SYNTH CHALLENGE 2022

Louis Kälble

Czech Technical University in Prague

31. December 2022

Abstract

This report describes the implementation of one solution to the Synth Challenge 2022 from the Czech Technical University in Prague. This year's challenge includes two tasks.

The first one is the synthesis of the sound of the electric vehicle Škoda Superb based on car motion data obtained from Škoda auto. For the synthesis, the important data variables were identified as the engine rpm and the state of the gear. The data was extracted and interpolated to the desired sampling frequency of the audio track. Then the sound was synthesized using frequency modulation based on the engine rpm and gear, where the frequency was modeled down when the gear was shifted up. The resulting frequency vector was used to create two sine wave sounds and their harmonics using additive synthesis. The minimum fundamental frequencies of the sound signals were chosen to be the frequency of the note F and that of D, which is the Phrygian mode to F. Together they create a harmonic sounding signal. The maximum fundamental frequency that would apply at maximum rpm was chosen to be 3 octaves higher than the minimum. Moreover, a clicking sound was created to simulate the sound of paddle-shifters using a two-pole filter. This signal was added at every gear shift. A light acoustic echo was also produced for the sound signal to create a fuller sound similar to inside a car.

The second task was to fully synthesize a composition based on its MIDI file. There was a choice between Barcarolle by Jacques Offenbach and Typewriter by Leroy Anderson. In this report Barcarolle by Jacques Offenbach was synthesized. The MIDI file could be decomposed and synthesized with using the provided MIDI toolbox for MATLAB as a frame and additionally adding functions to synthesize each note with its specified frequency, duration and amplitude for each instrument specified in the MIDI file. For Barcarolle, the file specified three channels with a total of two different instruments. Here, a violin was synthesized for two channels that originally specified a string ensemble and a piano was synthesized for the third channel. Synthesis of the violin was accomplished using additive synthesis and using the results of LPC-analysis of a violin. Additionally, a vibrato was added, and the violin was synthesized 20 times for each tone with slightly varying frequency to simulate multiple violins. For the synthesis of the piano, the Karplus-Strong algorithm was applied to physically model a plucked string. This was executed for each tone and the model was filtered to create the resonance of a piano. Lastly, an acoustic echo was added to further simulate the resonance of the body of a piano. Both instruments were shaped with an ASDR-envelope to fulfill their individual timbre.

The synthesized signals were both used to create .m4a audio files that accompany this report.

Contents

List of Figures

1. Introduction

This report describes the implementation of one solution for the Synth Challenge 2022 at the Czech Technical University in Prague. This year, the challenge consisted of two separate assignments:

- The first assignment was to synthesize the sound of the electric vehicle Škoda Superb based on sensor data obtained directly by the car manufacturer Škoda auto. Additionally, a video with a simulation of the car movement was provided to tailor the synthesis to it. In order to understand the sensor data, a full description of the submitted data is available.
- The second assignment was a musical instrument synthesis of one of two possible compositions based on their MIDI file. It was free to choose between the composition Barcarolle taken from the opera "The tales of Hoffmann" and composed by Jacques Offenbach or Typewriter, composed by Leroy Anderson. This report describes the synthesis of the composition Barcarolle. Along with the MIDI file, a descriptive file was included, describing the contents of the MIDI file in detail. It was free to change the instruments that were originally described in the MIDI file.

To complete the assignments, the synthesis could be performed in the MATLAB environment for both tasks. For the musical instrument synthesis, a MIDI toolbox was additionally provided that could be used to read a given MIDI file and decompose it into its instruments and individual notes with their given amplitude and duration. At this point an instrument synthesis function could be inserted to produce the synthesized composition. Furthermore, the rules stated that the resulting synthesis should be submitted in the .m4a format and all files used for the synthesis should be included.

The next chapters will conclude the methods used for this particular solution, a presentation of the results and a conclusion of the challenge.

