Audio Fingerprinting

Introduction to Audio Processing

Digitalization

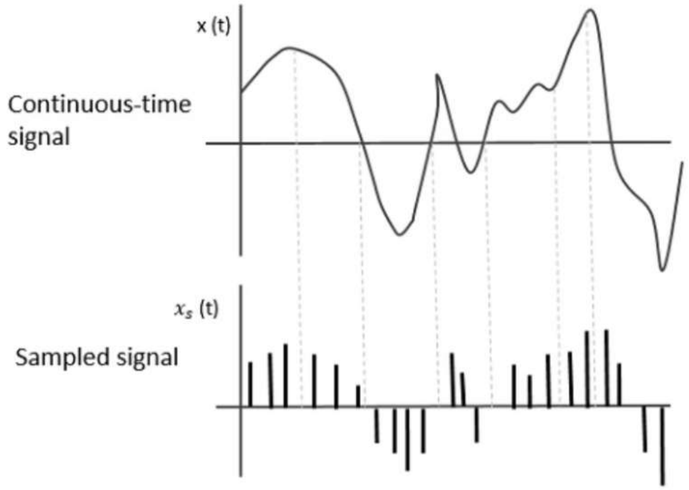
An audio signal is initially produced in analog form. However, the songs or audio files we listen to through our devices are the digitalized version of that audio signal. To analyze and manipulate the audio signal, it must be converted from its analog to digital form. This process of conversion is called digitalization.

1. Sampling

It is impossible to store a continuous analog signal as it contains an infinite amount of information. It needs to be discretized into segments that are small enough that information is not lost between these segments and, at the same time, large enough to reduce the space required to store the audio.

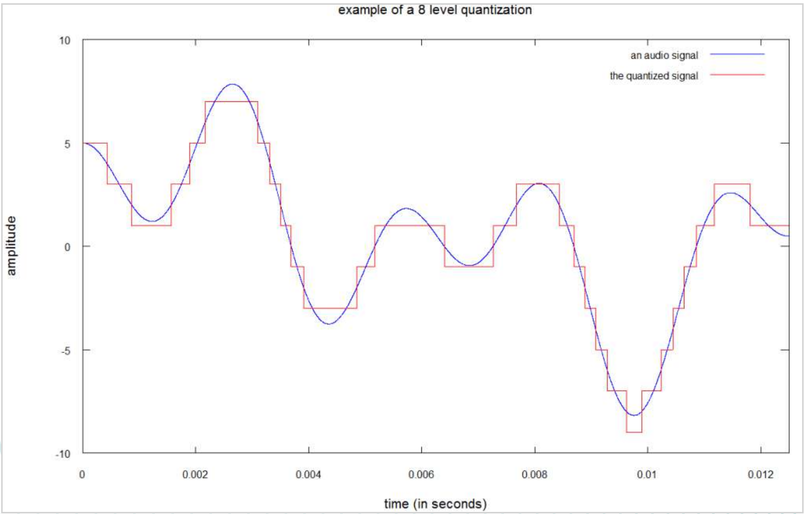
To digitalize a signal, we require the number of samples to be at least twice the frequency range, i.e., for signals ranging from 0 to 20kHz, we need at least 40000 samples per second to digitalize this signal efficiently.

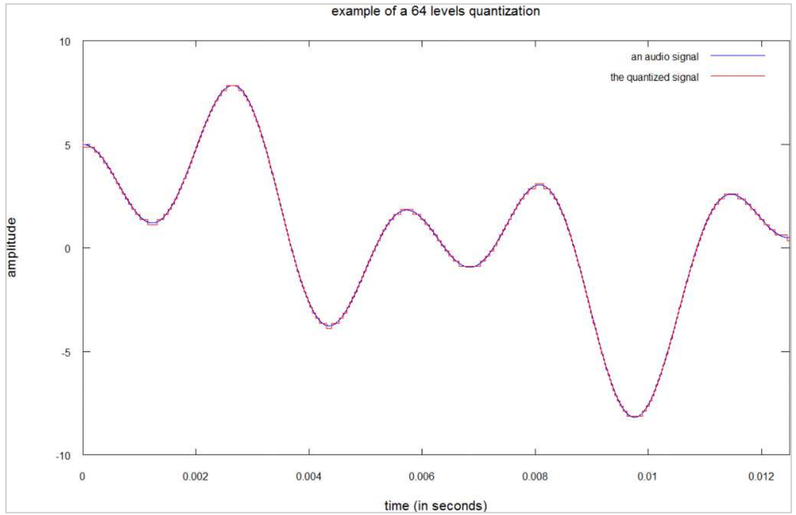
We'll take the standard sampling rate of 44.1kHz for our calculations.



