Project Report

EE-200

Frequency Mixer and Demixer

Course Project

Submitted by

Aviral Gupta

Instructor:

Dr. Tushar Sandhan

Date of Submission: 30/06/25

Introduction

1. Fourier Transform

The Fourier transform provides a mathematical framework for analysing signals or images in terms of their frequency content. By converting an image from its original spatial representation to a frequency-domain representation, we gain insight into the distribution of different spatial frequencies, such as edges, textures, and smooth gradients.
In general, the two-dimensional discrete Fourier transform (2D DFT) of an image $I(x, y)$ of size $M \times N$ is given by:

$$F(u, v) = \sum_{x=0}^{M-1} \sum_{y=0}^{N-1} I(x, y)\, e^{-j2\pi\left(\frac{ux}{M} + \frac{vy}{N}\right)}$$

The inverse Fourier transform, used to reconstruct the image from its frequency-domain representation, is defined as:

$$I(x, y) = \frac{1}{MN} \sum_{u=0}^{M-1} \sum_{v=0}^{N-1} F(u, v)\, e^{+j2\pi\left(\frac{ux}{M} + \frac{vy}{N}\right)}$$

The resulting frequency-domain representation $F(u, v)$ is complex-valued, where:

- **Magnitude** $|F(u, v)|$ describes the strength of each frequency component.
- **Phase** $\angle F(u, v)$ captures the positional information or spatial arrangement.

Typically, the magnitude spectrum is visualised to identify significant spatial frequencies present in the image, informing subsequent image processing steps such as filtering or fusion.

2. What is FFT and how it helps

The Fast Fourier Transform (FFT) is a computationally efficient algorithm to calculate the Discrete Fourier Transform (DFT), significantly reducing the complexity from $O(N^2)$ to $O(N \log N)$. The FFT converts a signal or image from its original domain (time or space) into the frequency domain, decomposing it into sinusoidal components characterised by amplitude and phase.
Using FFT helps by:

- identifying frequency components within an image.
- Facilitating effective filtering by allowing easy manipulation of specific frequency bands.
- Accelerating computations, making frequency-domain analyses practical for large images.
- Enabling straightforward reconstruction from frequency to spatial domains via inverse FFT.

In our project, the FFT allowed precise isolation and manipulation of image frequency bands, enabling the effective fusion of complementary image details and structures.

3. Magnitude Spectra

The magnitude spectrum of an image, defined as the absolute value of its Fourier transform, reveals the spatial frequency content present within the image. Formally, the magnitude spectrum is computed as:

$$|F(u, v)| = \sqrt{\text{Re}(F(u, v))^2 + \text{Im}(F(u, v))^2}$$

Where $\text{Re}(F(u, v))$ and $\text{Im}(F(u, v))$ represent the real and imaginary parts of the Fourier transform, respectively.

When attempting to visualise this magnitude spectrum directly on a linear scale, the resulting image typically appears almost entirely black. This occurs because the spectrum values span an extremely wide dynamic range, with a few dominant low-frequency components having significantly larger magnitudes than the high-frequency ones. Consequently, smaller but still important magnitude values become indistinguishable after normalisation.

To address this issue, a logarithmic (dB) scale is typically employed, which compresses the dynamic range and enhances visibility across all frequency magnitudes:

$$|F(u,v)|_{\text{dB}} = 20\ \log_{10}\left(|F(u,v)| + \epsilon\right)$$

Here, the small constant ϵ prevents logarithmic singularities. This representation makes it possible to clearly visualize both dominant and subtle frequency features simultaneously.
Interpreting magnitude spectra thus allows precise analysis and manipulation of frequency-domain components, which is critical to the successful implementation of our frequency mixer.

In our case, we used $\epsilon = 10^{-5}$ so that it does not affect our plot and solve the error.

Transfer Functions

In signal processing and system analysis, a **transfer function** represents the mathematical relationship between a system's input and output in the frequency domain. It is typically expressed as a ratio of two polynomials in the complex frequency variable s (for analog systems) or z (for digital systems). The transfer function characterizes how different frequency components of an input signal are *amplified, attenuated, or phase-shifted* as they pass through the system.

It plays a crucial role in analyzing and designing filters, as it provides insights into the system's **frequency response**, **stability**, and **dynamic behavior**. By studying the *poles* and *zeros* of a transfer function, one can predict how the system will respond to various inputs, making it a foundational concept in both control systems and digital signal processing.