2. Methods

2.1. Barcarolle synthesis

2.1.1. MIDI-File Analysis

To start off with the composition synthesis, the provided MIDI toolbox for MATLAB was analyzed and decomposed into its individual functions. The function to include any instrument synthesis was determined to be the *synth()*-function that takes in information about a single note to be played by a single instrument at a time, namely the note's *frequency*, its *amplitude*, *duration*, then the *sampling frequency* and information about *channel* and *synthtype* i.e., a number of the respective instrument taken from the MIDI instrument table. Next, the instrument types and channels included in the composition at hand were analyzed using the toolbox to decide for the instrument synthesis. According to the output, Barcarolle consists of three channels with instrument numbers 49 twice and 1. From the MIDI instrument table these correspond to an acoustic grand piano for number 1 and a string ensemble for number 49. The toolbox also determined that instrument number 1 accounts for 1238 notes of the file and channel one with instrument number 49 contains 129 notes while channel four with instrument 49 contains 162 notes (see Figure 1). After this analysis it was chosen to implement the instrument synthesis with a piano as instrument number 1 and a violin for instrument number 49. The next step was now to implement the instrument synthesis in a way that each note could be synthesized according to the MIDI file.

```
Your MIDI-file contains:
========================
Channel 1: instrument no. 49 (129 notes)
Channel 4: instrument no. 49 (162 notes)
Channel 7: instrument no. 1 (1238 notes)
========================
*** It's done! Enjoy the music :-) ***
```

Figure 1 Barcarolle MIDI-file Analysis using the MIDI toolbox

2.1.2. Instrument Synthesis – Violin

Starting with the synthesis of a violin meant to determine the type of instruments it belongs to, which in this case are string instruments. The sound of a violin is however not that of a plucked string as in a guitar but rather a continuous sounding string with many harmonics (see Figure 2).

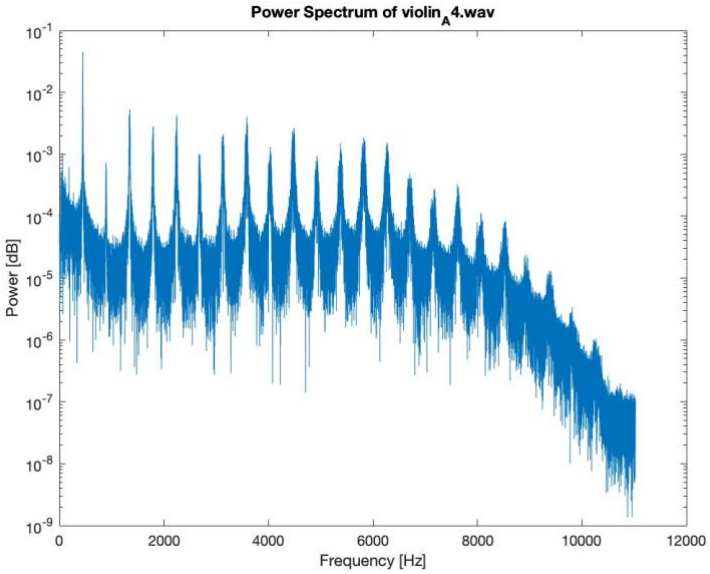

Figure 2 Power spectrum of a violin playing middle a, the harmonics can be clearly identified

To model this sound of a violin, additive synthesis was chosen [1]. Additive synthesis uses the simple technique of overlying sine tones of different frequencies over another, where each has one frequency of either the fundamental tone or the harmonics. To model the specific timbre of a violin, the harmonics and fundamental frequency need the correct scaling for a violin. This was achieved by analysis of a .wav file of a violin playing middle a at 440 Hz using linear predictive coding (LPC). With the LPC, the harmonics of a violin and their individual power as well as the resulting error signal for further resonance could be determined and saved for the instrument synthesis (see Figure 3). LPC analysis was executed using the file *lpc_analysis.m*.