2. Quantization

Loudness is the variation between the lowest and the highest level of an audio signal. To convert the loudness of a signal from a continuous to a discrete signal, we need to quantize the signal.





To reduce this quantization error, we used the standard 16-bit (2^16 = 65536 level) quantization for our calculations.

Fast Fourier Transform

The fingerprinting algorithm we are using works with the frequencies of the given audio file. Hence, we need to convert our audio signals from the time domain to the frequency domain.

It can be done by performing Discrete Fourier Transform on the audio signal. However, it will be a prolonged process because the calculation of DFT of a signal involves (2 * N^2) operations, where N is the window size, which can take days or months to calculate for an extensive database of audio files.

Hence, instead of DFT, we'll use Fast Fourier Transform to go from the time domain to the frequency domain.

Downsampling

To reduce the time complexity and number of operations required to calculate the FFT of the audio signal, we can reduce the window size while keeping the same frequency resolution by downsampling.
The frequency resolution will remain the same if we perform FFT on a 44.1kHz song with a 4096 sample window or an 11.025kHz song with a 1024 sample window. We can do so by dividing the sample into groups of 4 and averaging the 4 samples.

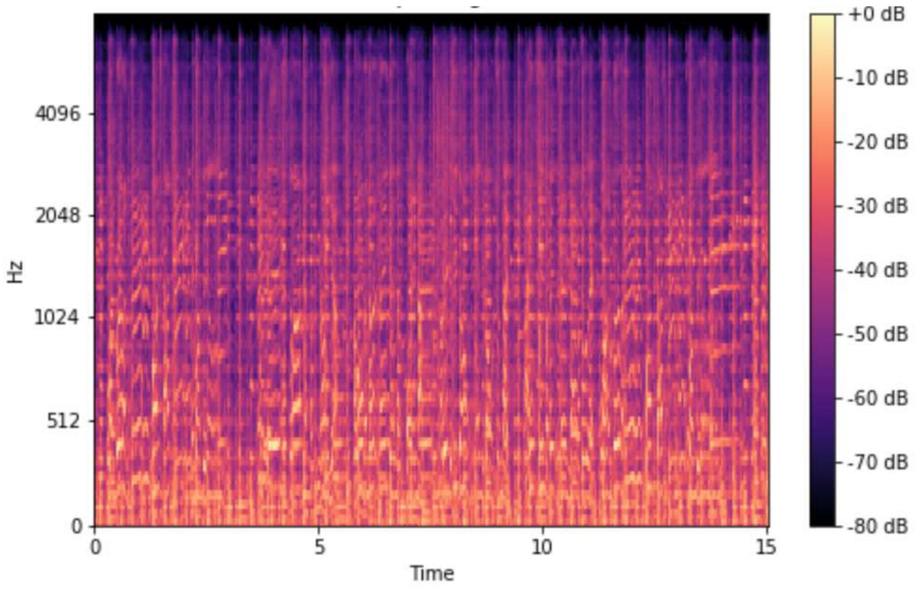
We can now perform the FFT of the audio by dividing the N samples into two groups, with N/2 samples, and computing the FFT for the two N/2 sample windows recursively, that is, by using the divide and conquer algorithm.

Spectrogram

A spectrogram is the visual representation of an audio, showing the spectrum of frequencies of a signal as they vary with time.

It is a three-dimensional graph displaying:

1. Time on the x-axis
2. Frequency on the y-axis
3. The amplitude of the frequency at a particular time described by a color



Audio Fingerprinting

What is Audio Fingerprinting?

