Study of timbre and compatibility between musical instruments

Luis Jonás Cuervo
Tutor: Amílcar Labarta Rodríguez

Abstract: Throughout this project we have studied the features of the sound produced by a series of musical instruments, as well as the way in which this sounds are heard and felt by listeners. Our main goal was to develop a numerical application that enables us to see how pleasant two or more sounds would be when reproduced together. This application will analyse the harmonic components of the sounds to provide a numerical assessment depending on how evenly the harmonics are distributed within the spectrum.

I. INTRODUCTION

Our work was inspired by my music theory teacher Fausto Murillo, who introduced me to the role that harmonics play in music and their relevance. He inspired me to try to develop a numerical application that would assess the compatibility among different musical instruments.

One of the original goals of this project was to record a series of musical instruments playing different melodies or chords and then analyse them. Unfortunately, due to COVID-19, this was not possible. Therefore, we used virtual instruments generated by a computer. The sounds obtained may be a little bit unrealistic, but since the software that we used was able to simulate a large variety of instruments, we were able to play some instruments that otherwise would have been very difficult to obtain.

Once we had created the sounds that we intended to analyse, we obtained the spectrum of this sounds for each instant (beat) of a musical piece, using the fast Fourier transform, zero-padding and windowing. And since we were working in a musical context, the spectrum obtained was then digitised: all obtained frequencies were turned into their corresponding nearest musical notes. Then a N x M matrix was written down with the spectrum data, where N corresponds to the number of beats in the song and M to the different notes of the spectrum.

The next step was to convolve the obtained data with human-ear response. To do this, we took into consideration the frequency response of the human ear (humans are not equally sensitive to all frequencies) and the masking effect, which will be explained in more detail later.

Finally, we developed a numerical application that measures how evenly the harmonics are distributed within the total spectrum when a series of instruments are played together. Then, we used this method to select the best combination of instruments to perform a certain song.

This numerical algorithm was developed using Matlab.

II. SOUND ANALYSIS

Before we get into the Fourier transform, it is necessary to discuss a bit the nature of audio files and recording. When a microphone records a sound, what it is really doing is to transduce the fluctuations of the air pressure into an electrical signal that is then digitised. All these data are then saved as a list of values, which are later encrypted in several ways. The rate of values that a microphone can detect per second is known as the sampling frequency (fs).

The sampling frequency must be at least twice the spectrum width in order to be able to perform a satisfactory analysis of the data [1]. That is why the value of fs at which our audio files were created was 44100Hz, twice the width of an average person's hearing frequency range (20-22000 Hz). We also used the maximum bit size possible for each sample, assuring the best audio quality. These data were then encrypted in .WAV format, which has very little data loss. An example of an audio signal is shown in Fig. 1.

Concerning data analysis, the method used to obtain the spectrum of our sounds was the fast Fourier transform (FFT). This is the most efficient method amongst the other discrete Fourier transform algorithms: it is hundreds of times faster than the 'simultaneous equations' or the 'correlation' method.

Thanks to this algorithm we could easily calculate the discrete Fourier transform:

$$Y(k) = \sum_{j=1}^{N} X(j) e^{\frac{-2\pi i}{N}(j-1)(k-1)} \tag{1}$$

Where X represents the data points, N is the length of the signal, k is the bin number index and j is the index of the Fourier summation.

But before applying this algorithm, we used some tools that allowed us to increase the accuracy and reduce the spectral leakage.

First, the data were treated by windowing: multiplying the signal by a gaussian-like function (with the same length as the signal) called the Hanning window. By doing so, the significance of the samples at the beginning and the end of the signal are diminished, which is useful because the FFT works by comparing the signal to sinusoidal wave-forms that have an integer number of oscillations along the time span. Since the signal's wave is probably cut at both sides, it will not have an integer number of oscillations, causing spectral leakage when compared with the algorithm's waves. By applying the window, the algorithm will focus mainly on the centre of the signal, where the sinusoidal components are not cut.

In order to improve the frequency resolution we used 'zero padding'. This method consists on adding zeros to the tail of the signal. Given that the frequency resolution in FFT is $\Delta f = fs/N$ (being N the length of the signal and fs the sampling frequency), for a larger N the resolution will improve. [2]
In Fig. 2, we can see the effects of windowing and zero-padding applied to the signal from Fig. 1.