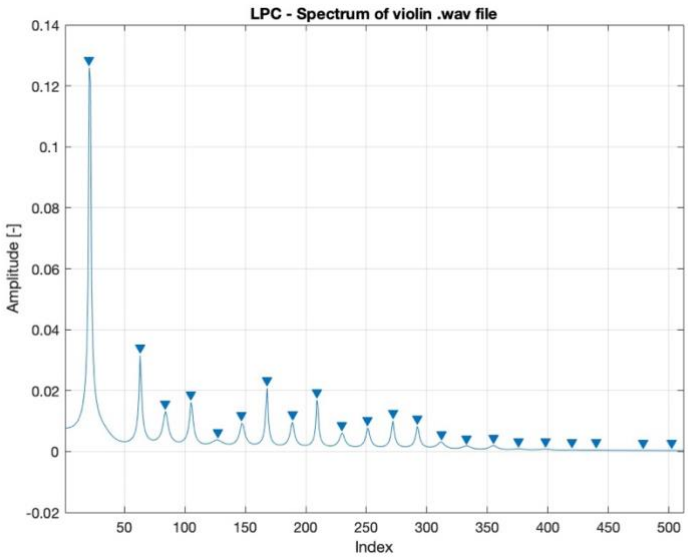

Figure 3 LPC spectrum of a violin with peak detection of harmonics

With the extracted characteristics of the harmonics, the function *violin_lpc.m* was written. This function takes in parameters *note_freq*, the frequency of the tone, *fs* as the sampling frequency and *dur* as the duration of the note and returns a signal of length dur with amplitudes between -1 and 1. In the function the harmonics characteristics are first loaded and then the additive synthesis is implemented in a for loop that loops over the number of harmonics. In each loop, the next harmonic with the corresponding multiplied fundamental frequency and corresponding amplitude is added. To avoid aliasing, the maximum number of harmonics is computed previous to the additive synthesis and the actual number of harmonics is adjusted according to it. To achieve a realistic synthesis, factors between fundamental frequency and harmonics are taken from the LPC analysis with an added random value, uniformly distributed between -0.001 and 0.001. Additionally, the vibrato of a violin was modelled by adding an alternating sine wave of a low frequency (here 4 Hz) and a modulation index of 0.1 to the fundamental frequency over the length of the note before the additive synthesis and thus also affecting each harmonic. Lastly, the result was multiplied with the desired envelope for a violin sound which was modeled as an ADSR envelope.

Outside of the *violin_lpc*-function the synth function specifies that each tone for the violin will be synthesized 20 times with a slight random change of the current tone frequency. All signals are added together and normalized and then multiplied with the specified amplitude.

2.1.3. Instrument Synthesis – Piano

Similar steps as for the violin were also made for the piano synthesis. The type of instrument is also a string instrument. However, the sound of a piano is closer to that of a plucked string. The choice of implementation, decided by personal preference, was the Karplus-Strong algorithm which physically models the plucked string as is part of waveguide synthesis [2]. In the resulting function *piano.m* similar arguments are required such as the sampling frequency *fs*, the duration *duration* and tone frequency *freq*. To then achieve the piano synthesis, first the algorithm is applied to create a signal that contains harmonics as a plucked string. This is achieved by filtering an exciting signal with the length of the tone that includes a burst of (here) white noise in the length of the determined string pluck model D (*sampling frequency / tone frequency*) using an IIR filter. The resulting signal has the harmonic components of a plucked string (see Figure 4) and is then multiplied in time domain with a sine wave of the tone frequency, essentially shifting the signal to the desired frequency. Furthermore, the signal is shaped by an ADSR envelope with a very short Attack phase to model is close to a real piano and then filtered to achieve realistic resonance using a two-pole filter.

Lastly, the signal is filtered in an IIR filter to achieve a slight acoustic echo that resembles the echoing sound of a piano before being normalized to the interval [-1,1]. Outside of the piano function, the returned signal is multiplied with the specified amplitude in the *synth*-function.

