ECE446F: Sensory Communication

Melody Improvisation Unit (μ)

Marko Ciric, Joey Zhang, Tangia Zhou

December 9, 2022

Course Instructor: Daniel Smieja

Teaching Assistant: Alexander Mertens

1.0 Abstract

As the world becomes more reliant on technology, many industries have begun an integration of automation into their products. The Melody Improvisation Unit (MIU or μ), serves to improve the music industry's ability to create new music by improvising a melody with variations in pitches and rhythms based on a series of input chords. This report will discuss the various methodologies and concepts utilized in creating a prototype product that is capable of generating melodies. It will discuss the objective of the project, specific implementation techniques, results, and key takeaways, and finally the future steps of this project.

2.0 Introduction

The Melody Improvisation Unit is a product that is inspired by a musician's ability to improvise over a set of chords. It attempts to simulate a musician's musical intuitions when creating a line of melody. MIU is a project that affords melody extemporization through the application of Fast Fourier Transforms (FFTs), digital signal processing (DSP) theories, and melody generation algorithms.

3.0 Objective

The objective of MIU is to extract the frequencies of various input acoustic signals and render a new melody based on the extracted rhythm and chordal harmony.

4.0 Design and Procedure

This section will focus on the comprehensive development process of MIU from the initial inspiration to implementation and technicalities. Additionally, this section will discuss the problems, reformulations, and reiterations encountered during the development of MIU. Also, the MIU algorithm is open-sourced and can be viewed on GitHub [1].

4.1 The Working Basis of MIU

The working principle of MIU is based on the improvisation techniques of musicians like John Coltrane. In the past, composers and performers have used these sophisticated improvisation techniques to create music and develop sensical melodies when playing or composing over a given set of chords. The general approach to improvisation is composed of two stages. The first stage is for the musician to recognize the given chord that is being played. The second stage is to extrapolate what notes or pitches should be used to play a melody in the given chordal context and rhythm durations. These two stages of music improvisation are used as guiding principles of MIU, aiding in the development and reiteration of the project throughout the semester.

4.2 Real Time Melody Generation

Various options and processes were explored for the project. In the beginning, the main goal was to analyze acoustic signals in real time and perform live melody generation on the chords, synonymous to a musician's ability to improvise during live performances. This rough idea allowed for the breakdown of project components into MIU's software and hardware counterparts: a microphone is needed for signal collection (a musician's ears), a microprocessor or computer is needed to perform computations on the received signals (a musician's music intuitions and brain), and finally a speaker is needed for the melody output (a musician's instrument).

The initial hurdle came when hypothesizing the pseudo code for the algorithm which would afford real time melody generation. The algorithm is broken into four main steps, as described in detail below.

1. Compute FFTs on the input signal as it is being collected by the microphone. This would allow for extraction of the frequencies that the chord is composed of. Not only is this dependent on the quality of signal collected by the microphone, but this is also limited by how efficient the FFT code runs. Any inefficiencies could contribute to possible latency issues of the melody generation.
2. Extract and recognize the dominant frequencies in the context of a chord. It must categorize the chord and be able to find the corresponding scales to be played over it. For

example, if a chord is a C major triad, then a melody that is composed of notes from the C major scale are sensible pitches that could be played. This would add further complications as then there is a need to create a pool of all possible scales and chords that the algorithm can look through. An illustration of the notes in the C major scale is shown in Figure 1.

3. Using the notes from the C major scale, the algorithm needs to generate sequences of random melodies which will then be outputted to the speaker.
4. Repeat steps 1-3 every time there is a new chord being picked up by the microphone. However, this is difficult to deal with as minimal changes in background noise could allow the FFTs to vary in its shape. This problem would require the algorithm to adopt some advanced heuristic. This would mean that a longer time for development and testing is needed.

The four steps listed above are a few major complications a real time algorithm will need to address. Due to this the real time melody generation version of MIU would be a difficult product to realize, with the most difficult part being the real-time processing of the input acoustic signal one segment at a time and knowing when a new chord has arrived. In the end, it was decided to approach the development of MIU from a different angle, which is processing the input signal before melody generation. Additionally, the microphone component of MIU has also been omitted due to the noise potentially being picked up from the environment. Ultimately, a DAW was used to generate the signal without unwanted background noise.

