Coding Project 2: Parsing musical frequency signatures

Wietse Vaes

Abstract

Ever wanted to listen to only one instrument of a certain song? Neither have I. Nevertheless, there is a solution to that! Say you don't know any musicians, the magic of mathematics can help you! Using a transformation on the digitization of a song, find the frequencies of certain instruments in that song. With those frequencies, we will be able to filter out the rest an play only one instrument.

1 Introduction

In order to filter out, or put emphasis on, certain instruments we need to know what differs them from other instruments or sounds in general. In order to do this, we need to ask ourselves what sound even is. In short, sounds are waves or vibrations. Our ears pick them up and our brains translate them so that we can hear what we hear. With these waves/vibrations come certain frequencies and these are what differs each sound. Every instrument plays at a certain frequency and even these frequencies are general ranges, since instruments can make different kinds of sounds. This goes from strong to silent, muffled to clear and also high or low notes. In this paper, we will be trying to distinguish two instruments: a baseline and a guitar. This will be accomplished using their different associated frequencies.

In order to know which frequencies are associated to the baseline and which to the guitar we will be using the Gabor transform on a song called "I'm shipping up to Boston" by the dropkick Murphys. However many instruments this has, we are going to focus on these two instruments. A way of mitigating the other instruments will be explained while implementing the method. The advantage of this song is that both instruments come in at different times and are thus more recognizable. After all is done, we look forward to only listening to the baseline and guitar.

2 Theoretical Background

In order to find the frequencies we would like to filter to, we will be developing the Gabor transform, otherwise known as short-time Fourier transform. As the name tells us, it will take the Fourier transform over a short-time or localized data. After this a discrete version will be explained such that it can be applied to a data set, the song.

2.1 The Gabor Transform

While the Fourier transform is a good solution to finding out which frequencies are generally used in a function or data, we might want to know when these frequencies come into play. For this, the Fourier transform is not a suitable option. In order to do this, one might want to look at the function around a certain time and dampen out the rest, after that we could look at the frequencies that happen around that time. This is where the Gabor Transform, or short-time Fourier transform, comes into play.

The name says it all: we want to find a Fourier transform over a shorter time. Therefore, we would like to dampen out the function before finding out which frequencies play a large part around a certain time. We're going to do this with the function $g(\tau - t)$ which serves as a time filter for localizing the function over a specific window of time. For this short paper we will mainly use the Gaussian-like function

$$g(\tau - t) = e^{-a(\tau - t)^2},$$

where a is chosen as a way to limit/expand the width of the function. This needs to be chosen carefully as we still want enough spectral information inside the window that we filter. After applying this time localisation, we take the fourier transform of the new function, this is called the Gabor transform. Therefore, the full general Gabor transformation is given by

$$\mathcal{G}[y](\tau, k) = \int_{-\infty}^{\infty} y(t) g(t - \tau) e^{-ikt} dt.$$

A visual representation of the localization is given in figure 1. While the

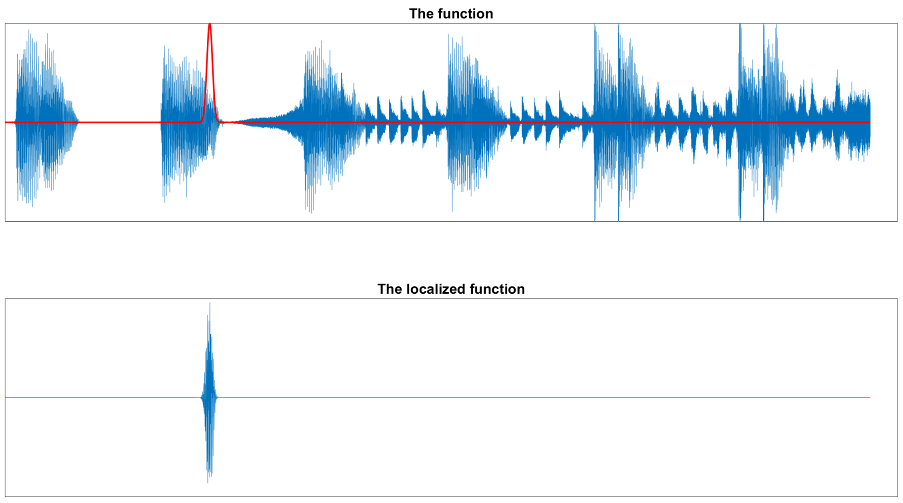

Figure 1: The effect of the localization on a function (blue) using a Gaussian smoothener (red).

visual representation of the effect in the frequency domain is given in figure 2.

