitch Shifting Naik 12 Semitone:")  
display(Audio(y_pitch_up_12, rate=sr))  
  
sf.write('data/soal-1-pitch-up-7.wav', y_pitch_up_7, sr)  
sf.write('data/soal-1-pitch-up-12.wav', y_pitch_up_12, sr)
```

Audio setelah Pitch Shifting Naik 7 Semitone:

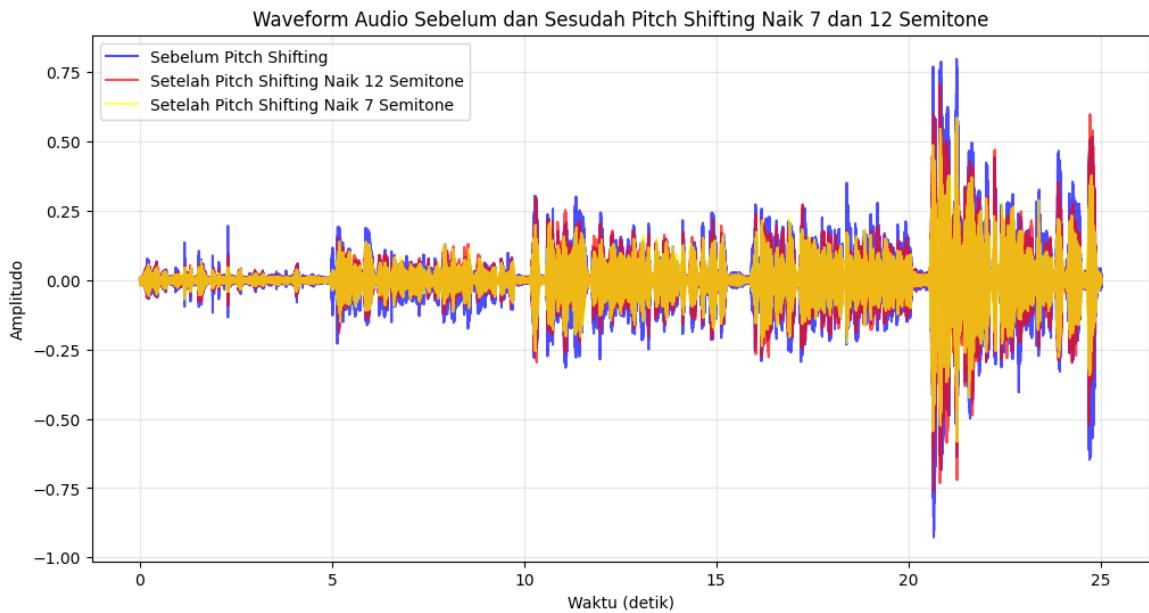
0:00 -0:25

Audio setelah Pitch Shifting Naik 12 Semitone:

0:00 -0:25

```
In [100...]: # Viualisasi waveform audio sebelum dan sesudah pitch shifting  
  
# Membuat array waktu untuk sumbu x dari detik 0 sampai durasi audio  
time = np.linspace(0, duration, len(y))  
time_pitch_up_7 = np.linspace(0, duration, len(y_pitch_up_7))  
time_pitch_up_12 = np.linspace(0, duration, len(y_pitch_up_12))  
  
plt.figure(figsize=(12, 6))  
  
# Membuat plot garis  
plt.plot(time, y, color='blue', alpha=0.7, label='Sebelum Pitch Shifting')  
plt.plot(time_pitch_up_12, y_pitch_up_12, color='red', alpha=0.7, label='Setelah Pitch Shifting Naik 12 Semitone')  
plt.plot(time_pitch_up_7, y_pitch_up_7, color='yellow', alpha=0.7, label='Setelah Pitch Shifting Naik 7 Semitone')  
  
# Menambahkan judul  
plt.title('Waveform Audio Sebelum dan Sesudah Pitch Shifting Naik 7 Semitone')  
  
plt.xlabel('Waktu (detik)')  
plt.ylabel('Amplitudo')  
plt.grid(True, alpha=0.3)  
plt.legend()
```

```
plt.show()
```



```
In [101]: # Visualisasi spectrogram audio sebelum dan sesudah pitch shifting

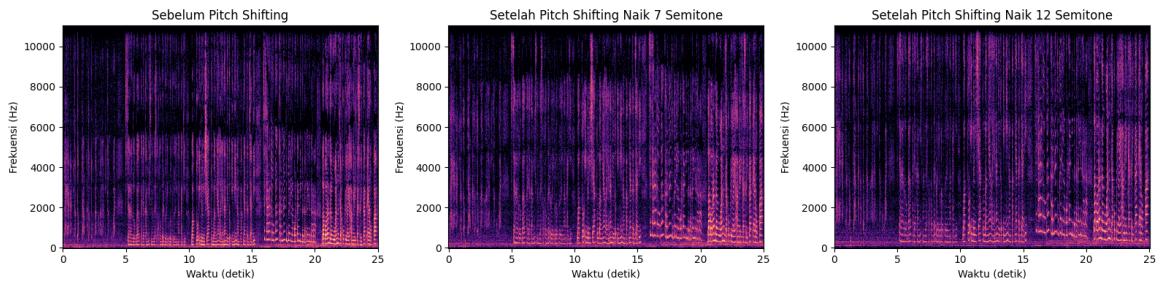
# Menghitung STFT
D = librosa.stft(y, n_fft=1024, hop_length=256)
D_pitch_up_7 = librosa.stft(y_pitch_up_7, n_fft=1024, hop_length=25)
D_pitch_up_12 = librosa.stft(y_pitch_up_12, n_fft=1024, hop_length=25)
DB = librosa.amplitude_to_db(abs(D), ref=np.max)
DB_pitch_up_7 = librosa.amplitude_to_db(abs(D_pitch_up_7), ref=np.max)
DB_pitch_up_12 = librosa.amplitude_to_db(abs(D_pitch_up_12), ref=np.max)

plt.figure(figsize=(16, 4))