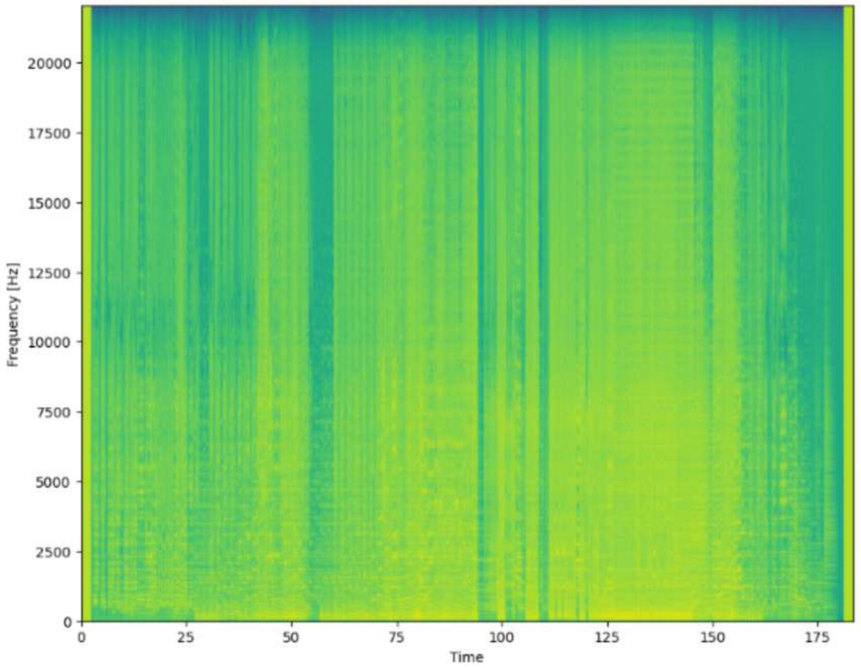
Spectrogram Filtering

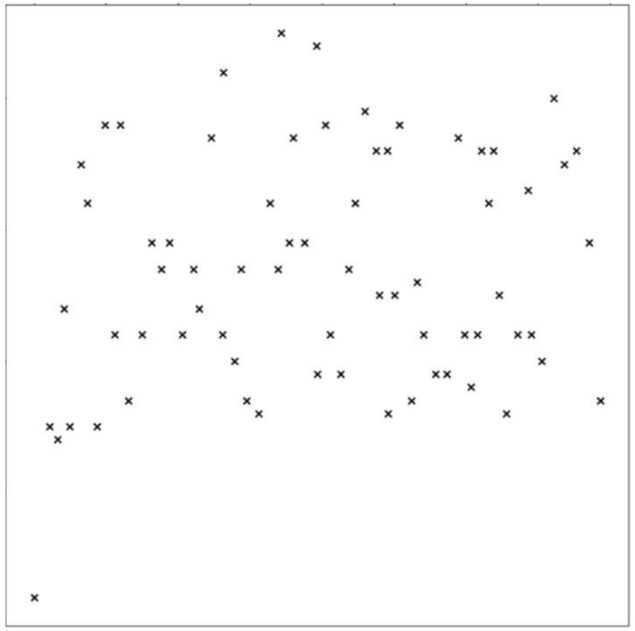
To generate the fingerprints of the audio files, we first filter out the spectrogram of the audio file by keeping only the peaks of energy for the points that represent the loudest notes in a neighborhood so that the fingerprints are noise tolerant.

- We can do so by dividing the total number of bins into a number of bands representing different loudness levels in the given audio file.
- Now we take the strongest bin of frequencies from each of these bands and compute the average of these six bins.
- And then, we mark only the bins of frequencies on the spectrogram above this mean bin value.

Spectrogram Filtering

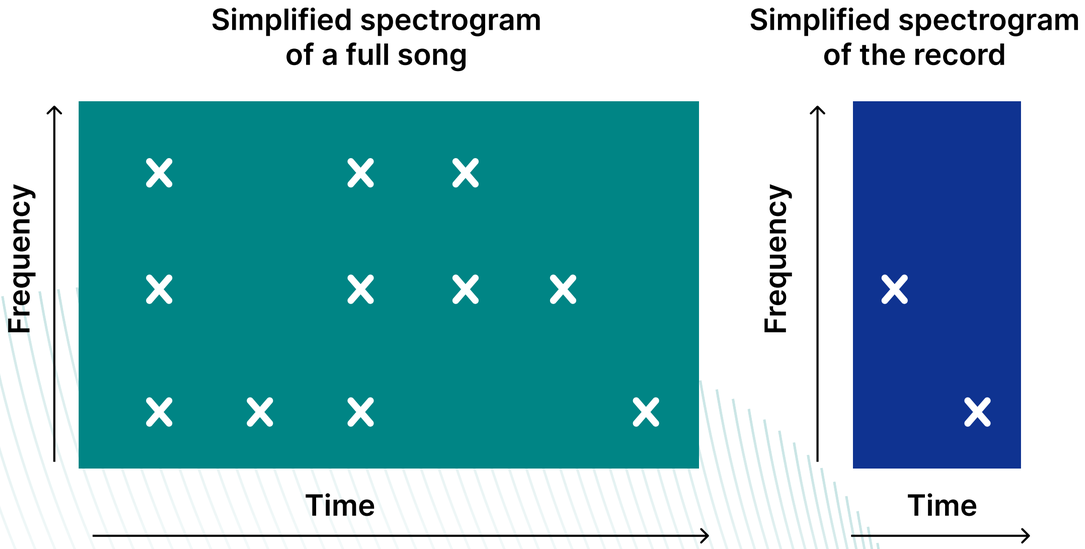
For each of these high-frequency points, we store their coordinates (time, frequency), which will be used later to generate the fingerprint.





