Assignment 2

Zaineb Amor
Erik Engelhardt

April 2019

1 Introduction

In this assignment we have made a feature extractor that extracts important features of the sound of people humming music. The extractor is designed to correspond with the way that humans percieve music and sound. In other words, it converts sound intensitity to a logaritmic scale and is quite robust against frequency tranposition. This extractor is meant to extract the important features of the humming sound recordings in order to enable a HMM to more easily model the sound and thereby make accurate predictions about what song was being hummed in the sound file.

2 Data investigation

We started the investigation by listening to the sound files and plotting the corresponding signals for intervals of 200 samples as indicated in the instructions. We then looked at the pitch and intensity profiles.

The figures 1 and 2 display the intensity profile respectively on the linear scale and on the logarithmic scale. This is done to account for the fact that human hearing percieves sound to be logarithmic rather than linear. This means that if you double the intensity of a sound a human will not percieve it as twice as loud, so our extractor accounts for that behaviour of human hearing by converting the intesity to log intensity.

The figures 3 and 4 display the pitch profile on the linear scale and on the algotithmic scale respectively. The pitch values on the logarithmic scale were restricted to the range [100 300]. The peaks that we observe can be explained by noise. The relevant parts of the pitch profile are then the somewhat constant segments that separate the noise peaks. We can observe some sort of proportinality between the pitch values of the first and second song on those segments. This behavior is expected since they are the same melody hummed in different pitchs. The pitch profile of the third song is different because it is a different song.

The previous plots were produced using the code displayed on the figures 5 and 6

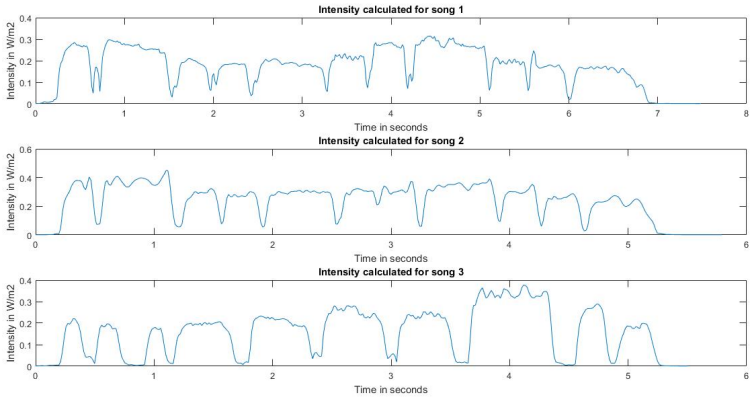

Figure 1: Intensity profile over time.