# --- Subplot 1: Sebelum Pitch Shifting ---
plt.subplot(1, 3, 1)
img0 = librosa.display.specshow(DB, sr=sr, hop_length=256, x_axis='Waktu (detik)')
plt.title('Sebelum Pitch Shifting')
plt.xlabel('Waktu (detik)')
plt.ylabel('Frekuensi (Hz)')
plt.tight_layout()

# --- Subplot 2: Setelah Pitch Shifting +7 ---
plt.subplot(1, 3, 2)
img1 = librosa.display.specshow(DB_pitch_up_7, sr=sr, hop_length=25)
plt.title('Setelah Pitch Shifting Naik 7 Semitone')
plt.xlabel('Waktu (detik)')
plt.ylabel('Frekuensi (Hz)')
plt.tight_layout()

# --- Subplot 3: Setelah Pitch Shifting +12 ---
plt.subplot(1, 3, 3)
img2 = librosa.display.specshow(DB_pitch_up_12, sr=sr, hop_length=25)
plt.title('Setelah Pitch Shifting Naik 12 Semitone')
plt.xlabel('Waktu (detik)')
plt.ylabel('Frekuensi (Hz)')
plt.tight_layout()
```



Penjelasan

1. Untuk melakukan pitch shifting, saya menggunakan library dari librosa dengan parameter:
 - y -> audio yang akan diproses
 - sr=sr -> sample rate dari audio
 - n_steps=7 & n_steps=12 -> pitch yang akan dishift adalah +7 dan +12 semitones/step
2. Perbedaan representasi visual:
 - Waveform -> setelah pitch-shifting, amplitudo menjadi lebih kecil secara umum dibandingkan sebelum pitch-shifting
 - Spectrogram -> setelah pitch-shifting, energi di frekuensi tinggi menjadi lebih besar atau intensitas pada frekuensi tinggi menjadi lebih tinggi dibandingkan sebelumnya. Yang sebelumnya pada frekuensi tinggi warna yang ditampilkan cenderung gelap, setelah pitch shifting warna gelap mulai diisi dengan warna cerah (biru).
3. Perubahan pitch memengaruhi kualitas dan kejelasan suara. Kualitas suara yang diberikan pitch-shifting menjadi kurang jelas.

```
In [102...]: # Menggabungkan kedua file audio hasil pitch shifting
y_combined = np.concatenate((y_pitch_up_7, y_pitch_up_12))
sf.write('data/soal-3.wav', y_combined, sr)

print("Audio setelah Menggabungkan Kedua File Audio Hasil Pitch Shi
display(Audio(y_combined, rate=sr))
```

Audio setelah Menggabungkan Kedua File Audio Hasil Pitch Shifting:

0:00 -0:50

Soal 4: Audio Processing Chain

```
In [103...]: # Mengatur path audio dengan os agar bisa dijalankan di mana saja
path_audio = os.path.join(os.getcwd(), 'data', 'soal-3.wav')

# Load file audio
y, sr = librosa.load(path_audio)
```

```

# Menampilkan sample rate
print(f"Sample Rate: {sr} Hz")

# Menghitung durasi
duration = librosa.get_duration(y=y, sr=sr)
# Menampilkan durasi
print(f"Durasi: {duration:.2f} detik")

# Menampilkan jumlah channel
print(f"Jumlah kanal: {'1 -> Mono' if y.ndim == 1 else '2 -> Stereo'")

# Menampilkan jumlah total sampel
print(f"Total Sampel: {len(y)}")

# Menampilkan audio player
display(Audio(y, rate=sr))

```

Sample Rate: 22050 Hz
 Durasi: 50.05 detik
 Jumlah kanal: 1 -> Mono
 Total Sampel: 1103500

0:00 -0:50

In [104...]