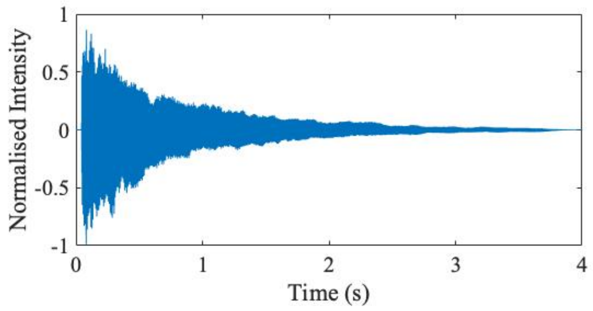


Figure 1: Audio signal of a C2 note played on a piano. The intensity is normalised to -1 (the minimum intensity that can be reproduced by the line driver) and 1 (maximum intensity), while the x-axis shows time in seconds.

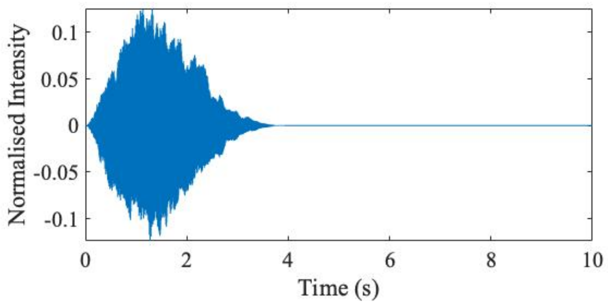


Figure 2: Same signal as in Fig. 1 after windowing and zero-padding. We padded enough zeros so that a total of 441000 data points were analysed, thus obtaining a frequency resolution $\Delta f = 0.1$ Hz.

A. Musical digitisation

With the spectrum defined for all possible frequencies, we wanted to digitise all these data into the musical notes. To do this we assigned a total of 108 notes (9 times the 12 musical notes in the different octaves) to their corresponding frequencies. Each peak of the spectrum located within a 2% of accuracy nearby to a note's frequency was arbitrarily assigned to that note. This way the spectrum was simplified so that it was easier to ascertain when two instruments have a common harmonic. An example of this is displayed in Fig. 3, where we can see the discrete spectrum of the previous signal shown in Fig. 2.

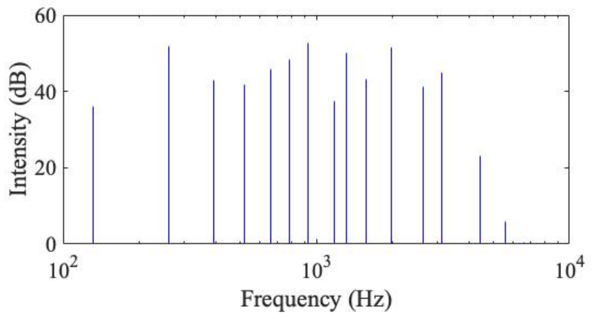


Figure 3: Musical Spectrum of the C2 note of Fig. 2. The intensity of the harmonic components is represented as a function of the frequency. Note that the frequency axis is displayed in a logarithmic form.

III. HUMAN HEARING

What we obtained until now was a mathematical description of the vibrations created by the musical instruments. But now we want to see how the brain deals with this information. There are two factors to be taken into account: the quiet threshold and the masking effect.

The first one relates to the frequency response of our ears, since the eardrum is not equally sensitive to all frequencies. Humans are specially sensitive to frequencies between 2 and 5 kHz, while other sounds outside of this range must be louder in order to be heard. The minimum intensity at which each frequency is heard is called the quiet threshold. The quiet threshold curve (depicted in Fig. 4 as a black solid line) is the minimum intensity at which each frequency must be reproduced to be heard. This curve is actually an average behaviour since there is a significant variability from person to person: for instance, as years go by we lose sensitivity to high and low frequencies, and people who are often exposed to loud noises might also lose some sensitivity. We worked with the average threshold of different subjects with ages between 20 and 25 years. [3]

The second one is masking effect. As we all know, loud sounds can sometimes hide other sounds. To study this phenomenon, scientists tried reproducing pure tones simultaneously to see when one of these tones masked the other. It was then discovered that, when two of these tones had close frequencies and one was significantly louder than the other, the quiet one was impossible to hear: it had been masked by the louder tone. Testing with different frequencies and intensities for these tones, it was possible to find curves through the frequency range of the spectrum around each masking peak representing the minimum intensity that nearby sound peaks must have in order to be heard. Because the masking curves are slightly different for each frequency range, we worked with three curves for three characteristic frequency regions (low, mid and high). These curves are shown in Fig. 4.

