1) Chroma related features

Chroma features are a powerful representation of music audio in which we use a 12-element representation of spectral energy called a chroma vector where each of the 12 bins represent the 12 equal-tempered pitch class of western-type music (semitone spacing). It can be computed from the logarithmic short-time Fourier transform of the input sound signal, also called a chromagram or a pitch class profile. Chroma features are commonly used in audio signal processing and music analysis to capture the tonal content of a sound signal. They are particularly useful in music information retrieval tasks such as genre classification, chord recognition, and audio similarity.

Applications:

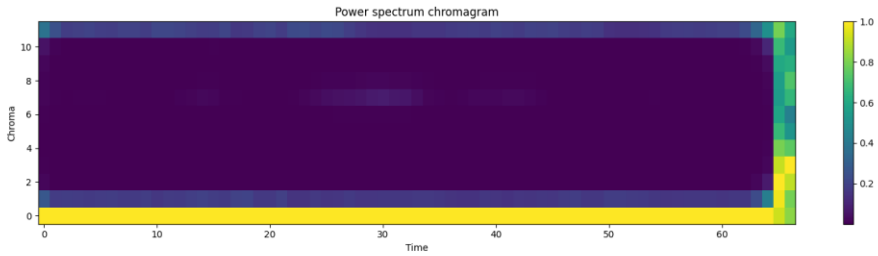
Music Analysis: Chroma features are used to represent the tonal content of music. Chroma vectors are computed to capture the distribution of pitch classes, providing a representation that is robust to changes in timbre and instrumentation. This is useful in tasks such as music genre classification, audio similarity matching, and chord recognition.

Chord Recognition: Chroma features are extensively used in chord recognition algorithms. By analyzing the chromagram systems can identify the chords played in a musical piece. This is valuable in music transcription, automatic accompaniment, and music information retrieval.

Melody Extraction: Chroma-based methods are employed in extracting melody information from audio signals. Melody extraction algorithms often use chroma features to identify the fundamental pitch of the main melody, aiding in tasks like melody transcription and generation.

Code:

```
# Load the audio file
y, sr = librosa.load(debussy_file)
# Compute the Chromagram
chroma_d = librosa.feature.chroma_stft(y=y, sr=sr)
# Plot the Chromagram
plt.figure(figsize=(15, 4))
plt.imshow(chroma_d, aspect='auto', origin='lower', cmap='viridis')
plt.colorbar()
plt.title('Power spectrum chromagram')
plt.xlabel('Time')
plt.ylabel('Chroma')
plt.tight_layout()
plt.show()
```



2)Tonality-based features

The tonal sounds of a harmonic stationary audio signal are actually the fundamental frequency (FF). The more technical definition of the FF is that it is the first peak of the local normalized spectrotemporal auto-correlation FF is an important feature for music onset detection, audio retrieval, and sound type classification. Tonality is the arrangement of pitches and/or chords of a musical work in a hierarchy of perceived relations, stabilities, attractions, and directionality. In this hierarchy the single pitch or triad with the greatest stability is called the tonic.

Applications:

Key detection and Music transcription: Tonality-based features are essential in determining the key of a musical piece. Key detection algorithms analyze tonal characteristics to identify the primary key or tonal center of a composition. This information is valuable for tasks such as music transcription and automatic music notation generation.