```

# Equalizer
# Low-pass filter di 5000 Hz
def butter_filter(data, cutoff, sr, filter_type='low', order=5):
    nyq = 0.5 * sr
    normal_cutoff = cutoff / nyq if filter_type != 'band' else [c /
        b, a = butter(order, normal_cutoff, btype=filter_type, analog=False)
    return filtfilt(b, a, data)

y_eq = butter_filter(y, cutoff=5000, sr=sr, filter_type='low')
print("Audio setelah Equalizer (Low-pass di 5000 Hz):")
display(Audio(y_eq, rate=sr))

# Gain / Fade
fade_in = int(sr * 5) # 5 s
fade_out = int(sr * 10) # 10 s
y_fade = y_eq.copy()
y_fade[:fade_in] *= np.linspace(0, 1, fade_in)
y_fade[-fade_out:] *= np.linspace(1, 0, fade_out)
print("Audio setelah Fade In/Out:")
display(Audio(y_fade, rate=sr))

# Normalization (peak)
peak = np.max(np.abs(y_fade))
y_norm = y_fade / peak
print("Audio setelah Peak Normalization:")
display(Audio(y_norm, rate=sr))

# Compression
def simple_compressor(x, threshold_db=-18, ratio=3.0):
    threshold = 10***(threshold_db / 20)
    x_out = np.copy(x)

```

```

mask = np.abs(x) > threshold
x_out[mask] = np.sign(x[mask]) * (threshold + (np.abs(x[mask]) - threshold))
return x_out

y_comp = simple_compressor(y_norm)
print("Audio setelah Compression:")
display(Audio(y_comp, rate=sr))

# Noise Gate
def noise_gate(x, threshold_db=-55, reduction_db=-60):
    threshold = 10**((threshold_db / 20))
    reduction = 10**((reduction_db / 20))
    gate = np.where(np.abs(x) < threshold, reduction, 1.0)
    return x * gate

y_gate = noise_gate(y_comp)
print("Audio setelah Noise Gate:")
display(Audio(y_gate, rate=sr))

# Silence trimming
y_trim, _ = librosa.effects.trim(y_gate, top_db=30)
print("Audio setelah Silence Trimming:")
display(Audio(y_trim, rate=sr))

# Loudness normalization ke -16 LUFS
meter = pyln.Meter(sr)
loudness = meter.integrated_loudness(y_trim)
y_lufs = pyln.normalize.loudness(y_trim, loudness, -16.0)
print("Audio setelah Normalisasi Loudness (-16 LUFS):")
display(Audio(y_lufs, rate=sr))

sf.write('data/soal-3-processed.wav', y_lufs, sr)

```

Audio setelah Equalizer (Low-pass di 5000 Hz):

0:00 -0:50

Audio setelah Fade In/Out:

0:00 -0:50

Audio setelah Peak Normalization:

0:00 -0:50

Audio setelah Compression:

0:00 -0:50

Audio setelah Noise Gate:

0:00 -0:50

Audio setelah Silence Trimming:

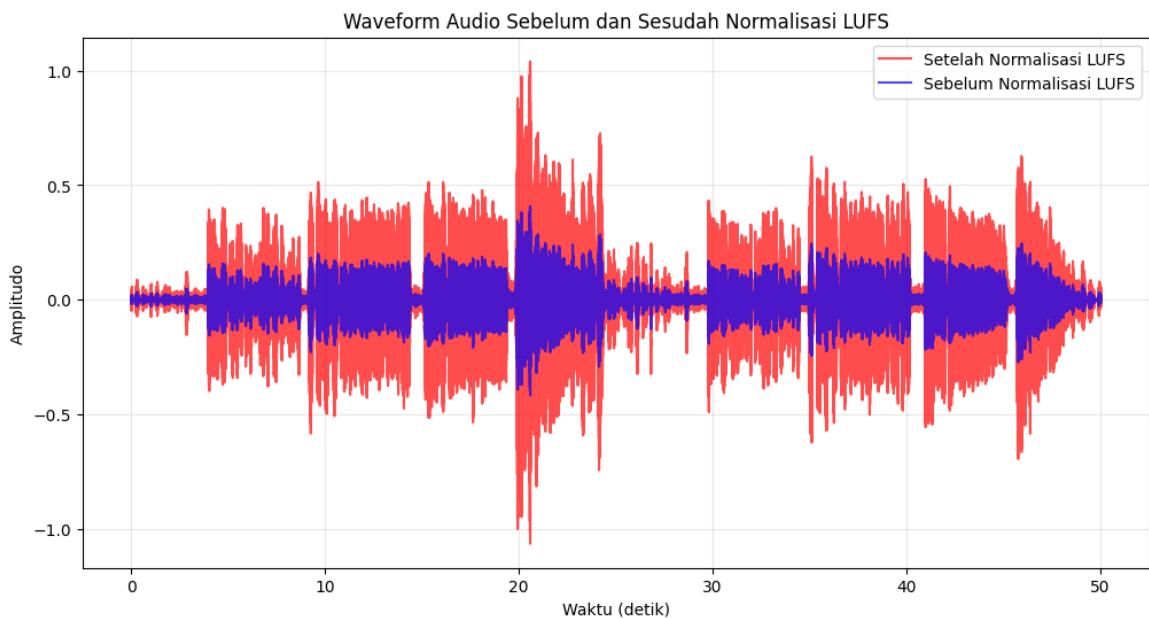
0:00 -0:48

Audio setelah Normalisasi Loudness (-16 LUFS):

```
/Users/nashwalaisya/multimedia-uv/lib/python3.13/site-packages/pylou  
dnorm/normalize.py:62: UserWarning: Possible clipped samples in output.  
    warnings.warn("Possible clipped samples in output.")
```

0:00 -0:48