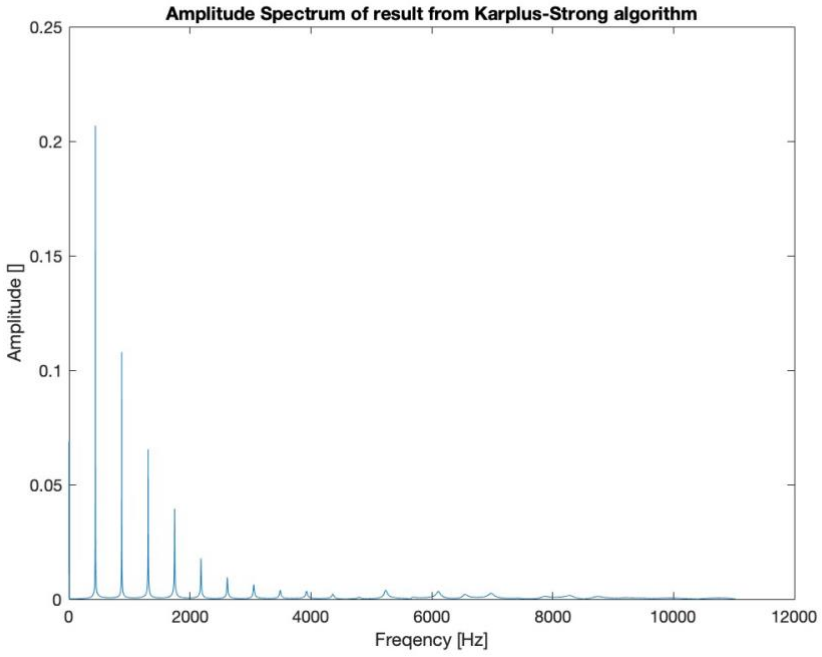

Figure 4 Result of Karplus-Strong Algorithm in the function *piano.m*

2.2. Electric Vehicle synthesis

The first step to start with the electric vehicle synthesis was getting to know the sensor data at hand and the materials at hand. Studying the sensor data description, the decision fell that the engine rpm might be the most valuable factor in determining the engine sound tone and frequency change. When the rpm is higher, the tone frequency should behave accordingly and vice versa. Additionally, it was found that information about the gear of the engine is also important for the car sound. Staying close to a realistic car sound, the higher the gear of the vehicle, the lower the frequency of the engine sound.

Thus, the first step in the sound synthesis was to extract this data and the time points from the file *control_signals.txt* (visualized in Figure 5) and interpolate them to the new sampling frequency. The sampling frequency was chosen to be at 48 000 Hz to provide a clear and accurate sound. To work with the signal more efficiently, the engine rpms have been normalized.

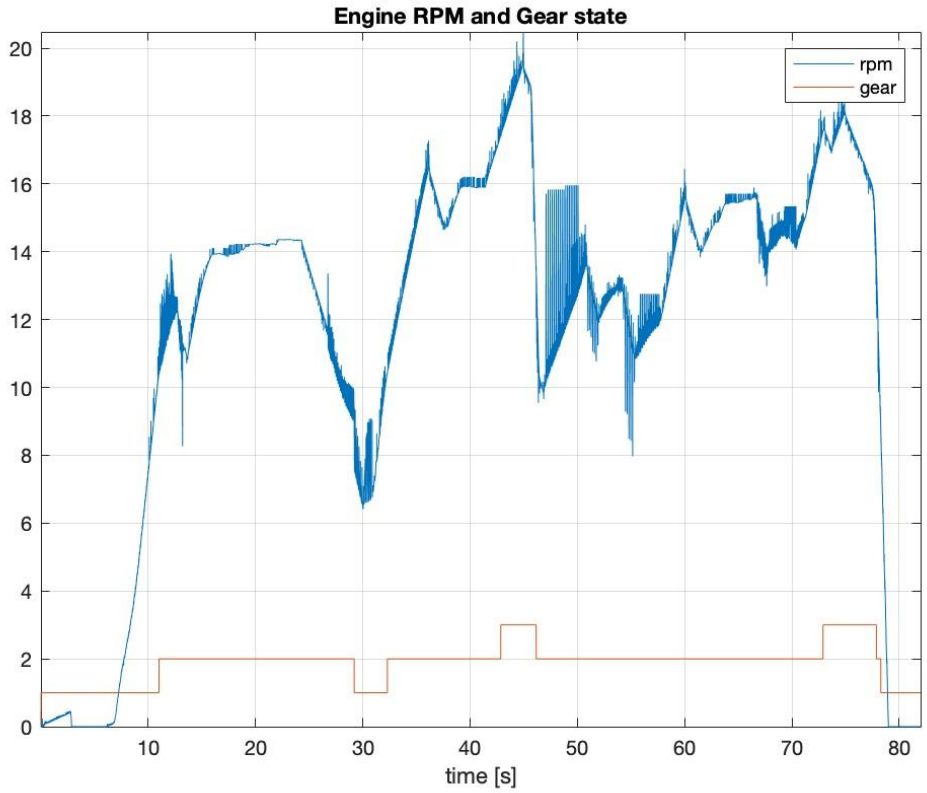

Figure 5 Engine rpm and gear state over the whole time period

The next step was to model a sound for the vehicle. Since it is an electric car, the decision fell on having a high frequency when the engine rpms are high, similar to a sports car or even a Formula 1 car so that it sounds rather like a fast car. To make the car sound more realistic and harmonics, partial harmonics are added to the fundamental frequency with additive synthesis. Additionally, a second frequency is added to model the engine more realistic. It was added based on personal preference of the sound and acts as a component higher in frequency than the

fundamental frequency. In fact, it was chosen as the Phrygian mode of the fundamental frequency in order to create a harmonic sounding signal and no dissonance. That means, the second sound signal was modeled to have a fundamental frequency 9 partial tones higher than the first sound signal's fundamental frequency.

