Image Processing - Exercise 2

Shir Hana Stern, shirste, 325945236

Introduction

This exercise focused on three tasks: embedding "good" and "bad" watermarks into audio files, grouping audio files based on their watermarks, and analyzing two audio files sped up using different methods (time vs. frequency domain). The goal was to understand the effects of subtle audio modifications and how these modifications can be detected or classified algorithmically.

The techniques which were used are:

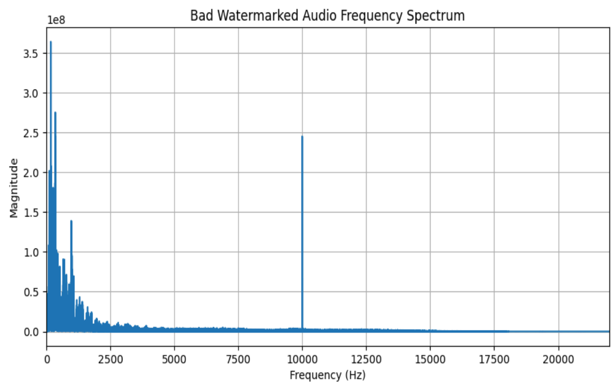
1. Watermarking: The concept of embedding imperceptible or noticeable information in audio signals, ensuring detectability by algorithms but not by the human ear.
2. Feature Extraction & Clustering: Extracting meaningful features from audio files to group them by similar watermark functions.
3. Time and Frequency Domain Analysis: Analyzing how speed-up transformations in both domains affect audio properties (duration and pitch) to identify the method and calculate the speed-up ratio.

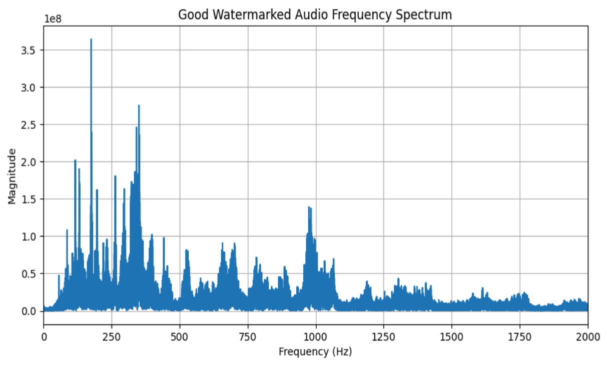
Adding Watermarks

In this exercise, two types of watermarks were designed: good (inaudible) and bad (audible). Both watermarks were applied by modifying the audio signal in subtle ways.

- Good Watermark: The "good" watermark was created by embedding a high-frequency tone (10 kHz) with a very low amplitude (1). While the frequency of 10 kHz is at the upper limit of typical human hearing, the low amplitude ensures that it remains imperceptible to the average listener. This type of watermark is designed to be subtle, leaving the original audio largely unchanged, while remaining detectable through computational spectral analysis.
- Bad Watermark: The "bad" watermark was also implemented using a high-frequency tone (10 kHz), but with a significantly higher amplitude (400). This amplitude ensures the watermark is intrusive and noticeable, causing distortion that can disrupt the listening experience. This deliberate choice makes the watermark audible to listeners, illustrating the contrast in perceptibility between "good" and "bad" watermarks.
  The graphs also illustrates that in the frequency domain, the "bad" watermark

reaches significantly higher amplitudes at 10 kHz compared to the "good" watermark, which is the primary reason it becomes audible to the human ear.