Q1: Frequency mixer: 'Beauty and the Blur'

1. Introduction

This exercise explores the principles of spatial-frequency analysis by implementing a frequency-domain image mixer. Inspired by hybrid images such as the classic Einstein–Monroe illusion, this method combines the fine details from one image (high frequencies) with the broad, smooth structural elements from another (low frequencies). The resulting fused image illustrates how our visual perception interprets different frequency bands differently, emphasising coarse structure at a distance and fine detail upon closer viewing. The experiment will involve spatial pre-processing, frequency-domain masking, and inverse frequency transformation.

2. Objective

The objective is to design and implement a frequency mixer that demonstrates the separation and recombination of spatial-frequency components. Specifically, this includes:

- Applying a Gaussian blur in the spatial domain to selectively suppress high-frequency content.
- Computing the 2D Fourier transforms (FFTs) of two complementary images.
- Designing and applying frequency-domain masks (low-pass and high-pass) to isolate specific frequency bands.
- Combining masked frequency components and reconstructing the resulting image via the inverse FFT.

This procedure highlights the relationship between spatial-frequency content and visual perception, providing insight into frequency-based image processing methods.

3. Libraries Used

The implementation of the frequency mixer and Fourier analysis was carried out using Python in a Jupyter Notebook environment. The following Python libraries were used:

- **NumPy**: Used for efficient numerical computations and array manipulation, including Fourier transforms via `numpy.fft`.
- **Matplotlib**: Employed for plotting and visualizing images, spectra, and graphs throughout the analysis.
- **OpenCV (cv2)**: Utilized for reading and preprocessing images, applying Gaussian blur, and handling image resizing and rotation.

These libraries provided a flexible and powerful framework to perform Fourier-based frequency decomposition, design and apply frequency masks, and reconstruct fused images efficiently.

4. Input Images

Input Images We were provided with two grayscale input images:

- `cat_gray.png` – a grayscale image of a cat, containing fine texture details and sharper features.
- `dog_gray.png` – a grayscale image of a dog, exhibiting smoother textures and broader structural patterns.

These two images were used as the base inputs for frequency decomposition and fusion. The objective was to extract high-frequency details from the cat image and low-frequency structure from the dog image, and then combine them into a single perceptually meaningful output using frequency-domain techniques.

Figure 1: Cat Image (High-Frequency)

Figure 2: Dog Image (Low-Frequency)

5. Fourier Transform and Magnitude Spectra

The first step involved computing the 2D Discrete Fourier Transform (DFT) of both input images using the `fft2` function from the `numpy.fft` module. This function efficiently converts spatial-domain images into their corresponding frequency-domain representations.
We then computed and visualized the magnitude spectra of the transformed images. Two types of magnitude visualizations were generated:

- **Linear Magnitude Spectrum:** The raw magnitude $|F(u, v)|$ was normalized and plotted. However, due to the extreme dynamic range in the frequency data, the linear plot appeared completely black. This occurred because a small number of frequency components had very large magnitudes (on the

order of 10^7), while the majority were much smaller—causing the normalization to suppress almost all visible contrast.

To diagnose the issue, we clipped the magnitude values at the 99th percentile before normalization. This revealed that the few high-valued components were located in the corners of the spectrum, corresponding to the DC and low-frequency components in the unshifted FFT.

- **Logarithmic (dB) Magnitude Spectrum:** To address the visualization challenge, we plotted the magnitude on a logarithmic scale using:

$$|F(u,v)|_{\text{dB}} = 20\log_{10}(|F(u,v)| + \epsilon)$$

where ϵ is a small constant to avoid logarithmic singularities. This transformation compressed the dynamic range, making both dominant and subtle frequency features visible. The dB spectrum confirmed that the most significant energy was concentrated near the corners.

These observations were instrumental in understanding the frequency behavior of each image and guided the design of masks for the fusion process.

6. Spectrum Center and Frequency Shifting

By default, the output of the 2D Fourier Transform (`fft2` from NumPy) places the lowest frequency (DC component) at the top-left corner of the spectrum. This layout is not intuitive for visual analysis, as the important low-frequency information is not centrally located.
To address this, we applied the `fftshift` function from `numpy.fft`, which reorders the spectrum so that the zero-frequency component is moved to the center of the image. This operation shifts the quadrants of the frequency domain image, resulting in a more interpretable visualization:

- The center of the shifted spectrum now represents low frequencies (coarse structures).
- Frequencies increase radially outward, with high frequencies (edges and fine details) located near the corners.

