Recognition of Musical Tones

Digital Signal Processing

Instituto Superior Técnico

Vicente Silvestre | 92730
Ivan Figueiredo | 93386

Group 3

April 2022

1. Basic Music Analysis

1.1. Visualization of the signal in the time domain

The signal stored in the file *greensleeves.wav* has a duration of 59.44 seconds, approximately. It was produced by a keyboard and then it was sampled at a frequency of F_s = 44 kHz. The time domain representation of the entire signal is shown in Figure 1.

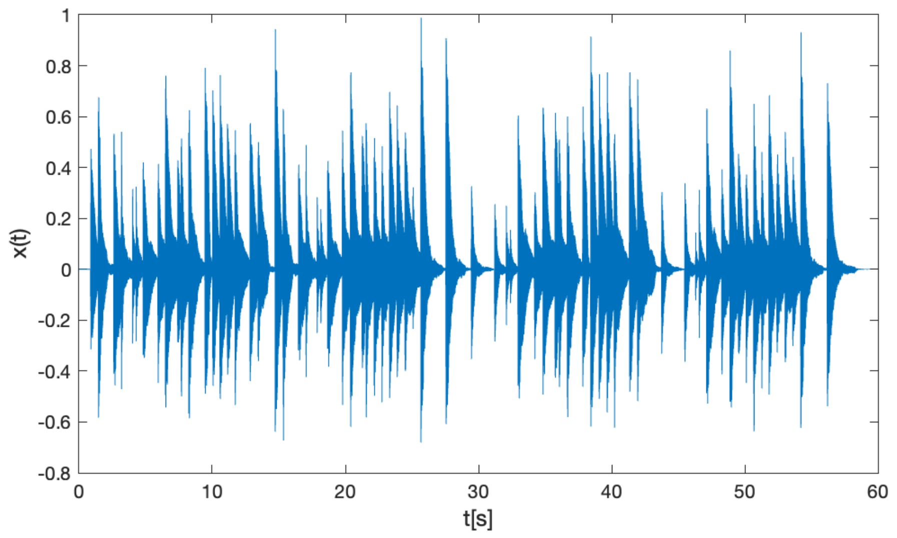


Figure 1 – Time domain representation of the entire signal

1.2. Shape of the signal associated to the first nine tones

As we will discuss later when doing the manual segmentation of the signal, the first nine tones are located in the time range $[0.87,\ 7.39]$ s. In the following figure, it is possible to visualize the shape of those first nine tones.

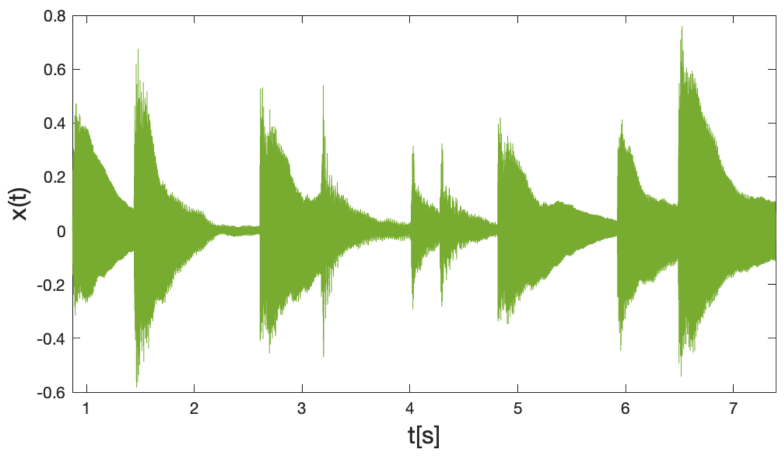


Figure 2 – Time domain representation of the first nine tones of the signal

1.3. Can a signal be down sampled before pitch recognition?

When sampling a signal, the sampling frequency Ω_S has to be, at least, twice the maximum frequency of the signal, Ω_M, i.e:

$$\Omega_S \geq 2\,\Omega_M. \tag{1}$$

Otherwise, information will be lost and the original signal can't be reconstructed. This way, the signal can't be down sampled before pitch recognition

1.4. Fourier spectrum and autocorrelation

Then, in order to obtain the frequency domain representation of this signal, we can apply the Discrete Fourier Transform, which is given by:

$$X(k) = \sum_{n=0}^{N-1} x(n) e^{-j\frac{2\pi}{N}kn}, \tag{2}$$

with $k = 0, ..., N-1$.