```
In [105...]: # Visualisasi waveform audio sebelum dan sesudah proses normalisasi  
# Membuat array waktu untuk sumbu x dari detik 0 sampai durasi audio  
time = np.linspace(0, duration, len(y_trim))  
time_lufs = np.linspace(0, duration, len(y_lufs))  
plt.figure(figsize=(12, 6))  
# Membuat plot garis  
plt.plot(time_lufs, y_lufs, color='red', alpha=0.7, label='Setelah Normalisasi LUFS')  
plt.plot(time, y_trim, color='blue', alpha=0.7, label='Sebelum Normalisasi LUFS')  
# Menambahkan judul  
plt.title('Waveform Audio Sebelum dan Sesudah Normalisasi LUFS')  
plt.xlabel('Waktu (detik)')  
plt.ylabel('Amplitudo')  
plt.grid(True, alpha=0.3)  
plt.legend()  
plt.show()
```



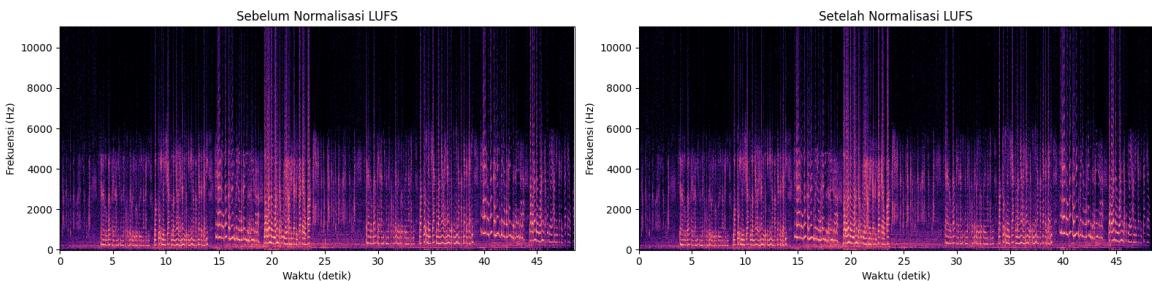
```
In [106...]: # Visualisasi spectrogram audio sebelum dan sesudah proses normalisasi  
# Menghitung STFT  
D = librosa.stft(y_trim, n_fft=1024, hop_length=256)  
D_lufs = librosa.stft(y_lufs, n_fft=1024, hop_length=256)  
DB = librosa.amplitude_to_db(abs(D), ref=np.max)  
DB_lufs = librosa.amplitude_to_db(abs(D_lufs), ref=np.max)  
plt.figure(figsize=(16, 4))  
  
# --- Subplot 1: Sebelum Normalisasi LUFS ---  
plt.subplot(1, 2, 1)  
img0 = librosa.display.specshow(DB, sr=sr, hop_length=256, x_axis='Waktu (detik)')  
plt.title('Sebelum Normalisasi LUFS')
```

```

plt.ylabel('Frekuensi (Hz)')
plt.tight_layout()

# --- Subplot 2: Setelah Normalisasi LUFS ---
plt.subplot(1, 2, 2)
img1 = librosa.display.specshow(DB_lufs, sr=sr, hop_length=256, x_a
plt.title('Setelah Normalisasi LUFS')
plt.xlabel('Waktu (detik)')
plt.ylabel('Frekuensi (Hz)')
plt.tight_layout()

```



Penjelasan

1. Perubahan suara yang terjadi setelah proses Equalizer adalah suara yang dihasilkan mengeluarkan frekuensi yang tidak setinggi sebelumnya, tetapi sedikit kehilangan kejelasan ucapan. Setelah fade in dan fade out, suara jadi memiliki efek dari pelan ke keras pada 5 detik awal dan efek keras ke pelan di 10 detik akhir. Setelah normalisasi peak, suara yang dikeluarkan tidak setinggi sebelumnya. Setelah compression, suara berbisik yang sebelumnya kurang terdengar menjadi lebih jelas. Setelah noise gate, suara yang dihasilkan lebih bersih. Setelah silence trimming, audio yang sebelumnya memiliki beberapa waktu yang hening, sekarang sudah tidak ada lagi dan durasi pun berubah menjadi lebih kecil. Setelah normalisasi loudness, suara menjadi lebih stabil.
2. Perbedaan normalisasi peak dan normalisasi LUFS adalah:
 - Untuk normalisasi peak, parameter utamanya adalah puncak amplitudo tertinggi dan mengatur volume agar puncak sinyal mencapai level tertentu.
 - Sedangkan normalisasi loudness (LUFS), parameter utamanya adalah persepsi loudness manusia dan mengatur volume rata-rata agar loudness terdengar sama dengan target.
3. Setelah proses normalisasi dan loudness optimization, volume terdengar lebih stabil, tidak ada yang terlalu pelan atau keras, suara/ucapan yang dihasilkan lebih detail.
4. Kelebihan dan kekurangan dari pengoptimalan loudness dalam konteks rekaman suara:
 - Kelebihan: Volume antar track seragam, tidak perlu ubah volume manual saat memutar; Lebih sesuai standar platform audio modern (YouTube, Spotify, broadcast); Meningkatkan persepsi "kejelasan" dan "profesionalitas" hasil rekaman; Memudahkan mastering dan

distribusi antar media.

- Kekurangan: Jika terlalu ekstrem, dapat merusak dinamika alami musik atau suara; "Loudness war" — rekaman jadi terlalu keras, cepat membuat pendengar lelah; Risiko clipping atau distorsi jika peak safety tidak dijaga dengan benar; Tidak selalu cocok untuk semua genre — misalnya musik klasik atau ambient butuh dinamika luas
-

Soal 5: Music Analysis dan Remix

In [107...]

