



Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

*Distributed
Computing*



Get Rhythm

Semesterthesis

Roland Wirz

wirzro@ethz.ch

Distributed Computing Group
Computer Engineering and Networks Laboratory
ETH Zürich

Supervisors:

Philipp Brandes, Pascal Bissig
Prof. Dr. Roger Wattenhofer

October 12, 2014

Acknowledgements

This project was chosen because of my interest in music in general and guitar in particular. It gave me the possibility to have look at the guitar from a completely different perspective.

I want to thank my supervisors Philipp Brandes and Pascal Bissig for their support during my work and Professor Wattenhofer for giving me the opportunity to do this project at his laboratory.

Abstract

The goal of this project is the development of a system for rhythm monitoring and chord detection in guitar playing. This can help musicians to improve their skills in holding the rhythm and to get a feedback of the played chords.

For rhythm monitoring we used a smartwatch to get acceleration data of the strumming hand. On one hand this data was processed offline on the computer to determine the rhythm and on the other hand it was used on the smartwatch itself, to give a live feedback about the calculated rhythm. In both cases the evaluation shows that the frequency, predicted by the algorithm, is near to the effectively played frequency, so it is possible to approximately determine the rhythm. For the chord detection the audio data was recorded with a smartphone and processed offline on the computer. The evaluation shows that the chord detection works with a good accuracy. To try to improve this accuracy value, the acceleration data was additionally used. The evaluation shows that this does not significantly improve the accuracy.

Contents

Acknowledgements	i
Abstract	ii
1 Introduction	1
2 Basics of music and guitar theory	2
2.1 Tones and chords	2
2.2 Parts of classical guitar	4
2.3 Playing techniques	4
3 System development	6
3.1 Rhythm monitoring	6
3.1.1 Offline analysis	6
3.1.2 Live application	10
3.1.3 Implementation	10
3.2 Chord detection	11
3.2.1 Audio recording	11
3.2.2 Analysis window	11
3.2.3 Fast fourier transformation	12
3.2.4 Chord determination	12
3.2.5 Implementation	14
4 Evaluation	18
4.1 Accuracy of the offline rhythm monitoring	18
4.1.1 Test sequence	18
4.1.2 Results	19
4.2 Accuracy of the live rhythm monitoring	19
4.2.1 Test sequence	19

CONTENTS	iv
4.2.2 Results	19
4.3 Chord detection	20
4.3.1 Test sequence	20
4.3.2 Results	20
Future work	22
Bibliography	23

Introduction

If people want to learn an instrument there are different ways. The conventional one is to go to a teacher, who can teach you how to play and, what is a very important part, to give you a feedback. Nowadays a lot of online tutorials and courses are available, which can partly replace the teacher for the teaching part. But what is still not common is the feedback part, especially for rhythm feedback. Of course there are metronomes, which can give you the rhythm, but when they stop, you do not know, if you are still right or not. Using new technologies like smartphones or smartwatches this can be changed. In this thesis we detect with the help of these tools the rhythm of guitar music on one hand and determine the played chords on the other hand. Chapter 2 shows an overview about basic music and guitar theory, especially about the themes, which are important to know for this thesis. In Chapter 3 the used hard and software are described as well as the implemented algorithms. Finally chapter 4 shows an evaluation about how precise the system works.

Since smartwatches are a new technology, there is no development about rhythm detection with smartwatches. Of course the rhythm detection with the help of the audio signal is well developed. Laroche in [1] or Ellis in [2] present methods to detect the rhythm which work quite well, but there are always different sources of errors. For example it is more difficult to determine the rhythm, when there is not a 4/4 beat. Another error occurs, when the double or the half frequency is detected. Of course detection via audio data is disturbed by environment noise. This is an advantage of the rhythm monitoring with acceleration data, like it is presented in this thesis. Environment noise (eg. other musicians in a band) has no influence on the movement of the hand respectively on the acceleration data.

Basics of music and guitar theory

This chapter gives an overview about the basics of music and guitar theory, especially these parts, which are important for this thesis. For the music theory it is a basic knowledge about the structure of different chords. Concerning the guitar theory, it is important to know, how the guitar is played.

2.1 Tones and chords

In western music theory there are fixed frequency steps, called tones. The factor between two steps is $2^{1/12}$. These tones are named by letters (order: A, Ais, H, C, Cis, D, Dis, E, F, Fis, G, Gis). Since there are more frequency steps than names, after 12 steps the names are repeated. That means if you double or halve the frequency, you reach a frequency step with the same name. This is called an octave. The difference between two neighboring tones is called a semitone. The frequency of 440 Hz belongs to the tone A. From here you can calculate all other tones.

If different tones are played harmonically together, this is called a chord. There are rules, which tones have to be played to get a certain chord. The chords are named after its basic tone and can occur in many different variants, depending on which other tones are played. In the following we take the C-chord as an example, but this can be adapted on all other chords by just playing another basic tone. The simplest variant of this chord is C-major. This means to play the following three tones together. The first one is the basic tone C. To find the second tone, we go four semitone steps higher and reach E. For the third tone we go again three semitone steps higher and reach G. So if C, E and G are played together, this is called C-major. With the same number of steps starting from another basic tone, we can construct other major chords. In the same way we can construct C-minor, starting from the basic tone C, but with a different number of steps. So the basic tone defines the name of the chord and

	chords with 3 tones				chords with 4 tones			
	C-major	C-minor	C-dim	C+	C-major7	C-minor7	Cj7	C-dim7
c	c	c	c	c	c	c	c	c
cis								
d								
dis		dis	dis			dis		dis
e	e			e	e		e	
f								
fis			fis					fis
g	g	g			g	g	g	
gis				gis				
a								a
ais					ais	ais		
h							h	
c								

Table 2.1: Different variants of the C-chord with three or four tones. It is also possible to construct chords with five or more tones, so this list is not complete.

the number of steps to the other tones defines the variant. In Table 2.1 you can see a few variants of the C-chord, with the corresponding number of steps.