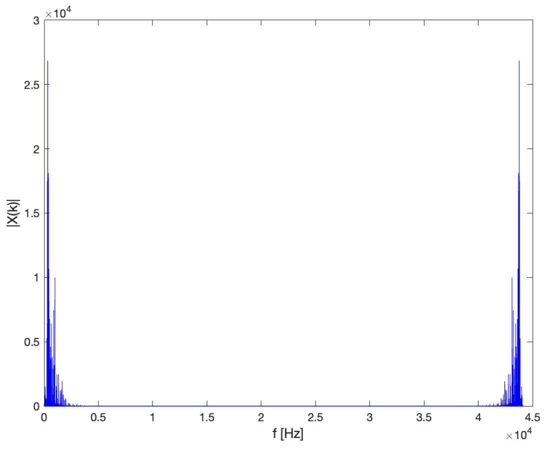


Figure 3 – Magnitude of the signal in the frequency domain

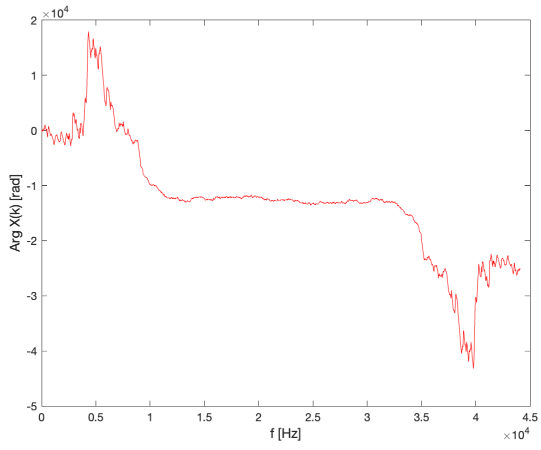


Figure 4 – Phase of the signal in the frequency domain

The following three figures represent, in the frequency domain, the magnitude of the first, second and fifth note of the signal. In these three cases, the fundamental frequency coincides with the peak with the largest magnitude. However, this is not always true: in the subsection 2.2, after segmenting the entire signal, we identify a note whose highest peak (after applying the DFT) does not correspond to the fundamental frequency, but to the first multiple of that frequency, i.e, it is a harmonic.

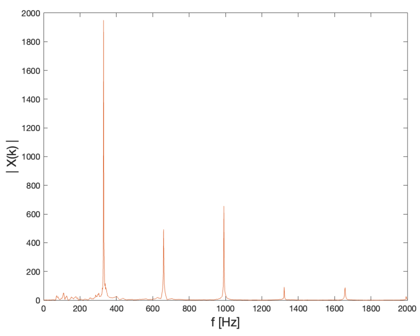


Figure 5 – |X(k)| for the first note

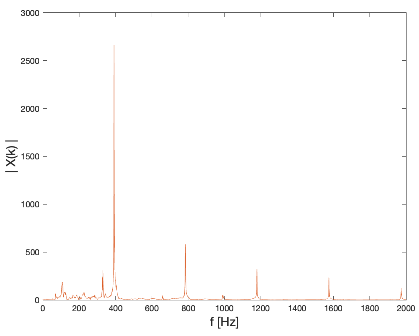


Figure 6 – |X(k)| for the second note

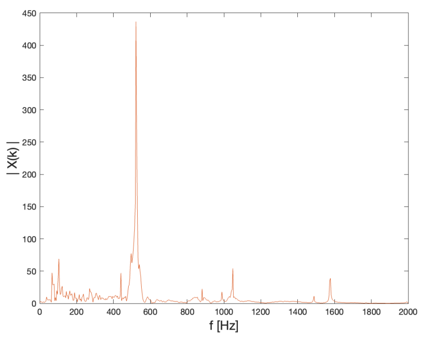


Figure 7 – |X(k)| for the fifth note

Then, we have computed the autocorrelation for each one of the first nine notes. Considering K lags, the autocorrelation for lag k is given by

