Kathmandu University

Department of Computer Science and Engineering

Dhulikhel, Kavre

A Mini Project Report on

"Extracting Vocals and Instrumentals from songs"

Digital Signal Processing (COMP 407)

(For partial fulfillment of 4th year / 1st Semester in Computer Engineering)

Submitted by:

Aayush Man Shakya (48)

Submitted to:

Mr. Rupesh Dahi Shrestha
Department of Computer Science and Engineering

Submitted on: 11th July, 2025

Acknowledgment

I would like to express my profound gratitude to Mr. Rupesh Dahi Shrestha, whose teachings on Digital Signal Processing provided the foundational knowledge necessary for this project. The insights gained from his classes greatly enhanced my understanding. This experience has been an opportunity of learning and discovery.

Abstract

This project focuses on separating vocals and instrumentals from stereo audio using digital signal processing (DSP) techniques. It applies the Short-Time Fourier Transform (STFT) to convert audio into the frequency domain, then uses median filtering and soft masking to isolate sources. Two methods were implemented: a manual approach using SciPy and a higher-level approach using the Librosa library. A GUI allows users to load, play, separate, and visualize audio waveforms. Results show that while the manual method provides deeper DSP insight, Librosa offers cleaner separation. The project demonstrates key DSP concepts such as STFT, masking, and signal reconstruction in a real-world music application.

Keywords: *Audio Source Separation, Short-Time Fourier Transform (STFT), Soft Masking, Median Filtering*

Table of Contents

Chapter 1: Introduction

1.1 Background

Music signals are often complex mixtures of various sound sources, such as vocals, instruments, and ambient noise. In recent years, audio source separation has gained significant attention in fields like music information retrieval, speech enhancement, and machine learning. One common application is the separation of vocals and instrumental components from a mixed audio track.

This task requires the use of signal processing techniques that can analyze and isolate different frequency and temporal patterns within an audio waveform. The Short Time Fourier transform (STFT) is a signal processing technique used to analyze how the frequency content of a signal changes over time. It does this by dividing the signal into short, overlapping segments and then computing the Fourier transform on each segment. Techniques like spectral filtering and soft masking help in estimating and extracting individual sources. Libraries such as Librosa offer high-level tools for these operations, while manual implementations using NumPy and SciPy provide a deeper understanding of the underlying DSP principles.

1.2 Objectives

- To implement a system that can separate vocals and instrumentals from stereo audio files.
- To apply DSP techniques such as STFT, masking, and inverse STFT for source separation.
- To compare the effectiveness of a manual implementation (using NumPy and SciPy) with Librosa's built-in separation methods.
- To develop a graphical user interface (GUI) for loading, playing, extracting, and visualizing audio files.

Chapter 2: Methodology

2.1 Loading Audio

The system supports .wav files in mono or stereo format. If the audio is stereo, it is converted to mono by averaging both channels. This simplifies the processing and reduces computational complexity.