4.3 Application of Harmonics

Another big component required by the algorithm is its ability to recognize and categorize chords. With the application of harmonics, there is also no additional need for generating a library or pool of functions for chord recognition. This idea allowed for drastic simplifications on the computational complexity of the algorithm. In a way, this mimicked the improvisation techniques of real musicians with one difference being that the algorithm itself did not have to actively be aware of what chord it is analyzing. All it had to do was take the FFT of that given segment of audio and extract its dominant frequencies for melody randomization. With

harmonics, there are parallels in the notes, with some exceptions to the accidentals like the B flats and A flats, as seen in the C major scale and the C harmonics diagram below (refer to Figures 1 and 2). Using the harmonic properties of a given note, its scale could be reconstructed. With this, the MIU algorithm could generate random melodies using the harmonics of a given chord.

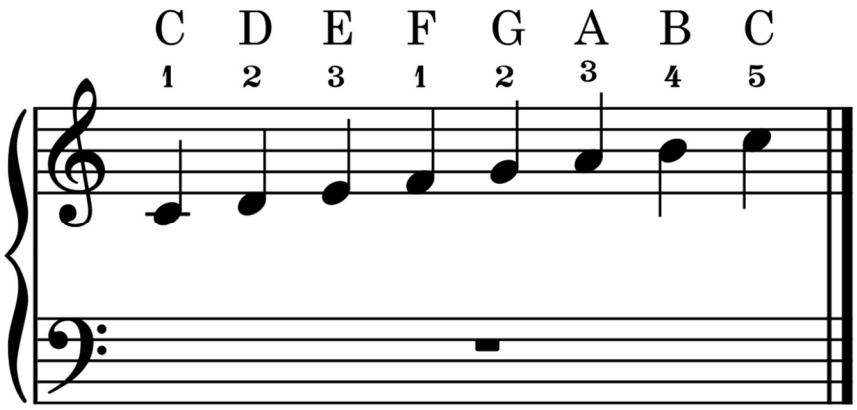

Figure 1. C Major Scale [2]

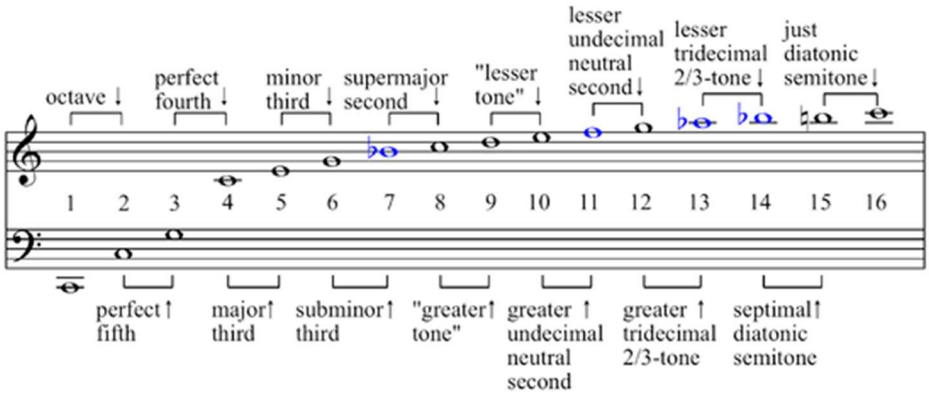

Figure 2: Harmonics of C [3]

4.4.1 Implementation of MIU in Software

The process of MIU begins by digitally generating a series of chord progressions using a digital audio workstation (DAW). This project utilises Logic Pro's assortment of software instruments for chord generation and the selection of the tempo of the chord progression. For this project, the

tempo is selected to be 60 beats-per-minute (BPM). The generated chord progression is then saved as a waveform audio file (WAV), to be processed by the algorithm. After the chord progression is saved as an audio file, it is imported into MATLAB, where the FFT is performed on the individual chord progressions (refer to Figure 3) so that the peak frequencies can be visualized and a general understanding of the FFT of the incoming signals can be gathered.

