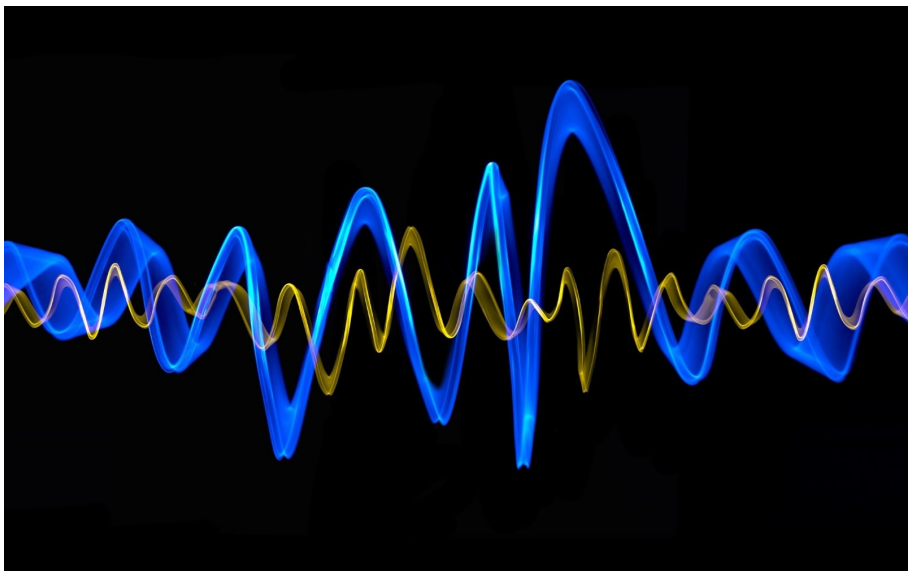
Project Report

How can we accurately identify the notes and scales in an audio track and add harmonies compared with the original melody?



Quentin Rolland and Clément Cazajous
EPFL

2024

Contents

1 Deviations from project proposal

We have adjusted our approach without changing our core question. Initially, we had not fully understood what was being asked. Our original idea was to create a code that could function like an application accessible to everyone, designed for public use. Now, our project focuses on a more relevant question and provides a clearer and more effective answer.

Our main focus is on comparing the audio output generated by our code with the actual harmonization of the input melody. This has become the primary objective of the project. To simplify the report, and because it is not the central question, we have decided to no longer include the analysis of the melody's tonality, range, and mode as initially proposed. Instead, we are displaying only the weighted comparison scores for each song in the outputs.

2 Introduction to the problem

Music, both a popular pastime and the profession of thousands, is an important and significant form of art worldwide. It serves as a means of expression and creation, essential in our demanding world filled with work and concerns. Furthermore, it has the power to unite people for various causes, whether political or otherwise. Humans have a deep need for it.

Music is also deeply rooted in mathematics: intervals are defined by simple frequency ratios, scales are built from specific divisions of the octave, harmonics consist of integer multiples of a fundamental frequency, rhythm is shaped by complex mathematical ratios, and sound analysis relies on mathematical principles, such as Fourier transforms. Given its mathematical foundation, you believed music could be generated through code. However, attempting to create a good song from scratch was overly ambitious, especially when the goal was to analyze the results and assess the song's viability. Indeed, despite its mathematical structure, music remains subjective. Any melody can serve as the basis for a song, but it's possible, even likely, that no one will enjoy it.

The final goal of your company is to develop a tool that could assist people in creating music through code. Users would simply need to play a melody on the piano to quickly hear how it would sound once harmonized. After researching, you found that there were no websites dedicated to harmonizing a melody, which is a crucial step in song creation. Harmonization involves adding chords to a melody to enhance its musicality, connecting the notes and adding depth to the song. This became the focus of your project.

That is when you approached us. As a starting point, you wanted to know if it is possible to accurately identify the notes and scales in an audio track and add harmonies compared with the original melody.

Therefore we developed a code that detects notes in a melody and adds chords in a rather simple way. We needed to assess whether the modifications sounded right to the ear and

if the tool could be used to create commercial songs. To do this, we compared existing pieces of music with their harmonization generated by our code, assigning a score based on how closely the harmonization matched the original. According to our grades we should say if your project to build an app for this seems feasible or not.

