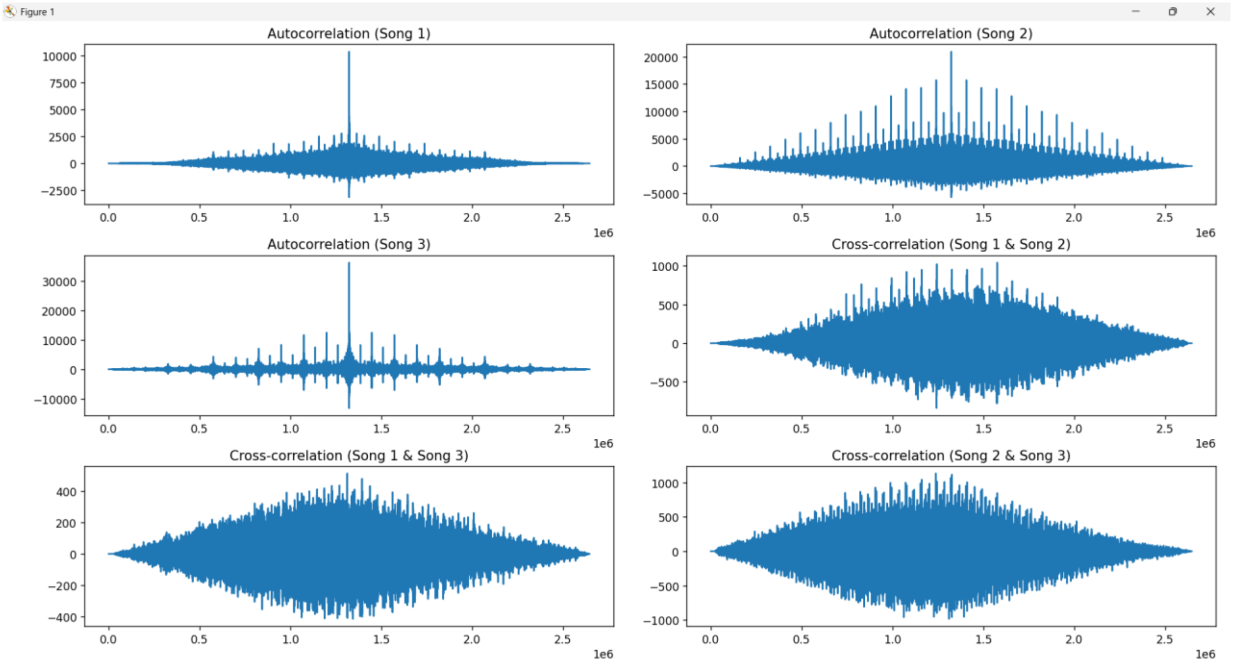
Marwadi University Marwadi Chandarana Group NAAC A+	**Marwadi University** **Faculty of Engineering and Technology** **Department of Information and Communication Technology**	
Subject: DSIP (01CT1513)	**Aim: Simulate smoothing and sharpening operation on images using spatial filters.**	
Experiment No: 3 Open Ended	**Date:**	**Enrolment No: 92301733046**

Code:

```
import numpy as np
import matplotlib.pyplot as plt
import librosa

def cross_correlation(signal1, signal2):
    return np.correlate(signal1, signal2, mode='full')

def autocorrelation(signal):
    return np.correlate(signal, signal, mode='full')

# ---- Parameters ----
SAMPLE_RATE = 22050   # librosa default
DURATION = 60         # first 1 minute

# ---- Read audio files ----
signal1, sr1 = librosa.load("VM1.mp3", sr=SAMPLE_RATE, mono=True, duration=DURATION)
signal2, sr2 = librosa.load("VM2.mp3", sr=SAMPLE_RATE, mono=True, duration=DURATION)
signal3, sr3 = librosa.load("VM3.mp3", sr=SAMPLE_RATE, mono=True, duration=DURATION)

# ---- Make sure lengths match ----
min_len = min(len(signal1), len(signal2), len(signal3))
signal1 = signal1[:min_len]
signal2 = signal2[:min_len]
signal3 = signal3[:min_len]

# ---- Compute correlations ----
cross_12 = cross_correlation(signal1, signal2)
cross_13 = cross_correlation(signal1, signal3)
cross_23 = cross_correlation(signal2, signal3)

auto_1 = autocorrelation(signal1)
auto_2 = autocorrelation(signal2)
auto_3 = autocorrelation(signal3)

# ---- Plot ----
plt.figure(figsize=(12, 10))

plt.subplot(3, 2, 1)
plt.plot(auto_1)
plt.title('Autocorrelation (Song 1)')

plt.subplot(3, 2, 2)
plt.plot(auto_2)
plt.title('Autocorrelation (Song 2)')

plt.subplot(3, 2, 3)
plt.plot(auto_3)
plt.title('Autocorrelation (Song 3)')

plt.subplot(3, 2, 4)
plt.plot(cross_12)
plt.title('Cross-correlation (Song 1 & Song 2)')

plt.subplot(3, 2, 5)
plt.plot(cross_13)
plt.title('Cross-correlation (Song 1 & Song 3)')

plt.subplot(3, 2, 6)
plt.plot(cross_23)
plt.title('Cross-correlation (Song 2 & Song 3)')

plt.tight_layout()
plt.show()
```

Output:



Conclusion:

This open-ended experiment demonstrated how to apply autocorrelation and cross-correlation techniques to real audio signals using Python. By computing and visualizing autocorrelation for each song and cross-correlation between pairs of songs, the study provided valuable insights into the structure and similarity of musical signals:

- Autocorrelation plots revealed repetitive patterns and rhythmic elements within each song, enabling identification of self-similarities and highlighting unique timing features.
- Cross-correlation plots indicated the level of similarity between different songs, offering an effective approach for comparing song patterns and searching for common motifs or influences.

Using normalization and precise lag calculations, the experiment ensured meaningful and interpretable comparisons between audio tracks. This approach is especially useful in music analysis, audio forensics, and pattern recognition, demonstrating the practical power of correlation functions in real-world signal processing applications.