To model the change in frequency over the engine rpm and gear state, frequency modulation (FM) was implemented. For this, a minimum frequency of the engine was defined as the above-mentioned fundamental frequencies for both signals and a maximum frequency was defined as the frequency for the highest rpm during the whole time period. This frequency was chosen to be 3 octaves higher than the fundamental frequency. For the execution of the FM, a for loop iterated over the rpm signal and adjusted the frequency at every 4 samples to the proportion of the current rpm to the maximum for both sound signals. To smooth the signal slightly and because the adjusting is done every 4 samples, the average of the current and the last 3 samples is taken as the factor for the frequency. On every other three samples the frequency is kept at the same value. To make the fundamental sine wave smooth, the argument *phi* was calculated for each sample while considering the last argument of the sine wave. Additionally, at each sample the gear state was controlled. If it changed, the frequencies at that sample would be adjusted to be either 1 ½ times higher or lower than the previous frequency depending on the shift being down or up. This creates an abrupt jump in frequency that clearly resembles an abrupt gear shift. On top of that, if the engine rpm is at zero, the sound frequency is modeled to be at the fundamental frequency. This was chosen because the simulation video still shows a running engine in the short periods where the rpm is at zero. Finally, a small vibrato was added to each phase/argument for the sine to resemble the small alternation of frequencies of an engine.

With all this considered, the frequencies over the whole time period were modeled (see Figure 6). The next step was to create the sine waves and their harmonics. This was done over a for loop of the length of a defined vector that contained the amplitudes of each harmonic. This vector was chosen to personal preference of the sound result and includes the fundamental frequency and 10 harmonics that are steadily descending in amplitude. This is not only because they are chosen to be less prominent but also because humans perceive higher frequencies as a higher amplitude [3], thus a less prominent high harmonic might still be perceived as equal or louder in amplitude.

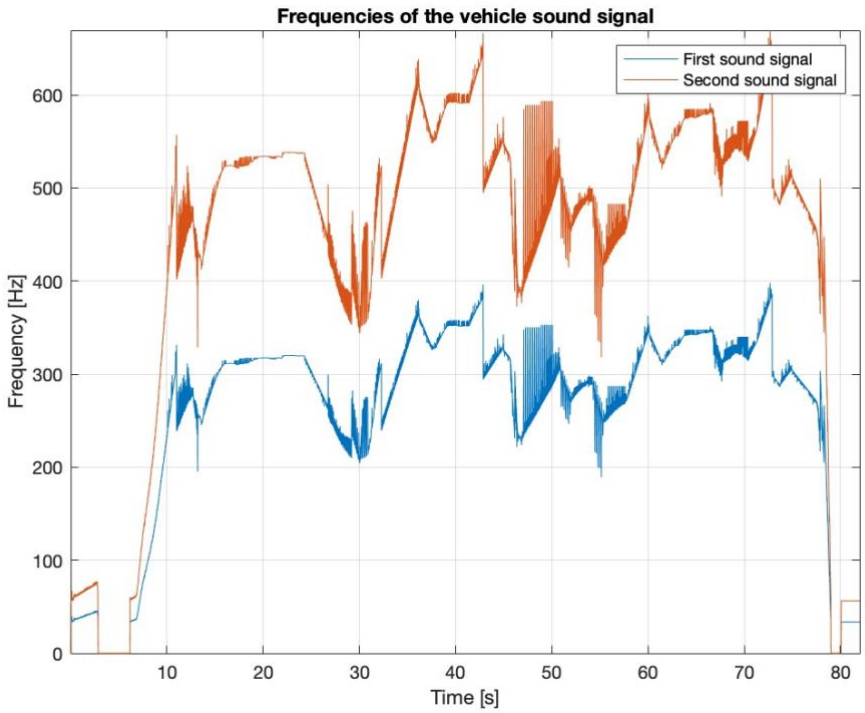

Figure 6 Modeled frequencies for both sound signals of the vehicle synthesis

After the sound signals have been created, a clicking sound was created to model the sound of a gear shift using paddle-shifters as in a sports car. For the shift sound, white noise of the duration of 0.5 seconds was filtered two times using an IIR-filter of two different resonance frequencies and the results were summed and multiplied with a fast exponentially decaying envelope. This created sound was then convolved with a vector of the length of the signal that contained ones at the indices of a gear shift. Before the shift sound was added, the complete sound signal was created as the sum of both individual sound signals and a light acoustic echo was added to fit to the simulation where the engine sound is perceived in the car.