Observations:

- After applying `fftshift`, the concentration of energy in the center of the spectrum became clearly visible.
- The visual structure of the magnitude spectrum aligned better with human intuition, showing smoother gradients at the center and sharper transitions outward.
- This visualization was instrumental in designing circular masks for filtering specific frequency ranges and confirmed that our fusion strategy would effectively isolate desired features.

All subsequent magnitude spectra were plotted with `fftshift` applied, ensuring that our frequency-domain operations aligned with the spatial characteristics of the images.

7. Effect of Image Rotation on Fourier Spectrum

To further analyze the behavior of frequency-domain representations, we rotated one of the input images (cat or dog) by 90° counter-clockwise in the spatial domain using OpenCV's `cv2.rotate` function. We then computed and plotted its 2D Fourier magnitude spectrum using the same procedure: applying `fft2`, `fftshift`, and logarithmic scaling.
Observations:

- **Identical Spectral Shape:** The overall structure of the magnitude spectrum—characterized by a bright central region and concentric lobes—remains unchanged after rotation. This indicates that the distribution of frequency magnitudes is preserved.
- **90° Rotation Match:** Every pattern or feature visible in the spectrum of the rotated image appears rotated by exactly 90° counter-clockwise compared to the original. This confirms that the frequency domain undergoes the same geometric transformation as the spatial domain.

- **Spatial–Frequency Correspondence:** The experiment validates a key property of the 2D Fourier Transform: a 90° counter-clockwise rotation in the spatial domain results in a corresponding 90° counter-clockwise rotation in the frequency domain, without altering magnitude values.

These observations reinforce the geometric consistency of the Fourier Transform and aid in understanding how spatial transformations reflect in frequency space—an essential insight for designing frequency-selective filters and image fusion systems.

8. Frequency Mixer: Combining Structural and Textural Features

The frequency mixer is the central component of our project. Its objective is to combine two images in such a way that one contributes the low-frequency (structural) content, while the other contributes the high-frequency (textural) detail. This mimics how the human visual system interprets multi-scale information — using coarse structures for overall shape recognition and fine textures for surface details.
In our implementation:

- The **dog image** (`dog_gray.png`) was used as the source of **low-frequency content**, representing broader shapes and structures.
- The **cat image** (`cat_gray.png`) was used as the source of **high-frequency content**, representing edges and fine textures.

To achieve this, we performed the following steps:

1. Computed the 2D FFT of both images and applied `fftshift` to center the frequency spectra.
2. Designed ideal circular frequency masks — a low-pass mask for the dog image and a high-pass mask for the cat image.
3. Optionally applied Gaussian blurring with $\sigma = 6$ in the spatial domain to enhance low-frequency dominance before transformation.
4. Multiplied each spectrum with the respective mask and summed the filtered spectra.
5. Applied inverse FFT to reconstruct the fused image.

This fusion method produces a hybrid image that appears differently depending on viewing distance — detailed textures from the cat dominate when viewed closely, while broader shapes from the dog emerge from afar. This effect highlights the power of frequency-based image manipulation.

9. Transfer Function Design and 2D Plots

The core of the frequency mixing system lies in the design of appropriate 2D frequency-domain transfer functions — specifically, binary masks that control which parts of each image's frequency spectrum contribute to the final output.

1. Transfer Functions Used

We designed two complementary transfer functions:

- **Low-Pass Mask (Dog Image):** A circular binary mask that passes only the low-frequency components (i.e., values near the center of the shifted spectrum) and blocks the high-frequency details.
- **High-Pass Mask (Cat Image):** A binary mask that is the complement of the low-pass mask. It passes high-frequency components (i.e., edges and fine textures) and blocks the structural, low-frequency content.

These transfer functions were applied to the Fourier-transformed images using element-wise multiplication. This frequency-selective filtering allowed us to retain smooth structures from the dog image and sharp details from the cat image.

2. Implementation Details

- The images were of the same size, say $N \times N$.
- After computing the 2D FFT and applying `fftshift`, we generated the masks centered at $(N/2, N/2)$.
- A circular region of radius r (empirically chosen, e.g., $r = 13$) was used to define the low-pass region.
- The high-pass mask was obtained as $1 -$ low-pass mask.