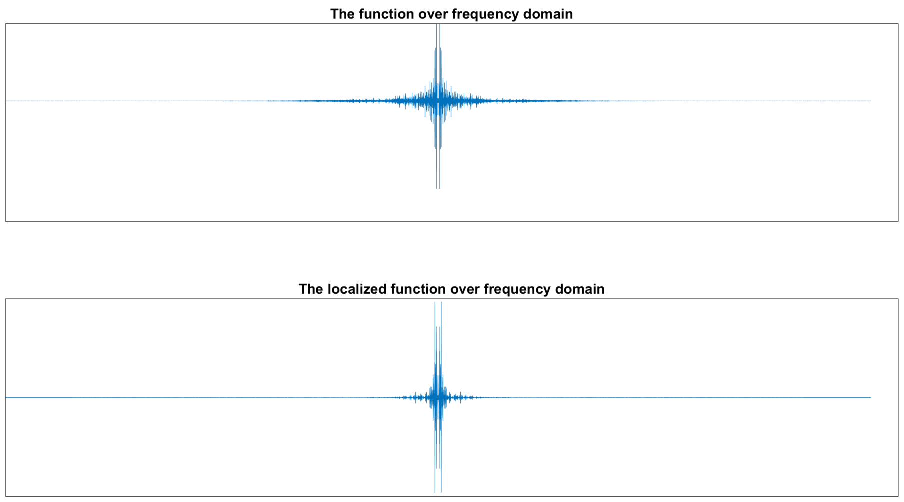

Figure 2: The effect of the localization on a function using a Gaussian smoothener over the frequency domain.

Here there is a clear difference and some frequencies don't play as big of a part as they did before. Note that this grid is very refined, thus smaller, but essential, changes are not very noticeable.

2.2 The Discrete Gabor Transform

Multiple characteristics of the Gabor Transform can be found, this is however not the interest of this short paper. One restraint which is important for this paper is that this is all defined on a continuous spectrum. This causes problems since we would like to use this transform in a numerical setting! That is why the discrete Gabor transform comes into play.

It's not very difficult to derive the discrete Gabor transform and is very much analogue to the discrete Fourier transform but now with that localization on a certain time. Looking at the discrete Fourier transform, one finds that the discrete Gabor transform on a finite grid becomes

$$\mathcal{G}[y](\tau, k) = \sum_{n=0}^{N} y(\frac{n}{N})g(t_n - \tau)e^{-\frac{2\pi}{N}ik}.$$

Now of course, we will be looking at only discrete frequency values and values for τ. Therefore we get a matrix

$$\mathcal{G}_{ml}[y] = \sum_{n=0}^{N} y(\frac{n}{N}) g(t_n - \tau_l) e^{-\frac{2\pi}{N} i k_m},$$

with τ_l and k_m equidistant grids. This will then be able to give something called a spectorgram, which we will go deeper in on in the next section.

3 Implementation

Now that the theory has been explained, it will be implemented into MATLAB in order to process the given data, also known as the song in this case, in order to get both instruments separated. The song is loaded in as a very large vector, in order to actually be able to grade it, we will cut the song into 4 equal parts. This was however not necessary, but it is possible due to the nature of the Gabor transform. Thereafter we discretize time, frequencies and the time points around which we will be localizing. Note that for all four segments we will be using the same time and localization points, this does not matter since they will only be used to created the Gaussian smoother. The only thing that matters is that they are consistent and that a consistent bandwidth is chosen.

Now that everything is loaded and discretized, we want to discover the frequencies at which both instruments operate. In order to do so, we take the Gabor transform of our data. We do this by going through all the localization points and localize our data. The smoothing function here is chosen by the Gaussian

$$g(t, \tau) = e^{-a(t-\tau)^2},$$

with a chosen such that the bandwidth is not too small, nor to large. Thereafter we calculate the Fourier transform using the `fft()` function. Using this, we could see the important frequencies, but there are multiple instruments in the song. Luckely the baseline and guitar are clearly the most dominant instruments, therefore we would like to soften the effect of the other frequencies. This can be done using another Guassian like filter on the spectral domain. The spectral domain of the data gives us the amplitudes of the frequencies. After finding the largest amplitude over only the positive frequencies, we get the most important frequency. Because of the nature of this data set, the frequencies are pretty much mirrored around the origin. There we would also have the negative of this frequency as an important frequency. We now want to smoothen around both frequencies, since the other has the opposite sign, we can just use the filter

$$f(k) = e^{-\frac{1}{L}(|k|-k_{peak})^2},$$

where L is specifically chosen for this domain. Usually a good guess for L is the length of the domain. After that we apply this filter, for visualization we take the absolute value of this vector and shift the vector. This is needed because of how MATLAB calculates the Fourier transform. This spectogram will be shown in the results section.