```
# Mengatur path audio dengan os agar bisa dijalankan di mana saja
path_lagu1 = os.path.join(os.getcwd(), 'data', 'lagu-1.wav')
path_lagu2 = os.path.join(os.getcwd(), 'data', 'lagu-2.wav')

# Load file audio
y1, sr1 = librosa.load(path_lagu1)
y2, sr2 = librosa.load(path_lagu2)

# Menampilkan sample rate
print (f"Sample Rate Lagu-1: {sr1} Hz")
print (f"Sample Rate Lagu-2: {sr2} Hz")

# Menghitung durasi
duration1 = librosa.get_duration(y=y1, sr=sr1)
duration2 = librosa.get_duration(y=y2, sr=sr2)
# Menampilkan durasi
print(f"Durasi Lagu-1: {duration1:.2f} detik")
print(f"Durasi Lagu-2: {duration2:.2f} detik")

# Menampilkan jumlah channel
print(f"Jumlah kanal Lagu-1: {'1 -> Mono' if y1.ndim == 1 else '2 -> Stereo'}")
print(f"Jumlah kanal Lagu-2: {'1 -> Mono' if y2.ndim == 1 else '2 -> Stereo'}")

# Menampilkan jumlah total sampel
print(f"Total Sampel Lagu-1: {len(y1)}")
print(f"Total Sampel Lagu-2: {len(y2)}")

# Menampilkan audio player
display(Audio(y1, rate=sr1))
display(Audio(y2, rate=sr2))
```

Sample Rate Lagu-1: 22050 Hz
Sample Rate Lagu-2: 22050 Hz
Durasi Lagu-1: 59.51 detik
Durasi Lagu-2: 60.06 detik
Jumlah kanal Lagu-1: 1 -> Mono
Jumlah kanal Lagu-2: 1 -> Mono
Total Sampel Lagu-1: 1312277
Total Sampel Lagu-2: 1324336

0:00

-0:59

5.1 Deteksi Tempo (BPM) dan Estimasi Kunci (Key)

In [108...]

```
# Nama-nama nada (pitch class)
PITCH_CLASS = ['C','C#','D','D#','E','F','F#','G','G#','A','A#','B']

# Profil Krumhansl (relatif) untuk mayor & minor
KRMHANSL_MAJOR = np.array([6.35,2.23,3.48,2.33,4.38,4.09,2.52,5.19
                           KRUMHANSL_MINOR = np.array([6.33,2.68,3.52,5.38,2.60,3.53,2.54,4.75

def _rotate(profile, k): # rotasi 12 nada (C=0, C#=1, ...)
    return np.roll(profile, k)

def estimate_bpm(y, sr):
    # Pakai komponen perkusif untuk tempo → lebih stabil
    _, y_perc = librosa.effects.hpss(y)
    onset_env = librosa.onset.onset_strength(y=y_perc, sr=sr)
    tempo, _ = librosa.beat.beat_track(onset_envelope=onset_env, sr=sr)
    return float(tempo)

def estimate_key(y, sr):
    # Pakai komponen harmonik agar pitch/chord lebih bersih
    y_harm, _ = librosa.effects.hpss(y)
    chroma = librosa.feature.chroma_cqt(y=y_harm, sr=sr, n_chroma=12)
    chroma_mean = chroma.mean(axis=1)
    chroma_mean = chroma_mean / (np.linalg.norm(chroma_mean) + 1e-12)

    best_key, best_mode, best_score = None, None, -1e9
    for k in range(12):
        maj = _rotate(KRMHANSL_MAJOR, k); maj = maj / np.linalg.norm(maj)
        minr = _rotate(KRMHANSL_MINOR, k); minr = minr / np.linalg.norm(minr)

        smaj = float(np.dot(chroma_mean, maj))
        smin = float(np.dot(chroma_mean, minr))
        if smaj > best_score:
            best_key, best_mode, best_score = PITCH_CLASS[k], 'major'
        if smin > best_score:
            best_key, best_mode, best_score = PITCH_CLASS[k], 'minor'
    return best_key, best_mode, best_score

def analyze_one(name, y, sr):
    bpm = estimate_bpm(y, sr)
    key, mode, conf = estimate_key(y, sr)

    # Sering terjadi 2x atau 1/2 tempo; tunjukkan kandidat alternatif
    alt_bpm = {'x0.5': bpm/2.0, 'x2': bpm*2.0}

    print(f"\n== {name} ==")
    print(f"Tempo (BPM) : {bpm:.1f} | alternatif: {alt_bpm['x0.5']} / {alt_bpm['x2']}")
    print(f"Key (estimasi) : {key} {mode} (score≈{conf:.3f})")
    return bpm, key, mode, conf

# === JALANKAN untuk Lagu-1 & Lagu-2 ===
```

```
bpm1, key1, mode1, conf1 = analyze_one("Lagu-1", y1, sr1)
bpm2, key2, mode2, conf2 = analyze_one("Lagu-2", y2, sr2)
```