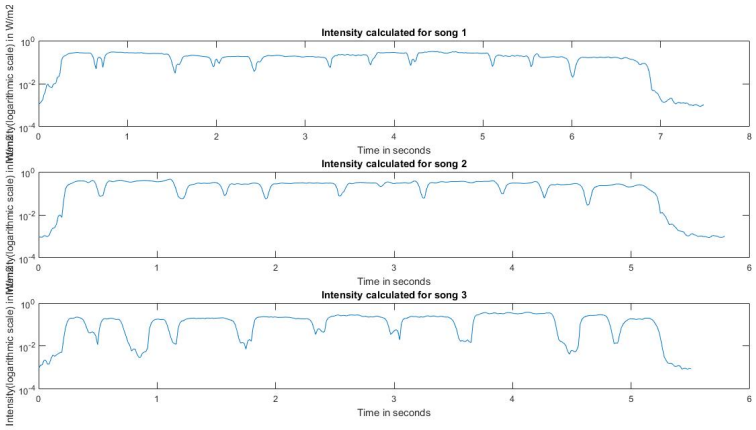

Figure 2: Intensity profile on logarithmic scale

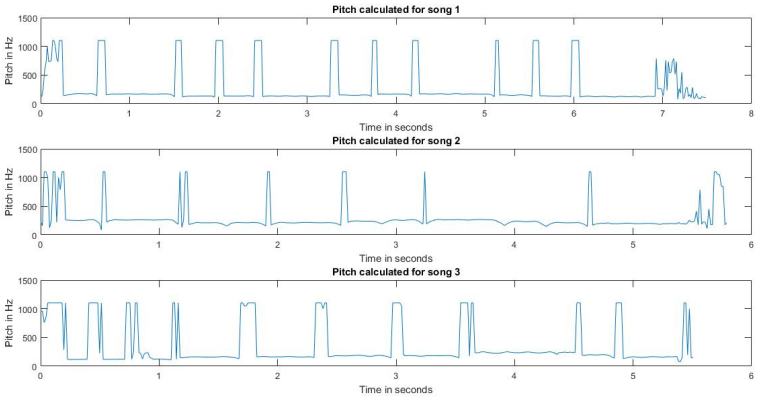

Figure 3: Pitch profile over time.

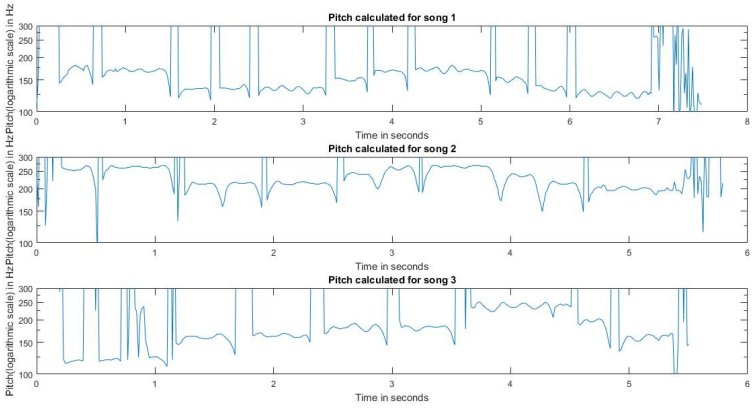

Figure 4: Pitch profile on a logarithmic scale

```
%% Pitch
ax1 = subplot(3,1,1)
plot(t1,P1)
xlabel('Time in seconds')
ylabel('Pitch in Hz') % on the logarithmic scale
title('Pitch calculated for song 1')
ax2 = subplot(3,1,2)
plot(t2,P2)
xlabel('Time in seconds')
ylabel('Pitch in Hz')
title('Pitch calculated for song 2')
ax3 = subplot(3,1,3)
plot(t3,P3)
xlabel('Time in seconds')
ylabel('Pitch in Hz')
title('Pitch calculated for song 3')
gca=[ax1 ax2 ax3]
%set(gca, 'YScale', 'log')
%set(gca, 'YLim', [100 300]);
```

Figure 5: Code to plot the pitch

```
%% Intensity
ax1 = subplot(3, 1, 1);
plot(t1,I1)
xlabel('Time in seconds')
ylabel('Intensity in W/m2') % in the logarithmic scale
title('Intensity calculated for song 1')
ax2 = subplot(3, 1, 2);
plot(t2,I2)
xlabel('Time in seconds')
ylabel('Intensity in W/m2') % in the logarithmic scale
title('Intensity calculated for song 2')
ax3 = subplot(3, 1, 3);
plot(t3,I3)
xlabel('Time in seconds')
ylabel('Intensity in W/m2') % in the logarithmic scale
title('Intensity calculated for song 3')
%set([ax1 ax2 ax3], 'YScale','log');
```

Figure 6: Code to plot the intensity

To understand the relationship between the noise, the pitch and the intensity we plotted the intensity and pitch on a logarithmic scale, while constraining the pitch values to the interval [100 300] as shown in 7. We can see that the peaks in the pitch profile coincide with a decrease in intensity, this reinforces our belief that the peaks are due to noise.

An important step when extracting the features is to distinguish between the real signal and the noise. We therefore look at the behaviour of the pitch, intensity and correlation to determine which parts are noise and which are signal.

The boundaries displayed on figure 8 will be used to filter the noise when extracting the features.