This resulting shift sound signal was added to the sound signal after a first normalization to ensure that the shift sound is clearly audible. The final signal was then again normalized. As the final step, the synthesized sound signal is written into an .m4a audio file using *audiowrite()*.

3. Results

3.1. Barcarolle synthesis

As a result of the above-described methods, a full synthesized composition of Barcarolle can be found in the file *Barcarolle.m4a*. The spectrogram which shows the course of all frequencies over a set time window throughout the signal is shown in Figure 7.

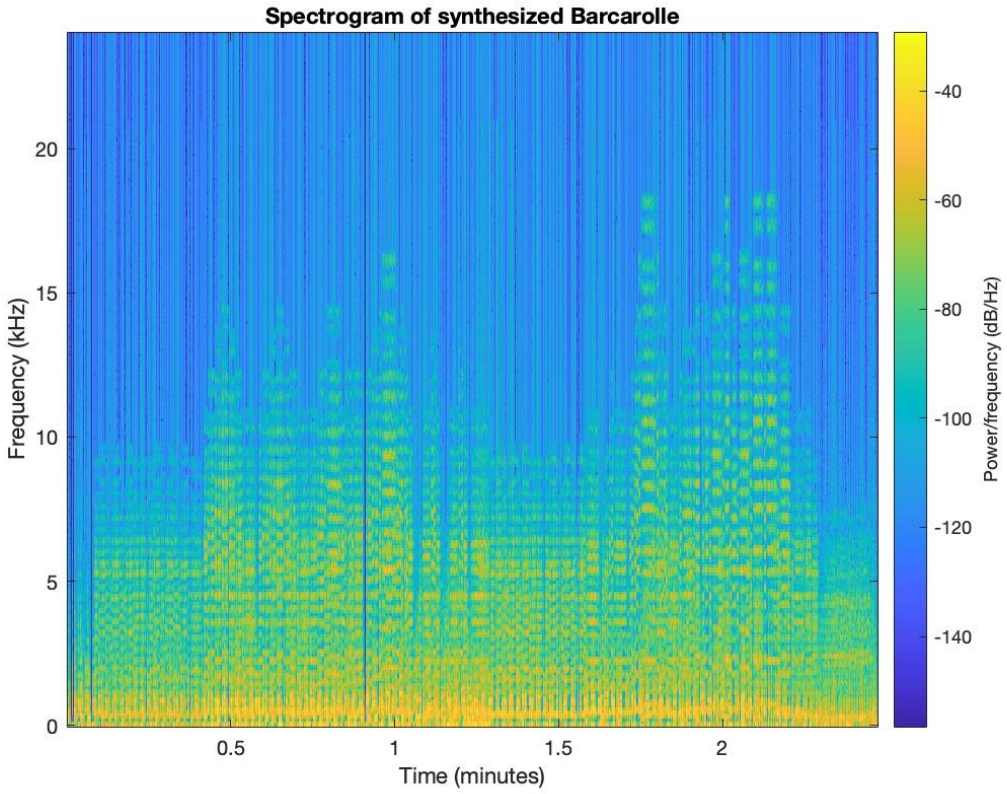

Figure 7 Spectrogram of synthesized Barcarolle

3.2. Electric Vehicle synthesis

After executing the above-described script *car_synthesis_kalbllou.m* the full sound signal can be found in the file *electric_vehicle_synthesis_kalbllou.m4a*. It corresponds to the full-length data and can be added as the audio track of the simulated video. The spectrogram showing the course of frequencies of this synthesis can be found in .