3. 2D Transfer Function Plot

The visual appearance of the transfer function is shown below:

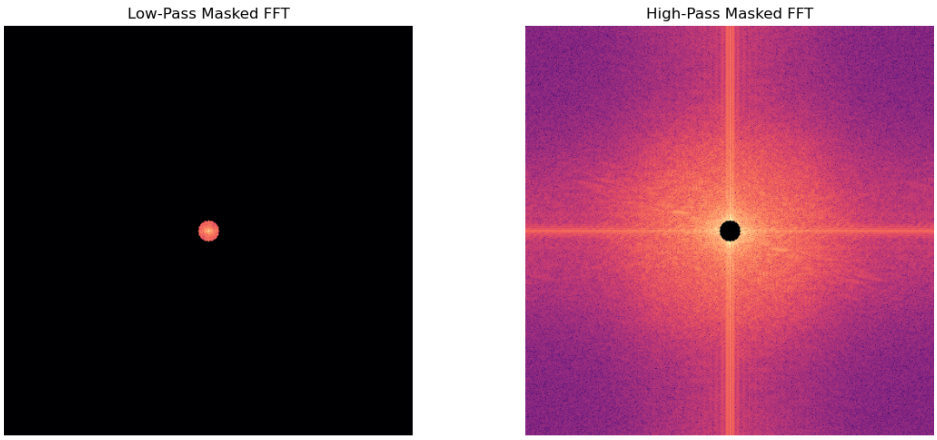

Figure 3: Result of Transfer function on masking

This 2D binary mask act as frequency selector and form the functional core of the mixer system. Their design directly controls the spatial features preserved in the final hybrid image.

Figure 4: Fused Image: Combination of Dog (Low-Frequency) and Cat (High-Frequency) Features

Q2. Frequency de-mixer: 'Unwanted Solo'

1. Introduction

Digital Signal Processing (DSP) plays a fundamental role in the analysis and modification of signals such as audio, images, and sensor data. In the context of audio processing, DSP techniques allow us to emphasize, suppress, or isolate specific frequency components to enhance sound quality, remove noise, or extract meaningful information.
This experiment aims to investigate the impact of various filter design approaches on an audio signal. Both time-domain and frequency-domain representations are examined to interpret the impact of filtering. Interactive tools are used to dynamically explore parameter changes, allowing for real-time observation of their influence. The experiment culminates with a phase-reversed subtraction technique, demonstrating how specific frequency bands can be selectively attenuated from the original audio.

2. Objective

The objective of this experiment is to explore the fundamentals of digital signal processing by analysing an audio signal in both time and frequency domains. The task involves designing and applying digital filters using different design methodologies to manipulate specific frequency components within the signal. The experiment also aims to visualize the system behavior through frequency responses, power spectral density plots, and pole-zero representations.

3. Noise and Its Removal in Signal Processing

In signal processing, **noise** refers to any unwanted or irrelevant component that contaminates the original signal and degrades its quality or intelligibility. Noise can originate from various sources such as electronic interference, environmental disturbances, or limitations in recording equipment. In the frequency domain, noise often appears as additional energy spread across a broad range of frequencies, particularly in regions where the true signal has low or no activity.

To remove or reduce noise, filtering techniques are employed. Filters are designed to pass frequency components that belong to the desired signal while attenuating those associated with noise. For instance, if noise primarily occupies higher frequencies, a low-pass filter can be used to suppress it. Conversely, if noise is localized in a narrow band, a band-stop filter may be more effective.

4. Libraries Required

The following Python libraries were used to implement the signal processing operations, visualization, and interactive user interface for the experiment:

- **NumPy**: For numerical operations and array manipulations.
- **SciPy.signal**: For designing and applying digital filters, computing frequency responses, power spectral density, and pole-zero plots.
- **Matplotlib**: For visualizing time-domain waveforms, frequency responses, and spectrograms.
- **IPython.display.Audio**: For audio playback of original and filtered signals within the notebook.
- **librosa**: For loading and preprocessing audio files (e.g., reading audio, converting stereo to mono).
- **ipywidgets**: For creating interactive user controls such as sliders and dropdown menus to dynamically update filter parameters.

5. Preprocessing and Initial Analysis

The experiment began by loading an audio signal using the `librosa` library. If the input signal was in stereo format, it was converted to mono by averaging the channels to simplify further processing.

Next, the signal was analyzed in the time domain by plotting its waveform. This provided insight into the amplitude variations over time. The corresponding frequency domain representation was obtained by computing and plotting the magnitude spectrum using the Fourier Transform.

To visualize how the frequency content of the signal evolves over time, a spectrogram was also generated. The spectrogram offered a time-frequency representation that helped identify dominant frequency bands and regions potentially affected by noise.