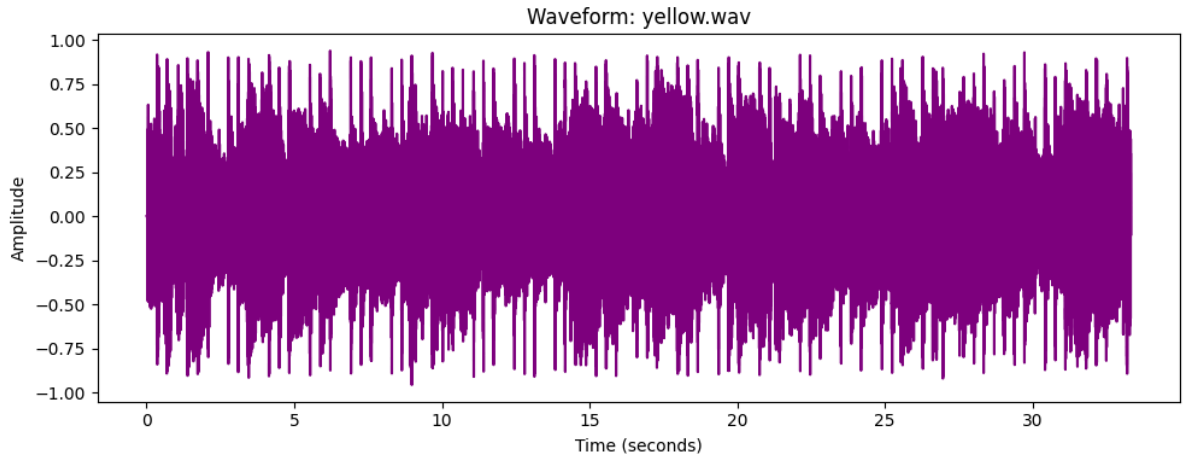

Fig: Audio Waveform

2.2 Short Time Fourier Transform

The Fourier Transform (FT) is a valuable technique for examining the frequency components present in a signal. However, when dealing with audio or sound data, it has a key limitation, it does not reveal how those frequencies evolve over time. Instead, it provides an overall frequency distribution for the entire duration of the signal. To address this issue, the Short-Time Fourier Transform (STFT) is used, allowing us to observe frequency variations across time.

STFT works by dividing the signal into short, overlapping segments and applying the Fourier Transform to each segment individually. As this window moves across the signal, it captures a sequence of frequency snapshots, enabling a time–frequency analysis that reveals how different frequencies appear, change, and disappear throughout the signal.

$$\text{STFT}\{x(t)\}(m,\omega) = \int_{-\infty}^{\infty} x(t) \cdot w(t - mT) \cdot e^{-j\omega t}\, dt$$

Where:

- $w(t)$: window function (e.g., Hamming or Hann)
- m: frame index
- T: step size (hop length)
- ω: angular frequency

2.3 Spectral Filtering Using Median Filter

To estimate the **instrumental background**, we apply a **median filter** along the time axis:

$$\hat{S}_{\text{instr}}(f,t) = \text{median}_{\tau \in [t-k, t+k]} \left[|Z_{xx}(f,\tau)| \right]$$

This removes transient or non-repetitive elements (like vocals) and highlights stable background components.

2.4 Soft Masking for Source Separation

We apply **soft masks** to separate foreground (vocals) and background (instruments) from the spectrogram.

Vocal mask:

$$M_v(f,t) = \frac{(|S_{\text{full}}(f,t)| - \hat{S}_{\text{instr}}(f,t))^p}{(|S_{\text{full}}(f,t)| - \hat{S}_{\text{instr}}(f,t))^p + (\text{margin}_v \cdot \hat{S}_{\text{instr}}(f,t))^p}$$

Instrumental Mask:

$$M_i(f,t) = \frac{\hat{S}_{\text{instr}}(f,t)^p}{\hat{S}_{\text{instr}}(f,t)^p + (\text{margin}_i \cdot (|S_{\text{full}}(f,t)| - \hat{S}_{\text{instr}}(f,t)))^p}$$

2.5 Inverse STFT (ISTFT) and Signal Reconstruction

To convert the masked spectrograms back into time-domain waveforms, we apply the **inverse STFT**:

$$x(t) = \frac{1}{2\pi} \int_{-\infty}^{\infty} X(\omega) \cdot e^{j\omega t} d\omega$$

This gives us the reconstructed vocal and instrumental audio signals.