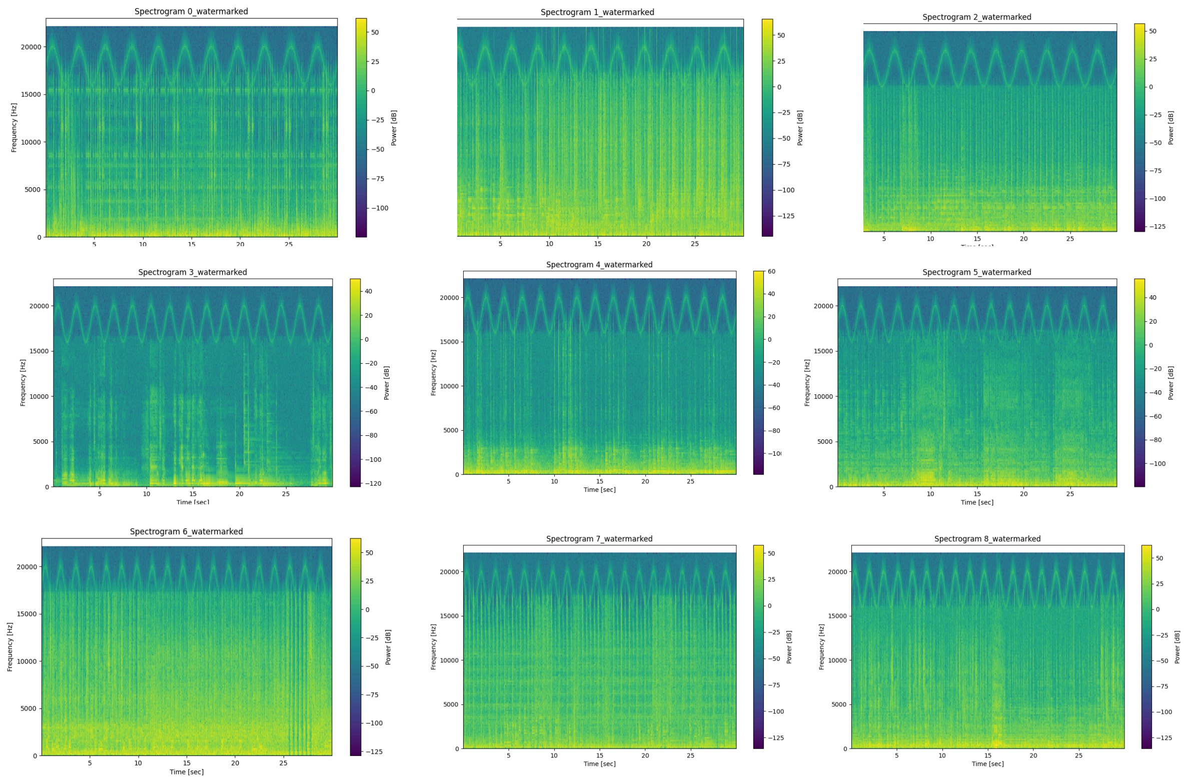
I initially experimented with embedding the "good" watermark as a mid-frequency tone (around 5 kHz) with low amplitude. While it was less intrusive, it was still perceptible in quiet segments and harder to detect algorithmically. For the "bad" watermark, I tried random noise, but it lacked consistency and was less audibly intrusive in some cases.

The "bad" watermark was less successful because its high amplitude at 10 kHz caused noticeable distortion, making it easily audible and reducing the audio's quality. This intrusive tone overshadowed subtle details and overpowered natural frequencies, compromising the listening experience and the integrity of the original signal.

To design a watermark for images, I would embed it in a way that balances visibility and resilience.
Compared to the audio watermark, a frequency domain watermark for images works similarly to the "good" audio watermark. It modifies transform coefficients, such as in the Discrete Cosine Transform (DCT), embedding the watermark in mid-frequency components to keep it imperceptible to the human eye while making it detectable algorithmically.

Classifying watermarks

My approach was to load each audio file and plot it as a spectrogram to identify sine wave-like patterns peaking around 20,000 Hz. By counting the peaks in these patterns, I determined the wavelength of the watermark. Based on this analysis, I grouped the audio files into 3 clusters, each corresponding to a distinct watermark.



- First cluster: [0_watermarked.wav, 1_watermarked.wav, 2_watermarked.wav] with function: **sin(2π * 11/30 * x)**
- Second cluster: [3_watermarked.wav, 4_watermarked.wav, 5_watermarked.wav] with function: **sin(2π * 15/30 * x)**
- Third cluster: [6_watermarked.wav, 7_watermarked.wav, 8_watermarked.wav] with function: **sin(2π * 19/30 * x)**

Implementation of the Classification Algorithm:

To classify the audio signals and identify the watermark functions, I used the spectrogram to analyze the frequency content of each audio file. The peaks in the spectrogram around 20,000 Hz were detected, and I calculated the cycle time and the number of peaks to identify the watermark's frequency. The watermark function was approximated as a sine wave based on the number of peaks and the cycle time, which was calculated using custom code.

Libraries Used:

- scipy.signal: Used for computing the spectrogram and detecting peaks.
- numpy: Used for numerical calculations like array manipulation and mean calculations.
- matplotlib.pyplot: Used for visualizing the spectrograms.
- scipy.io.wavfile: Used to load and read the audio files.

These libraries were chosen because they provide efficient and straightforward tools for signal processing and data visualization.

Parts Implemented from Scratch:

- Peak detection and filtering: I implemented a custom function to detect peaks above a specified threshold and reduced the noise by considering only significant peaks.
- Watermark function extraction: I used the peak count and cycle time to approximate the sine function for each watermark.

