PROJECTE FINAL DE CARRERA

Alignment of the recording of a violin performance with the corresponding musical score using multimodal information

Enginyeria de Telecomunicació

Jordi Bartolomé Guillén

Director: Dr. Xavier Serra

Barcelona, Juny 2011

Collaborations

This project has been developed in the Music Technology Group, a department of the Universitat Pompeu Fabra.

Acknowledgments

First of all, I would like to thank Xavier Serra for opening the doors of the Music Technology Group and give me the possibility to do this thesis in this group.

Secondly, I would like to thank Marco Marchini and Panos Papiotis for their support and guidance with the project and also for reading the document and making valuable suggestions.

Also I would like to thank you, Laura, for being here all time.

Finally, I would like to thank my family for their unconditional support not only in this thesis, but during all my university studies. Without them, none of this would have been possible.

A tots, moltes gràcies.

Resum del projecte

La recerca en la manera en que els músics executen les peces musicals ha comportat el desenvolupament d'eines per tal de poder mesurar alguns aspectes relatius a la interpretació.

En aquest projecte es pretén estudiar una part de la conducta dels músics, com és la variació del tempo al llarg de la cançó. Per fer-ho, s'ha d'obtenir la durada de les notes durant la interpretació de les peces. Les notes tenen una durada determinada a la partitura, però aquesta varia substancialment respecte la durada de les notes tocades pels músics, ja que aquests toquen sense l'ajut de metrònom. És per això que existeix una variació en el tempo de la cançó respecte el tempo teòric. Obtenint la durada real de les notes, es pot calcular el tempo seguit pel músic.

En el nostre cas els músics estudiats són violinistes. El procediment per analitzar els temes musicals enregistrats consisteix en la gravació del so, i en la utilització d'uns sensors per obtenir la posició, el moviment, la força i la velocitat de l'arc durant la peça. Amb tots aquests paràmetres és possible aconseguir un model definit de la interpretació del músic.

Per calcular el tempo s'ha implementat un algorisme, anomenat *detector de polsos* o *beat detector,* que detecta el tempo seguit pel violinista durant la interpretació. Posteriorment, s'ha implementat un altre algorisme, anomenat *alineador* o *aligner* que, aprofitant els resultats de la variació del tempo, crea una partitura amb les duracions reals de cada nota.

Actualment, els sistemes existents que detecten el tempo de les cançons utilitzen només les dades de la gravació del so. Per tal de millorar l'eficiència d'aquests algorismes, proposem un nou sistema que, a part d'utilitzar el so, incorpora les dades provinents del moviment de l'arc.

Per determinar la qualitat del nostre sistema, s'han comparat els resultats amb els algorismes presentats al certamen anual Music Information Retrieval Evaluation eXchange (MIREX), on es presenten, entre d'altres, sistemes detectors dels polsos de les cançons. Després de realitzar diferents experiments, s'ha determinat que l'ús del moviment de l'arc incrementa notablement l'efectivitat de l'algorisme.

Finalment, per tal de millorar els resultats obtinguts, s'ha proposat un sistema que detecta els polsos de les cançons utilitzant les dades provinents d'un duet, trio o quartet com a senyals d'entrada. El fet que hi hagi més d'una veu en una cançó implica que hi hagi una relació implícita entre els tempos que segueixen els violinistes, que és la que utilitza l'algorisme proposat.

Resumen del proyecto

La investigación en la forma en la que los músicos ejecutan las piezas musicales conlleva al desarrollo de herramientas para poder medir algunos aspectos relativos a la interpretación.

En este proyecto se pretende estudiar una parte de la conducta de los músicos, como es la variación del tempo a lo largo de la canción. Para hacerlo, se obtiene la duración de las notas durante la interpretación de las piezas. Las notas tienen una duración determinada en la partitura, pero ésta varia sustancialmente respecto la duración de las notas tocadas por los músicos, ya que éstos tocan sin la ayuda de metrónomo. Es por eso que existe una variación en el tempo de la canción respecto al tempo teórico. Obteniendo la duración real de las notas, se puede calcular el tempo seguido por el músico.

En nuestro caso los músicos estudiados son violinistas. El procedimiento para analizar los temas musicales grabados consiste en la grabación del sonido y en la utilización de unos sensores para obtener la posición, el movimiento, la fuerza y la velocidad del arco durante la pieza. Con todos estos parámetros es posible conseguir un modelo definido de la interpretación del músico.

Para calcular el tempo se ha implementado un algoritmo, llamado *detector de pulsos* o *beat detector*, que detecta el tempo seguido por el violinista durante la interpretación. Posteriormente, se ha implementado otro algoritmo, llamado *alineador* o *aligner* que, aprovechado los resultados de la variación del tempo, crea una partitura con las duraciones reales de cada nota.