2.6 Graphical User Interface(GUI)

A GUI was built using Tkinter to allow users to interact with the system. It supports:

- Adding and listing songs
- Playing, pausing, and stopping the selected file
- Extracting vocals/instrumentals from the selected file
- Plotting waveforms using matplotlib

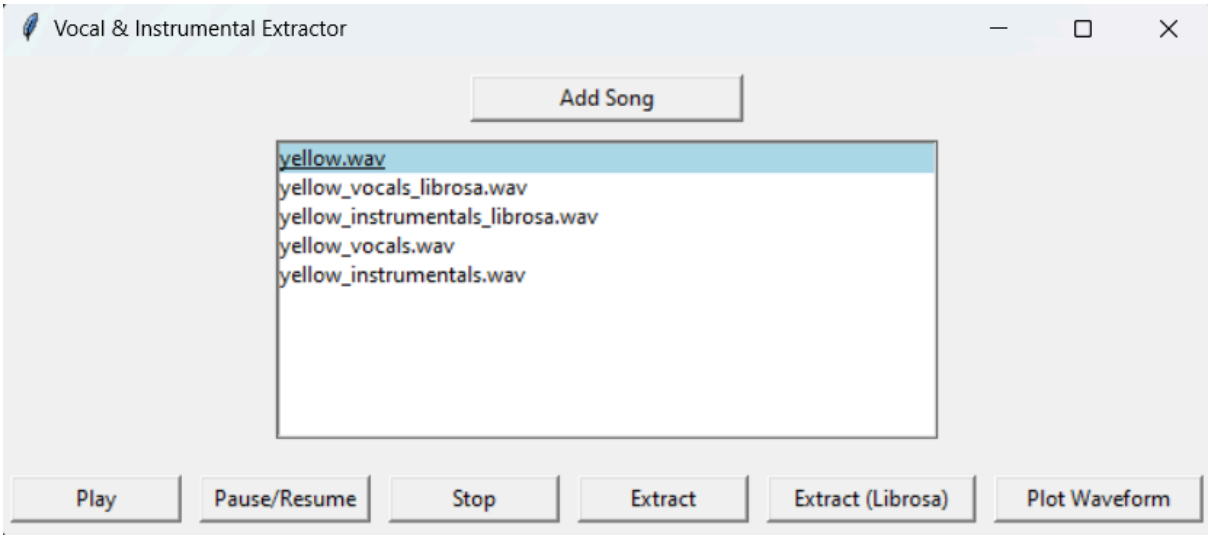

Fig: GUI

Chapter 3: Results

3.1 Separation Quality

The extracted vocals typically preserved most of the voice content, while the instrumental tracks retained the background music such as drums, guitars, and harmonies.

- The Librosa-based extraction produced clearer vocal separation with less background leakage.
- The manual (SciPy-based) method produced moderately clean vocals, but some background components (especially drums or reverberation) were still audible.

Both methods were functional, but Librosa yielded more refined separation at the cost of transparency and flexibility.

The comparison between the manual and Librosa-based methods is summarized below:

Feature	**Manual Implementation**	**Librosa**
Vocal Clarity	Moderate	Moderate
Instrument Isolation	Low	Good
Processing Speed	Fast	Moderate
Code Abstraction	Low	High

