Voice - Frequency Encoding and Decoding System

Israa Haj Ali 1201506, Rivan Jaradat 1200081, Masa Itmaizi 1200814

Birzeit University

1201506@student.birzeit.edu, 1200081@student.birzeit.edu, 1200841@student.birzeit.edu

Abstract

This Digital Signal Processing project involves a two-phase implementation of an English alphabet character voice-frequency encoder and decoder system. The encoder represents each character with a distinct combination of low, middle, and high voice-band frequencies. The subsequent decoder designed using frequency analysis (e.g., Fourier transform) and bandpass filters, retrieves the original text string from an encoded multi-frequency signal. The system operates on audio files (.wav), analyzing frequency components in 40ms intervals to decode characters. The outcome is a GUI display of the decoded text, showcasing the integration of signal processing techniques for efficient English character communication through voice-frequency encoding.

1. Introduction

The English alphabet character voice-frequency encoder and decoder system will be implemented in two phases as part of this digital signal processing project. During the initial stage, a voice-frequency encoder is created to uniquely combine the three voice-band frequency components (low, medium, and high) to represent every English character. An example of a signal having frequencies (100Hz, 1100Hz, 2500Hz) could be used to encode the letter "a." Any character can be represented by a corresponding signal having the necessary frequency components thanks to this encoding system.

In the second phase, a decoder is developed to recover the original text string from an encoded multi-frequency signal. Two approaches are employed for decoding:

- **Frequency Analysis Approach:** The decoder use Fourier transform and other techniques to examine the frequency components of the input signal at 40 ms intervals. It then uses the frequencies with the largest amplitudes to decode the associated characters.

- **Bandpass Filters Approach:**Filters designed to represent the assigned frequencies are used to route the input signal. Character decoding is aided by the information contained in each 40 ms interval about which frequencies are accepted by the filters and which are rejected.

The final system takes an audio file (.wav) as input, employs the specified decoding approaches, and displays the decoded text on a graphical user interface (GUI) screen. This project combines principles of signal processing, encoding, and decoding to achieve efficient and accurate communication of English characters through voice-frequency encoding.

2.Problem Specification

The project intends to address the problem of employing voice-frequency components for both encoding and decoding English alphabet letters in the field of digital signal processing (DSP). Creating an encoder system that can represent every English character using a particular mix of the three voice-band frequencies (low, middle, and high) is the first phase of the work. For example, encoding 'a' as a signal including frequencies (100Hz, 1100Hz, 2500Hz) implies mapping characters to distinct frequency groups. The second phase of the project then concentrates on developing a decoder system that makes use of bandpass filters and frequency analysis through the application of techniques like Fourier transform. The encoded multi-frequency signal must be decoded by the decoder within 40 ms of the original text string being recovered. The highest amplitude frequencies must be interpreted.

3. Data

The project will create and assess an audio frequency coding and decoding system using both synthetic and real audio inputs. Real-world data could be speech that has been recorded and contains all of the letters in the English alphabet. It could also include real sound frequency patterns that have been used for encoding and the implementation of particular techniques to reduce the impact of noise on the encoding and decoding process. Furthermore, it is possible to create artificial signals in order to systematically verify and test the system's resilience. The system's versatility and accuracy in a variety of contexts are ensured by these synthetic signals, which may mimic various scenarios such as varied accent patterns, background noise levels, and signal aberrations. In order to produce a complete and trustworthy dataset for the efficient design, implementation, and evaluation, real-world and synthetic data are combined.

4. Evaluation Criteria

- **Character Recognition Accuracy:** Percentage of correctly decoded characters compared to the total number of characters in the input audio.

- **Word Decoding Accuracy:** Percentage of correctly decoded words compared to the total number of words in the input text, and evaluates the system's effectiveness in reconstructing complete words, which is crucial for practical communication.

- **Frequency Analysis Precision:** the accuracy of identifying the correct frequencies in the Fourier transform analysis and evaluate the accuracy of the frequency analysis approach in extracting the information for character decoding.

- **Filtering Accuracy in Bandpass Approach:** Percentage of correctly identified frequencies using bandpass filters and measuring the effectiveness of the bandpass filter approach in isolating and decoding specific frequency components.