2.2 Parts of classical guitar

As shown in Figure 2.1, a guitar consists of three main parts, the body, the neck and the head [3]. There are six strings, which are fixed at the bridge with one end and at the head with the other end. The guitar produces basic frequencies from 80 Hz up to 660 Hz (without harmonics).

2.3 Playing techniques

In guitar playing the two hands have different functions. In the following these are shown from the view of a right-handed person. Of course there are a lot of different techniques how to play the guitar, but this overview is concentrating on basic techniques. The left hand is responsible to press the strings towards the frets, shortening the length of the oscillating string. Depending on how long the oscillating part is, a specific tone is generated [4]. Each of the six strings can produce a different tone, controlled by the left hand finger positions.

The right hand is responsible for strumming the strings. It touches the strings in the area of the soundhole, so the strings start oscillating and creating a tone. There are different styles how to play with the right hand.

Strumming Strumming means to strike all strings (excepting one or two, which do not belong to the chord) immediately after each other, so that different tones sound together, resulting in a chord. This can be done by hand or, what is more common for professional music, with the help of a plectrum. Strumming can be done up- and downward and can differ for every song, giving rise to different, so-called strumming patterns. The movement of the hand can be periodic and approximately be a harmonic oscillation, or it can be irregular. In this thesis, as a requirement, the movement has to be regular, to determine the rhythm.

Picking Another method is the finger-picking. This means to touch or pick single strings with the finger to play just single tones of a chord. This is used to give a song a special characteristic or to play a melody. This method is not monitored in this thesis.

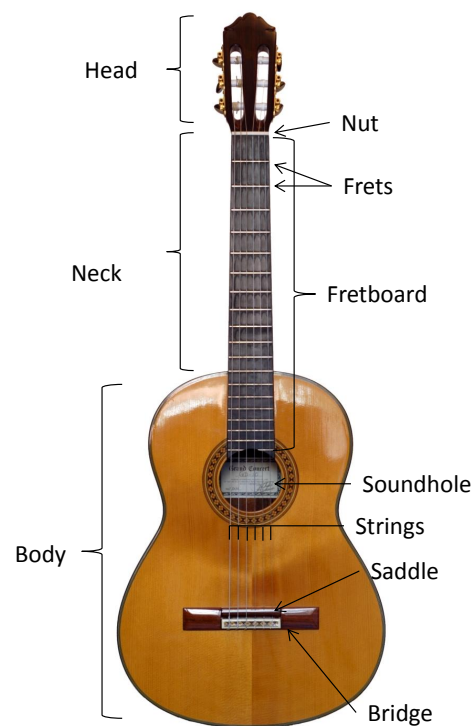


Figure 2.1: Parts of a classical guitar. It is divided into three main parts; head, neck and body. *source: <http://de.wikipedia.org/wiki/Gitarre>, visited: 7.10.2014*

System development

3.1 Rhythm monitoring

To determine the frequency of a played song, the smartwatch has to be fixed at the wrist of the right hand. This hand is responsible to strum the strings and gives the rhythm (from the view of a right handed person). To analyze the movement of the right hand, the accelerometer of the smartwatch is used. The accelerometer provides acceleration information in three dimensions. Since the movement of the right hand goes just up and down, only one dimension is needed to recognize this movement (see Figure 3.1).

In the following, a methods for rhythm monitoring is presented. It is implemented in two different ways. The first one is an offline analysis, where the smartwatch stores the captured data to use it later implicit. The second one is a live calculation of the frequency performed on the smartwatch .

3.1.1 Offline analysis

The data of the accelerometer is stored together with a time stamp on the smartwatch, from where it can be loaded on the computer for further calculation. In Figure 3.2 the raw data of all three dimensions can be seen. The main movement of the watch is along the y-axis, so this one is used for further calculation. If we take a closer look at the acceleration graph of the y-axis, the regular movement of the right hand, which is approximately a harmonic oscillation, is easy to see. In Figure 3.3 the corresponding positions of the hand are marked. To get a clearer view the the gravity is subtracted. The highest positive respectively the highest negative velocity probably appear when the strings are strummed and it is zero, when the hand is at the top or the bottom. So it is plausible for the acceleration that its peaks correspond to the top and bottom position of the hand and its zero crossing to the moment, when the strings are strummed.

The first step of calculation is to find out where the playing starts and where it ends. This can be done by calculating the variance. The variance is taken because

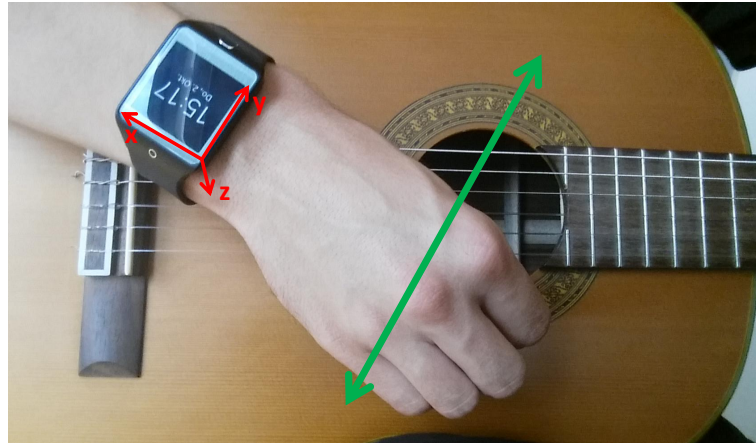


Figure 3.1: The green arrow shows the movement of the hand while playing. This movement is parallel to the y-axis of the watch.