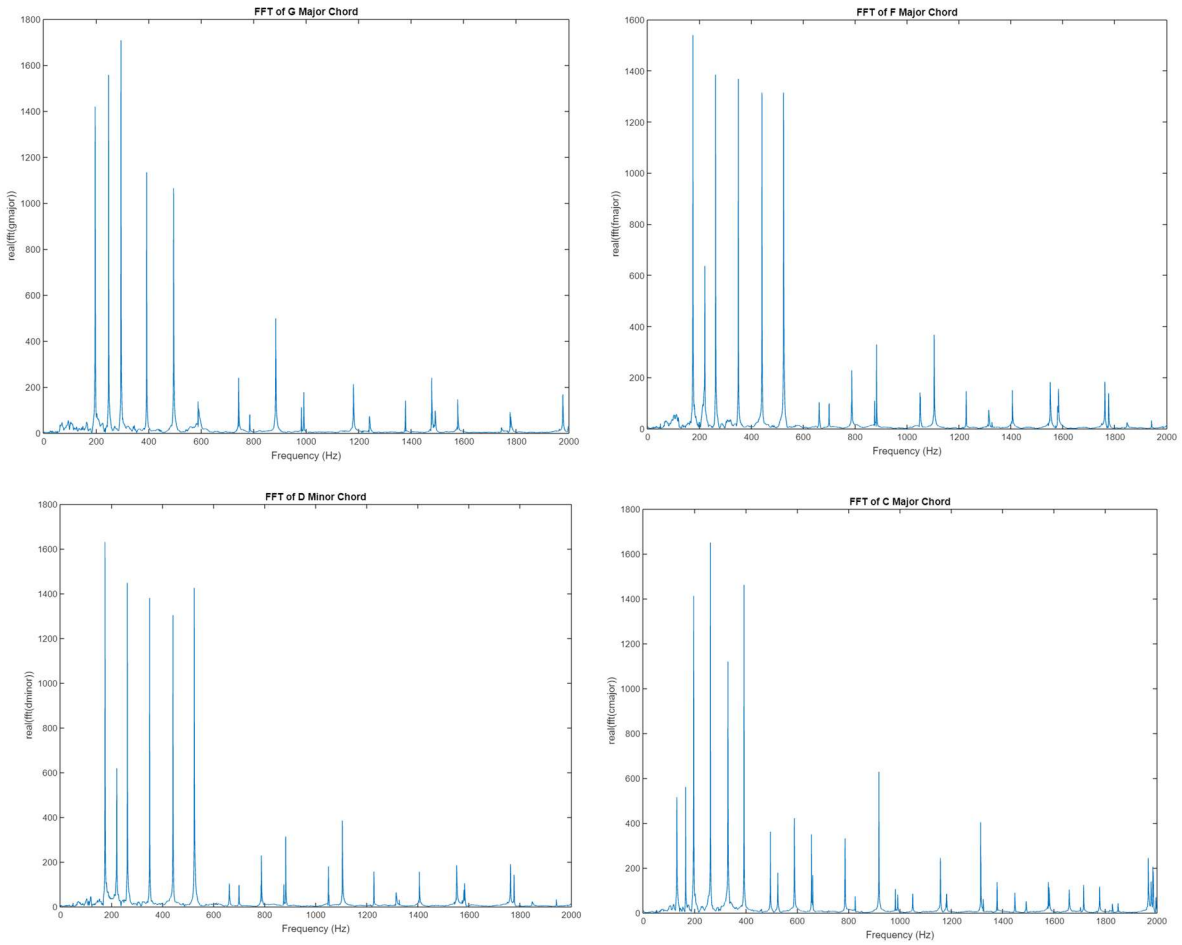

Figure 3: FFT Analysis of Individual Chord Progressions

A spectrogram is also created from the full progression to understand how the frequencies progress throughout the full chord progression, as seen in Figure 4. For the C Major progression, it is clear that each chord is clearly split in the spectrogram, signifying that the signals generated from the DAW software are clearly defined with limited noise contributions.