Actualmente, los sistemas existentes que detectan el tempo de las canciones utilizan sólo los datos de la grabación del sonido. Con tal de mejorar la eficiencia de estos algoritmos, proponemos un nuevo sistema que, a parte de utilizar el sonido, incorpora los datos provenientes del movimiento del arco.

Para determinar la calidad de nuestro sistema se han comparado los resultados con los algoritmos presentados en el certamen anual Music Information Retrieval Evaluation eXchange (MIREX), donde se presentan, entre otros, sistemas detectores de pulsos de las canciones. Después de realizar diferentes experimentos se ha determinado que el uso del movimiento del arco incrementa notablemente la efectividad del algoritmo.

Finalmente, con tal de mejorar los resultados obtenidos, se propone un sistema que detecta los pulsos de las canciones utilizando los datos provenientes de un dueto, trío o cuarteto como señales de entrada. El hecho de que haya más de una voz en una canción implica que haya una relación implícita entres los tempos que siguen los violinistas, que es la que utiliza el algoritmo propuesto.

Abstract

The research on the behavior of the music performance has led to the development of tools to measure some aspects about the interpretation.

This work aims to study some of the behavior of the musicians regarding the change of the tempo throughout the song. In order to do it, the length of the notes during the performance of the pieces has to be computed. Although the notes have a fixed length marked in the score, it varies substantially with the duration of the notes of the song. Because they play without the help of a metronome, this is why there is a variation of the tempo of the songs compared with the theoretical one. The actual tempo followed by the musicians can be computed by calculating the actual duration of the notes.

In our case, the musicians studied are violinists. The procedure to analyze the recorded songs consists of using some sensors to obtain the position, the velocity, the movement and the force of the bow used in the performance. With all these parameters it is possible to define an accurate model of the musical performance.

In order to calculate the tempo, a *beat detector* algorithm has been implemented. This algorithm detects the tempo followed by the violinist during the performance. Next, another algorithm has been implemented, named *aligner*. This one uses the results of the variation of the tempo obtained by the beat detector, and creates a new score with the real durations of the notes.

Currently, the existing systems that detect the tempo of the songs only use the recording data as an input. In order to improve the efficiency of these algorithms, we propose a new system that not only uses the recording, but also the data of the bow displacement.

In order to determine the quality of our system, the obtained results have been compared with the results of the algorithms submitted to the annual evaluation campaign Music Information Retrieval Evaluation eXchange (MIREX), in which different beat detector algorithms are presented and analyzed, among others. After conducting different experiments, it can be concluded that the use of the bow displacement increases significantly the efficiency of the algorithm.

Finally, in order to improve the obtained results, we propose another system that detects the beats of the songs using the data of a duet, trio or quartet as inputs. In these cases, there is more than one violin playing at the same time, and there is an implicit relation between the tempos followed by the violinists. The proposed algorithm takes advantage of this relation to obtain the actual tempo.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Context

There is currently a project in the European Union called SIEMPRE (Social Interaction and Entrainment using Music PeRformance Experimentation) that aims to develop new theoretical frameworks, computational methods and algorithms for the analysis of creative social behavior within small groups of people. The areas of research are focused on ensemble musical performance. Their goal is to create models, techniques and algorithms for the extraction of social features based on the analysis of synchronization processes underlying expressive movements, audio and biometric signals during interpersonal creative communication. The Music Technology Group (MTG), a department of Universtitat Pompeu Fabra, is one of the partners of this project among others (University of Genova, Queens University of Belfast, University of Geneve and Italian Institute of Technology). The tasks of the MTG in the project are the acquisition, analysis and modeling of bowed-string performance gestures involved in sound production mechanisms.

1.2 Motivation

The need to model a system that synchronizes a recording with its score comes from the research on how the musician plays. The essential idea is to study the expressivity of the musician, in this case a violinist, when he/she is playing, and identify the differences between the produced sound and the score.

During the performance, the tempo followed by the violinist does not correspond exactly with the tempo marked in the score. This difference of tempo is one of the characteristics of the expressivity of the musicians. In order to calculate this difference, it is necessary to develop a system that finds the best correspondence between the audio and the score. To do that, it is indispensable to process the audio signal in order to determine the variation between the duration of the notes played by the violinist and their theoretical duration.

The systems developed so far try to make the synchronization by only processing the audio recording; they do not take into account the movements of the musicians. The challenge of this thesis is to develop a system that synchronizes the audio with the score, using the audio recording and also multimodal information extracted from the performance of the musician. This method aims to improve the results of the existing algorithms, and demonstrating that the movements of the musicians are an important part of the performance, and can be useful in the synchronization.

