Image Processing - Exercise 2

Areen Mansour, areen0507, 212900211

Introduction

- This exercise focuses on enhancing audio denoising techniques using Python and various libraries like NumPy, Matplotlib, SciPy, and Soundfile. The goal is to implement and compare denoising algorithms for two audio files, q1.wav and q2.wav.

- **Differences between Noises**

  q1 Noise: Dominated by a specific frequency, addressed using targeted removal through Fast Fourier Transform (FFT).

  q2 Noise: Spread across a range of frequencies, mitigated using Short-Time Fourier Transform (STFT) and a selective band-reject filter.

Understanding these noise distinctions guides customized denoising strategies for each audio signal. Further details about the algorithmic implementation, challenges, and insights can be found in the following sections.

Algorithm

(a) Description

q1 Denoising Algorithm:

1. **Read Audio File:**
   - Utilized `wavfile.read` to load the audio file, acquiring the sampling rate (`fs`) and audio data (`data`).
2. **Identify Dominant Frequency:**
   - Defined `find_max_frequency_index` to determine the index of the maximum frequency magnitude in the audio data.
3. **Frequency Removal:**
   - Applied Fast Fourier Transform (`np.fft.fft`) to analyze and manipulate frequency components.
   - Removed the identified frequency and its symmetric counterpart.
4. **Inverse FFT:**
   - Used Inverse Fast Fourier Transform (`np.fft.ifft`) to reconstruct the denoised audio signal (`y_denoised`).

q2 Denoising Algorithm:

1. **Load Audio File:**
    - Employed `wavfile.read` to load the WAV file, capturing the sampling rate (`sample_rate`) and audio data (`data`).
2. **Short-Time Fourier Transform (STFT):**
    - Computed the STFT of the audio data using `scipy.signal.stft`, allowing frequency analysis within localized time windows.
3. **Frequency Analysis:**
    - Extracted magnitude and phase information using `np.abs` and `np.angle` functions.
4. **Define Frequency Bins:**
    - Employed `np.fft.fftfreq` to establish frequency bins corresponding to the STFT result.
5. **Frequency Range Specification:**
    - Specified a frequency range between 1100 Hz and 1250 Hz and found corresponding indices in the frequency bins.
6. **Band-Reject Filter:**
    - Applied a band-reject filter by zeroing out the magnitude values within the specified frequency range.
7. **Inverse STFT:**
    - Reconstructed the time-domain signal using `scipy.signal.istft` with adjusted parameters.

(b) Implementation

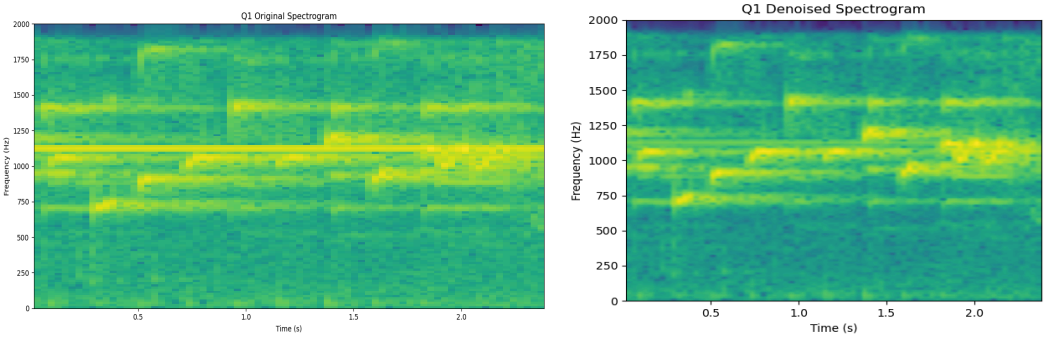
q1 Function:

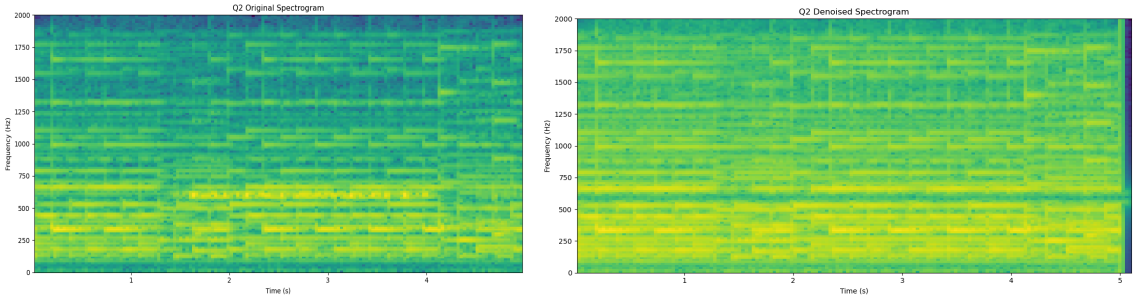
1. **Implementation Details:**
    - Reads a WAV audio file using `scipy.io.wavfile.read`.
    - Utilizes the `find_max_frequency_index` function to identify the index of the maximum frequency magnitude.
    - Zeros out the identified frequency component and its conjugate symmetric counterpart in the FFT domain.
    - Applies the inverse FFT (`np.fft.ifft`) to obtain the denoised audio signal.
2. **Hyperparameters and Choices:**
    - The only notable hyperparameter is the choice of the frequency index to be removed. The code uses the maximum magnitude frequency index, which is a common and reasonable approach.
    - The implementation doesn't explicitly define other hyperparameters, keeping the denoising process simple.
3. **Challenges Faced and Addressed:**
    - The main challenge is in choosing the appropriate frequency to remove. The code tackles this by selecting the index with the maximum magnitude. However, more advanced techniques for frequency identification could be explored for robustness.

q2 Function:

1. **Implementation Details:**
    - Reads a WAV audio file using `scipy.io.wavfile.read`.
    - Computes the Short-Time Fourier Transform (STFT) using `scipy.signal.stft`.
    - Applies a band-reject filter to the specified frequency range in the time-frequency representation.
    - Reconstructs the filtered signal using the inverse STFT (`scipy.signal.istft`).
    - Optionally saves the filtered audio to a new file using `soundfile`.
2. **Hyperparameters and Choices:**
    - `nperseg` (set to 1000): Determines the number of samples per segment in both STFT and ISTFT calls. Adjusting this parameter can affect the trade-off between frequency and time resolution.
    - Frequency range (`start_frequency` and `end_frequency`): Defines the range of frequencies to be filtered out. Adjusting these values influences the denoising frequency range.
3. **Challenges Faced and Addressed:**
    - Choosing appropriate parameters for STFT (`nperseg`) is critical for achieving a balance between frequency and time resolution.
    - Selecting the frequency range for denoising may require careful consideration based on the characteristics of the input audio.

In both functions, the code leverages existing libraries (`numpy`, `scipy`, `soundfile`) for FFT, STFT, and file I/O functionalities. The chosen hyperparameters appear reasonable, but their effectiveness depends on the specific characteristics of the audio being processed. Consideration of more advanced techniques for frequency identification and parameter tuning could further enhance the denoising algorithms.





Conclusion

In summary, the provided code presents two denoising algorithms for audio: `q1` removes a specific frequency component, while `q2` uses Short-Time Fourier Transform (STFT) for more selective denoising. Key insights include the importance of tuning hyperparameters, visualizing denoised audio through spectrograms, and enabling audio playback for comparison. Challenges include frequency selection and parameter adjustments. Recommendations include exploring advanced techniques, creating user-friendly interfaces, and evaluating algorithm performance on diverse audio data. The code forms a foundation for audio denoising, and further refinement and exploration can enhance its effectiveness.