$$r_k = \frac{1}{T\sigma^2}\sum_{t=1}^{T-k}(x_t-\mu)(x_{t+k}-\mu), \tag{3}$$

where μ and σ^2 stand, respectively, for the mean and variance value of the signal that is being analyzed.

In the following figure, we represent the autocorrelation for the first 5 notes, considering 150 lags.

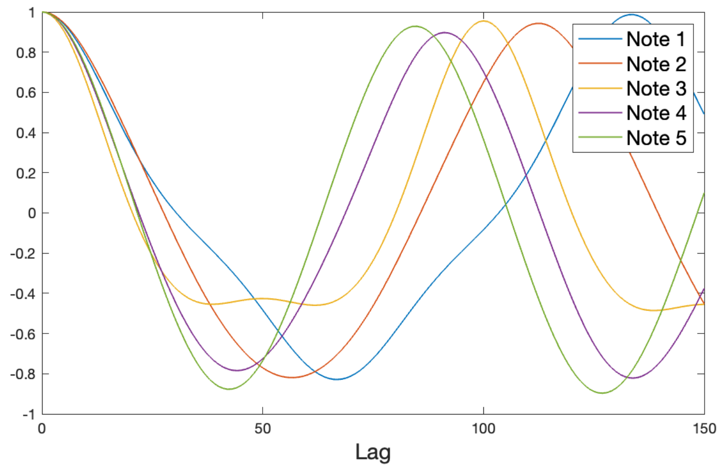


Figure 8 – Autocorrelation for the first 5 notes of the signal

Autocorrelation is an alternative method to compute the fundamental frequency of each note, and it could be used instead of the DFT.

1.5. Time-frequency representation of the signal

The spectrogram we obtained is the following:

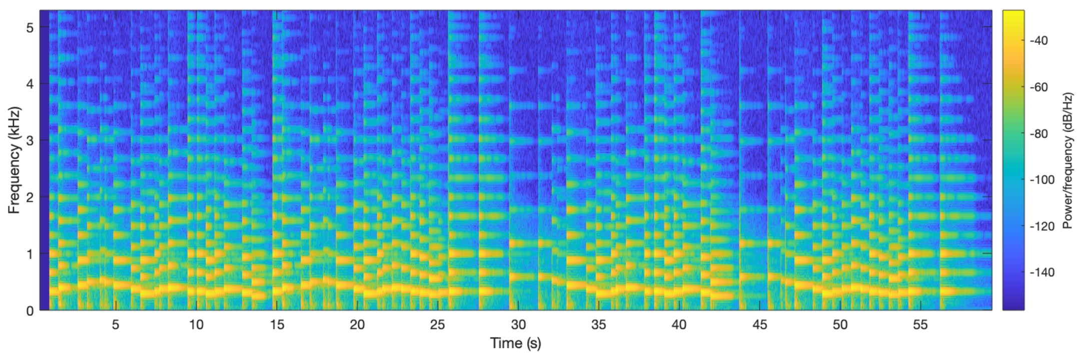


Figure 9 – Spectrogram of the entire signal

The spectrogram proves to be a useful visual representation of the spectrum of frequencies of a signal as it varies with time. The fundamental frequencies of each note, as well as their harmonics, are easily recognized, attending to the colors of the heatmap.