From this spectogram we can see that the frequencies for the base are around $0-300$ and the guitar frequencies are around $300-\ldots$. This resonates with the actually frequencies on which a base and guitar operate, respectively, $60-250$ and $80-1200$. Therefore we want to only listen to these frequencies of the song. We do this by transforming the song to its frequency domain and then filtering with a heavy-side function which is comprised of ones on only the aforementioned frequencies we want and 0 everywhere else. Doing this will result in data representing the baseline and guitar respectively.

4 Results

Since technology is not as advanced to have widely available paper with speaker, we will only be showing the spectograms and the results from that. The reason this song was chosen for this paper was that the instruments come along separately. Therefore this should be also seen in the spectogram. The spectogram $\mathcal{S}$ of the first section can be seen on figure 3, for visual purposes, we show $\log(\mathcal{S}+1)$.

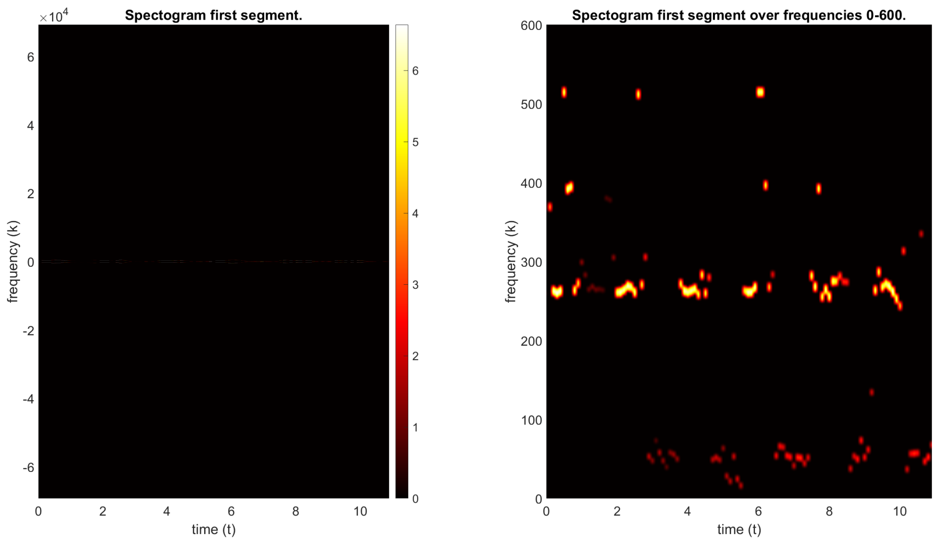

Figure 3: The spectogram of the first segment of the song. The left figure is the full spectrum. If one were to squint their eyes, one can see that there is a thin line of red. On the right we have zoomed in on this image to only see the positive frequencies up to 600. The colorbar holds for both figures and one can see clear activities around the 300 hertz mark.

Here we see, after zooming in, that most frequencies are around and under the 300 mark, A couple of high amplitudes can be seen above, but most are around and under this mark. If we were to now plot the one of the next segment, we get figure 4:

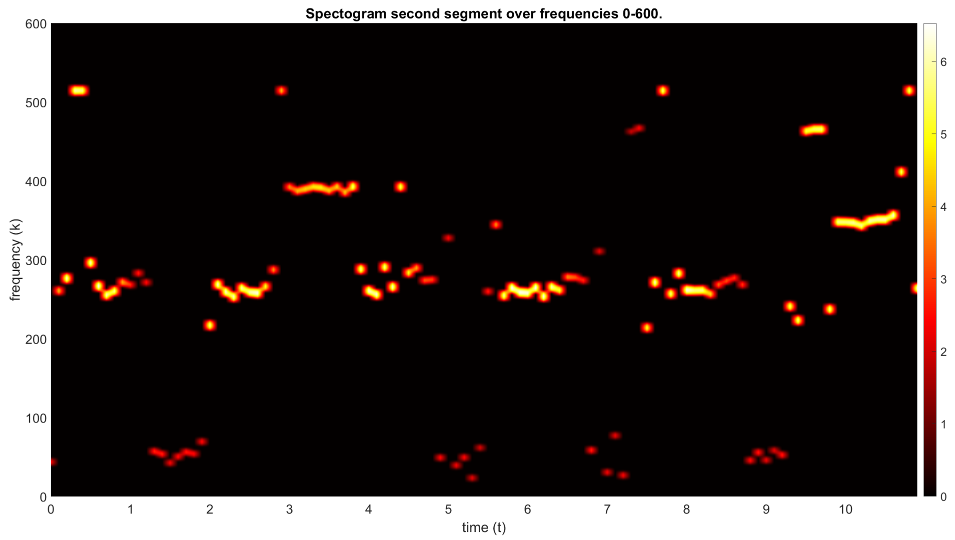

Figure 4: The spectogram of the second segment of the song zoomed in over the positive frequencies up to 600 hertz. Certain activities are noticeable above 300 hertz. This coincides with the beginning of the guitar.

Here we see more activities above the 300 mark. There were some activities above the 300 mark, but these were minimal and necessary since theory and real life don't always line up. After the second segment begins there is, in general, more activity and as said before, this song has been chosen since there is a clear beginning to the guitar part. In order to further prove that the frequencies under 300 are the baseline, we show the last segment:

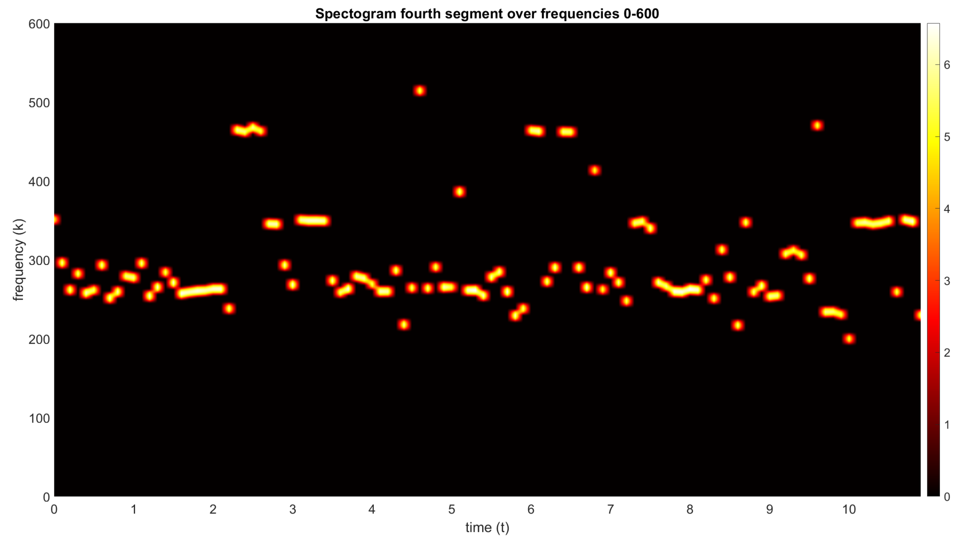

Figure 5: The spectogram of the fourth segment of the song zoomed in over the positive frequencies up to 600 hertz. As in the song, the frequencies around the 300 hertz mark have an increased amount of activities. As in the song, the baseline ramps up.

Here we see the importance of the frequencies under 300 ramp up, in the song the baseline also really ramps up. Further proving our point. Because of this, certain thresholds have been chosen in the implementation.

5 Conclusion

Using the a localized Fourier transform, otherwise known as the Gabor transform, we were able to distinguish the frequencies associated to different musical instruments and filter them out, such that we can hear the other(s). Now everyone can listen to only the baseline or the string instruments of ”I’m shipping up to Boston”.

However basic this may seem, a general insight in the structure of the song can be given through this. The applications of this are immense. Say we can do this for more complex structures, such as voices, we would have their digital expression and therefore we might be able to filter them and only them out. This would be revolutionary for married men/women, teenagers, etc. The filtering out of a certain sound would, in general, be far-reaching.

There are, however, multiple limitations. The first is that we manually had to decide the thresh holds. This is not a difficult blockade to overcome. But then again, this song had a clear difference between when each instrument came into play. Unless we have a situation where only the certain sounds we want to filter out, are played, we need human interaction to distinguish the different things. Other drawbacks may include that, even though it is localized, we still need a bit more information before we can decide which frequencies can be associated to which noise. Also, what if we have a complex noise that deals over all frequencies, or certain noises have overlapping frequencies. A last drawback is that the noise after filtering does not perfectly sound like it is. We would need pretty precise frequencies in order to find the perfect filter. Perhaps AI is a solution to fine tuning this filter.

Acknowledgment

I would like to acknowledge the professor for giving us the parameters used in the smoothening functions instead of letting us suffer.