Proseminar Computer and Music – Pitch and Chord Recognition

Nikola Balog

February 15, 2018

Contents

1 Introduction

In Pitch and Chord Recognition, we try to find automated methods, with which we can extract the Information out of an audio signal and find notes or chords that represent this signal. Methods used for this problem can often be used for other problems. The last method, the Hidden Markov modell, can be used for every problem, where we want our software to be able to label data.

2 Musictheory

We will only look at the western notation and consider equal temperament tuning. Depending on which region we look at, the notation can vary and in some genres, the tuning can vary.
Equal temperament tuning means, that the ratio between two adjacent notes is always the same. In the western notation we have 12 notes:
$\{A, A\#, B, C, C\#, D, D\#, E, F, F\#, G, G\#\}$. Every note has a given frequency. For example, $A = 2n \cdot 27.5Hz$ where n is a non-zero integer. Let's say $A_0 = 1 \cdot 27.5Hz$. Then $A_1 = 2 \cdot 27.5Hz$ is an octave above A_0.
So if we consider that there are 12 notes in our notation and that we use equal temperament, we can find out, that the distance between two adjacent notes is $\sqrt[12]{2}$. So $A\#_1 = \sqrt[12]{2} \cdot A_1 = \sqrt[12]{2} \cdot 55Hz$.
So if we know what frequency corresponds to what note, shouldn't we be able to just look at the frequency and determine the note? Because the note A doesn't really exist like that in nature, this is not possible. This is because of harmonics. Harmonics are integer multiples of a given note, the fundamental frequency, that are present when we play or sing the note. These harmonics pollute our signal and therefore we can't just look at the frequency by itself, we have to extract the fundamental frequency out of the signal, which in the case of A_1 would be $55Hz$.

3 Pitch Recognition

Before we start, we will need to define some variables: pitch range, accuracy/resolution, pitch sample rate, and processing time.

3.1 Pitch Range

The pitch range should be chosen with the application in mind. Having a smaller pitch range helps us with the prediction and also eliminates unimportant sounds. But it should also have all the possible pitches we want to identify. The human voice can span over 39 notes, so 80-800 Hz should be more than enough. But for some instruments, we should choose another interval.

3.2 Accuracy/Resolution

As mentioned above, we use the western notation, therefore the difference between two adjacent notes is about $\sqrt[12]{2}$ or 6%. So the accuracy of our method should lie somwhere between 98-99% for our system to work perfectly. Less than that

would lead to big problems where a note could be predicted halfway between two adjacent notes.

3.3 Pitch Sample Rate

If we look at the human voice, the shortest often used note is the sixteenth note. With a metronome set at 90 bpm and in 4/4 time, the duration of this note is 166 msec. This is about 6 notes per second or $6Hz$. Now if we consider the Nyquist-Shannon sampling theorem, our sample rate should be at least $2 \cdot 6 = 12Hz$. Therefore we will choose a sample rate of $12 - 24Hz$.

3.4 Processing time

We can either focus on real-time operation methods, which can almost immediately give feedback, without the need to record the musical piece, or we can focus on non-real-time operation methods, which have to record the signal, and can't give immediate feedback, but are much stronger then real-time operation methods. Real-time operation methods can be used for speech recognition, electric tuners, or games and learning software, like karaoke and Rocksmith®2014, while non-real-time recognition methods can be used for things that require better, stronger methods. These things could be normal musical pieces, that we want to break down in its chords and notes.
For pitch recognition, real-time operation methods are better, therefore we will now focus on these.

3.5 Autocorrelation Method

Now that we have looked at all the variables, let's start with the first method. The autocorrelation method uses the fact, that the human voice has a pattern when we say some letters. These patterns are independent of the human voice. Therefore, if we could just look at the pattern, we could determine what letter the person just said. For a human, this task is time-consuming but also very easy. But the computer doesn't know, when the pattern starts to repeat, therefore it doesn't know, what interval it should look at. To find the interval, after which the pattern repeats, we will map the signal onto itself, but with a slight delay. Then we will check, how similar they are. If they are identical, we have found our period. Then we only would have to look the pattern we see up and map a letter to it. Using statistics, we could then find out, what words these letters could represent in the language.