2. Pitch Recognition in Segmented Sounds

2.1. Manual segmentation of the signal

Here, we proceeded to manually segment the notes by observing the signal in the time domain. For the sake of brevity, only the first nine notes were considered.

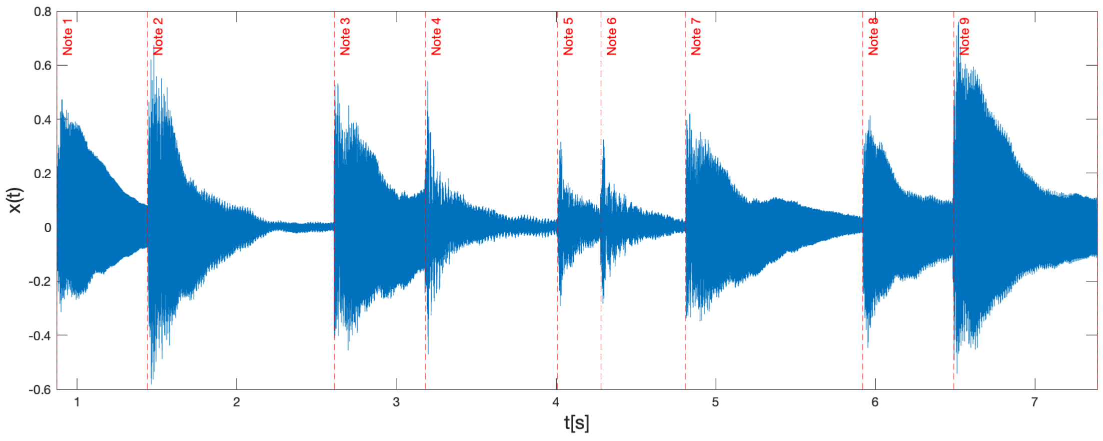


Figure 10 – The first nine segmented notes in the time domain

Tone	$t_{\text{beginning}}$ (s)	t_{end} (s)
1st	0.87	1.44
2nd	1.44	2.61
3rd	2.61	3.18
4th	3.18	4.01
5th	4.01	4.28
6th	4.28	4.81
7th	4.81	5.92
8th	5.92	6.49
9th	6.49	7.39

Table 1: The instants of time that correspond to the beginning and the end of the first 9 notes, obtained manually

2.2. Pitch Recognition Algorithm

2.2.1. Computing the frequency of each note

For the pitch detection algorithm, a fast Fourier transform is obtained in the time domain of each note.

For the Fourier spectra of the first notes analyzed, we considered that the frequency associated with the maximum amplitude is the fundamental frequency of the note played.

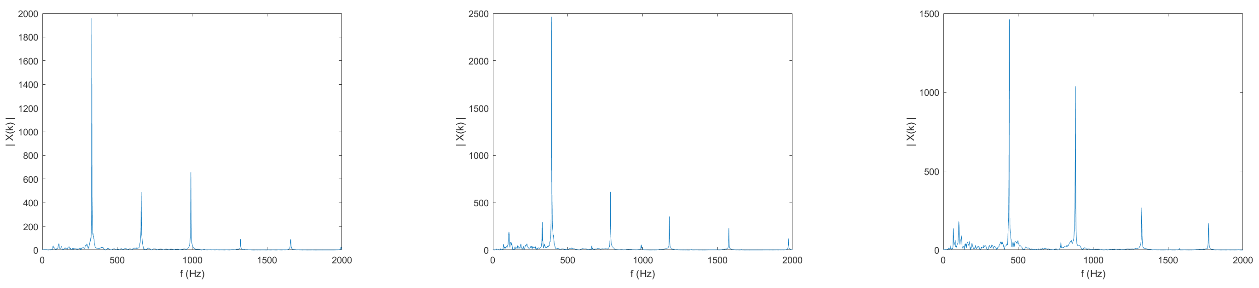


Figure 11 – The Fourier spectra for the first three notes

Although this allowed to correctly identify the pitch of most notes, some were misidentified. For example, note 35, supposedly a C4, was identified as C5 due to the fact that in this particular case, the amplitude of the first harmonic was larger than of the fundamental.