3 Approach used

First, we needed to identify the notes in the melody from the input audio. To achieve this, we opened the audio file as a signal and analyzed the data to detect the frequencies being played and their timing.

We divided the signal into small windows (0.05 seconds) and started to apply Fourier 6.1.1. Finally we applied the autocorrelation [3]. This detection of frequency process consists of comparing the signal with its lagged versions; it reveals how values at one time are related to values at another.

Formula of the discrete autocorrelation R for time interval j and signal x[n] is :

$$R[j] = \sum_n x[n]x[n-j] = \mathbb{E}\big(x[n]x[n-j]\big),$$

This step was initially implemented in Matlab, but to improve running speed because of the important data quantity of the signal, we translated it into C. However, the running process turned out to be longer than expected, probably because the advantage of vectorization in Matlab was certainly more important than the advantage of memory management in C, so we ultimately decided to keep it in Matlab.

When the frequencies are detected, we automatically transpose them to an octave roughly ranging between 254 Hz and 508 Hz.6.2

To identify the start of each note, we relied on changes in mean amplitude and frequency. A significant change indicates the beginning of a note. Once detected, we assume the note continues until the next change. During this interval, we calculate the average of all frequencies to determine the played frequency as accurately as possible. We then search our chromatic scale data and assign the closest note to each detected frequency.

Once we have identified all the notes played in the audio, we need to determine which scale they belong to. 6.1.2

We also need to establish the time grid. To do this, we calculate the average duration each time the smallest duration (more or less not much) of a note appears. The average serves to ensure that the determination of this duration is as precise as possible. This duration is either a multiple or a submultiple of the quarter note. We have set a tempo range between 80 and 160 bpm, as these are the most common tempos used in songs. We then multiply or divide the smallest duration until the resulting tempo falls within this range. While this method could lead to errors, such as mistakenly interpreting the quarter note as an eighth or a half note, such mistakes are unlikely and have little impact

on the overall result. Once this is done, we change a little bit the start of the first note and the duration of the quarter note to find the combination of two where there is the most note played that starts at a logical moment, in all possible eighth notes in the grid. When we have determined the start of the melody and the quarter note, we have the grid time perfectly set on the notes played.

For the attribution of the chords we have chosen to place them only at the beginning of measures, based on the note played at that moment or the closest note. 6.1.3

We cannot assign chords solely based on the note played. Once we identify the three possible diatonic triads, we need to determine which one to use. To do this, we rely on patterns. 6.1.3 We must therefore decide which one to apply. For the beginning, a function determines which chord patterns align with the three possible chord options available at the start of each measure. Once this is done, we perform the same check for the end of the piece. Afterward, if there is room to fit patterns in between, the same process is applied throughout the rest of the song: at each step, we search for a matching pattern. We based ourselves on pre-existing and commonly used patterns [2].

If no suitable pattern is found, we move on, leaving those positions without chords. Once we have processed the entire piece, any remaining empty positions are filled with the chord corresponding to the note played (e.g., a C note gets a C chord) for the sake of simplicity. If two empty positions are adjacent, the second chord is determined based on the first and so on. In this case, we choose the chord with the most notes in common with the previous one, under one condition: it cannot be the same chord.

Once each measure has been assigned a chord, we reconstruct an audio file combining the initial melody with the determined chords so that we can get a sense of the musical rendering.

Next, we need to check if the elements detected by our code, along with the chords it adds, align with the existing music. To do this, we assign a score at each key step of our program. The score is weighted according to the factors involved. 6.1.4

In the end, we combine these three scores to calculate a final grade, with 30% assigned to note detection, 30% to scale detection, and 40% to chord determination. We allocate 40% of the grade to chords (evaluating their placement and the number of corresponding notes) because the primary objective of this project is harmonization. Our approach is based on what we consider important and relevant for a humanized comparison between two songs. Ultimately, for a human listener, the determining factor will be the chords and their accuracy. Since everything stems from the detected notes and scale, we chose to assign these scores significant weight in the final grade.

We rely on these scores, which are the result of our work, to answer the question you posed.