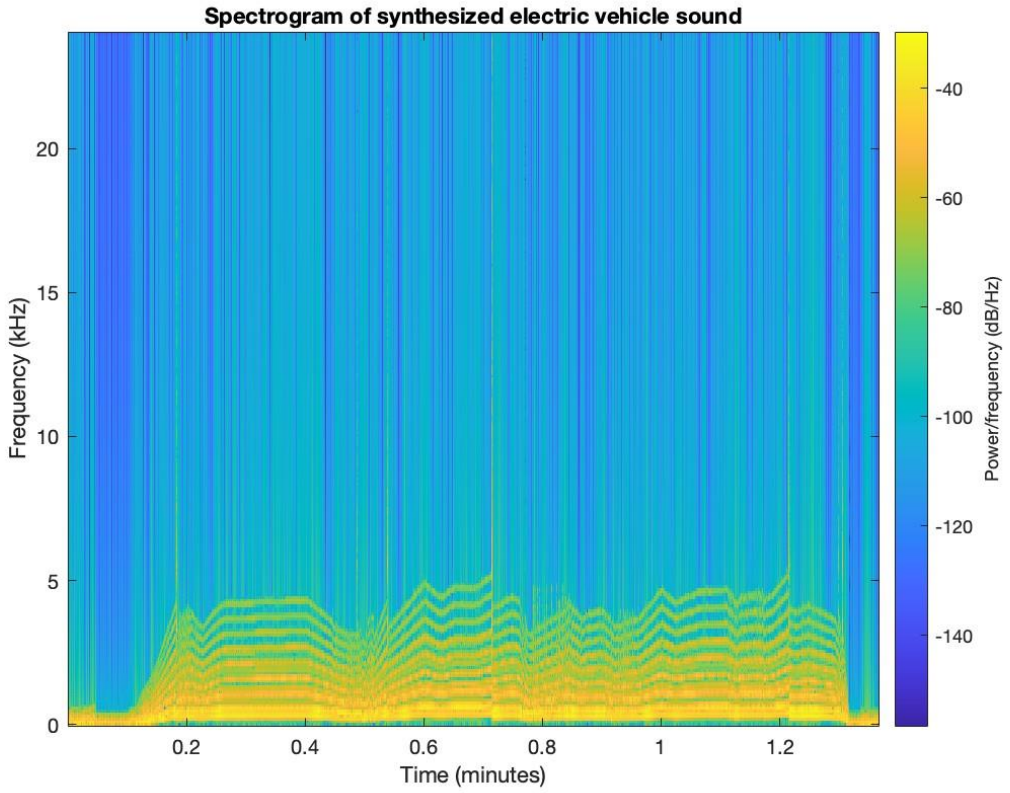

Figure 8 Spectrogram of synthesized electric vehicle sound

4. Conclusions

4.1. Barcarolle synthesis

The described synthesis of the composition Barcarolle by Jacques Offenbach was successfully completed using MATLAB functions to synthesize a violin and a piano and the provided MIDI toolbox. The result resembles the song Barcarolle however, it is noticeable that more string instruments are missing to give it the necessary full sound over many frequency ranges of different string instruments. Therefore, as a next step, more string instruments should be added, or a string ensemble can directly be synthesized as the MIDI file originally described. Additionally, the piano could be improved with a more complex version of the Karplus-Strong algorithm [2], here a simple version was implemented.

4.2. Electric Vehicle synthesis

The synthesis of the sound from car motion sensor data was successfully implemented in the above-mentioned MATLAB script. It automatically creates a synthesis based on the proved data.

The resulting sound of the electric Vehicle slightly resembles a sports car or even a formula one car as it goes very high in frequency. This was personal preference as it makes the electric car sound very fast. In this implementation, the resulting car sound was modeled using simple additive synthesis. In the future, this model could be improved to include actual resonance of a car's chassis or a different modeling that may improve the sound based on the determined frequencies. It might even be beneficial to measure the spectrum of an actual engine and model the electric vehicle sound similarly. Additionally, this synthesis was created with the knowledge over the full data signal. In the future, adaptive filtering could be implemented to allow the synthesis to work in real time and make it stable e.g., the proportion of the max rpm could be predicted based on past values with an adaptive filter.

References

[1] R. C. Maher, ‘Sinewave Additive Synthesis Revisited’, presented at the Audio Engineering Society Convention 91, Oct. 1991. Accessed: Dec. 31, 2022. [Online]. Available: https://www.aes.org/e-lib/browse.cfm?elib=5588

[2] M. Karjalainen, V. Välimäki, and T. Tolonen, ‘Plucked-String Models: From the Karplus-Strong Algorithm to Digital Waveguides and beyond’, *Computer Music Journal*, vol. 22, no. 3, pp. 17–32, 1998, doi: 10.2307/3681155.

[3] I. Johnston, *The Interplay of Physics and Music, Third Edition*, 3rd ed. Boca Raton: CRC Press, 2011. doi: 10.1201/9781439894675.