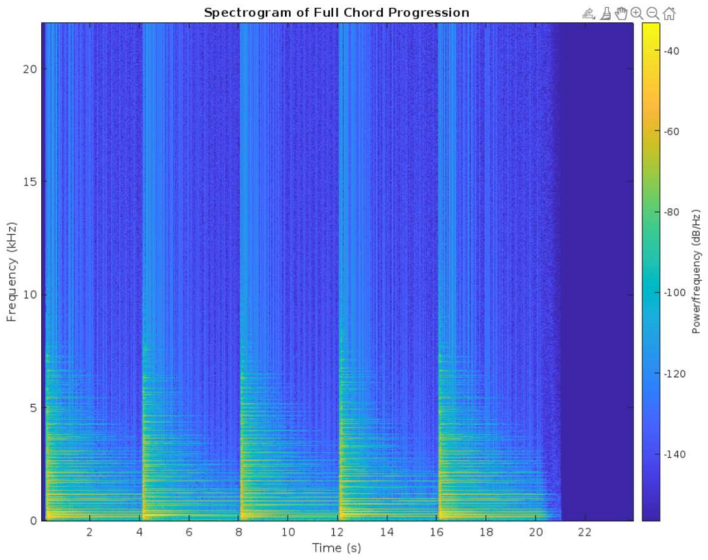

Figure 4: Spectrogram of Full Chord Progression

After the chord progression is saved as an audio file, it is imported to the Python program to be split into equally sized segments, where the higher the number of segments, the more gradual the frequency spectrum would be. Therefore for this project, the signal is processed with ten equally sized segments. After the signal is split, FFTs are performed on each of the segments to extract the dominant frequencies of each segment. The dominant frequencies are the main harmonics of the chords within the segment; where the highest peaks in the FFT occur. They are sorted and stored into vectors, where they will be further processed. The *random function* in Python is then used on each of those vectors to select random dominant frequencies where they will be generated as pure tone signals. Rather than using a more robust algorithm that is based on a musical artificial intelligence (AI) model, the *random function* in Python is used to reduce complexity. The duration of each pure tone signal is selected based on the length of the segment. A random rhythm is chosen between a quarter, eighth, and sixteenth note. For example, if a given segment is four seconds long in total, then the melody generated for this segment would need to consist of four quarter notes. The same idea can be applied to eighths and sixteenth notes. After the rhythm and frequencies are generated to fill each segment, they are appended with one another to form the generated melody vector. Finally, the main melody vector is then

normalized and added to the normalized original input chord vector to create the final product of MIU — a WAV file of a melody that is being played over a given input chord progression.

4.4.2 Implementation of MIU in Hardware

To create MIU into a portable unit that can be marketed as a product, a microprocessor is used to execute the MIU algorithm and output the new melody through a speaker. For this project, a Raspberry Pi 3 allows MIU to be a portable unit. The first step is to install the Raspberry PI Operating System. The MIU program is then uploaded to the Raspberry Pi 3 using a USB flash drive. After the user runs the program and an audio file is generated, it is played through the speaker, as illustrated in Figure 5 below.