4 Results

Here are our results presented in the form of a grade table for five musics :

- Mariah Carey - All I Want for Christmas is You [6]
- Imagine dragon - Birds [7]
- Daft Punk - Get Lucky [8]
- Vanessa Paradis and M - La seine [9]
- Richard Sanderson - Reality [10]

Name	Note detection	Scale detection	Chord determination	Final grade
All I Want for Christmas is you	74.9	100.0	69.8	80.4
Birds	90.2	80.0	76.8	81.8
Get lucky	86.7	100.0	66.0	82.4
La Seine	72.1	100.0	73.4	81.0
Reality	84.6	0.0	40.2	41.4
Average	81.7	76.0	65.24	73.4

Figure 1: Table of results by music

To provide a scale and understand the true meaning of each note, here is a reference table to guide you:

Meaning	Note detection	Scale detection	Chord determination	Final grade
Perfect	95<	100	80<	90<
Good	85< <95	100	60< <80	80< <90
Average	70< <85	80	40< <60	60< <80
Bad	<70	0	<40	<60

Figure 2: Table of intervals used

We based our evaluation on the previously defined weights and the variations in scores that occur when minor changes are made to the results.

- For note detection, within the 40% allocated to timing, a score of 38-39% is considered perfect, 35% is good, and anything below 30% is poor. For note identification, a score of 57-58% is perfect, 53% is good, and below 40% is poor. The sum of these two components determines the final note detection score.
- For scale detection, since there are only three possible outcomes, we distribute the scores accordingly. On average, the program should be able to identify at least one scale with the correct notes, which is why 80% is set as the average value.

- For chord determination, within the 20% allocated to timing, a score of 19% is considered perfect, 17% is good, and below 15% is poor. For chord selection, if all three notes in a chord match, the program receives the maximum score of 80%. If two notes match, the score is 53.3%; if one note matches, it is 26.6%; and if no notes match, the score is 0%. The average score is calculated across all chords in the piece of music. The final grade for chord determination is perfect when all chords have 2-3 notes in common with the actual chords, good when approximately 2 notes match per chord, and poor when only one note matches.

In each case, we define an interval around the central value to account for slight variations. Finally, the same weighting is applied to these intervals to establish the scale for the final grade.

On average, the note detection grade is not particularly good. Similarly, the average grade for scale detection is rather poor based on our criteria. However, this might be due to the significant gap between the "poor" and "average" categories, for the first four pieces of music the average would have been much better. The average grade for chord detection, on the other hand, is quite satisfactory as it falls within the "good" range. Overall, the final grade is not too bad; on average, it falls within the "average" interval.

5 Conclusion & Outlook

Based on our work, it appears that creating an application for harmonizing melodies is feasible, though it still requires further refinement. While the note detection and scale identification components did not perform at an ideal level, the chord determination process showed promising results, which is the core focus of this project. The system was able to detect the melody's notes and apply diatonic triads. However, the performance varied across different songs, with some pieces showing more accurate results than others.

The results for note detection and scale identification were not as high as expected. The average grade for note detection was lower than ideal, which may have been influenced by the difficulty in accurately identifying every note within a melody, even if we choose to reproduce pieces of music with the software FL studio [1]. The scale detection was also not perfect, which indicates that further improvements are needed to ensure the system can consistently identify the correct scale or mode of the melody. Despite these challenges, the chord determination aspect of the project performed reasonably well, with a satisfactory level of accuracy in most cases.

We made several simplifications for the sake of the project's simplicity. The audio input used was simple, consisting only of a piano and without background noise. We set the tempo range between 80 and 160 BPM, as this is the most common for music. We assumed the first note starts on the first beat of the first measure and that the piece is in binary time. We also decided to place chords only at the start of each measure and used only diatonic triads for harmonization.

Thus several areas can be improved to enhance the overall effectiveness of the tool:

- Incorporation of Seventh Chords: The current system only works with basic diatonic triads, which limits the harmonic complexity of the generated harmonization.
- Differentiated Patterns for Major and Minor Scales: At present, the chord patterns used by the system are not differentiated based on whether the melody is in a major or minor scale.
- Tempo and Rhythm Considerations: The current system relies on an average tempo range of 80-160 bpm, but real-world music often varies outside this range. Implementing more sophisticated tempo detection and rhythm analysis could help the system better adapt to different musical styles and tempos, making it more versatile.
- Support for Ternary Rhythmic Grids: The current rhythmic grid assumes a binary structure, which may not accurately reflect the timing of pieces written in ternary time. By introducing ternary rhythmic grids, the tool could align the detected notes and chords more precisely with the structure of ternary compositions.
- Chords Within Measures: At present, chords are placed only at the start of each measure. To create richer and more dynamic harmonizations, future versions could assign multiple chords within a single measure.