Although the masking curves were found while studying pure tones, they are also valid for complex sounds. We must have in mind the Fourier theorem, which states that every periodic function (such as a sound wave) can be expressed as a sum of a series of sine or cosine functions (pure tones) [4]. So if we were to reproduce simultaneously all the frequencies obtained via the analysis of a given sound as pure tones with their corresponding intensities, we would actually hear that same sound.

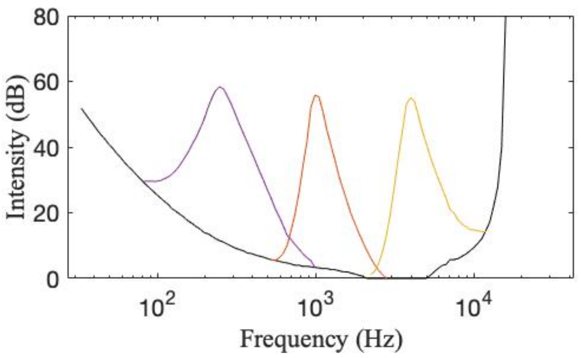


Figure 4: Masking and quiet threshold curves. The black line represents the quiet threshold, while the colored lines represent the masking curves for pure tones with frequencies equal to 250Hz (purple), 1kHz (orange), and 4kHz (yellow). All of these masking tones have an intensity of 60dB.

We applied these concepts to the analyses of the spectra in the following way: first we deleted those frequencies that were not audible due to the masking effect. To do this we created a list sorting all of the spectrum peaks by intensity. Starting with the highest peak we adjusted the masking curve to that peak's intensity and location, and then deleted every peak with intensity below that of the masking curve. All the following peaks went through the same procedure, if they had not been masked already.

Finally we subtracted from the whole spectrum the quiet threshold, obviously setting to zero all negative values. Following this procedure we obtained the spectrum in the way that a person really hears it.

IV. MUSICAL COMPATIBILITY

It is well known that, when we listen to a sound, our eardrums vibrate as a response to that sound. This vibration is then sent to the cochlea (inner ear) where the basilar membrane is located. The basilar membrane is thick on one side and progressively thinner on the other, in a way that the thick part resonates with low frequencies and the thin part with high frequencies. The basilar membrane acts as a support for 260,000 (aprox.) sensory cells that compose the cochlear nerve, which send an impulse to the brain when their supporting region resonates [5]. All in all, the whole system acts as a a frequency spectrum analyser. This is why it is easier for us to distinguish two sounds when they belong to different frequency ranges: the vibration of each sound takes place in different regions of the basilar membrane.

Following this concept we developed our hypothesis, consisting on the assumption that two sounds are more compatible with each other when their spectra minimise their mutual overlapping; thus making it easier for the auditory system to distinguish the sounds. To prove it, we developed a function that computes how much the harmonics of two different musical sounds are overlapped.

This was actually straightforward. After finding the discrete spectrum of each sound in the form of a vector (as explained before) we calculated the scalar product of these vectors. This way, if the two sounds have many intense harmonics in common, the number obtained by the product will be larger and vice-versa. We also normalised the scalar product dividing it by the norms of the two vectors, so that the final result was independent of the intensity of the sounds. In such a way, the results obtained varied between 0 (no overlap) and 1 (complete overlap).

Later on, we made a little adjustment: instead of dividing by the norm of both vectors, we only divided the product by the norm of the shortest vector. Since we mostly used the application to compare a single instrument with all of its companions combined in a single spectrum, when the product was divided by the norm of both vectors the result was usually very small, thus making it harder to see significant differences between different combination of instruments. However, after this adjustment, the obtained values are not limited like before. In order to know which values were to be expected, we applied the function to some fragments of classical pieces, in which several instruments were involved. Fig. 5 shows a histogram created with this results, where we can see that the computed values for what we called the *compatibility parameter* (CP) spread between 10 and 140.