3 Extractor specifications

When extracting the noise we account for the nature of human hearing by considering the logarithm of the intensity and the logarithm of the pitch. The first step is then to convert the intensity and the pitch to their logarithm.

The second step is to detect the noise. To do so, we use the three features already extracted: the pitch, the intensity and the correlation. A vector containing boolean values indicating the existence of noise or not for each sample is created according to the figure 9.

To define the new features, we start from the definitions of an octave and a semitone. In fact, if we consider p1 as our reference pitch, an octave is defined

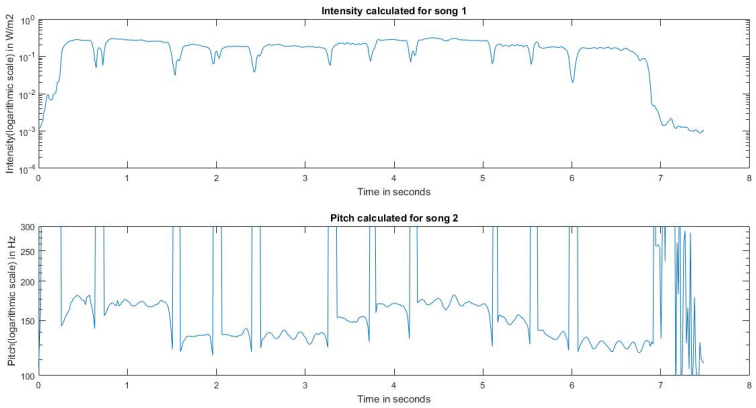

Figure 7: Intensity and pitch profile on a logarithmic scale

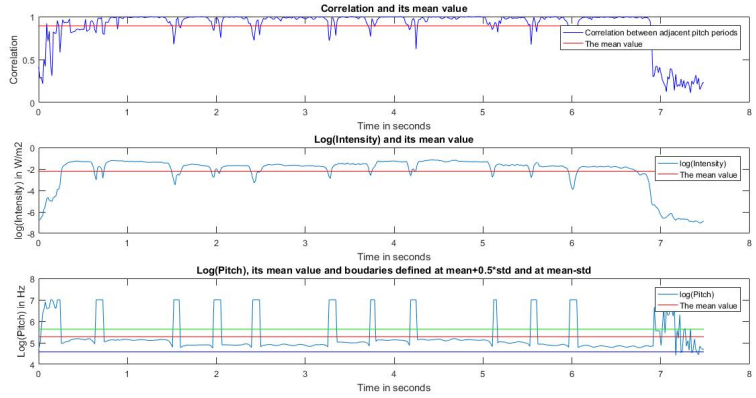

Figure 8: Correlation, log(Intensity) and log(Pitch) with the values of their statistics for the first song

```
meanI = mean(I);
meanR = mean(R);
noise = zeros(1, length(tl));
for i=1:length(noise)
    if ((I(i) < meanI) & (R(i) < meanR))|(P(i) < mean(P) - std(P))|(P(i) > mean(P) + 0.5*std(P)) ;
        noise(i) = 1;
    end
end
```

Figure 9: Detection of the noisy samples

```
P_estim2 = 12*log(P/minP)/log(2)+1;
P_estim2(ind_noise)=0.1*rand(1,length(ind_noise));
```

Figure 10: Semitones and noise

as pO = 2×pmin. An octave can be divided in 12 semitones such that each semitone i can be defined (using pmin and pO) as

$$log(pi) = log(pmin)+(i-1)\times\frac{log(pO/pmin)}{12} = log(pmin)+(i-1)\times\frac{log(2)}{12} \quad (1)$$

In the case where we only have one octave each semitone i associated to the pitch pi is defined as

$$i = 1 + 12 \times \frac{log(pi/2 \times pmin)}{log(2)} \quad (2)$$

We can prove that this holds when we have more than just one octave. Lets assume that i=1 in the second octave i.e. when considering the reference pitch as twice the pmin already defined, we can show that i= 12 when considering the reference pitch as pmin (first semitone of the second octave coincides with the last semitone of the first octave). This can be generalized via

$$log(pi) = log(2\times pmin)+(i-1)\times\frac{log(2 \times pO/2 \times pmin)}{12} = log(pmin)+(11+i)\times\frac{log(2)}{12}$$

We define the semitone according to 1. After that we replace the noise values with small random values definied in figure 10. This way i=0 will indicate silence.