The overall results, particularly for chord determination, indicate that the project has strong potential, but further fine-tuning and development are needed for better accuracy in note detection and scale identification. With improvements, the tool could become a valuable asset for music creation and harmonization.

6 appendix

6.1 complementary

6.1.1 First approach for detect frequencies

For the detection of frequency we started to use Fourier transforms :

$$\mathcal{F}(f) : \xi \mapsto \hat{f}(\xi) = \int_{-\infty}^{+\infty} f(x) e^{-i\xi x}\, dx.$$

We applied the Fourier transform [4] to each window to determine the dominant frequency. However, the smaller the window size, the less precise the result, as the frequency resolution is determined by deltaf=framerate/window_size. The autocorrelation was found to be a better middle ground for the precision on small windows, conversely, with this method, the larger the window the less precise was the frequency resolution.

6.1.2 How we determined the scale

We look for the scale with the greatest number of matching notes. If multiple scales have the same number of matches, we examine the first and last note of the audio. We consider that if a melody starts or ends with a note, it is the tonic note of the scale. For example, if the melody starts with a D and we detect that the audio is either in D or F, we consider the scale to be D.

6.1.3 How we assigned the chords

1) It is important to know that chords are generally composed of three notes, called triad chords [5]. To simplify because many more are existing, we have used only this type of chord. Furthermore, our chords are diatonic, meaning that all the notes used belong to the same scale, the scale which is determined earlier. To create these chords, we select three notes separated by one note each. For example, to create the chord of degree 1, we select the notes 1-3-5 of the scale. The fact that we use diatonic triads means that for each measure start, we have the choice of three chords because we can only use a chord where the note belongs. That greatly simplifies our approach.

2) These are groups of chords known to sound harmonious and appropriate for specific contexts, such as the beginning or the end of a song. This becomes evident when listening to the end of a song, as we can often recognize it solely from the melody and the chords being played.

6.1.4 Weighting of each score

- The scoring for note detection is distributed as 40% for correct timing and 60% for identifying the correct note.
- For scale detection, the score is 100% if the scale is correct, 80% if it corresponds to the related major or minor scale, and 0% otherwise. This distinction is made because, while the type of scale does not alter the notes within it, it influences the patterns used, with major scales sounding more joyful and minor scales more somber.
- Finally, the scoring for chord correspondence is weighted at 20% for correct timing and 80% based on the number of matching notes. The timing weight for chords is reduced to 20% because chords are more diffuse, and their exact timing is less critical.

References

[1] *FL Studio*. URL: https://www.image-line.com/.

[2] Michael Miller. *The complete idiot's guide to music composition*. 2005.

[3] Wikipedia. *Autocorrelation*. 2024. URL: https://en.wikipedia.org/w/index.php?title=Autocorrelation&oldid=1246127858.

[4] Wikipedia. *Fourier Transform*. 2024. URL: https://en.wikipedia.org/w/index.php?title=Fourier_transform&oldid=1263036586.

[5] Wikipedia. *Triad*. 2024. URL: https://en.wikipedia.org/w/index.php?title=Triad_(music)&oldid=1220827825.

[6] Youtube. *All I Want for Christmas is You*. URL: https://www.youtube.com/watch?v=aAkMkVFwAoo.

[7] Youtube. *Birds*. URL: https://www.youtube.com/watch?v=vOXZkm9p_zY.

[8] Youtube. *Get Lucky*. URL: https://www.youtube.com/watch?v=5NV6Rdv1a3I.

[9] Youtube. *La Seine*. URL: https://www.youtube.com/watch?v=5NV6Rdv1a3I.

[10] Youtube. *Reality*. URL: https://www.youtube.com/watch?v=8ejtIwBpqK4.