These initial visualisations formed the foundation for designing appropriate filters to manipulate or remove specific frequency components from the signal.

6. Observations from Initial Analysis

- The time-domain waveform revealed clear amplitude variations, indicating segments of high and low energy throughout the signal.
- The magnitude spectrum showed that the dominant frequency components were concentrated in the ranges of approximately **100–400 Hz**, **around 1000 Hz**, and a prominent band between **1500–2500 Hz**.
- The spectrogram provided a detailed time-frequency representation, highlighting when and for how long different frequency bands were active. Notably, an additional frequency component was visible in the **1024–2048 Hz** band, possibly indicating structured noise or a distinct tonal element.
- Regions with low amplitude and high-frequency content in the spectrogram appeared in yellow, suggesting transient or low-energy noise.
- These insights guided the design of filters to either attenuate or isolate specific frequency bands for further processing and analysis.

7. Filter Design and Implementation

Following the initial analysis of the audio signal, targeted filtering was required to suppress undesired components while preserving the quality of the underlying signal. To achieve this, a flexible digital filter framework was developed using Python, incorporating both FIR (Finite Impulse Response) and IIR (Infinite Impulse Response) designs.

Filter Types and Structure

Two core approaches to digital filter design were explored:

- **Finite Impulse Response (FIR) Filters:** These filters were constructed using the window method with Blackman windows to ensure effective sidelobe attenuation. FIR filters offer the benefit of a strictly linear phase response, which is particularly desirable for audio processing where phase distortions must be minimized.
- **Infinite Impulse Response (IIR) Filters:** Butterworth filters were chosen for IIR design due to their smooth and monotonic frequency response in both passband and stopband. IIR filters provide more compact implementations for the same level of attenuation but introduce non-linear phase shifts.

Each filter was designed to suppress or isolate specific frequency bands identified through spectral analysis. To avoid phase distortion, zero-phase filtering was achieved using the `filtfilt()` function from the `scipy.signal` module, which applies forward and reverse filtering.

Interactive User Interface

To provide real-time control over the filtering process, interactive widgets were created using the `ipywidgets` library. These sliders and dropdown menus allowed dynamic adjustment of:

- Cutoff frequencies (`low_cut`, `high_cut`)

- Filter order (for IIR)
- Filter method (FIR or IIR)

Each user interaction triggered a re-computation of the filter, application to the audio signal, and generation of updated plots and audio outputs. This allowed immediate feedback and iterative tuning of the filter characteristics.

Visual and Auditory Feedback

For each filtering configuration, the following analyses and visualizations were generated:

- **Frequency Response:** Gain vs frequency plots illustrated the passbands and stopbands of the filter.
- **Time-Domain Signals:** Comparison of original and filtered waveforms provided insight into amplitude changes and signal shaping.
- **Power Spectral Density (PSD):** Welch's method was used to estimate and visualize the power distribution of the filtered signal.
- **Pole-Zero Plot:** The pole-zero configuration of each filter was computed using the transfer function coefficients and plotted to analyze system stability and behavior.
- **Audio Playback:** Side-by-side audio players enabled subjective evaluation of the filtering effect.

8. Filter Analysis and Observations

A series of frequency-selective filters were applied to analyze and manipulate different components of the audio signal. Each filter helped in isolating or suppressing specific elements, leading to a better understanding of the audio composition and guiding noise removal decisions.

Low-Pass Filter

A low-pass filter with a cutoff at 1000 Hz resulted in an output dominated by the vocal elements of the song. Most high-frequency instruments, including the unwanted flute component, were effectively removed. This further confirmed that the vocals primarily reside in the lower frequency range, whereas the intrusive elements lie above 1000 Hz.

High-Pass Filter

Applying a high-pass filter with a cutoff at 1500 Hz resulted in audio that retained only the instrumental components of the song. The vocals, which primarily lie in the lower frequency range, were effectively suppressed. This confirms that the instrumental noise—particularly the piccolo flute—is concentrated in the higher frequency spectrum.

Band-Pass Filter

A band-pass filter tuned to 1200–2500 Hz allowed only a narrow frequency range to pass. The output primarily contained the intrusive sound of the piccolo flute, with all other musical elements removed. This indicated that the flute's energy was highly localized in this band, making it an ideal candidate for targeted suppression.

Band-Stop Filter

By designing a band-stop filter to suppress the 500–4500 Hz range, the previously identified intrusive flute component was significantly attenuated. The resulting audio preserved the desired vocals and background music with minimal distortion, thereby recovering the original song quality with reduced noise.