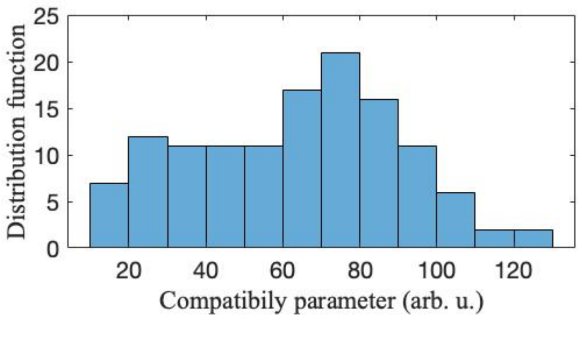


Figure 5: Histogram of the compatibility parameter when comparing various melodies within classical music pieces (*'String quartet Op. 76 no. 2'* by J. Haydn and *'Pomp and circumstance'* by E. Elgar). $\overline{CP} = 63$ and $\sigma = 27$.

A. Study of timbre

A musical note has its harmonics frequencies fixed, independently of what instrument reproduces it. What will differentiate one instrument form another is the intensity of each of these harmonics (intensity might also be zero in some harmonics). Therefore, when we analysed the spectra of a musical piece played by a series of different instruments, we observed some variations in the CP.

That is why we thought that it would be interesting to develop a function that would enable us to see which instruments would fit together in a multi-score musical piece (a score is a set of melodies and/or chords written in musical language). To do this we recorded into audio files each score being played by different instruments, and obtained its spectra for all the beats. All these spectra were then saved into a 3D array, for which the first two dimensions represent the frequency spectrum in each beat and the third dimension specifies the instrument that is being played.

Finally, we wanted to obtain a CP value for every possible combination of instruments. To do so we compared the spectrum of each score with the rest of the scores played together, obtaining a CP value for each instrument that participates in the song (actually we obtained a list of CP values for each beat, but they were averaged to become a single value). Then we computed the average of all these values (there were as many values as instruments played in the song) and repeated the procedure for the rest of possible instrument combinations.

V. RESULTS

The music piece that we analysed was a self-written piece that contains 4 scores: ‘bass’, ‘chords’, ‘melody’ and ‘high-pitch’. We recorded each score being interpreted by a series of different instruments, which we thought that sounded good when played by themselves, and proceeded to compare them.

Before performing the calculations, we ensured that all of the instruments had the same relative intensity, meaning all of the sounds had the same intensity at the highest peak. Otherwise the results would depend on the intensity at which they were recorded, which was inevitably different for each instrument.

Then we had the program running for almost three hours, comparing all possible instrument sets as described on section IV-A. A total of 1,680 different instrument combinations were compared, whose average CP were then saved into a 4D matrix of size 7x5x8x6, each index representing a certain instrument and each dimension a certain score (the code developed to compute this is shown in the appendix). From all these results we chose two: one with high compatibility (set A) and the other one with low compatibility (set B).

The values of CP for each beat of the song are shown in Fig. 6, where Fig. 6(a) shows the most compatible instrument set ($\overline{CP} = 21$) and Fig. 6(b) the less compatible instrument set ($\overline{CP} = 57$). The histograms of the data from Fig. 6 are shown in Fig. 7.