```
/var/folders/ds/tzk88yl5705g6nytd56kbr10000gn/T/ipykernel_11858/282
7593516.py:16: DeprecationWarning: Conversion of an array with ndim
> 0 to a scalar is deprecated, and will error in future. Ensure you
extract a single element from your array before performing this oper
ation. (Deprecated NumPy 1.25.)
    return float(tempo)
==> Lagu-1 ===
Tempo (BPM)      : 117.5 | alternatif: 58.7 / 234.9
Key (estimasi)   : D major (score≈0.949)

==> Lagu-2 ===
Tempo (BPM)      : 117.5 | alternatif: 58.7 / 234.9
Key (estimasi)   : C major (score≈0.931)
```

Analisis

- Tempo numerik ≈117 BPM untuk keduanya, tapi persepsi tempo berbeda karena pola ritme dan energi musik.
- Lagu 1 (D major) → "moderato tapi emosional", cocok untuk suasana tenang/sedih.
- Lagu 2 (C major) → "moderato tapi cerah", cocok untuk suasana gembira/optimis

5.2 Remix

```
In [109...]: # REMIX PIPELINE: Time-stretch → Pitch-shift → Crossfade (+ audio p

# -----
# 0) PARAMETER
# -----
path_lagu1 = os.path.join(os.getcwd(), 'data', 'lagu-1.wav')
path_lagu2 = os.path.join(os.getcwd(), 'data', 'lagu-2.wav')

COMMON_SR      = 44100
TARGET_BPM     = 118.0
TARGET_KEY_NAME = 'C'
CROSSFADE_SEC  = 8.0
APPLY_FILTER   = True

# -----
# 1) UTILITAS
# -----
PITCH_CLASS = ['C', 'C#', 'D', 'D#', 'E', 'F', 'F#', 'G', 'G#', 'A', 'A#', 'B'
def key_to_index(name): return PITCH_CLASS.index(name.upper())

def detect_bpm(y, sr):
    _, y_perc = librosa.effects.hpss(y)
    onset = librosa.onset.onset_strength(y=y_perc, sr=sr)
    tempo, _ = librosa.beat.beat_track(onset_envelope=onset, sr=sr)
    return float(tempo)
```

```

KRMHANSL_MAJOR = np.array([6.35,2.23,3.48,2.33,4.38,4.09,2.52,5.19]
KRMHANSL_MINOR = np.array([6.33,2.68,3.52,5.38,2.60,3.53,2.54,4.75
def rotate(v,k): return np.roll(v,k)

def detect_key_basic(y, sr):
    y_harm, _ = librosa.effects.hpss(y)
    chroma = librosa.feature.chroma_cqt(y=y_harm, sr=sr)
    cmean = chroma.mean(axis=1); cmean /= (np.linalg.norm(cmean)+1e-12)
    best = (-1e9, None, None)
    for k in range(12):
        maj = rotate(KRMHANSL_MAJOR,k)/np.linalg.norm(KRMHANSL_MAJOR)
        minr= rotate(KRMHANSL_MINOR,k)/np.linalg.norm(KRMHANSL_MINOR)
        smaj, smin = np.dot(cmean, maj), np.dot(cmean, minr)
        if smaj > best[0]: best=(smaj,PITCH_CLASS[k],'major')
        if smin > best[0]: best=(smin,PITCH_CLASS[k],'minor')
    return best[1], best[2], best[0]

def time_stretch_to_bpm(y, sr, detected_bpm, target_bpm):
    rate = target_bpm / (detected_bpm + 1e-12)
    return librosa.effects.time_stretch(y, rate=rate)

def pitch_shift_to_key(y, sr, src_key, tgt_key):
    src, tgt = key_to_index(src_key), key_to_index(tgt_key)
    n_steps = (tgt - src) % 12
    if n_steps > 6: n_steps -= 12
    return librosa.effects.pitch_shift(y, sr=sr, n_steps=n_steps),

def normalize_peak(y, target_dbfs=-1.0):
    peak = np.max(np.abs(y)) + 1e-12
    return y * (10***(target_dbfs/20) / peak)

def highpass(y, sr, fc=100.0, order=2):
    b, a = butter(order, fc/(0.5*sr), btype='high')
    return filtfilt(b, a, y)

def crossfade_concat(a, b, sr, cf_sec=8.0):
    N = int(cf_sec * sr)
    N = min(N, len(a)//2, len(b)//2)
    fade_out = np.linspace(1, 0, N)
    fade_in = np.linspace(0, 1, N)
    head, tail, rest = a[:-N], a[-N:]*fade_out + b[:N]*fade_in, b[N:]
    return np.concatenate([head, tail, rest])

# -----
# 2) LOAD AUDIO
# -----
y1, sr1 = librosa.load(path_lagu1, sr=None, mono=True)
y2, sr2 = librosa.load(path_lagu2, sr=None, mono=True)
sr = COMMON_SR
if sr1 != sr: y1 = librosa.resample(y1, sr1, sr)
if sr2 != sr: y2 = librosa.resample(y2, sr2, sr)