Specialized Phase-Reversal Subtraction

While the band-stop filter was effective, it did not preserve the phase integrity of the signal or isolate the flute component for separate analysis. To overcome this, a custom approach was implemented:

- The intrusive band was extracted using a band-pass filter.
- The phase of this component was inverted.
- The inverted signal was subtracted from the original.

This technique enabled more surgical removal of the flute component without significantly affecting adjacent frequencies. Additionally, it allowed the isolated component to be studied or reused separately. This method proved especially valuable when preserving the rest of the audio with high fidelity was important.

9. Challenges Encountered

A significant challenge encountered during the filtering process was the overlap between the noise and the desired audio content in the frequency domain**. The intrusive sound, primarily due to a piccolo flute, shared a frequency band (approximately 1200–2500 Hz) with some musical elements of the original song. As a result, applying traditional filters such as band-stop or low-pass to remove the noise also inadvertently attenuated parts of the desired signal. This made it difficult to achieve clean separation without compromising audio quality. The frequency overlap limited the effectiveness of purely frequency-selective filtering, highlighting the need for more nuanced approaches like phase inversion subtraction or time-frequency masking to better preserve the integrity of the original audio.

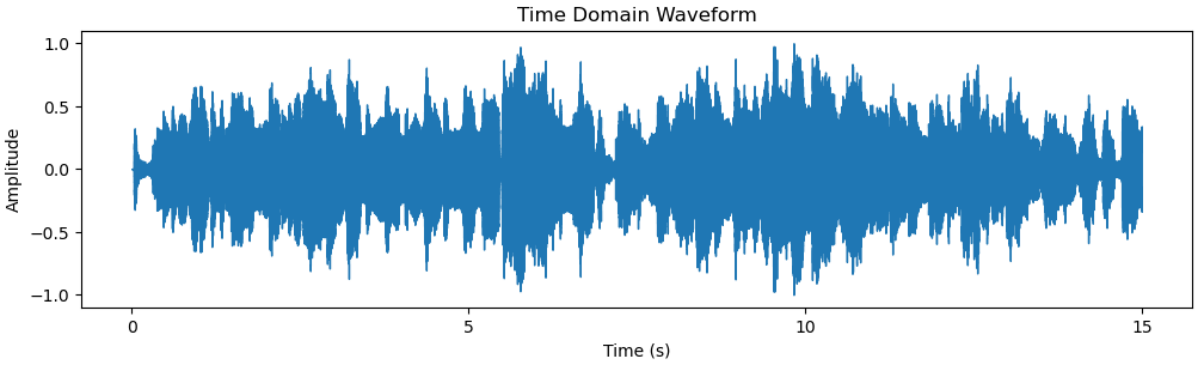

Figure 5: Time-Domain Waveform of the sample audio

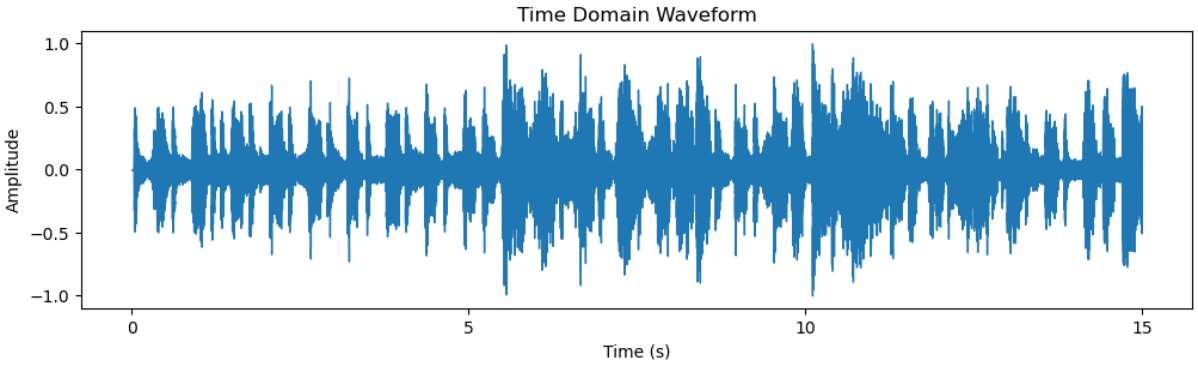

Figure 6: Time-Domain Waveform of the recovered audio