Hyperparameters and Choices:

- Target Frequency: Set to 20,000 Hz based on visual inspection of the spectrograms.
- Threshold in dB: Set to -6 dB to detect significant peaks in the high-frequency range.
- Cycle Time Threshold: I used a dynamic threshold based on the previous cycle time. The dynamic adjustment of the Cycle Time Threshold allows the algorithm to fine-tune its detection of watermark patterns. By recalculating with the base threshold when a significant change in cycle time is observed, the algorithm ensures that each watermark group is correctly identified based on its unique periodic characteristics.

One of the first challenges was determining where to start. Initially, I was unsure of the best approach, but I decided to begin by visually inspecting the spectrograms of each audio file. This turned out to be very helpful, as it allowed me to observe that the watermarks were distinguishable by the sine wave patterns visible in the spectrograms.

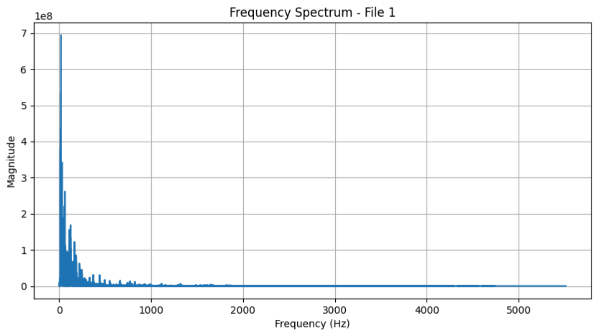
The second challenge was calculating the sine function representing the watermark. While I could identify the sine wave, the peaks in the spectrogram were often noisy, especially in some files. This noise made it difficult to clearly detect the watermark. To address this, I applied a dynamic threshold to filter out the "close peaks," which were likely caused by noise or irrelevant fluctuations. I only kept the significant peaks that were well-separated, which helped improve the accuracy of my calculations.

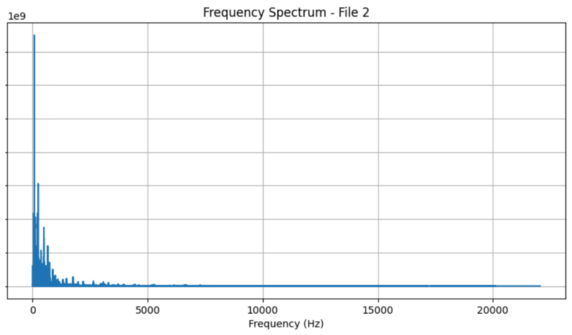
By combining these strategies—visual inspection, peak detection, and noise filtering—I was able to correctly identify and approximate the sine wave functions of the watermarks.

To remove the watermark, we can use the Fourier Transform (FFT) to convert the audio to the frequency domain. Once in the frequency domain, we identify the watermark frequency (e.g., 20,000 Hz) as a peak and remove or attenuate it by setting its corresponding frequency components to zero. After removing the watermark, we apply the Inverse FFT (IFFT) to convert the signal back to the time domain.

Determining speedup method

The two speedup methods differ both perceptually and technically.
Perceptually, the time-domain speedup makes the audio sound faster and higher-pitched, as compressing the signal in time shifts all frequencies upwards. In contrast, the frequency-domain speedup plays the audio faster while preserving its original pitch, resulting in a more natural sound.
Technically, the time-domain method shortens the signal by resampling or removing parts of the waveform, increasing the frequency and pitch. The frequency-domain method, however, transforms the signal into the frequency domain using a Fourier Transform, modifies it, and then converts it back, adjusting the playback speed without altering the pitch.





File1 uses the Time Domain speedup, and File2 uses the Frequency Domain speedup. The Speed-up Factor (x) for both files is 4.00.
Based on the frequency spectra of both figures, we can observe key differences that explain the speed-up methods used:
File 1: The frequency spectrum shows that the peak is located at lower frequencies, with the magnitude dropping significantly as frequency increases. This indicates that file 1 underwent a time-domain speed-up followed by a slow-down. In time-domain speed-up, the signal's time duration is reduced, but this causes the frequency content to spread out (lower frequencies become more prominent), resulting in a slower appearance in the time domain after the manipulation.
File 2: The frequency spectrum of File 2 shows a much higher magnitude at lower frequencies compared to File 1, indicating that File 2 underwent a frequency-domain speed-up. In this case, the frequency components of the signal are scaled, shifting the content to higher frequencies, while the time duration remains unaffected.

Conclusion

The key findings show that a "good" watermark is inaudible and detectable through algorithmic analysis, while a "bad" watermark is audible and easily noticeable. The classification of audio files based on their watermarks was successful, using frequency spectra to identify distinct watermarking functions. Lastly, the speedup methods for two audio files were distinguished by their frequency characteristics, with one file undergoing time-domain speedup and the other frequency-domain speedup, along with a calculated speedup ratio.