If the person has a high voice, the delay has to be small enough, not to skip the first time, the pattern repeats. But fixing this would just sterngthen the next problem; the computational load gets too big. But because companies like Gogle, Microsoft, IBM, Apple, and Amazon use methods, that are based on deep learning, which is probably the future for most problems like ours, this method is not used very often.

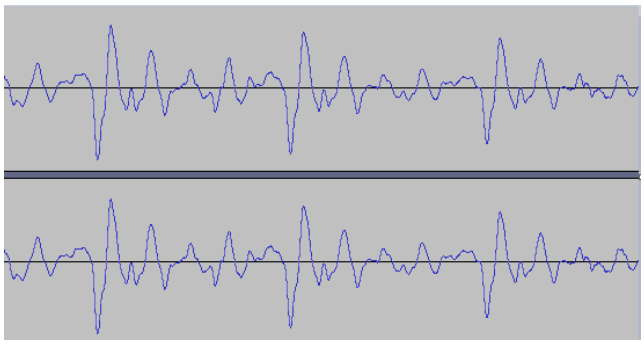
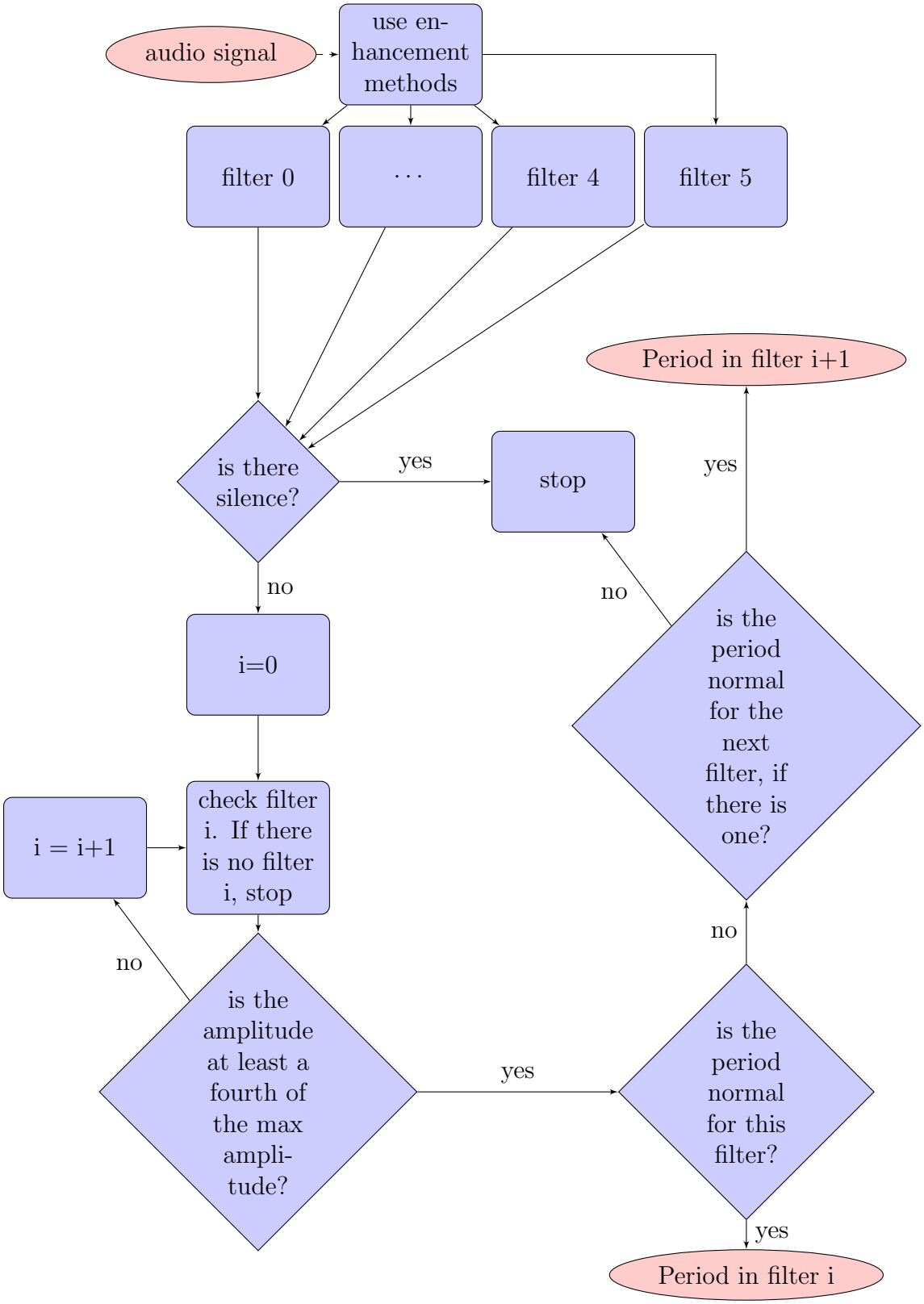