it is approximately zero, when the hand does not perform large movements and it has a high value, when the hand is strumming. So for each acceleration value the variance is calculated with window size of 20 data points (0.4 sec). This value has been found empirically. For too low values of the window size single peaks in the phase before playing can be classified wrong. Too large values have a bad influence on the precision. When the variance reaches a certain threshold the acceleration can be classified as playing until the variance falls under the threshold again. This threshold has been found empirically. In Figure 3.4 the variance can be seen and it is obvious where the playing starts and where it ends. The condition for this method is, that there are not large movements of the hand before the playing starts.

To determine the frequency, a fast fourier transformation (FFT) is performed. It gives an overview about all the frequencies and how they appear in the presented data set. The most dominating frequency is the searched resulting frequency. It leads to the plot in Figure 3.5, from which the frequency can be determined. Another method to get the local frequency is to measure the duration of one oscillation, in this case the duration from one zero crossing to the one following the next. In Figure 3.6 both methods are shown in the same plot. This recording was made by an amateur musician and with the help of a metronome. It is interesting to see, that the given rhythm can not be held perfectly during the song. Instead the movement of the hand has to be adjusted permanently, to stay more or less on the given frequency of the metronome.

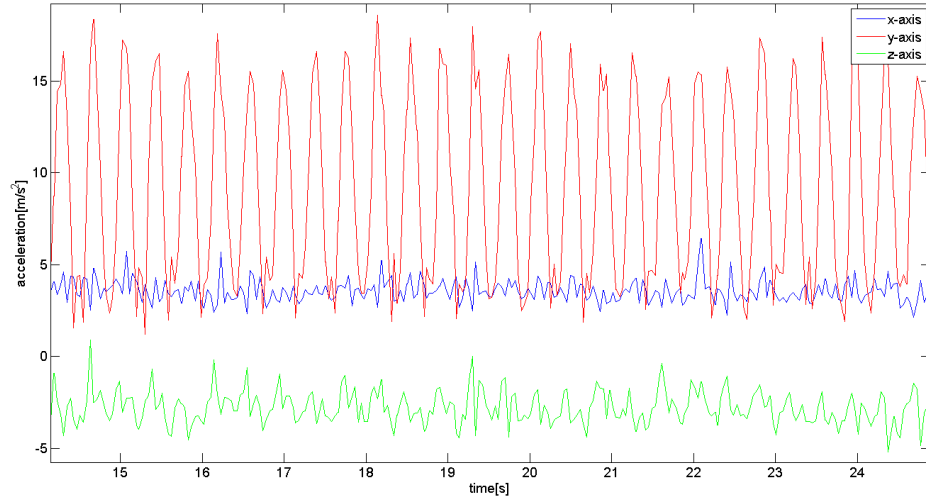


Figure 3.2: This plot shows the acceleration measurement of all three dimensions. Since the main movement of the hand is parallel to the y-axis (red line), it has the largest deflections.

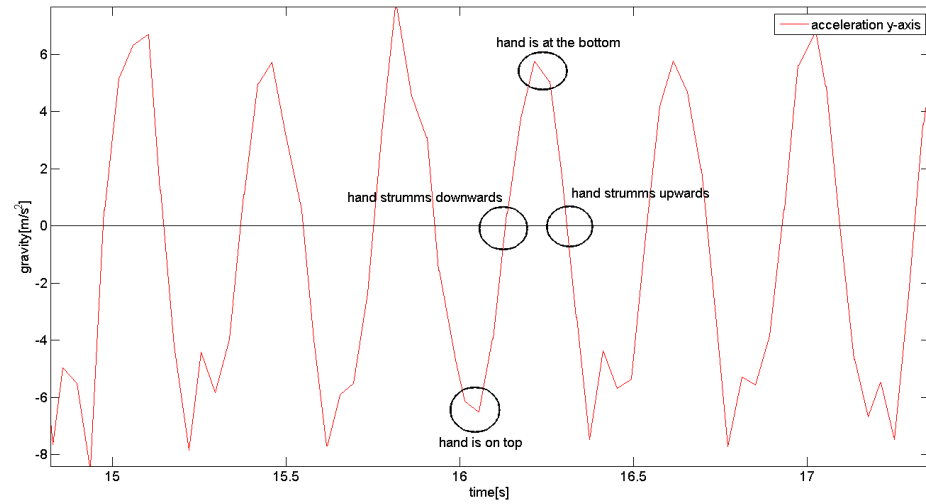


Figure 3.3: The acceleration of the y-axis with the corresponding positions of the hand. To get a clearer view the gravity is subtracted.