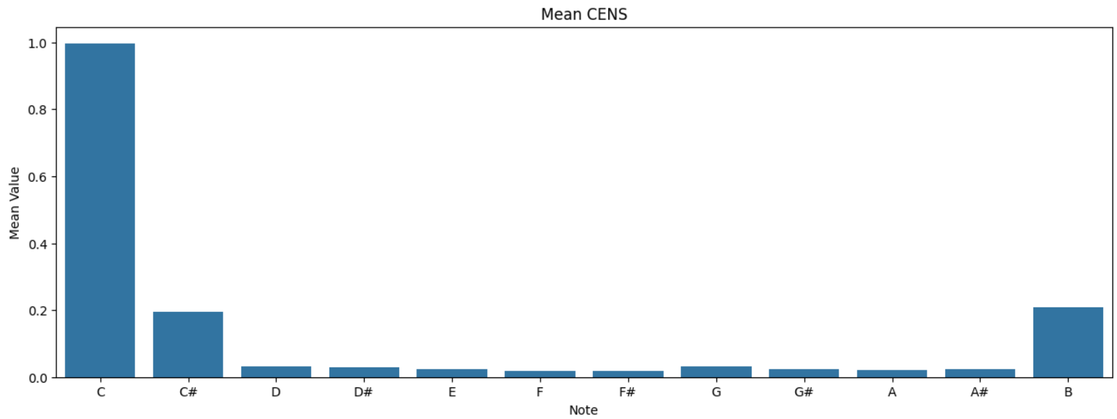
Pitch Shifting and Time Stretching: Tonality features are used in algorithms for pitch shifting and time stretching. By analyzing the tonal content of the audio signal, these algorithms can adjust the pitch without affecting the perceived tonality, allowing for musical adjustments without significant distortion.

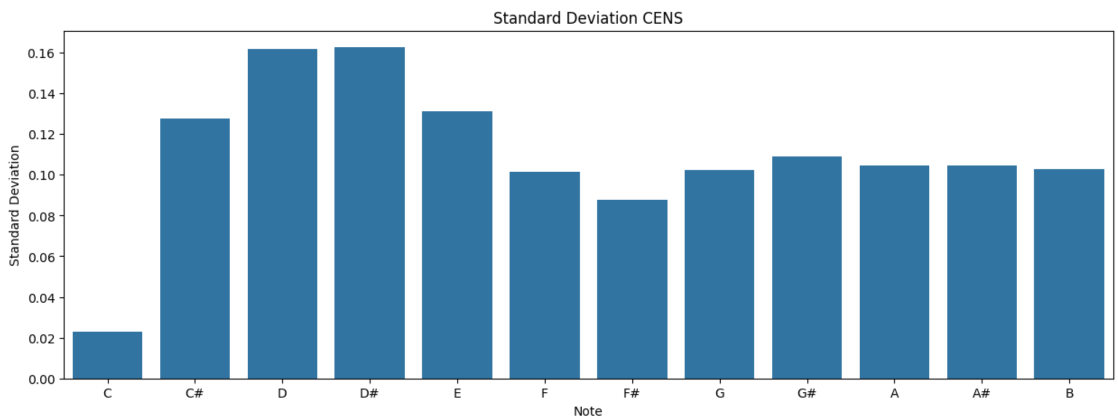
Automatic Accompaniment: Tonality-based features contribute to the development of automatic accompaniment systems. These systems use tonal analysis to understand the harmonic structure of a musical piece, enabling the generation of accompanying chords or musical elements.

Code:

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

# Compute the Chromagram
chroma = librosa.feature.chroma_stft(y=y, sr=sr)

# Compute the mean and standard deviation of CENS
chroma_mean = np.mean(chroma, axis=1)
chroma_std = np.std(chroma, axis=1)

# Plot the summary
octave = ['C', 'C#', 'D', 'D#', 'E', 'F', 'F#', 'G', 'G#', 'A', 'A#', 'B']

# Plot Mean CENS
plt.figure(figsize=(15, 5))
plt.title('Mean CENS')
sns.barplot(x=octave, y=chroma_mean)
plt.xlabel('Note')
plt.ylabel('Mean Value')
```

```
plt.show()