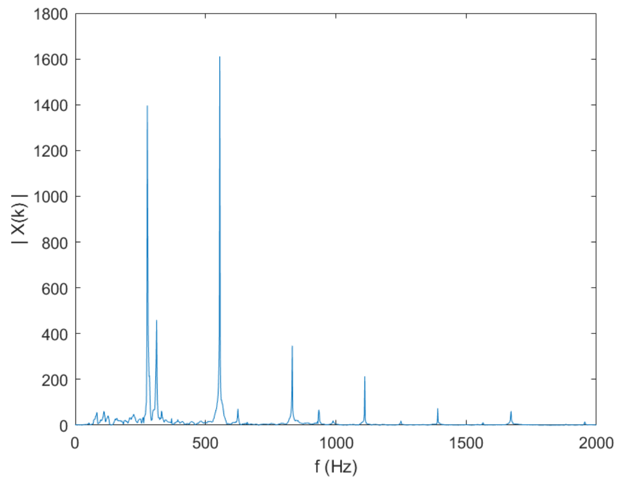


Figure 12 – Fourier spectrum of note 35 (C4)

In order to account for this, we improved the algorithm so as to after finding the frequency with the largest amplitude f_0, the ’subharmonics’ were searched to make sure that the frequency found actually corresponds to the fundamental. The rationale for this is that if we find a relevant peak on the range of frequencies of the first subharmonic ($f_{-1} = f_0/2$), then this frequency f_{-1} is a candidate for the fundamental frequency, and the frequency f_0 found was actually an harmonic of f_{-1}.

Regarding the implementation, we fixed a threshold amplitude of a quarter of the (maximum) amplitude of f_0 in order to consider the subharmonic region a candidate to the fundamental (and thereby discarding the noise effects that are always present). The algorithm performs the search until the second subharmonic ($f_{-2} = f_{-1}/2 = f_0/2^2$), although in this project there never was the case of finding the fundamental in this region.

2.2.1. Computing the pitch in each segment

After using the Fourier transform in order to obtain the frequency of each segmented note, we proceeded to identify the note that corresponds to that frequency. To do that, we created two functions: one of them to compute the octave associated to each note and the other to compute the tone. We considered the octave 4 as the reference octave: this octave includes the frequencies between 261.63 Hz, which corresponds to a C, and 493.88 Hz, which corresponds to a B. This two values can be computed in the following way: since the reference frequency f_0 corresponds to the A4 note, and using the fact that the frequency of the i-th tone can be computed as

$$f_i = 2^{i/12} f_0, \tag{4}$$

it follows that $f(C4) = 2^{-9/12} f_0 = 2^{-9/12} \times 440 = 261.53$ and $f(B4) = 2^{2/12} f_0 = 2^{2/12} \times 440 = 493.88$ Hz, because the C4 note is 9 tones under the A4 note and the B4 note is 2 tones above the B4 note.

Initially, we were assigning the C4 note if the frequency value obtained using the Fourier method was in the interval [f(C4), f(C4#)[. The same was done for the other notes. However, we realized that, for example, a note that was supposed to be a C4 was being classified as a B3, because the corresponding frequency was something like 260 Hz, which is near to the beginning of a C4, although it is still a B3. So, in order to solve this problem, which probably is associated to noise in the corresponding segmented signal, we decided to associate an error of δ = 5 Hz to each note: this way, the beginning of each note corresponds to a frequency which is 5 Hz smaller than the supposed frequency obtained with the previous computations. So, in order to clarify this, we present in the following table the range of frequencies considered to each note of the octave 4.