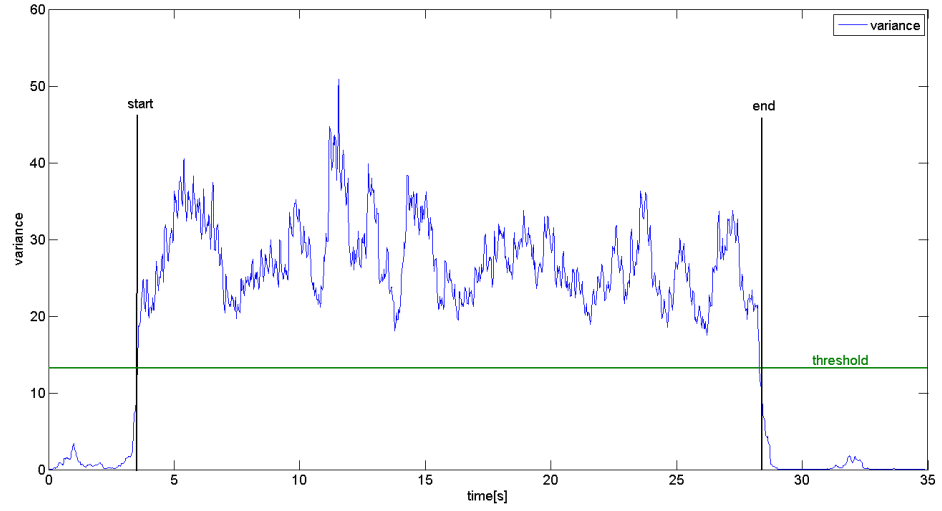


Figure 3.4: The variance of each data point calculated with a window size of 20 data points (0.4 sec). The start and end point of the strumming is clearly visible.

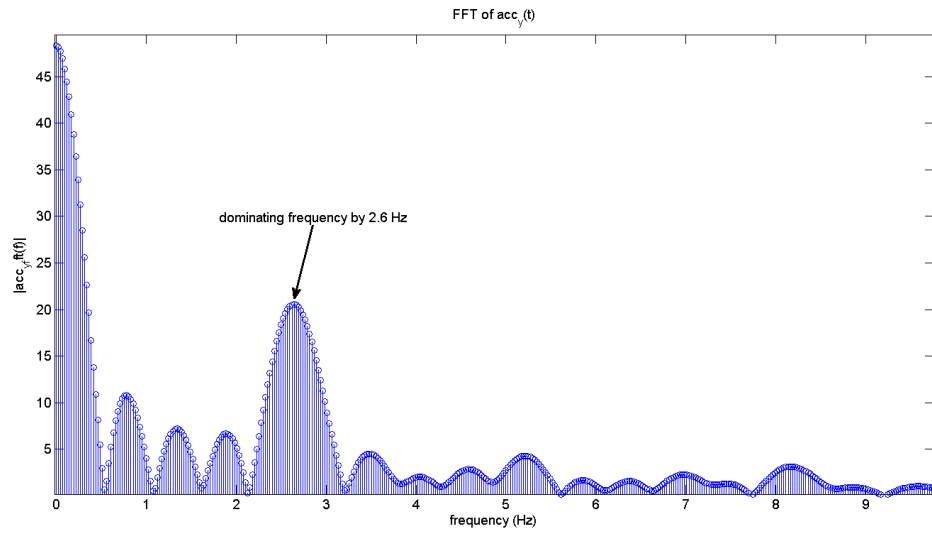


Figure 3.5: This plot shows the FFT about the whole acceleration data with its dominating frequency by 2.6 Hz.

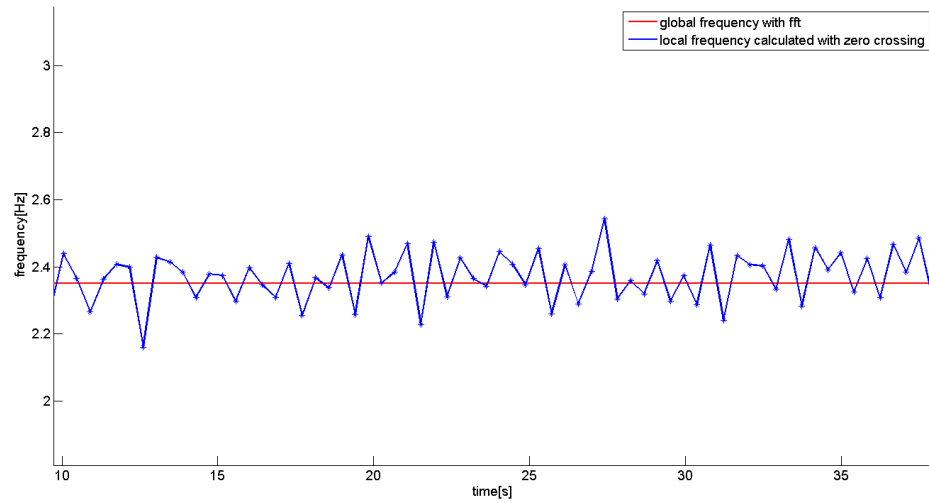


Figure 3.6: The global and local frequency measured with two different methods. It is interesting to see, that the given rhythm can not be hold perfectly during the song. Instead the movement of the hand has to be adjusted permanently, to stay more or less on the given frequency of the metronome

3.1.2 Live application

The sequence of calculation for the live application is similar to the offline analysis but the whole calculation takes part on the watch itself. Since it is a live calculation, only past data is available, so we can give some information about the frequency of the past few seconds. For the live application the variance part is not necessary. The data set used for the FFT is defined by the past, heuristically found 120 data points, what correspond with the past 2.4 seconds. In Figure 3.7 the running application is shown. The top line informs about the actual value for the acceleration of the y-axis. The middle line shows the resulting frequency of the FFT of the past 2.4 seconds. The bottom line shows the frequency in bpm (beats per minute), which is the standard unit in music to describe the velocity of a song.