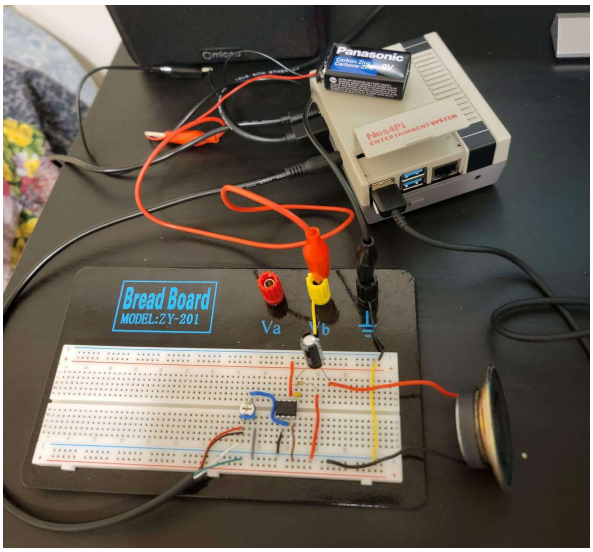

Figure 5: Raspberry Pi 3 connected to homebrew speaker

4.4.3 Homebrew Speaker Design

A LM386 low voltage audio power amplifier is used to output the new melody output from the Raspberry Pi 3. The volume is adjustable by varying the resistance of the potentiometer and the bypass capacitor serves as a filter to reduce high-frequency noise. This model is only a prototype as the gain is set as 20, but an additional capacitor can be connected between pins 1 and 8 of the LM386 that would increase the gain, which for simplicity is neglected in this model.

5.0 Results

The final results produced a successful project that met the initial objective, which is to extract the frequencies of various input acoustic signals and render a melody based on the extracted harmony. Through the FFT, specific frequencies were extracted and outputted to the melody generation algorithm to create a sequence of melodies based on the input chord progressions. This produced the output WAV file that contained the original chord progression along with the generated melody, ultimately satisfying the objective of the project. Some of the final generated melodies included Fantasie in C, Jams in C, Cool Beats, Sad Final Season, and Rainy Day which can be accessed along with other recorded songs available on MIU github [1]. These pieces were all generated using chord progressions of either major or minor keys, allowing a specific "mood" to be produced in each generated melody that corresponded with the "mood" of the input chord progression. Since the minor keys contained dissonant harmonies, the frequencies extracted from those chords played together would also create a dissonant melody. Overall, the generated melodies complete the goal of generating a melody based on the characteristics of a sequence of input chords.

6.0 Future Steps

The generated melodies met the objective of the project, which is generating melodies using harmonics of the segments from the original chord progression. However, based on the results, there are improvements and changes that can be taken into account to improve the project in the future. Firstly, due to the fact that the tones were randomized, frequencies often jumped at large intervals, not creating a smooth melody. This could be improved by developing a more robust algorithm that uses AI music models, such as JukeBox from OpenAI. To reduce the jumps in frequency, the algorithm could store frequencies close to those of the input chords, and generate a melody with a pattern more similar to the original chord progression, so that the final result is more smooth and transitions better with the input chord sequence. The algorithm could also potentially be more flexible by taking in more input parameters, generating the melody based on music genre, type of instrument, dynamics, active melody resolution, etc. This would create a more accurate improvisation model and generate a melody more similar to the input chord progression. Initially, MIU was also to be implemented in real-time. Due to the scope of this project, this was unable to be implemented. However, this project can be improved in the future

by implementing this algorithm in real-time. This would involve recording the signal and generating the melody all in real-time, immediately available to the user. Finally, to continue the idea of using a Raspberry Pi to read and output the generated melody, MIU can be marketed as a portable unit. On top of this, an application version of MIU will also be developed for personal devices with the conceptual logo prototype, seen in Figure 6.

Figure 6: Application Logo for MIU

7.0 Conclusion

Through extensive planning and implementation, the final project successfully met the objectives to render a melody based on a series of input chords. This was achieved through the usage of FFTs, simple properties of harmonics, randomization of frequencies and rhythms, and the development of the homebrew speaker. In the end, each component of the project, whether it be the project planning, technical implementations, or reiterations, all converged towards the final product that is now known as MIU – the Melody Improvisation Unit.

8.0 References

[1] Ciric, M., Zhou, T., Zhang, J., MIU, (2022), GitHub repository, https://github.com/markociricilic/MIU

[2] "Harmonic series (music)," *Wikipedia*, 29-Nov-2022. [Online]. Available: https://en.wikipedia.org/wiki/Harmonic_series_%28music%29. [Accessed: 09-Dec-2022].

[3] Lidia, "C major scale: The Ultimate Beginners Guide," *Piano Blog by Skoove - Piano Practice Tips*, 01-Aug-2022. [Online]. Available: https://www.skoove.com/blog/c-major-scale/. [Accessed: 09-Dec-2022].