Figure 1: This is a recording of the letter A. For humans it's easy to see the pattern.

3.6 Fundamental Period Measurement Method

As we know the fundamental frequency is the important part of the note and is also the deepest noise. This method uses this information to manipulate the signal, so it can use an easier matchmaking step.
This method consists of three important steps:

- The first step tries to enhance the audio signal. Using bass boost and automatic level control to give the deeper frequencies more weight.
- The second step is very important. We have a filter bank of about 6 (or more) lowpass or half-octave bandpass filters, beginning at 136 Hz. The enhanced audio signal is sent through all these filters, where we try to "cut off" the harmonics above our fundamental frequency, which should leave us with the fundamental frequency.
- In the third step, we extract the amplitude and period information out of the signal and use our pitch decision algorithm to determine the pitch. On the next page there is a flow chart, that shows how the algorithm works.


audio signal
use enhancement methods
filter 0
· · ·
filter 4
filter 5
Period in filter i+1
is there silence?
yes
stop
yes
no
no
i=0
is the period normal for the next filter, if there is one?
i = i+1
check filter i. If there is no filter i, stop
no
no
is the amplitude at least a fourth of the max amplitude?
yes
is the period normal for this filter?
yes
Period in filter i

4 Chord Recognition

It's not easy to define chords, but for our purpose, we will say, that chords are three or more notes, that are played (almost) simultaneously and have a given distance between each other. Let's take a look at the table underneath.

Distance	Interval	Ratio
0	(perfect) unisons	1/1
1	Minor second	16/15
2	Major second	9/8
3	Minor third	6/5
4	Major third	5/4
5	(perfect) fourth	4/3
6	Tritone	45/32
7	(perfect) fifth	3/2
8	Minor sixth	8/5
9	Major sixth	5/3
10	Minor seventh	9/5
11	Major seventh	15/8
12	(perfect) Oktave	2/1

The smaller the integers in the ratio seem, the more pleasent the chord sounds. Therefore, the minor third and major third are considered pleasent. The most often used chords are the major and minor chords, which consist of a minor third and a major third. The major chords consist of the first note, the root note, then a major third and then a minor third. The minor chord consists of the root note, then a minor third and then a major third. The major chords are used to express happiness and festivity, while the minor chords represent gloomier and sadder, sometimes even creepier feelings. Both kind of chords are often used in popular music.

4.1 Important Chracteristics

It is important to know, that the intervals between the notes are the same for every major chord and every minor chord respectively. Because if we want to get the information from every major or minor chord, we don't have to look at every chord. We only have to look at one major and one minor chord. Then we can just shift the information about a note one semitone higher or deeper to get new information about a new chord.