3.1.3 Implementation

Data capturing The data capturing for both, the offline and the live method, is performed on the Samsung Gear2 neo, which is identical to the Samsung Gear2 concerning the firmware and differentiates only in the missing camera. The operating system of this smartwatch is Tizen. The accelerometer has a samplerate of about 50 Hz. The captured data is stored on the local storage,



Figure 3.7: The screen of the smartwatch while running the application. The middle line shows the resulting frequency of the FFT.

from where it can be loaded to the computer via USB.

Data analysis On the computer the calculation takes part in Matlab. The FFT is performed with the FFT-function provided by Matlab itself. For the live application, the FFT algorithm has to be implemented, since Tizen does not provide an FFT-function.

3.2 Chord detection

Parallel to the acceleration capturing with the smartwatch, the audio data is recorded with a smartphone. For the chord detection only an offline method has been developed. In the following it is described, how the played chords are detected from the audio data and how the acceleration data can help doing this.

3.2.1 Audio recording

The audio signal is recorded by an application of the smartphone and stored on the local storage, from where it can be loaded to the computer for further calculation.

3.2.2 Analysis window

To determine which chords has been played, the data set has to be split up. To do so, two different methods have been tested.

The first one is to define a fixed window size and an analysis interval. Now for every interval step the data within the window is analyzed. Depending on the parameters, these windows can overlap. The analysis interval can be chosen as small as required, it has just a bad influence in the calculation time. For the window size a too small value means to have a less accurate result, caused by noise or missing frequencies during this small time window. With a too wide time window transitions between two chords can not be determined exactly.

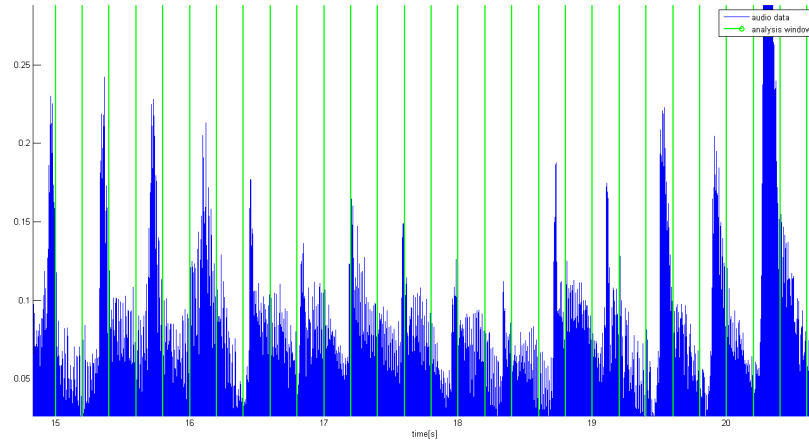
The second method is to integrate the acceleration data to split up the audio data. Analyzing the acceleration, it can be determined, when the strings are possibly strummed. Between two of these moments, no other chord can appear, so we take this period of time as our analysis window. The idea of this method is that in one analysis window only tones of one chord can appear. Compared to the fixed window size method, where in one window one chord can end and the next chord begins. In Figure 3.8 examples of both methods are shown. You can see that in Figure 3.8(b) before every peak of the audio data a new window begins.

3.2.3 Fast fourier transformation

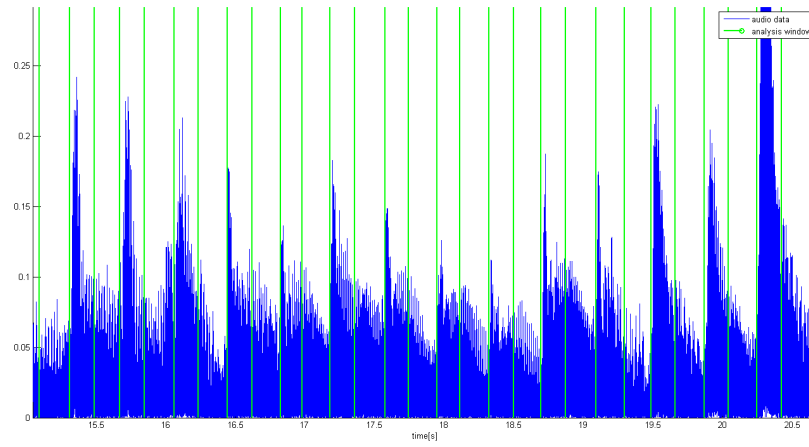
For each time window a fast fourier transformation (FFT) is performed. This provides an overview about the frequencies that occur. The interesting frequency band starts by 80 Hz up to 660 Hz, the frequencies, which the guitar produces. In Figure 3.9 the FFT of a C-major-chord is shown. Above the dominant frequencies, the tones, which were effectively played by the guitar are marked but also the harmonics. A C-major-chord contains the tones C, E and G. In the plot we see one of the difficulties of the chord detection. The third harmonic of the tone E with a basic frequency of 164.8 Hz is the tone H with a frequency of 494.4 Hz. But the tone H does not belong to the C-major-chord. So there is a correctly detected tone, which can badly influence the final prediction.

3.2.4 Chord determination

To determine the chord, a list is created, which includes every chord (with its variation). For every chord the tones respectively frequencies which belong to this chord are listed. Every frequency is compared with the resulting FFT and gets a value, depending on how dominant this frequency is. Now every chord gets its own score by calculating the mean of the FFT-values of all frequencies of this chord. This results in a ranking of all chords, where we find the most probable chord on top. In Figure 3.10 an example of this score calculation is shown. The plot shows the FFT of a C-major-chord. In Figure 3.10(a) the score for the C-major-chord is shown. The C-major-chord contains the tones C, E and G. These tones respectively their frequencies are dominant in this FFT, so



(a) Windows with a fixed size (0.2 sec) and a fixed interval (0.2 sec).