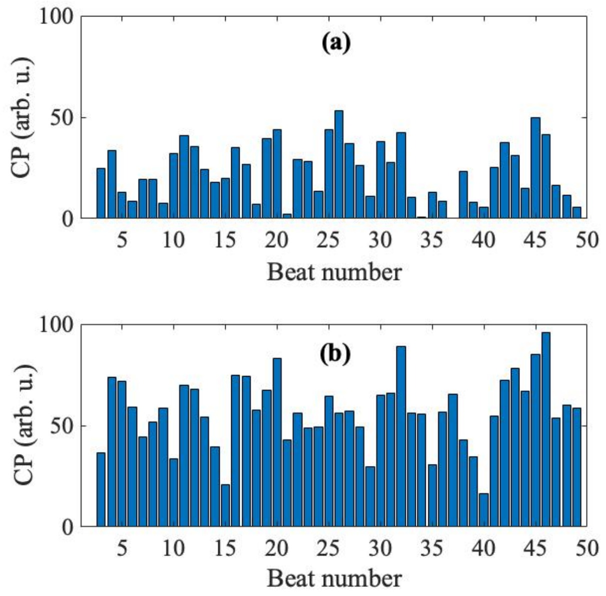


Figure 6: Bar plot of the compatibility parameter of each beat within the piece. In panel (a) ‘bass’ is being played by a bass guitar, ‘chords’ by a balalaika (a type of Russian guitar), ‘melody’ by a vibraphone and ‘high-pitch’ by a violin. In panel (b) ‘bass’ is interpreted by a tuba, ‘chords’ by a Chinese zither, ‘melody’ by a harpsichord, and ‘high-pitch’ by an organ.

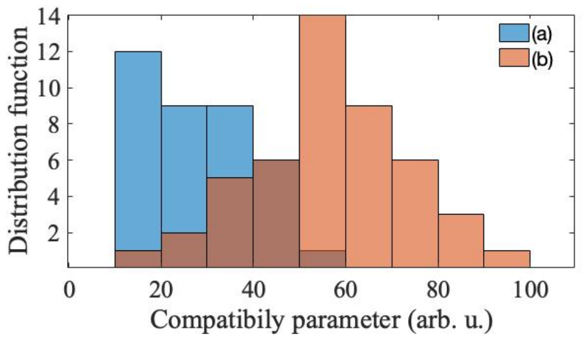
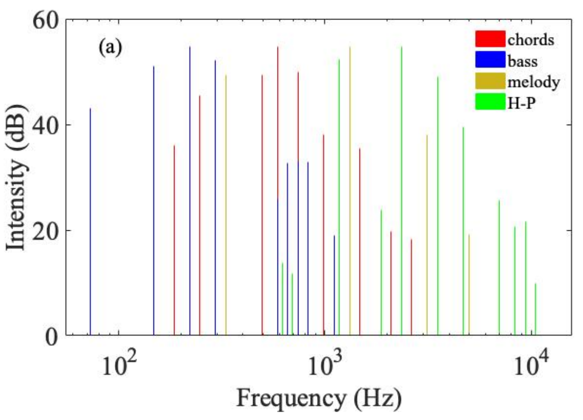


Figure 7: Histogram of the compatibility parameter per beat, where the most compatible set is shown in blue and the less compatible set in orange. The brown area indicates the overlapping between the two histograms. $\sigma = 14$ for set (a) and $\sigma = 17$ for set (b).

Most of the obtained CP values are smaller than the average value from Fig. 5 (63). This is because in the previous example (see Fig. 5) we compared orchestral pieces where many instruments were involved, thus presenting higher spectra overlap.



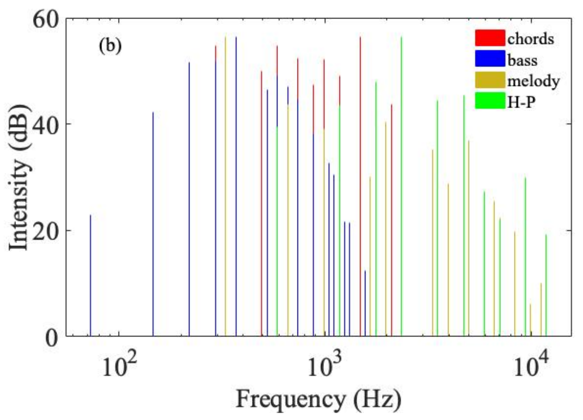


Figure 8: Complete spectrum corresponding to beat number five for instrument sets with good compatibility (a) and bad compatibility (b). The spectrum corresponding to each score is represented in different colors, as shown in the legend. It is worth noting the large spectral overlapping at the central region of frequencies for set b.

We will pay special attention to beat number five. Here set A returns a CP of 13, while set B returns a value of 72. We used this beat to illustrate a clear case of good and bad compatibility. Fig. 8 shows the spectrum of set A and set B at this specific beat, allowing us to easily evaluate the overlap between spectra.

Looking at Fig. 8, we can see that the main reason why set B is not very compatible is that all of its instruments have a lot of spectral information. Specially when we take a look at the central part of the spectrum, we can see many intense harmonics very close to each other. This excess of harmonics causes a lot of spectral overlap, which provokes a feeling of nastiness on the listener.