4.2 Short Time Fourier Transformation (STFT)

To get the information needed for the two methods, we will use the STFT to get the chroma features of the musical piece. Chroma features are features that are robust to changes in instrumentation, timbre or dynamics so that the methods work independently of outside variables. These features also capture enough information for us to be able to identify the chord.
First, we will divide our signal X into n equal sized frames $x_1, x_2, \ldots, x_n$. We apply the Fast Fourier transformation on every frame, to get a spectrogram, with an x-axis with the time in seconds and a y-axis with the frequency in Hz. The colour determines the intensity in dB. We will then use a logarithmic function to change the spectrogram into the log-frequency spectrogram, where the distance between two notes is not exponential anymore, but linear. We could, theoretically, discard the octave information if we are not interested in that. Afterwards, we will normalize the intensity and emphasize on the attack of the sound, which will give us the chromagramm.

4.3 Template Based Chord Recognition

What we want to do for the first method, is to split the audio signal into small frames. Then we will asign a chord label onto each of these frames.
So the recording X is split into $(x_1, x_2, \ldots, x_N)$ of feature vectors $x_n \in \mathbb{R}^{12}$, $n \in [1 : N]$. We get these feature vectors from the chromagramm, where $x_i = (percentage(C), percentage(C\#), percentage(D), \ldots, percentage(B))$.
Λ is the set of all chord labels we are intrested in. If we have a big set our information extraction and matchmaking step have to be strong, if we have a small set we could map the chord labels falsly. For example, if we only look at the set that contains all major and all minor chords, then the chord $C7$ could be mapped onto Em or C, because they only difer by one semitone.
Then we will need $\mathcal{T}$, the set of all pre-calculated feature vectors for the $\lambda \in \Lambda$. For example,
$Cmajor7 = (1, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 1)$
$Em = (0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 1)$ $Cmajor = (1, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0)$.
For the matchmaking step we will use this formula:

$$\frac{\langle x, y \rangle}{||x|| \cdot ||y||} = \frac{\sum_{n=1}^{12} x_n \cdot y_n}{\sqrt{\sum_{n=1}^{12} x_n \cdot x_n} \cdot \sqrt{\sum_{n=1}^{12} y_n \cdot y_n}}$$

With that we will have a percentage, how similar two chords are. If we use this formular for $x = Cmajor7$ and $y \in \{Cmajor, Em\}$ we see, that there is no

difference. This is the first problem we encounter, a small set of considered chords. Other problems are, that we haven't considered the harmonics present in every chord we play. This problem is most often fixed with mashine learning, where we give our software information about one major and one minor chord and the software tries to find patterns. Then we only shift each note a semitone and with that, we can get a good set $\mathcal{T}$,

4.4 Hidden Markov Modell

We try to deliver some kind of message through music. Therefore a chord is more likely to follow some chords than others. This, of course, depends on the genre we are looking at, but independent of that, there is a structure that is not only used in music but literature and film as well. This fact is very important for our last method that uses the Hidden Markov modell.
The Hidden Markov modell, or HMM, is a statistical model where the probability for the next observation, depends on the current observation, which is caused by some underlying, hidden states. Therefore it's called the Hidden Markov modell. For our chord recognition software we will define states and observations as follows:

- $\mathcal{A} = \{\alpha_1, \ldots, \alpha_N\}$, where α_i is a possible chord that is hidden from us.
- $\mathcal{B} = \{\beta_1, \ldots, \beta_I\}$, where β_i is a possible observation which could be a chord or some other noise.

The problem is that $\mathcal{B}$ is not a discrete set, there are almost infinitely many possible observations. Therefore we could map every observation onto a prototype vector. These vectors are not the same vectors from the set $\mathcal{T}$ because these prototype vectors don't correspond to any chord. Now we can map an infinite set onto a finite one.
Let also:

$$\mathrm{A} = \begin{pmatrix} a_{11} & a_{12} & \ldots & a_{1N} \\ a_{21} & a_{22} & \ldots & a_{2N} \\ \vdots & \vdots & \ddots & \vdots \\ a_{N1} & a_{N2} & \ldots & a_{NN} \end{pmatrix} \quad \mathrm{B} = \begin{pmatrix} b_{11} & b_{12} & \ldots & b_{1I} \\ b_{21} & b_{22} & \ldots & b_{2I} \\ \vdots & \vdots & \ddots & \vdots \\ b_{I1} & b_{I2} & \ldots & b_{II} \end{pmatrix}$$

$$a_{ij} = \alpha_i \to \alpha_j \qquad b_{ij} = \alpha_i \to \beta_j$$

$a \to b$ describes the probability to change from a to b. So A describes all the probabilities, that we change from one chord to another and B all the probabilities, that the considerd hidden chord acutaly caused the observation.
Let the HMM now be $\Theta = (\mathcal{A}, A, C, \mathcal{B}, B)$, with:
$C = (c_1, c_2, \ldots, c_n)$

c_j = the probability that α_j is the first hidden chord in the sequence.