(b) Windows defined by the acceleration data. Before every peak of the audio data a new window begins.

Figure 3.8: Two different methods to define the analysis windows. In (b) the window size and the interval are fixed. In (a) the windows are defined by the acceleration data.

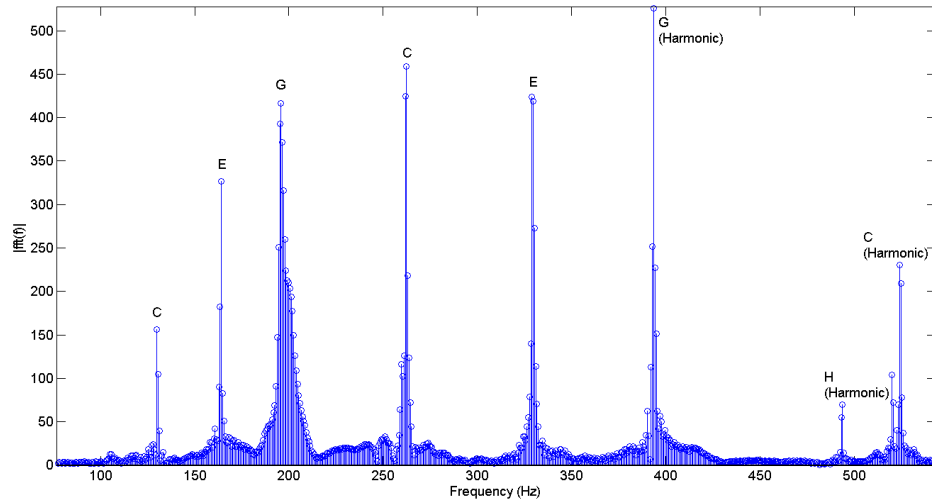


Figure 3.9: The FFT of a C-major-chord. Above the dominant frequencies, the tones, which were effectively played by the guitar are marked but also the harmonics.

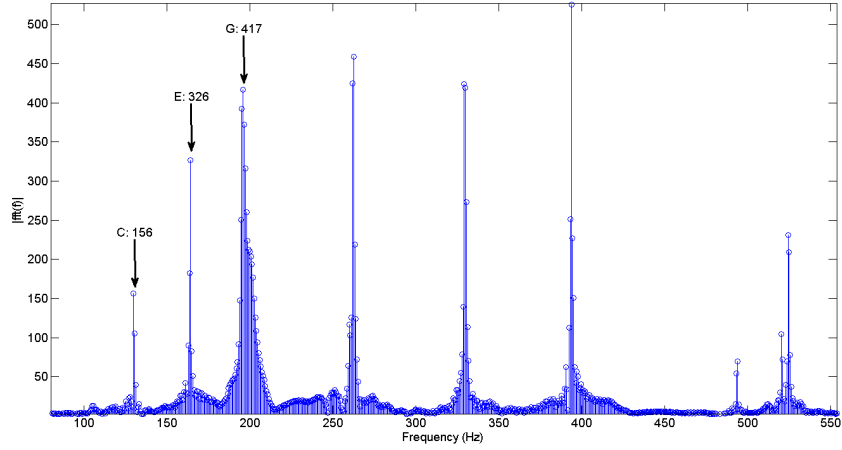
the C-major-chord gets a high score. The score is the mean value of these three FFT-values, in this case 300. Figure 3.10(b) shows the score of the D-minor-chord. The D-minor-chord contains the tones D, F and A. Because these tones do not reach a high FFT-value, the score of the D-minor-chord is only 12.

During some tests the following problem appeared. There were single misinterpreted chords between a row of the same correct chords. From the view of human analyst it is obvious, that this is a fault of the program. This errors are improved by implementing a simple algorithm. This algorithm assume, that if a chord is played, the next chord is probably the same one, especially when it has still a high score. In this case the previous chord is taken as final prediction. In Figure 3.11 an example of such an error correction is shown. In Figure 3.11(a) the G-chord is the only one with a high score, so it is the winner. In Figure 3.11(b) there a two chords with a high score. The previous chord was an Em (E-minor). Since the previous chord has still a high score, Em is the winner.

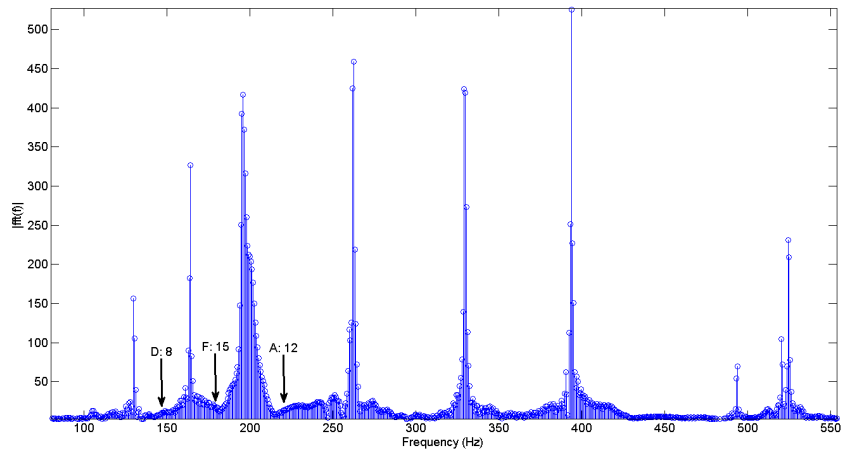
The final product of the program is a plot, where you can see which chord was played in which time period (see Figure 3.12).