4 Results

The figures 11, 12, 13 display the results of the extraction. To generate 12, we used the funtion dtw (dynamic time warping) in matlab so that the two signals have the same length and the comparaison becomes more meaningful.

Since human hearing interprets sound frequencies as differences between each note, by changing the musical scale a song is sung in we can change all the frequencies but still perceive it as the same song. So we want our feature extractor to account for that. By multiplying the frequencies by 1.5, we can see that the transposition does not affect the features extracted much. The mean distance between song 1 and its transposition is 0.0519 whereas the distance (distance between each two corresponding features and summed for all features) between them is 25.8440. We can see on the figure 12 that the feature extractor extracts the semiphone of the frequencies and that these coindice to a large degree for song 1 and 2 which is expected since they are the same song. The distance calculated between these two sequences of features is

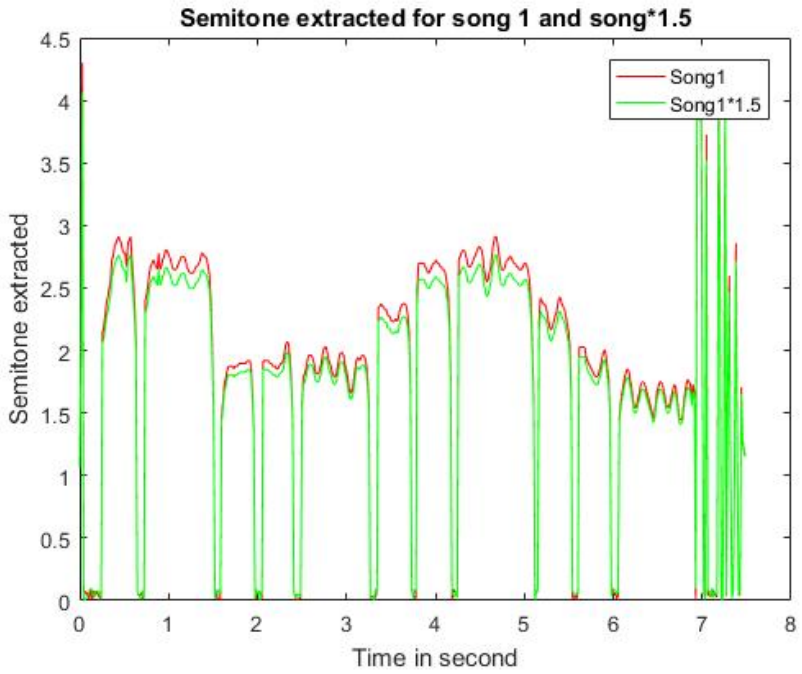

Figure 11: Semitones extracted for the first song and the first song transposed by 1.5