# Plot Standard Deviation CENS
plt.figure(figsize=(15, 5))
plt.title('Standard Deviation CENS')
sns.barplot(x=octave, y=chroma_std)
plt.xlabel('Note')
plt.ylabel('Standard Deviation')
plt.show()
```





Spectrum shape based features:

3)Spectral centroid:

The spectral centroid is a measure to characterize the "center of mass" of a given spectrum. Perceptually, it has a robust connection with the impression of sound "brightness". Timbre researchers formalize brightness as an indication of the amount of high-frequency content in a sound. The spectral centroid is calculated as the weighted means of the frequencies present in a given signal, determined using a Fourier transform, with the frequency magnitudes as the weights.

$$Centroid, \mu = \frac{\sum_{i=1}^{N} f_i \cdot m_i}{\sum_{i=1}^{N} m_i}$$

Where m_i is the magnitude of the bin number i and f_i is the center frequency of the bin i.

Applications:

Timbre Analysis: The spectral centroid is a crucial feature for characterizing the timbre of a sound. It provides information about the tonal color or brightness of a sound. Sounds with higher spectral centroids are often perceived as brighter or more treble-heavy, while lower spectral centroids are associated with darker or bass-heavy sounds.

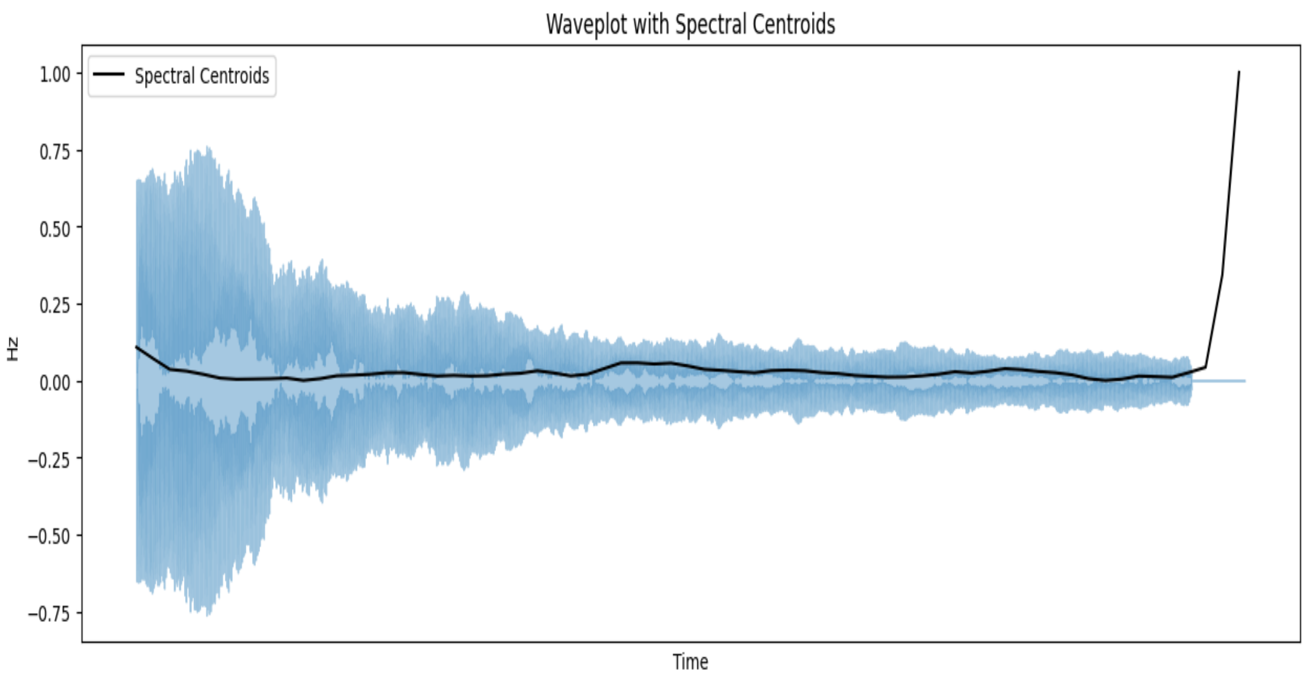
Speech Processing: In speech processing, the spectral centroid can be useful for differentiating between different speech sounds and speakers. It contributes to tasks such as speaker identification and emotion recognition by capturing aspects of the spectral distribution related to vocal characteristics.

Music instrument Recognition: Spectral centroid is used in the identification of musical instruments. Different instruments exhibit unique spectral centroid characteristics due to variations in their harmonic content and tonal qualities. This feature can be employed in algorithms for recognizing and classifying musical instruments.

Code:

```
# Load the audio file
y, sr = librosa.load(debussy_file)
# Compute spectral centroids
spectral_centroids = librosa.feature.spectral_centroid(y=y, sr=sr)[0]
# Compute the time variable for visualization
frames = range(len(spectral_centroids))
f_times = librosa.frames_to_time(frames)
# An auxiliary function to normalize the spectral centroid for visualization
```

```
def normalize(x, axis=0):
    return sklearn.preprocessing.minmax_scale(x, axis=axis)
# Plot the waveplot and spectral centroids
plt.figure(figsize=(15, 5))
# Plot the waveplot
plt.subplot(1, 1, 1)

librosa.display.waveshow(y, sr=sr, alpha=0.4)

# Plot the spectral centroids
plt.plot(f_times, normalize(spectral_centroids), color='black', label='Spectral Centroids')

plt.ylabel('Hz')
plt.xticks([])
plt.legend()
plt.title('Waveplot with Spectral Centroids')
plt.show()
```



4)Spectral contrast

Octave-based Spectral Contrast (OSC) was developed to represent the spectral characteristics of a musical piece. It considers the spectral peak and valley in each sub-band separately.

In general, spectral peaks correspond to harmonic components and spectral valleys correspond to non-harmonic components or noise in a music piece. Therefore, the difference between spectral peaks and spectral valleys will reflect the spectral contrast distribution.

Applications:

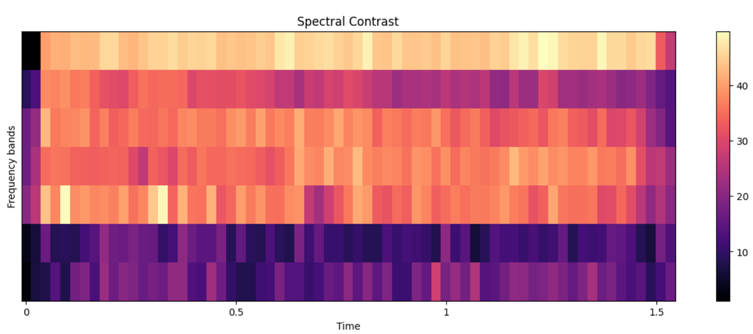
Audio Scene Analysis: Spectral contrast contributes to audio scene analysis by helping to differentiate between sounds in a scene. This is useful in applications like automatic scene recognition, where understanding the acoustic characteristics of different environments is essential.

Music Information Retrieval: Spectral contrast is utilized in music information retrieval tasks, such as content-based audio retrieval. By capturing the differences in spectral characteristics, it helps in matching and retrieving audio content based on timbral features.

Audio Quality Assessment: Spectral contrast is employed in audio quality assessment to evaluate the perceived quality of audio signals. It can be used to analyze the tonal balance and timbral characteristics of audio recordings, providing insights into factors that may affect the perceived quality of the sound.

Code:

```
# Assuming y_harmonic is the harmonic component of your audio signal
y_harmonic, _ = librosa.effects.hpss(y)
# Compute spectral contrast
contrast = librosa.feature.spectral_contrast(y=y_harmonic, sr=sr)
# Plot the spectral contrast
plt.figure(figsize=(15, 5))
librosa.display.specshow(contrast, x_axis='time', sr=sr)
plt.colorbar()
plt.ylabel('Frequency bands')
plt.title('Spectral Contrast')
plt.xlabel('Time')
plt.show()
```



5)Spectral Rolloff

Spectral rolloff point is defined as the Nth percentile frequency of the power spectral distribution, where N is usually 85% or 95%. The rolloff point is the frequency below which the N% of the magnitude distribution is concentrated.

$$\underset{f_c \in \{1,\ldots,N\}}{\arg\min} \sum_{i=1}^{f_c} m_i \geq 0.85 \cdot \sum_{i=1}^{N} m_i$$

Where m_i is the magnitude of the ith frequency component of the spectrum and f_c is the rolloff frequency.

Applications:

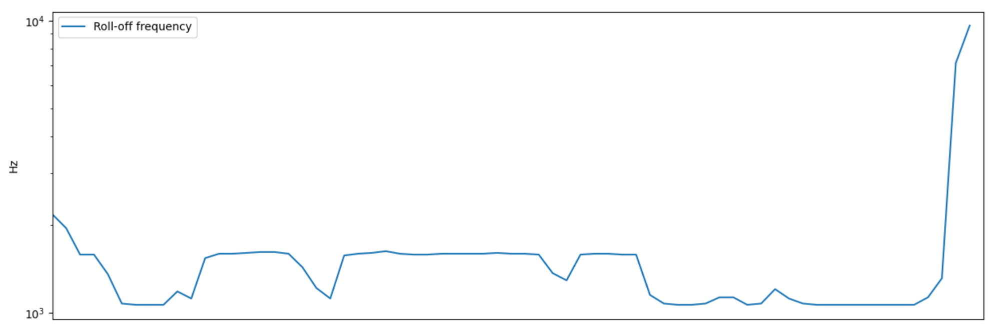
Speech and Speaker identification: Rolloff frequency can be used in speech processing and speaker recognition systems. It helps in distinguishing between different speakers and speech patterns based on the distribution of energy in the high-frequency components of the audio signal.

Detection of Anomalies or Events: Changes in the spectral rolloff frequency can indicate anomalies or events in audio signals. Sudden shifts in rolloff frequency might be indicative of certain sound events, making it useful in applications like audio surveillance, intrusion detection, or abnormal sound detection.

Audio Compression: Rolloff frequency information can be considered in audio compression algorithms. Understanding the spectral rolloff can help in designing compression strategies that prioritize or allocate bits more efficiently, particularly in the high-frequency regions where the human auditory system may be less sensitive.

Code:

```
rolloff = librosa.feature.spectral_rolloff(y=y, sr=sr)

plt.figure(figsize=(15,5))
plt.semilogy(rolloff.T, label='Roll-off frequency')
plt.ylabel('Hz')
plt.xticks([])
plt.xlim([0, rolloff.shape[-1]])
plt.legend()
```



6)Mel-frequency cepstral coefficients (MFCCs)

Mel-frequency cepstral coefficients represent the short-time power an audio clip based on the discrete cosine transform of the log power spectrum on a non-linear mel scale. The difference between a cepstrum and the mel-frequency cepstrum (MFC) is that in the MFC, the frequency bands are equally spaced on the mel scale to more closely resemble the human auditory system's response as opposed to the linearly-spaced bands in the normal spectrum. This frequency warping allows for better representation of sound and is especially useful in audio compression.

Applications:

Audio Content Based Retrieval: MFCCs are used in content-based audio retrieval systems where users search for audio content based on its acoustic features. By representing the spectral content of audio clips, MFCCs help in matching and retrieving similar sounds or music.

Environmental Sound Analysis: MFCCs can be applied to the analysis of environmental sounds. They are used to characterize and classify sounds in the environment, facilitating applications such as audio scene recognition, surveillance, and detection of specific events.

Noise Reduction and Speech Enhancement: In noise reduction and speech enhancement applications, MFCCs are used to analyze and distinguish between the speech and noise components in an audio signal. This information helps in designing effective algorithms for reducing noise and enhancing the intelligibility of speech.

Code:

```
mfccs = librosa.feature.mfcc(y=y_harmonic, sr=sr, n_mfcc=13)

plt.figure(figsize=(15, 5))
librosa.display.specshow(mfccs, x_axis='time')
plt.colorbar()
plt.title('MFCCs')
```

Text(0.5, 1.0, 'MFCCs')