The multimodal information used in this system is obtained by the sensors placed on the violin and on the bow. They extract the information of the position of the bow, so that the speed of the movement of the bow, the force applied by the bow on the strings, the string used by the violinist and other related information can be computed. With all this information, and the recording, it is going to be implemented the algorithm.

The steps of the algorithm are:

- Firstly, the tempo followed by the violinist, which is continuously changing along the song, is tracked and identified by the beat detector.
- Secondly, the detected tempo is compared to the theoretical one, so that each single note can be identified and matched to the one written in the score.
- Finally, the aligner algorithm is applied to create a score with the real duration of the notes.

1.3 Objectives

In order to improve the results of the methods already existing, some of the gesture data extracted from the violinist (such as the movement of the bow) will be used. One way to implement the method that detects the tempo of the song is to find the beginning of its beats, so it is necessary to process the audio and detect changes in frequency and peaks of the signal. Thus, along with the musical recording, some gesture data extracted from the violinist - such as the movement of the bow - is processed, as it can be helpful in determining the beginning of each beat.

This implementation presents several challenging problems. On the one hand, the fact that the violinists are recorded without metronome causes a considerable variation in the tempo during the performance. This makes it important for the algorithm to be able to adapt its decisions to these possible changes. On the other hand, the accuracy of the audio processing is an important requirement. As opposed to a symphony orchestra, in which the brass instruments and the percussion define the intrinsic tempo of the song, the signal produced by a violin does not have sharp attacks, making it more difficult to detect the tempo. Thus, an accurate processing method is required for the current system.

The main goal is to process the available data by weighting the different parameters in order to decide the position of each beat. This task is performed by the pulse tracking method, which gives the actual tempo as an output. Afterwards, another algorithm will match the audio recording with the score.

As a final evaluation, predictions will be compared with manual annotations. It is going to be used a cross fold validation method, and the results of the implemented algorithm will be compared to the ones obtained by the existing methods submitted to the Music Information Retrieval Evaluation eXchange (MIREX).

The last part of the project consists of improving the algorithm so that it can be applied to other musical ensembles, such as duets, trios or quartets. This is accomplished by taking advantage of the implicit synchronization among performers.

Chapter 2

Theoretical framework

Although this project is based on the analysis of the audio signal and data processing, it is also necessary to know some music concepts to understand how the system works. In this chapter the theoretical fundamentals are explained, as well as the procedure used to extract the performance data from the violinist.

2.1 Basic concepts of music

2.1.1 Musical figures

A basic introduction to the musical figures starts showing the most used figures in music. The *whole note* is the longest note, and as from this, the other musical figures were created. As the name of the figure indicates, the duration of each figure is the half of the duration of the higher one (see Figure 2.1).

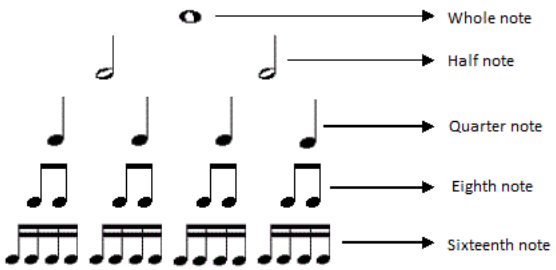

Figure 2.1: Musical figures.

For each musical figure there is a rest of the same duration. The correspondence of them is shown in the Figure 2.2.

Figure 2.2: Musical rests.

2.1.2 Beats per minute (BPM)

Beats per minute or *BPM* is the unit used to measure the tempo in music. It indicates the number of beats in one minute. Each beat is a pulsation of the song, and it is typically what listeners tap with their foot or clap with their hands along with a piece of music. To indicate it in a score, a musical figure is used followed by an *equal sign* and a number (see Figure 2.3). This means that, in one minute, there are as many figures as the number indicated after the equal sign.

Figure 2.3: Example of BPM.

Although any musical figure can be used to indicate the BPM, the *quarter note* is the most commonly used. The time interval between beats can be calculated from the BPM. In the numerical example illustrated in the Figure 2.3, if in one minute fit 60 quarter notes, each quarter note takes one second, so there is one second between beats.

For simplicity, the time interval between beats is used several times in this thesis instead of the BPM. More specifically, the parameter used is the *Inter-Onsets Interval* or *IOI* which indicates the interval in seconds between two consecutive onsets of the notes (see Figure 2.4).