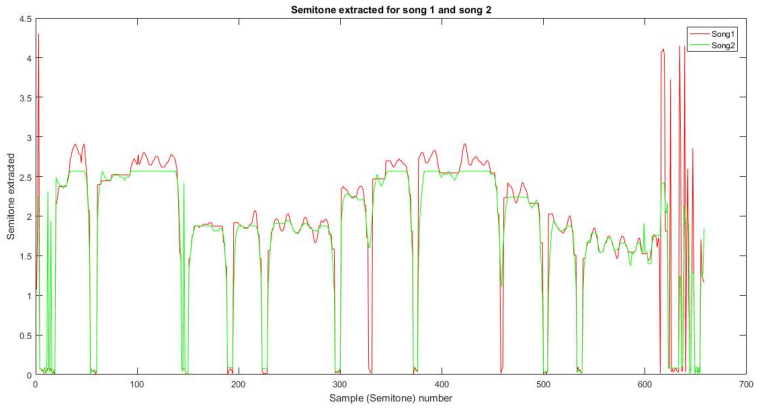

Figure 12: Semitones extracted for the first song and the second song

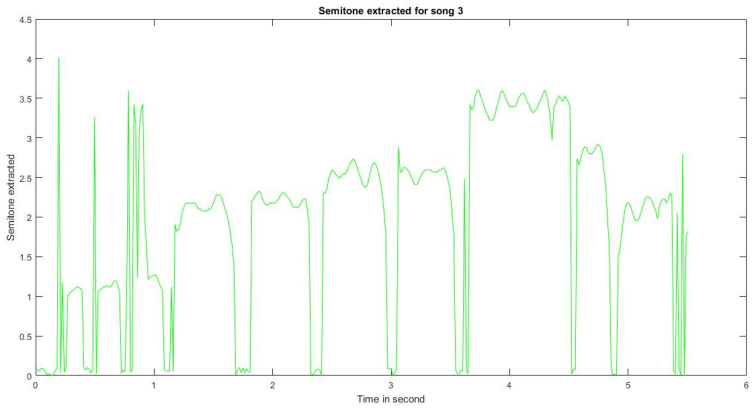

Figure 13: Semitones extracted for the third song

1. Mean distance between song 1 and song 2 is 0.0867.
2. Distance (distance between each two corresponding features and summed for all features) between song 1 and song 2 is 59.7191.

The distance calculated between the third song and the two other songs is

1. Mean distance between song 1 and song 3 is 0.3823.
2. Distance (distance between each two corresponding features and summed for all features) between song 1 and song 3 is 256.9211.
3. Mean distance between song 2 and song 3 is 0.3849.
4. Distance (distance between each two correspondinf features and summed for all features) between song 2 and song 3 is 214.0256.

5 Discussion

5.1 Link with the HMM

Our features are continuous, they can be generated from subsources that generate Gaussians centered on k with standard deviation equal to 0.1 (used to define the noise), k=0,1,2,... and depends on how many octaves are present in the song, k=0 being the silence.

5.2 Requirements

Our extractor allow distinguishing between dierent melodies since each semitone is definied using the retio between its pitch and a reference pitch. If two melodies

contain different ratios between note frequencies, the semitones calculated for each of the two will be different too.

Two note sequences with the same pitch track, but where note or pause durations differ will have similar profiles but duration in time for the features corresponding to the different parts (longer or shorter) will be different.

Our extractor is insensitive to the transposition of the pitch since it uses the ratio between each pich and a reference pitch: a transposition will translate all pitch values by the same scale.

The noise effect is decreased by filtering the signal and replacing the noise values by values very close to 0 (semitone k=0 being the silence). This is done using a condition on the correlation, the pitch and the intensity.

We do not use the intensity to define the semitones therefore the volume should not have any effect.

5.3 Adversarial input for the extractor

While the extractor aims to imitate human hearing, there are ways to confuse it and cause it to extract features differently than a human would. For example, humans are very good at distingishing between voices and keeping track of different conversations. But the extractor is not as capable at that as a human. So if you used the extractor to try to extract the sound of someone humming in an environment with a lot of human noise, ex where many people are talking at a similar intensity and frequency as the person humming, the extractor would likely not be able to extract the data from the noise. So a song hummed in such a situation would sound the same to a human listener with or without the noise, but the extractor would likely return very different results because of an unability to remove the noise correctly. You can also do the same thing in reverse, i.e. confuse the extractor to think that very different sounds produce similar feauture sequences. This could for example be done by tricking the extractor to listen to the noise instead of the actual humming. That way you could have different songs being hummed but because the extractor listens to the noise it will produce similar feature sequences if the noise sounds the same. This could be done by producing noise that is very correlated over time (since correlation is used for determining noise) and to keep the noise at a fairly consistent intensity and have the humming be significantly more or less intense than the noise (since that also is used to determine what is noise and what is data).