# -----
# 3) ANALISIS BPM & KEY
# -----
bpm1 = detect_bpm(y1, sr); key1, mode1, _ = detect_key_basic(y1, sr)

```

```

bpm2 = detect_bpm(y2, sr); key2, mode2, _ = detect_key_basic(y2, sr)
print(f'Lagu-1: {bpm1:.1f} BPM, {key1} {mode1}')
print(f'Lagu-2: {bpm2:.1f} BPM, {key2} {mode2}')

# -----
# 4) TIME-STRETCH
# -----
y1_ts = time_stretch_to_bpm(y1, sr, bpm1, TARGET_BPM)
y2_ts = time_stretch_to_bpm(y2, sr, bpm2, TARGET_BPM)

# -----
# 5) PITCH-SHIFT
# -----
y1_ps, n1 = pitch_shift_to_key(y1_ts, sr, key1, TARGET_KEY_NAME)
y2_ps, n2 = pitch_shift_to_key(y2_ts, sr, key2, TARGET_KEY_NAME)
print(f'Pitch Shift: Lagu-1 {n1:+} semitone, Lagu-2 {n2:+} semitone')

# -----
# 6) FILTER OPSIONAL
# -----
if APPLY_FILTER:
    y2_ps = highpass(y2_ps, sr, fc=60.0)

# -----
# 7) NORMALISASI & CROSSFAADING
# -----
y1_ps = normalize_peak(y1_ps, -3.0)
y2_ps = normalize_peak(y2_ps, -3.0)
remix = crossfade_concat(y1_ps, y2_ps, sr, CROSSFADE_SEC)
remix = normalize_peak(remix, -1.0)

sf.write('data/soal-4-remix.wav', remix, sr)

# Audio Player
display(Audio(remix, rate=sr))

```

```

/var/folders/ds/tzk88yl5705g6nytd56kbr10000gn/T/ipykernel_11858/307
7755455.py:25: DeprecationWarning: Conversion of an array with ndim
> 0 to a scalar is deprecated, and will error in future. Ensure you
extract a single element from your array before performing this oper
ation. (Deprecated NumPy 1.25.)
    return float(tempo)

```

Lagu-1: 120.2 BPM, D major
Lagu-2: 120.2 BPM, C major
Pitch Shift: Lagu-1 -2 semitone, Lagu-2 +0 semitone

0:00 -1:53

Penjelasan

1. Time-stretch (penyesuaian tempo)
 - Tujuan: menyamakan kecepatan lagu agar beat-nya seirama.
 - Fungsi: librosa.effects.time_stretch()
 - Parameter: rate = TARGET_BPM / BPM_asli → mengatur

percepatan/perlambatan.

2. Pitch-Shift (Penyesuaian Kunci Lagu)

- Tujuan: menyamakan nada dasar kedua lagu (misalnya ke C).
- Fungsi: librosa.effects.pitch_shift()
- Parameter: n_steps = selisih semitone antara key asal dan key target.

3. Crossfade (Transisi Halus)

- Tujuan: menggabungkan dua lagu dengan overlap halus.
- Teknik: bagian akhir lagu pertama difade-out, lagu kedua difade-in.
- Parameter: CROSSFADE_SEC = durasi overlap (contoh: 8 detik).

4. Filter Opsional (High-Pass)

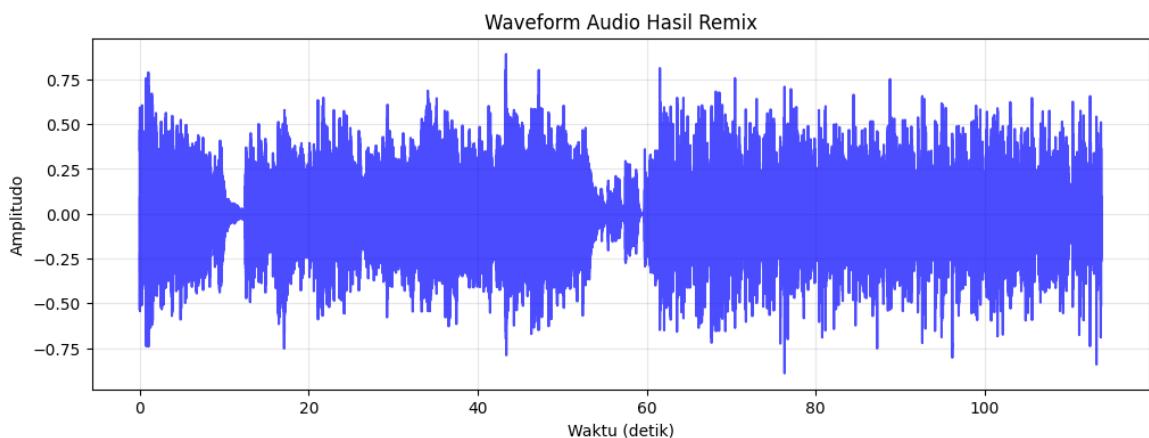
- Tujuan: menambah efek kreatif atau membersihkan frekuensi rendah saat transisi.
- Fungsi: scipy.signal.butter() → filter high-pass sederhana.
- Parameter: fc (cutoff) = 60 Hz, order = 2.

5. Normalisasi (Peak Normalization)

- Tujuan: menjaga level suara agar tidak clipping dan seimbang.
- Parameter: target_dbfs = -3 dB (sebelum crossfade) dan -1 dB (akhir).

In [110...]