Tone	Range of frequencies (Hz)
C4	[256.63, 272.18[
C#4	[272.18, 288.66[
D4	[288.66, 306.13[
D#4	[306.13, 324.63[
E4	[324.63, 344.23[
F4	[344.23, 364.99[
F#4	[364.99, 387.00[
G4	[387.00, 410.30[
G#4	[410.30, 435.00[
A4	[435.00, 461.16[
A#4	[461.16, 488.88[
B4	[488.88, 518.25]

Table 2: Range of frequencies associated to each note of the octave 4

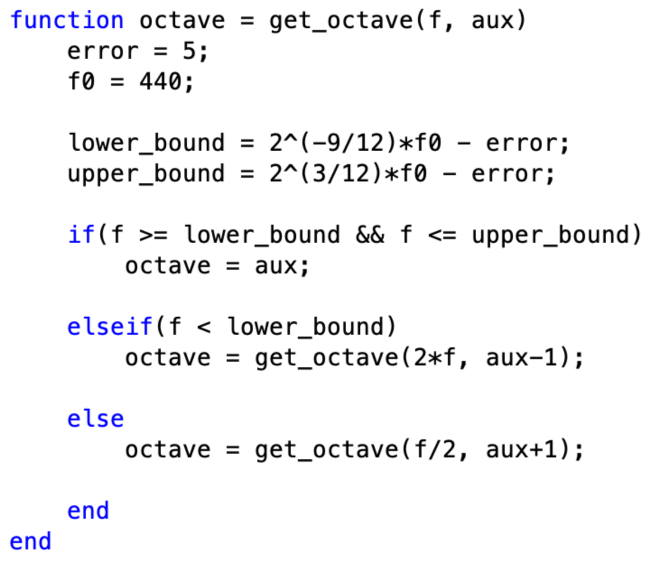
The upper bound presented in the table above for the B4 note is computed as: f(C5)$-\delta$ = $2^{3/12} \times 440 - 5 = 518.25$ Hz.

The function created to detect the octave of each note is presented in figure 14. This function has as inputs the frequency of the segmented note and, as default, we consider that the corresponding octave is the octave 4, so we assign the variable `aux` to 4 before calling this function. So, if the frequency is in the admissible range corresponding to the octave 4, the output is immediately 4. Otherwise, the function is called recursively until the frequency received as the input is in the admissible range: if the initial frequency is bigger than the `upper_bound` of the reference octave, we call this function again with a frequency given by an half of the previous frequency; if the initial frequency is smaller than the `lower_bound`, this function is called again with a frequency given by the double of the previous frequency. This can be done because the frequency of the same “notes” of different octaves are related by powers of 2. In fact, since the number of tones between the same “note” in different octaves is 12, it comes that

$$f(X)\big|_{\text{octave } i} = 2^{i-j} f(X)\big|_{\text{octave } j}, \tag{5}$$

where X can be A, A#, B, C, C#, D, D#, E, F, F#, G or G#. As an example, we have that f(C6) = $2^{15/12} f_0$ because it is 15 tones above the fundamental frequency, and f(C4) = $2^{-9/12} f_0$ because it is 9 tones under the fundamental frequency. And, in fact, f(C6)/f(C4) = 2^2, which can be also written as

$$f(C)\big|_{\text{octave } 6} = 2^2 f(C)\big|_{\text{octave } 4}. \tag{6}$$



```
function octave = get_octave(f, aux)
    error = 5;
    f0 = 440;

    lower_bound = 2^(-9/12)*f0 - error;
    upper_bound = 2^(3/12)*f0 - error;

    if(f >= lower_bound && f <= upper_bound)
        octave = aux;

    elseif(f < lower_bound)
        octave = get_octave(2*f, aux-1);

    else
        octave = get_octave(f/2, aux+1);

    end
end
```

Figure 13 – `get_octave()` function

Once the octave is computed, the code calls the function `get_tone()` with a frequency in the admissible range of octave 4. To do this, we use the computed value of the octave to convert that note in a note that belongs to the octave 4. In fact,

$$f(X)\big|_{\text{octave } 4} = 2^{4-i} f(X)\big|_{\text{octave } i}, \tag{7}$$

with $0 \leq i \leq 8$. Then, we assign tone 1 to a C, tone 2 to a C#, tone 3 to a D, and so on, until tone 12, which corresponds to a B. So, after using these two functions, `get_octave` and `get_tone()`, the note of each segmented tone is finally identified.