The instruments in set A, however, do not have as many harmonics. Specially when looking at the 'melody' score: the vibraphone from set A only has four harmonics (that is why it resembles a pure tone), and it does not overlap with any other instrument. The rest of the instruments have more harmonics, but they are arranged in a way that they do not disturb one another.

VI. CONCLUSIONS

After studying the behaviour and features of musical sounds, as well as the newly defined concept of musical compatibility, we feel very satisfied with the results. We believe that we have developed a software that, when used properly, can help music composers choose the instruments to perform in their creations.

Although we worked on a very subjective field, where we did not discard the possibility that some people could enjoy noisy music with a lot of spectrum overlaps (such as hard-rock, punk, metal...), we feel very satisfied with our results, which have proved to work out properly. At the very least in a classical music context.

We must never forget that the compatibility parameter, as we defined it in this project, should only be used to compare instruments playing the same score. If we want to know the amount of spectral overlap within a song for a specific set of instruments, we should instead use the scalar product normalised between 0 and 1. This value will then tell us the percentage of overlapping harmonic components.

To conclude, we believe that working further on the concept of musical compatibility could lead to more sophisticated tools like: a new CP applied to musical notes, an automatic equaliser (frequency balancer), or maybe even an AI performer who instantly reproduces sounds that fill the heard spectrum in an optimal way.

Acknowledgments

I would like to express my gratitude to my tutor Amílcar Labarta, whose thorough support and supervision made this research possible, and my music teacher Fausto Murillo, who inspired this project that I have enjoyed so much.

[1] Claude E. Shanon *Communication in the presence of noise.* Proceedings of the IRE, England (1949).
[2] Steven W. Smith *The scientist and engineer's guide to digital signal processing.* Analog Devices, Inc, England (1997).
[3] H. Fastl, E. Zwicker *Psychoacoustics: facts and models.* Springer (2007).
[4] Elias M. Stein *Fourier analysis: an introduction.* Princeton University Press, (2003).
[5] Mariano J. Merino, Loida Muñoz-Repiso *La percepción acústica, física de la audición.* Universidad de Valladolid (2013).

APPENDIX

```
%First we create the 3D matrixes with all the instruments. The function
%cat() concatenates all the matrixes in the third dimension.

C=cat(3,Abalalaika,Acitarra,Aguitarra,Aharpa,Aharpicordio,Apiano,Asinte);
%7 instruments playing the chords

B=cat(3,Bsinte,Bcellos,Bbasses,Bbass,Btuba);
%5instruments playing the bass

M=cat(3,Mharpa,Mguitarra,Mharpicordio,Mpiano,Msinte,Mcelesta,Mcitarra,Mvibraphone);
%8 instruments playing the melody

HP=cat(3,Pvibraphone,Pflauta,Ppiano,Pviolin,Poboe,Porgan);
%6 instruments playing the high-pitch



Z=zeros(7,5,8,6);   %We will store our results in this matrix

for i = 1:7
    for j = 1:5
        for k = 1:8
            for l=1:6
                %The function juntarR() joins the spectrum of two scores,
                %taking into account masking and that our values are
                %logarithmic.

                %We created 4 matrixes that contain the whole spectrum except for
                %one instrument.

                NoC=juntarR(juntarR(HP(:,:,l),B(:,:,j)),M(:,:,k));
                NoB=juntarR(juntarR(HP(:,:,l),C(:,:,i)),M(:,:,k));
                NoHP=juntarR(juntarR(C(:,:,i),B(:,:,j)),M(:,:,k));
                NoM=juntarR(juntarR(HP(:,:,l),B(:,:,j)),C(:,:,i));

                %Next we compare each instrument with the rest combined.
                %The function comparar returns a CP value for each beat.

                Afit=comparar(C(:,:,i),NoC);
                Bfit=comparar(B(:,:,j),NoB);
                Mfit=comparar(M(:,:,k),NoM);
                Pfit=comparar(HP(:,:,l),NoHP);


                %and finally we compute the average of the values for every beat
                %and save the result into the matrix Z.
                Z(i,j,k,l)= (mitjana(Afit) + mitjana(Bfit) + ...
                    mitjana(Pfit) + mitjana(Mfit))/4;
            end
        end
    end
end
%---------
```