6.2 Graphs

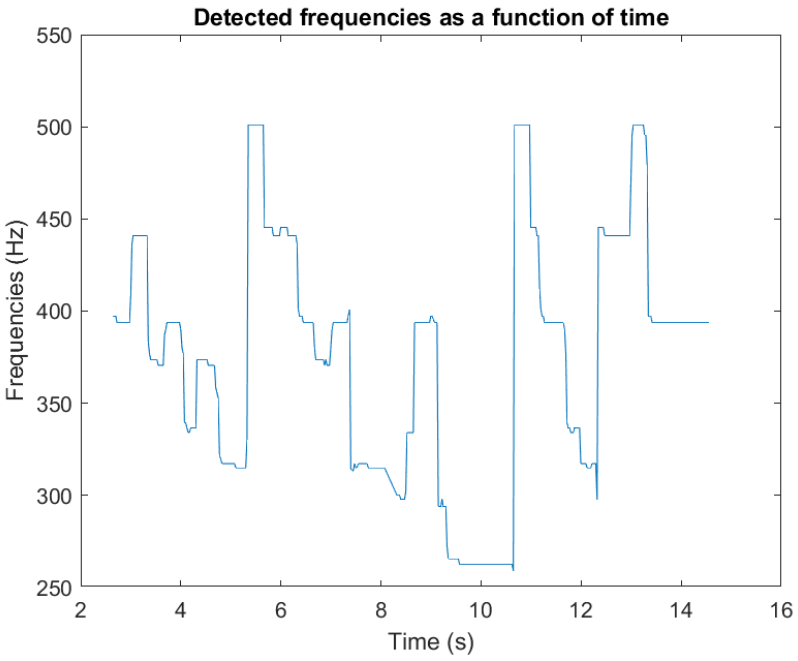


Figure 3: Plot of detected frequencies in *All I Want for Christmas is You*

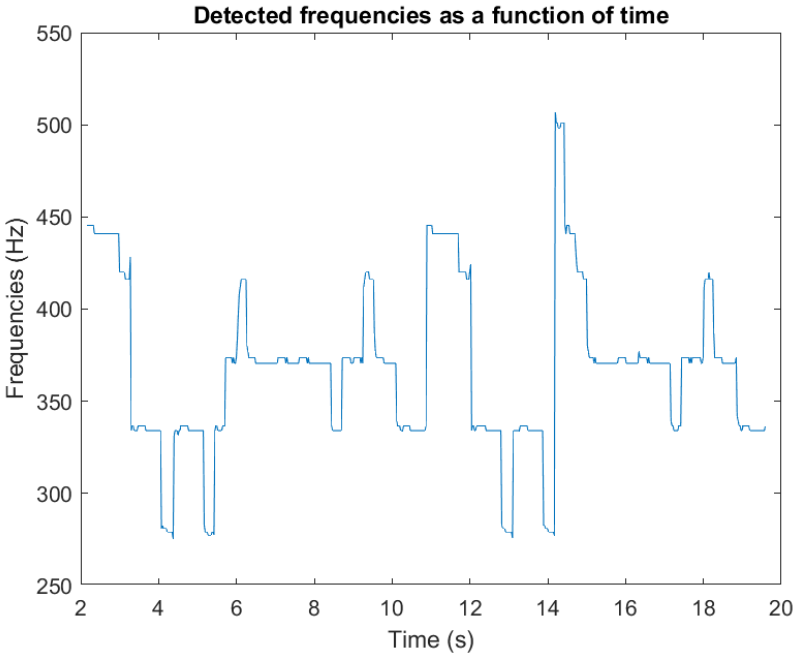


Figure 4: Plot of detected frequencies in *Birds*

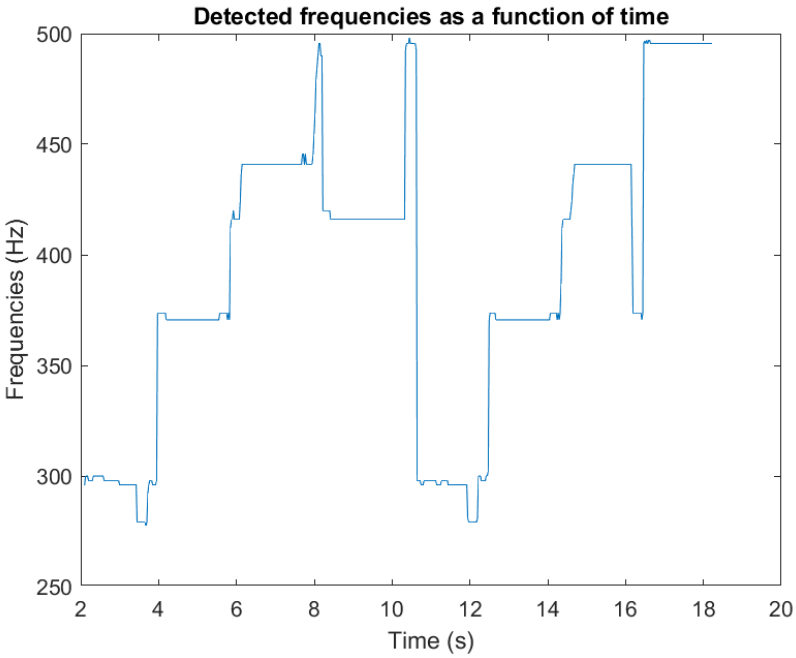