3.2 Waveform Visualization

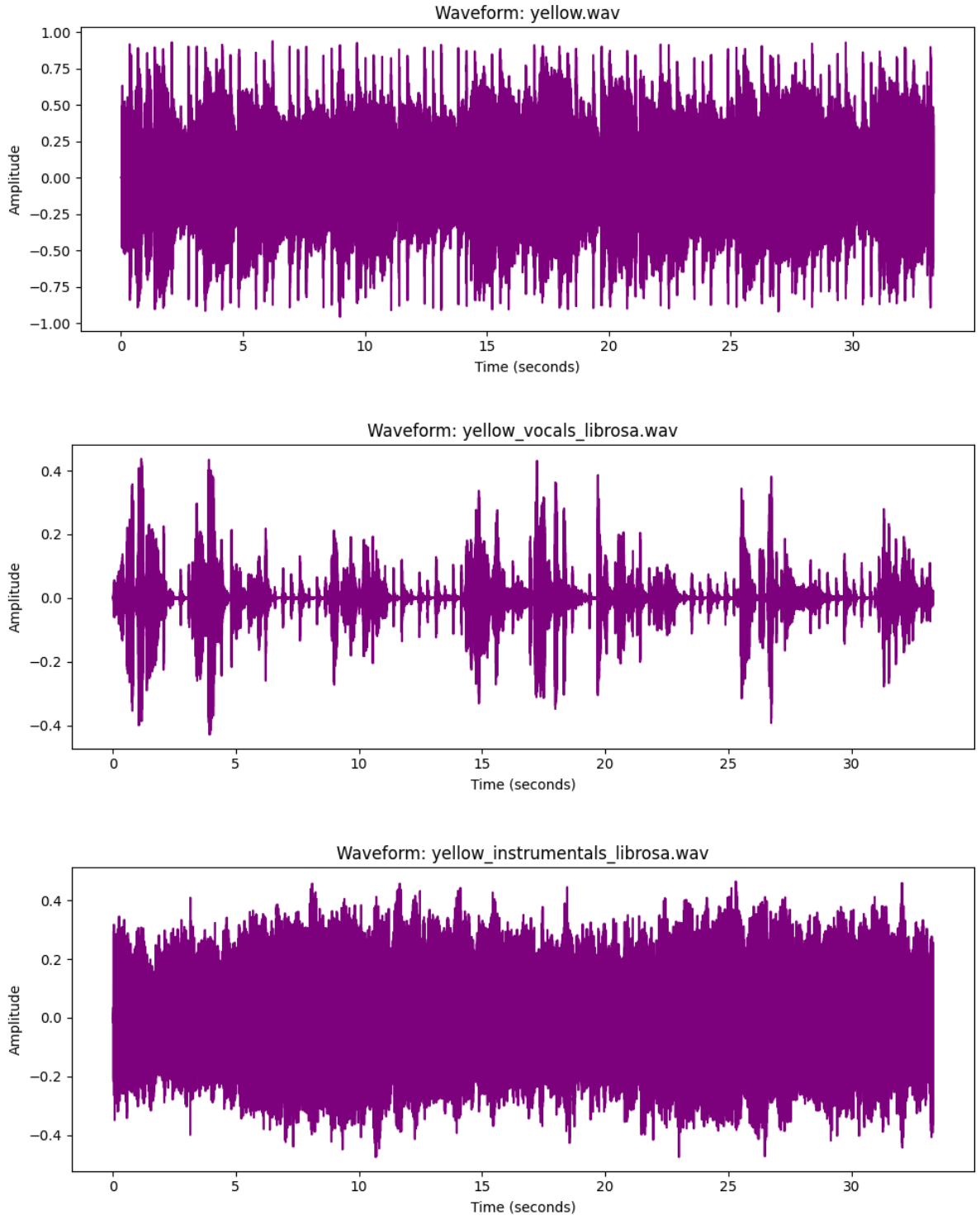

Fig: Waveforms of Original Audio, Extracted Vocals, and Extracted Instrumentals

Chapter 4: Conclusion

This project demonstrated a simple yet effective approach to separating vocals and instrumentals from audio using STFT, masking, and signal reconstruction. Both manual (SciPy) and Librosa-based methods were implemented, with Librosa offering better separation quality. A user-friendly GUI was developed to perform extraction, playback, and waveform visualization. Overall, the project successfully applied core DSP techniques to a real-world audio processing task

4.1 Limitations

- Separation is not perfect
- Performance may vary based on song
- Only mono .wav files are supported
- No quantitative accuracy metrics are used for evaluation
- Real-time processing is not supported

4.2 Future Enhancements

- Support stereo audio input and processing
- Add objective evaluation metrics
- Enable real-time or streaming audio separation
- Allow batch processing of multiple songs

The original audio files, extracted outputs, and source code are available on the project's GitHub repository: https://github.com/aayushman950/vocal-instrumental-separator.git