3.2.5 Implementation

Audio recording The audio recording was done with a Samsung Galaxy S3, running an android operation system. The sample rate of the recording is 44.1 MHz.

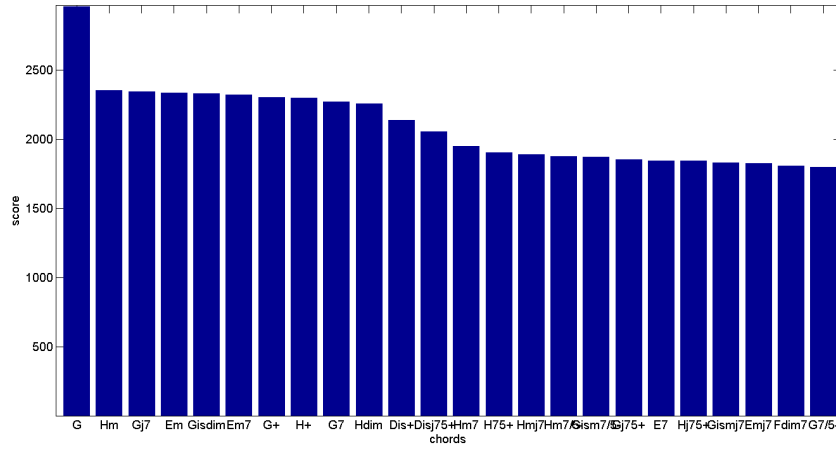


(a) Determination of the score for the C-major-chord by calculating the mean of the three FFT-values. The mean value for the C-major-chord is 300.

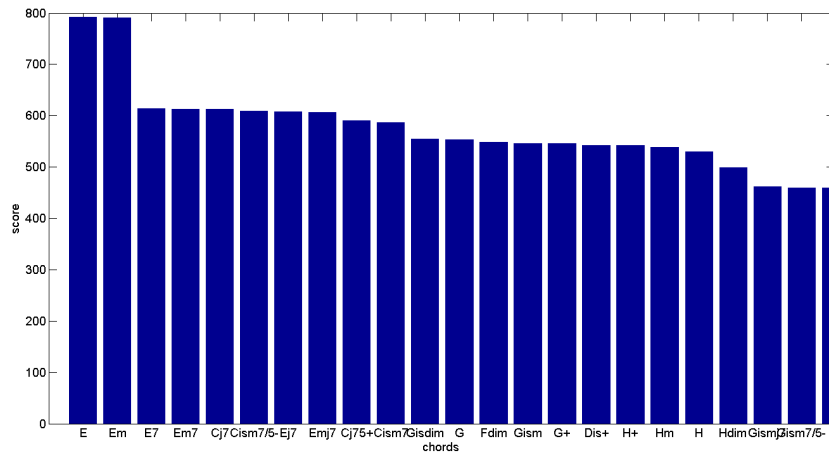


(b) Determination of the score for the D-minor-chord by calculating the mean of the three FFT-values. The mean value for the D-minor-chord is 12

Figure 3.10: Two examples how to calculate the score for a specific chord. In plot (a) the C-major-chord gets a high score, because the tones, which belong to C-major, are dominant. In plot (b) the D-minor-chord gets a low score, because its tones are not dominant.



(a) A scoreboard with an obvious winner (G).



(b) A scoreboard with two chords with a high score (E and Em).

Figure 3.11: Two different scoreboards. The first one (a) has an obvious winner. The second one (b) has two chords with a high score. Here the error correction algorithm defines Em as winner, because it was the previous chord and has still a high score.

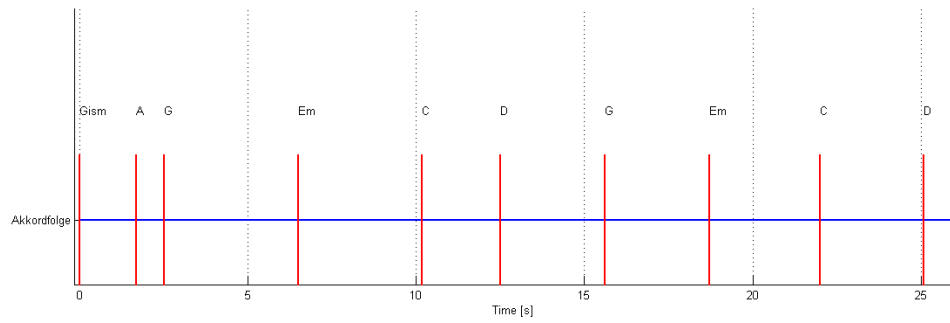


Figure 3.12: Visualization of the played chords. This plot shows, which chord was played in which time period.

Further calculation All the calculation is done in Matlab. The FFT is done by the FFT-function provided by Matlab.

Evaluation

To know how precise the system works, it has been evaluated. This was made for both implementations of the rhythm monitoring as well as for the chord detection.

4.1 Accuracy of the offline rhythm monitoring

4.1.1 Test sequence

To test the accuracy of the offline rhythm monitoring the following test was performed. Four different songs were played, keeping the rhythm with the help of a metronome. The strumming pattern was different for each song, but the movement of the hand was regular. In Table 4.1 the songs with there velocity are listed. You can see the setting of the metronome and the velocity which was effectively played. This was determined by counting the number of beats and measuring the time by hand out of the plotted audio and acceleration data. As you can see, the effectively played velocity does not perfectly match with the velocity given by the metronome. The velocity calculated by hand is used to determine the accuracy of the rhythm monitoring.