Figure 5: Plot of detected frequencies in *Get Lucky*

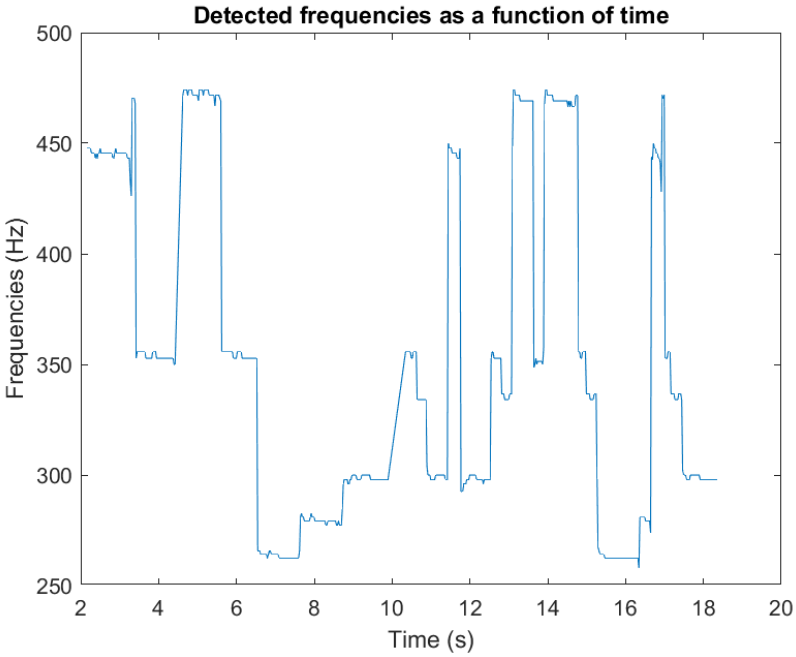


Figure 6: Plot of detected frequencies in *La Seine*

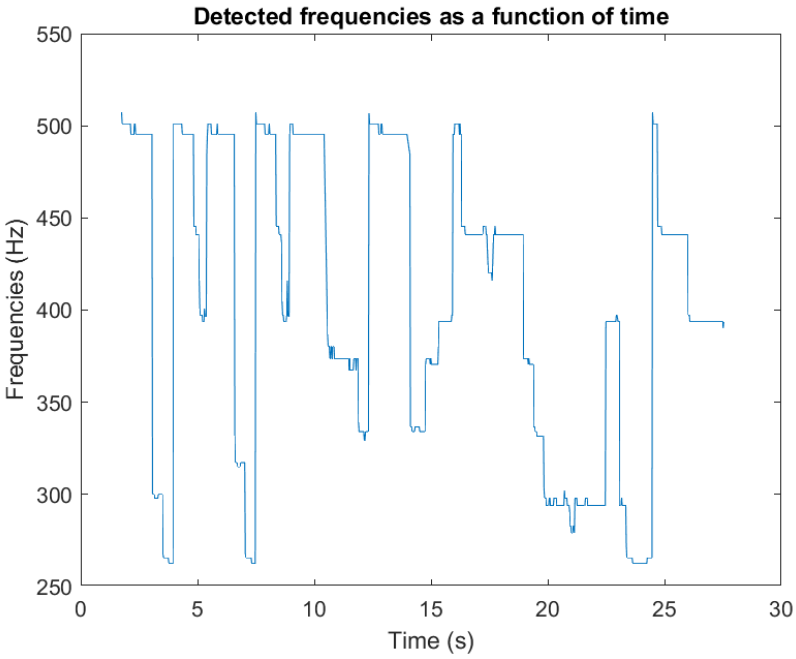


Figure 7: Plot of detected frequencies in *Reality*