We can use the Baum-Welch algorithm[4], which can determine the locally optimal solution for choosing A, B, and C.
To determine the optimal path we use the Viterbi algorithm[4] which looks like this:

Input: $\Theta = (\mathcal{A}, A, C, \mathcal{B}, B)$ and observations $\beta_{k_1}, \ldots, \beta_{k_N}$
Output: Optimal sequence of chords

```
D= new float[I][N];
E= new float[I][N-1];
D[i][1] = c[i] * b[i][k[1]];
for (int n=0; n<N;n++){
   for (int i=0; i<I;i++){
      D[i][n] = max(a[j][i] * D[j][n-1])) * b[i][k[n]];
      if (n=N){
         continue;
      }
      E[i][n-1] = maxindex(j, a[j][i] * D[j][n-1]);
      // maxindex(j,f) returns the index stored in j, which maximizes f
      }
    }

i[N]=maxindex(j, D[j][N]);
i[n]=E[i[n+1][n];
```

Now I want to dissect this algorithm:
D and E are both matrices that have the possible hidden chords on the left side and the observed sequence above. D stores the highest probabilities that a hidden chord causes an observation, depending on the previously hidden chords. E stores the values of the hidden chords that are used in 2a).
1. Here we initialize the first row of D with the probabilities, that we start with the hidden chord α_i and that this caused the observation β_{k_1}.
2a) Here we maximize the probability, that the chord α_i causes the observation β_{k_n} while also considering the probability to get to the chord α_i from another chord α_j. Of course, the same thing applies to α_j, so we also have to know the probability to get this chord.
2b) Here the values are stored, which are responsible for the maximal probability.
i_N is the last chord that corresponds to the last observation.
i_n then looks in E to find the hidden chord, that was responsible for the existence of i_{n+1}. Let's look at an example:

A, B, and C are given.

A	*Gmajor*	*Am*	*Cmajor*
Gmajor	0.1	0.45	0.45
Am	0.45	0.1	0.45
Cmajor	0.45	0.45	0.1

B	β_1	β_2	β_3
Gmajor	0,8	0,2	0,0
Am	0,3	0,7	0,0
Cmajor	0,0	0,1	0,9

C	*Gmajor*	*Am*	*Cmajor*
	0,6	0,2	0,2

D, E and i are calculated

D	β_1	β_2	β_1
Gmajor	0.48	0.0096	0.054432
Am	0.06	0.1512	0.002484
Cmajor	0.0	0.0216	0.0

E	β_1	β_2
Gmajor	1	2
Am	1	2
Cmajor	1	2

$i_1 = 1,\ i_2 = 2,\ i_3 = 1$

So in this example, the chord sequence is $Gmajor, Am, and\ Gmajor$. This method and algorithm is very important, because it's not only usefull for chord recognition, but also for every problem, where we have to be able to categorize data. So out of all these methods, the HMM and Viterbi algorithm are most often used.[1]

References

[1] Ai project: Harmonizing pop melodies using hidden markov models, April 2017.

[2] William B. Kuhn. A real-time pitch recognition algorithm for music applications. *Computer Music Journal*, 14(3):60–71, 1990.

[3] Meinard Müller. Music processing using chroma features, November 2012.

[4] Meinard Müller. *Fundamentals of Music Processing: Audio, Analysis, Algorithms, Applications*. Springer Publishing Company, Incorporated, 1st edition, 2015.

[1] For more information about this method and an python implementation of this, you can visit this site: https://luckytoilet.wordpress.com/tag/viterbi-algorithm/ .