No.	Velocity by metronome [Hz]	Velocity measured by hand [Hz]
1	2.333	2.357
2	2.333	2.330
3	1.833	1.830
4	2.833	2.654

Table 4.1: Velocity of the test sequences. You can see the setting of the metronome in column two. In column three you can see the velocity which was effectively played. This was determined by counting the number of beats and measuring the time by hand.

No.	Velocity measured by hand [Hz]	Velocity predicted by the system [Hz]	Divergence [%]
1	2.357	2.351	0.25
2	2.330	2.344	0.60
3	1.830	1.849	1.04
4	2.654	2.656	0.08
			ø0.49

Table 4.2: Comparison of predicted data by the system and measured by hand. The mean divergence is 0.49%.

4.1.2 Results

In Table 4.2 the predicted velocity of the system is listed for each song as well as the calculated divergence. As you can see the system has a mean divergence of 0.49 %.

4.2 Accuracy of the live rhythm monitoring

4.2.1 Test sequence

To test the accuracy of the live rhythm monitoring the following test was performed. The smartwatch was moved by hand by keeping a fixed velocity with the help of a metronome. The live algorithm provides every 2.4 seconds a new result. This result is the velocity of the previous 2.4 seconds. While moving the hand constantly, all results were noted. This test was performed for three different velocities and for 40 seconds each.

4.2.2 Results

In Table 4.3 the results of the live monitoring evaluation are listed. For all predicted velocities of one given velocity the mean and the standard deviation is calculated. Compared to the offline rhythm monitoring system, the divergence between the given velocity and the mean of the predicted velocities is high. The standard deviation is on a low level. The high divergence is probably caused by other reasons than just a not perfect working algorithm. In this evaluation scenario the ground truth can not be determined perfectly, because there is a human element.

No.	Velocity given [Hz]	predicted velocities [Hz]		divergence [%]
		mean	standard deviation	
1	2.333	2.368	0.014	1.50
2	1.833	1.977	0.0067	7.86
3	2.833	2.768	0.012	2.29
				ø3.88

Table 4.3: This table shows the given velocity of the metronome compared with the mean and standard deviation of the predicted velocities. The fifth column shows the divergence between the given velocity and the mean of the predicted velocities.

4.3 Chord detection

4.3.1 Test sequence

As mentioned in section 3.2.2, two different methods to define the analysis windows were implemented. On one hand the analysis windows are chosen with a fixed window size in a fixed interval. On the other hand the acceleration data is used to define the analysis windows. In the following test sequence both methods are evaluated.

To evaluate the chord detection system, the same test sequence as in section 4.1 is used. Four songs with different velocity and strumming pattern are played. To calculate the accuracy it has been verified for every analyzed time window, if the predicted chord correspond with the played chord.

4.3.2 Results

Table 4.4 shows the results of the evaluation with the fixed window size. The mean accuracy is 88.0%. It is strikingly that most of the wrong predictions occur, when there is a change from one chord to another. This is an explanation for the low accuracy of song number two, in which partially many different chords are played during a short time interval.

In Table 4.5 the results of the evaluation with the analysis windows defined by the acceleration data are listed. The mean accuracy is 87.2%.

Comparing both methods of analysis window positioning, there is no significant difference, so there is probably no advantage in using the acceleration data for detection the chords.

No.	#analyzed windows	#correct predictions	accuracy [%]
1	268	249	92.9
2	403	301	74.7
3	262	242	92.4
4	240	221	92.1
			ø88.0

Table 4.4: This table shows the results with the fixed window size method. In the second column the number of analyzed windows is listed. The third column shows the number of correct predictions and the fourth column the accuracy.

No.	#analyzed windows	#correct predictions	accuracy [%]
1	254	236	92.9
2	383	290	75.7
3	206	182	88.3
4	256	235	91.8
			ø87.2

Table 4.5: This table shows the results with the window size defined by the acceleration data. In the second column the number of analyzed windows is listed. The third column shows the number of correct predictions and the fourth the accuracy.

Future work

Concerning the live rhythm monitoring on the smartwatch, the feedback part is still missing. The existing application shows just the actual velocity, but it is impossible to read it while playing. One idea is to develop a standalone application on the smartwatch, which gives a haptic or an audio feedback, whether you're playing too slow or too fast. Another idea is to connect the smartwatch with the smartphone, so it would be possible to get also a visual feedback. Another improvement of the rhythm monitoring system can be realized by implementing the detection of irregular hand movement.

Concerning the chord detection the algorithm can be improved and it can be implemented in a smartphone application. One of the improvements that can be done is the positioning of the analysis windows. Maybe there is another way to use the acceleration data than the one presented in this thesis. Possibly the audio data has to be analyzed for a better positioning. Another improvement can be done in the error correction. The idea is to implement some kind of Markov chain, with probabilities about the following chord, if the actual chord is known. This can be done based on music theory, where it is known, which chords normally follow each other.

Bibliography

- [1] Jean Laroche. Efficient tempo and beat tracking in audio recordings. *Journal of the Audio Engineering Society*, 2003.
- [2] Daniel P. W. Ellis. Beat tracking by dynamic programming. *J. New Music Research*, 2007.
- [3] Richard Mark French. *Technology of the Guitar*. Springer Science+Business Media, 2012.
- [4] Richard Mark French. *Engineering the Guitar*. Springer Science+Business Media, 2009.