2.3. Evaluation of the recognition algorithm performance

For the first 9 notes, the correspondent frequency and pitch are presented in the following table.

Note	Frequency (Hz)	Pitch
1	330	E4
2	392	G4
3	441	A4
4	494	B4
5	522	C5
6	495	B4
7	440	A4
8	369	F4
9	294	D4

Table 3: Frequency and pitch obtained for the first 9 notes

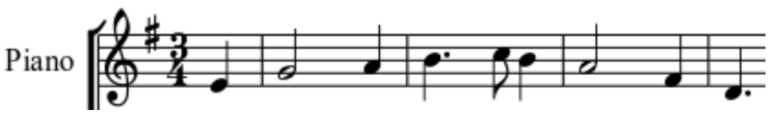


Figure 14 – The first nine notes of the Greensleeves song

The notes presented in the sheet music coincide with the ones computed with the developed algorithm.

3. Pitch recognition and sound segmentation

3.1. Tone segmentation and pitch recognition algorithm

Our approach to automatically segment the tones was done on the time domain and consisted in detecting the sudden rise of intensity associated with the attack of a new note being played.

One of the crucial points in the development of this algorithm was the initial stage of pre-processing the sampled signal. Firstly, we filtered the signal by selecting the maximum value within the range of a moving window of a particular size ($w = 2000$), and dropping all other values (Figure 15).

An extra step for the correct functioning of the algorithm had to be done. In some particular cases, such as the one on Figure 16 (left), the maximum value of the attack isn't reached instantly, passing through intermediary values. Since the algorithm considers the difference

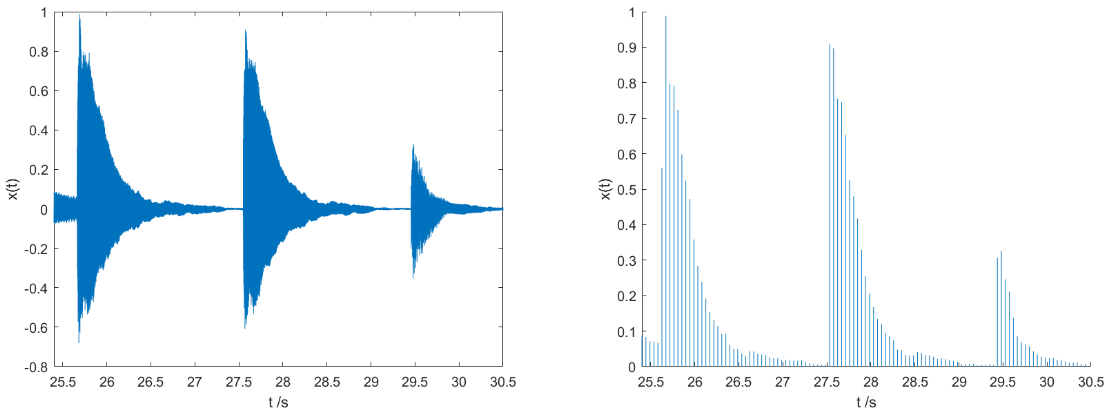


Figure 15 – Three notes of the original signal (left) and the same notes after applying the filtering described (right)

between consecutive points and might misidentify this consecutive increases as two different attacks, these intermediary values were dropped (as seen in Figure 16, right). This was the last step of the preprocessing.