- **User Interface Interaction Time:** Time taken for users to input an audio file, process it, and view the decoded text on the GUI, which reflects the user-friendliness and efficiency of the graphical user interface

- **Overall System Accuracy:** Composite score considering all relevant metrics, such that it provides an aggregated measure of the system's overall performance.

5. Approach:

The approach to solving the problem of encoding and decoding English alphabet characters through voice-frequency components in this Digital Signal Processing (DSP) project involves a two-phase implementation.

5.1. Voice-Frequency Encoder Design:

Frequency Definition: Define specific low, middle, and high voice-band frequencies for each English character

- **Encoding Function:** Generate encoded signals for individual characters using a combination of sine waves with the specified frequencies, and convert the signal to 16-bit integer format.

- **String Encoding:** Concatenate the encoded signals for each character in the input string, save the final concatenated signal as a WAV file.

5.2. Voice-Frequency Decoder Design:

- **Frequency Analysis Approach:** Read the input WAV file and process it in frames, utilize Fourier transform to analyze the frequency components of each frame, map the identified frequencies back to corresponding English characters based on predefined frequency sets, finally , assemble the decoded characters to reconstruct the original text.

- **Bandpass Filters**

  Approach:

  Design bandpass filters for each character, apply these filters to the input signal in frames, determine the character with the maximum energy in the filtered signals for each frame, and assemble the decoded characters to reconstruct the original text.

5.3. User Interface:

- Provide a GUI using the “Tkinter” library for user interaction.

- Allow users to input sentences for encoding and specify WAV files for decoding.

- Implement buttons for encoding, decoding using frequency analysis, decoding using filters, playing the encoded sound, and clearing input fields.

5.4.Visualization:

- Use Matplotlib to plot the time-domain and frequency-domain representations of the encoded signal.

- Display the plots for visualizing the signal characteristics.

5.5. Audio Playback:

- Enable the playback of the encoded sound using PyAudio

5.6. Organization and Structure:

- Divide the code into functions and classes for better organization and reusability.

- Follow coding standards for readability and maintainability.

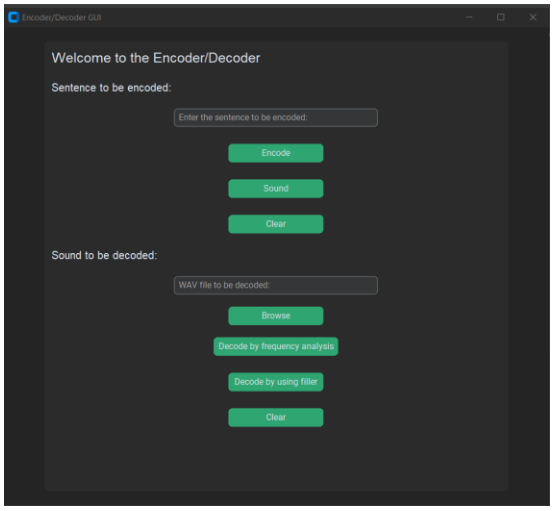

Figure 1: Graphical User Interface

6. Results and Analysis:

First, we created a dictionary in the Python language so that the dictionary contains lowercase English letters, and each letter is associated with a small frequency, a medium frequency, and a high frequency.

We have created a function that encodes English letters and converts them into a signal using the frequencies associated with each letter. The resulting signals from each letter are then appended to the string signal, and a complete audio signal is created related to the string that was entered by the user.

We saved the resulting audio signal as a .wav file, and we have plotted the resulting signal in the time & frequency domain.

It was worth creating a GUI for the user to make it easier for him to interact with the system.

For example, we tried the string "masa israa rivan":

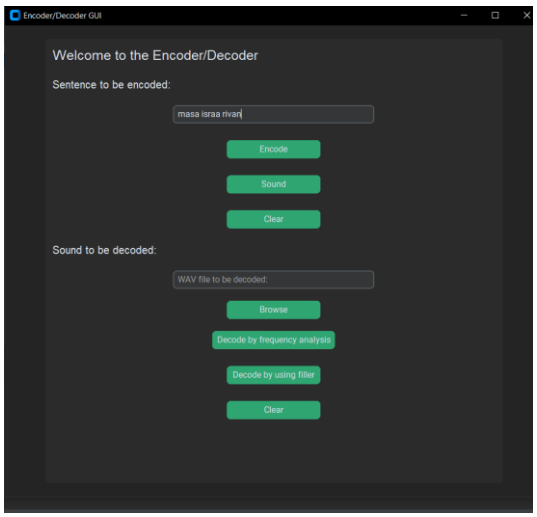

Figure 2.Using GUI to enter the input string

We obtained the following encoding results:

- The signal obtained from the encoding the string "masa israa rivan" – time domain:

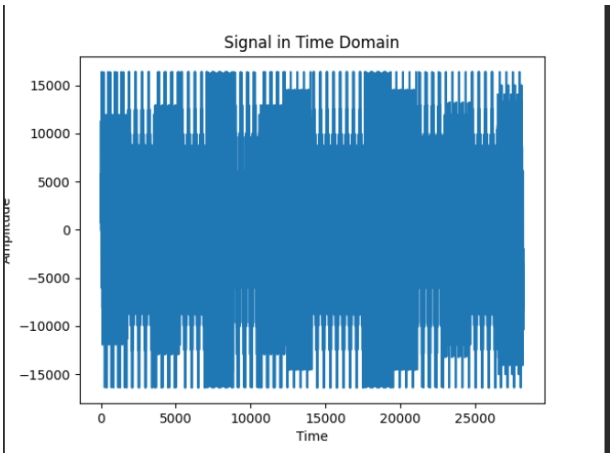

Figure 3.The signal obtained from the encoded string - Time domain

- The signal obtained from the encoding the string "masa israa rivan" – frequency domain:

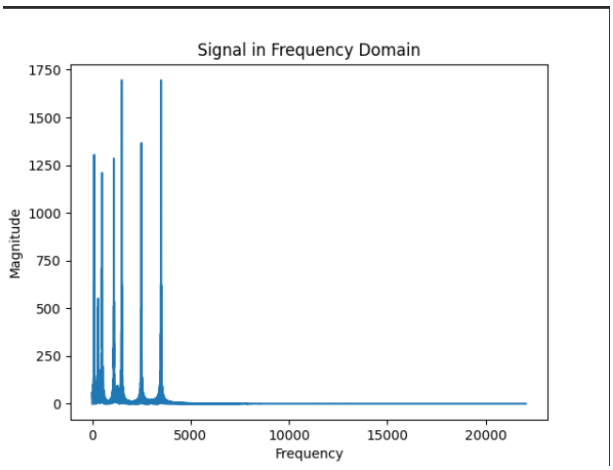

Figure 4.The signal obtained from the encoded string - Frequency domain

After we have encoded the string, and converted the resulting audio signal into an audio file in .wav int16 format, we come to the decoding, and this was done in two ways, the first one was using bandpass filter, and the second was using frequency analysis (e.g. FFT).

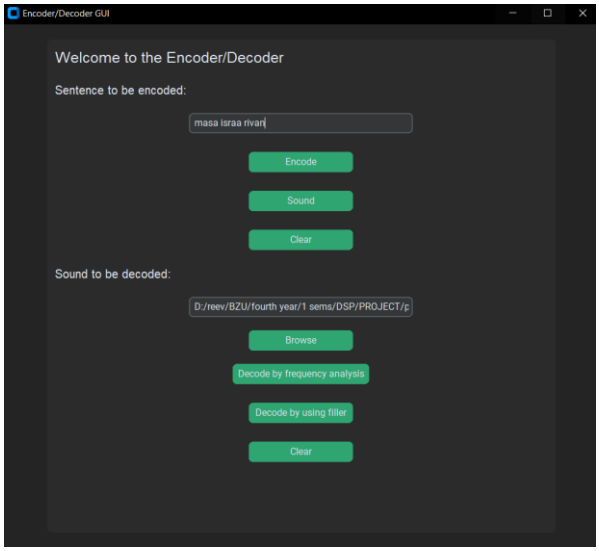

Figure 5.Adding the .wav file to be decoded

After adding the audio file to the decoding system, we obtained the following output, which represents the recovery of the original string that was encoded after it was converted into an audio signal:

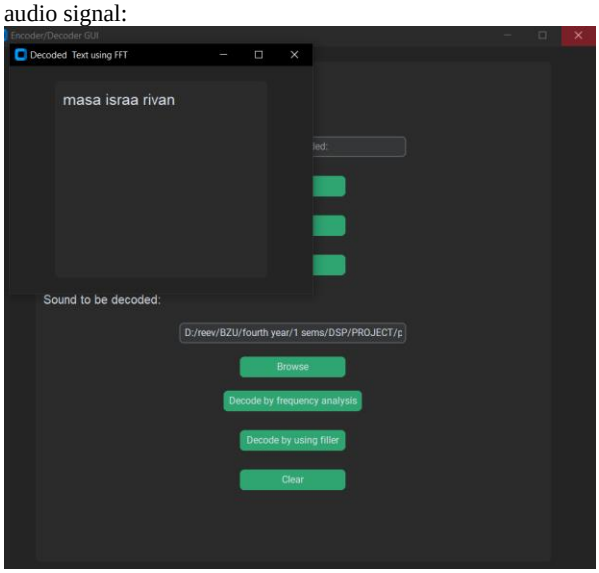

Figure 6.The output of the decoding system using FFT

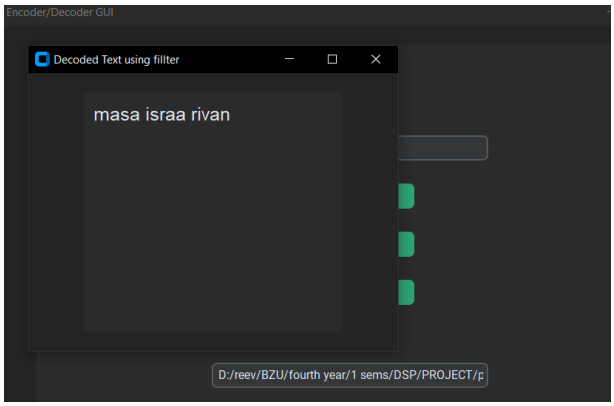

Figure 7.The output of the decoding system using fillter

Through the previous example that was presented, which explains the mechanism of the encoding and decoding system, it was shown that the system works very well according to the standers mentioned previously, which showed the effectiveness and accuracy of the system in the encoding and decoding process.

7. Development:

The aim was focused on reducing the effect of noise during project implementation. Strategies included optimizing frequency sets for noise-resistant coding, implementing context-aware decoding to reduce the effect of noise, in addition to improving Fourier transform techniques to enhance accuracy in the presence of noise. In addition, adaptive band-pass filters are introduced to selectively treat specific frequency components affected by noise. The results and analysis highlighted improvements in noise resistance, emphasizing the project's commitment to reducing unwanted distortions and enhancing overall system accuracy.

8. Conclusion

In conclusion, the Digital Signal Processing (DSP) project provided insights into encoding and decoding English letters using audio frequency components. It took into account challenges such as noise interference, pronunciation differences, and overlapping frequency bands. The project demonstrated the importance of using diverse signal processing techniques, including frequency analysis and bandwidth filters, to develop a robust and adaptable system. Overall, the project demonstrated the complex nature of audio frequency encoding and decoding while laying the foundation for future improvements and advances in this field.

2. References

[1] NumPy Developers, 2022. [Online]. Available: https://numpy.org/doc/stable/reference/generated/numpy.fft.fft.html. [Accessed 20 1 2024].

[2] "SciPy v1.12.0 Manual," [Online]. Available: https://docs.scipy.org/doc/scipy/reference/generated/scipy.signal.lp2bp.html. [Accessed 18 1 2024].

[3] "How to convert a .wav file to a spectrogram in python3," stackoverflow, 22 7 2021. [Online]. Available: https://stackoverflow.com/questions/44787437/how-to-convert-a-wav-file-to-a-spectrogram-in-python3. [Accessed 12 1 2024].

[4] curlS, "Get Secret Message from an Audio File," medium, 12 10 2019. [Online]. Available: https://medium.com/analytics-vidhya/get-secret-message-from-audio-file-8769421205c3. [Accessed 12 1 20224].

[5] "tf.audio.decode_wav," tensorflow, 11 1 2024. [Online]. Available: https://www.tensorflow.org/api_docs/python/tf/audio/decode_wav. [Accessed 13 1 2024].

[6] "docs.scipy.," scipy, [Online]. Available: https://docs.scipy.org/doc/. [Accessed 14 1 2014].