```
# Menampilkan waveform dari file audio hasil remix
# Menghitung durasi
duration_remix = librosa.get_duration(y=remix, sr=sr)
# Membuat array waktu untuk sumbu x dari detik 0 sampai durasi audio
time_remix = np.linspace(0, duration_remix, len(remix))
plt.figure(figsize=(12, 4))
# Membuat plot garis
plt.plot(time_remix, remix, color='blue', alpha=0.7)
# Menambahkan judul
plt.title('Waveform Audio Hasil Remix')
plt.xlabel('Waktu (detik)')
plt.ylabel('Amplitudo')
plt.grid(True, alpha=0.3)
plt.show()
```



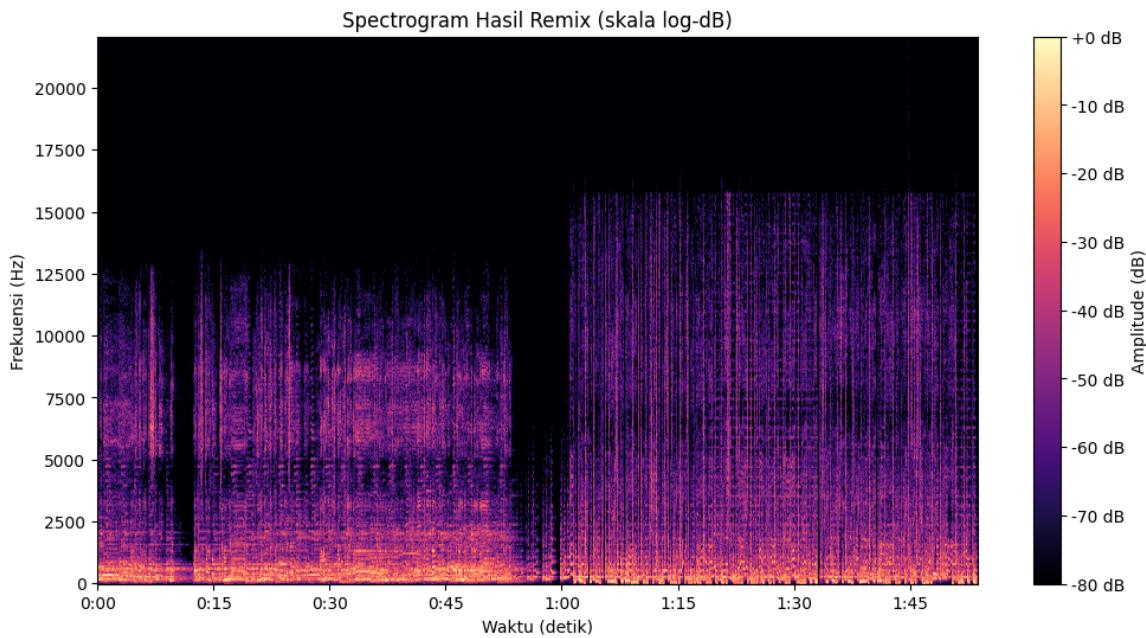
In [111...]

```
# Menampilkan spectrogram dari file audio hasil remix
# Menghitung STFT
D = librosa.stft(remix, n_fft=1024, hop_length=256)
```

```

# Mengubah ke skala log-dB (amplitudo ke dB)
DB = librosa.amplitude_to_db(abs(D), ref=np.max)
# Menentukan ukuran figure
plt.figure(figsize=(12, 6))
# Menampilkan spectrogram dalam skala log-dB
img = librosa.display.specshow(DB, sr=sr, hop_length=256, x_axis='t')
# Menambahkan colorbar
plt.colorbar(img, format='%+2.0f dB', label='Amplitude (dB)')
# Menambahkan label dan judul
plt.title('Spectrogram Hasil Remix (skala log-dB)')
plt.xlabel('Waktu (detik)')
plt.ylabel('Frekuensi (Hz)')
plt.show()

```



Penjelasan Hasil Remix

1. Tempo telah diseragamkan -> Kedua lagu sekarang memiliki kecepatan yang sama yaitu sekitar 118 BPM, sehingga ketukan dan ritme terdengar lebih sinkron saat digabung.
2. Kunci lagu disamakan -> Setelah pitch-shift ke C major, perbedaan nada dasar antara lagu sedih (D major) dan lagu ceria (C major) hilang. Sekarang transisi antar lagu terdengar lebih harmonis dan tidak "fals".
3. Transisi halus dengan crossfade -> Bagian akhir lagu pertama perlakan fade-out sementara lagu kedua fade-in selama ± 8 detik. Hasilnya, transition terasa lembut dan menyatu, tanpa jeda tiba-tiba.
4. Filter tambahan (high-pass) -> Saat lagu kedua mulai masuk, frekuensi rendah sedikit dipotong (sekitar 60 Hz) untuk memberi efek "terbuka" dan menonjolkan energi masuknya lagu baru.
5. Normalisasi akhir -> Volume keseluruhan diseimbangkan dan dijaga di sekitar -1 dBFS agar tidak clipping, membuat hasil akhir terdengar bersih dan konsisten.

Credit dan Referensi

ChatGPT: <https://chatgpt.com/share/68f1dfb0-8c38-8004-b03d-b9c64ca9bc21>