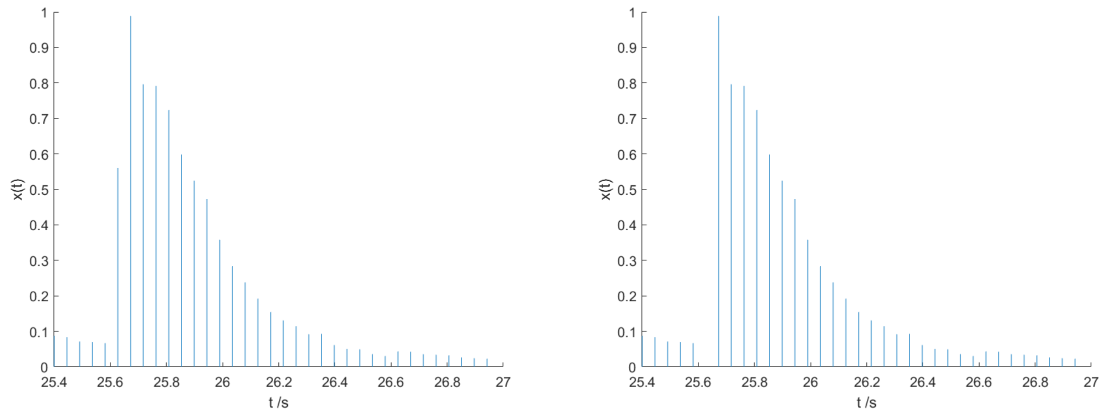


Figure 16 – Signal before dropping (left) and after dropping (right)

In this algorithm, to detect the beginning of a new note (and, consequently, the end of the previous one) we considered that the variation between consecutive maximums had to be equal or greater than +0.1.

With this processing complete, the algorithm is finally able to detect the attack of a new note by finding considerable rises in intensity between consecutive points: if this difference is larger than the threshold value, then it is considered the begin of a new note.

Additionally, a constant offset of 0.05 s was added to the time intervals to account for systematic errors associated with the nature of the signal, as well as the sampling and filtering used.

After being automatically segmented, we used the algorithm previously described in section 2.2. to identify the pitch associated to each segmented note.

3.2. Evaluation of the tone segmentation and pitch recognition algorithm performance

The automatic segmentation algorithm identifies 72 notes in the entire signal, which equals the number of notes that we count manually. Then, we compared the obtained pitches with the notes in the sheet music of this song, and we identified 4 different notes: our algorithm identifies 2 notes D that should be D#, accordingly to the sheet music, and 2 pitches C that were supposed to be C#. However, then we used a mobile app to detect the frequencies corresponding to those 4 pitches in the provided signal, and, in fact, our algorithm was right, which means that those 2 supposed D# were, in fact, played as D in the keyboard, and those 2 supposed C# were played as C.

At last, we synthesized a signal using only the fundamental frequency for each note. This was done by assigning to the range between the beginning and the end times of each note a sinusoidal function with frequency equal to the fundamental one.

The synthesized signal can be heard ***here***.

Note	Pitch	Note	Pitch	Note	Pitch	Note	Pitch	Note	Pitch
1	E4	16	E4	31	G4	46	E4	61	F#4
2	G4	17	F#4	32	F#4	47	F#4	62	D4
3	A4	18	D#4	33	E4	48	G4	63	E4
4	B4	19	B3	34	D#4	49	E4	64	F#4
5	C5	20	E4	35	C#4	50	E4	65	G4
6	B4	21	G4	36	D#4	51	D4	66	F#4
7	A4	22	A4	37	E4	52	E4	67	E4
8	F#4	23	B4	38	E4	53	F#4	68	D#4
9	D4	24	C5	39	D5	54	D#4	69	C#4
10	E4	25	B4	40	D5	55	B3	70	D#4
11	F#4	26	A4	41	C5	56	D5	71	E4
12	G4	27	F#4	42	B4	57	D5	72	E4
13	E4	28	D4	43	A4	58	C5	-	-
14	E4	29	E4	44	F#4	59	B4	-	-
15	D4	30	F#4	45	D4	60	A4	-	-

Table 4: Detected pitch for each one of the 72 notes