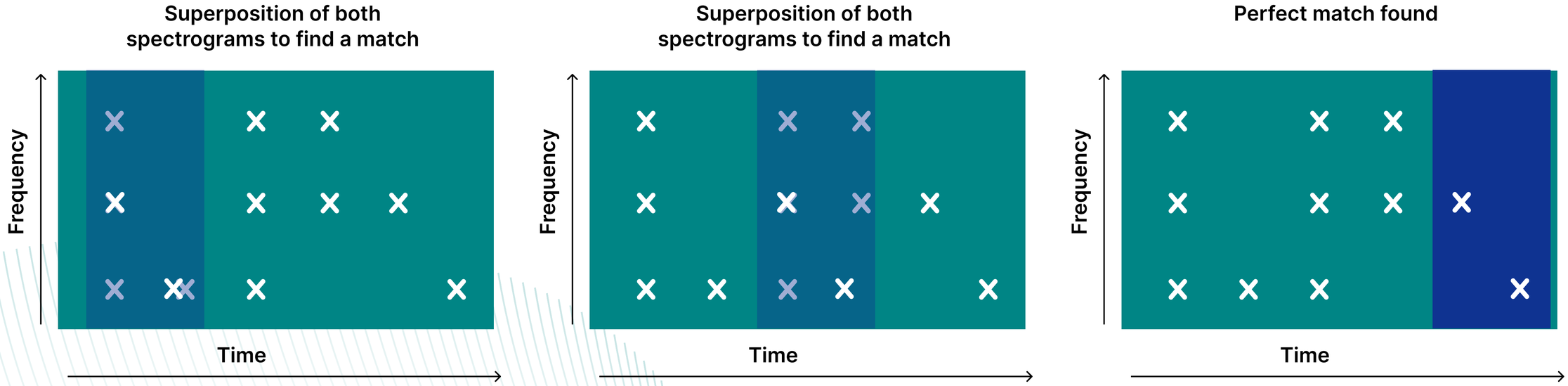
Generating Fingerprints

Suppose we generated the filtered spectrogram as follows for the full song and for the recording that we need to identify. Now the program needs to match the peaks in the record with every song in the database to see which song matches it perfectly.



Generating Fingerprints

This will be done by comparing the spectrogram for the record at every interval of a song for each song in the database. But it requires a lot of time.



Generating Fingerprints

Instead of comparing each peak of the record to the peaks of the song, we analyze a group of points together. In our program, we are considering 15 peaks at a time in one group and a unique anchor point for each group which is different from these 15 points. Then we see if the time difference between any two peaks in this group for the record is the same as that in the song by matching the fingerprint.

The fingerprint of a peak contains two parts [hash value, time offset]:

1. The hash value is a unique hashcode for each peak which depends on [the frequency of the anchor point, the frequency of the peak, and the time difference between the peak and the anchor point].
2. The time offset is the absolute time of the anchor point of that group.

Matching Fingerprints

Now we compare these fingerprints for the original song and the record. and calculate the match percentage.

1. We extracted the index of the first matching hashcode of the song and the record.
2. Now we calculate the frequency of occurrence of the time offsets in the fingerprints and store them in a new separate array for the song and the record.
3. Then we find the longest common string between these new arrays and calculate the percentage match by the formula:

$$\text{percentage match} = \frac{\text{length(longest common string)}}{\text{length(fingerprint array of record)}} * 100\%$$

Applications

This algorithm to match a snippet of audio to its original audio can have various applications:

1. Identification of a song
2. Copyright monitoring

In other industries like education, travel, healthcare, etc. audio fingerprinting can be used provide suggestions or related information from audio or video files.

Thank You