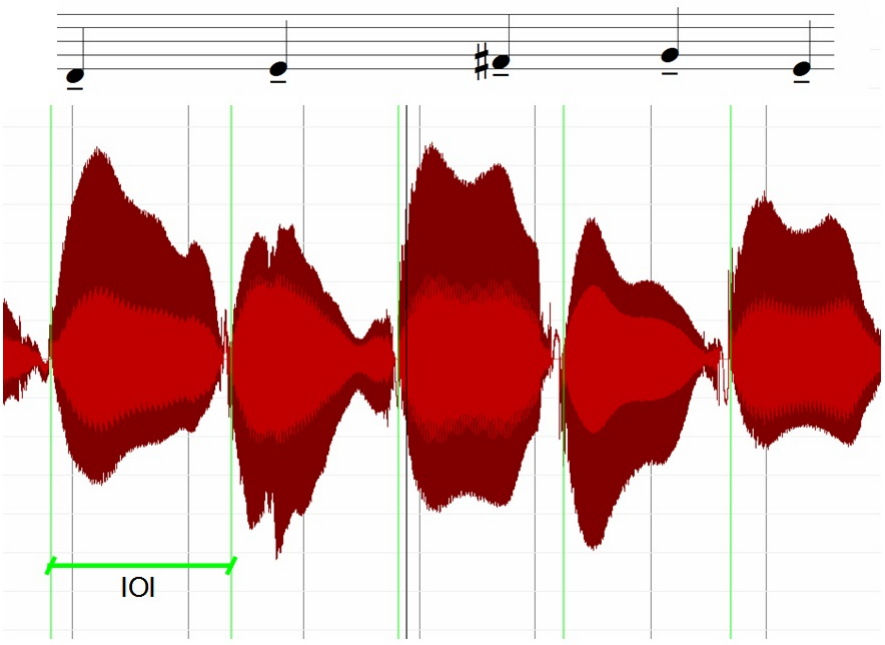

Figure 2.4: Example of an inter-onsets interval.

Musically, the range of beats per minute varies approximately from 40 to 208 (see Table 2.1). It is important to take it into account in order to choose the resolution of the windows used in the beat detector, as it will give a reference of the duration in seconds of the notes.

Name	BPM
Largo	40 - 60
Larghetto	60 - 66
Adagio	66 - 76
Andante	76 - 108
Moderato	108 - 120
Allegro	120 - 168
Presto	168 - 200
Prestissimo	200 - 208

Table 2.1: Relation of tempo names and BPM.

To simplify calculations, the quarter note has been taken as the musical reference. However, the implemented beat detector has been designed to work with any of the existing musical figures as a reference.

2.1.3 Downbeat and number of beats per bar

The *downbeat* is the impulse that occurs at the beginning of a bar in measured music. It frequently carries the strongest accent of the rhythmic cycle. However, in some cases, the downbeat may not be emphasized.

The number of beats per bar is indicated by two numbers at the beginning of the score (see Figure 2.5). The top number means the number of notes of the type of the down number, that fit in a bar. The down number indicates the specific type of note; 1 represents the longest note (a whole note) and the successive notes are indicated by a power of 2 (2^n). The relation $\frac{4}{4}$, which means four quarter notes per bar, is the most commonly used.

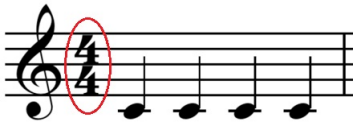

Figure 2.5: Beats per bar.

2.1.4 On-beat and off-beat

When using the $\frac{4}{4}$ time, the first beat of the bar often carries the strongest accent of the melody, while the third beat carries the next strongest accent. These beats are called *on-beats*. The second and fourth accents, called *off-beats,* are weaker. Subdivisions of the beats are even weaker than the own beats, and they also follow the same rule. Figure 2.6 illustrates an example, with the importance (in numbers) of each beat, and the importance of every subdivision of each beat.

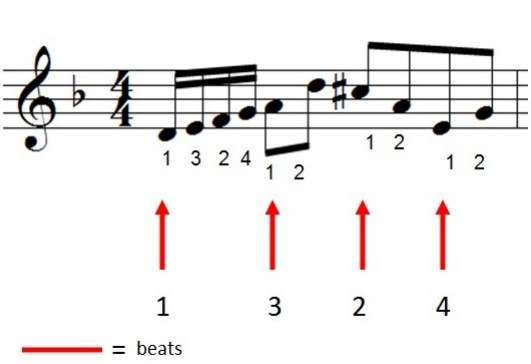

Figure 2.6: Example of the importance of each beat and every subdivision.

This concept must be correctly taken into account because the beat detector tries to detect the first on-beat of the subdivision of each beat. The intensity of the accent carried by the first on-beat, will be used to predict the beat.

2.1.5 Difference between tempo and timing

It is important to differentiate between *tempo* and *timing* concepts. Given a metrical structure, *tempo* is defined as the rate of beats at a given metrical level, such as the quarter note level in the score.

Timing is the position where the onsets of the beats are located. Figure 2.7 shows the difference between tempo and timing changes. A change in *timing* does not represent a change in *tempo*; it is only a punctual variation in the onset of the beat, without modifying the beat rate.

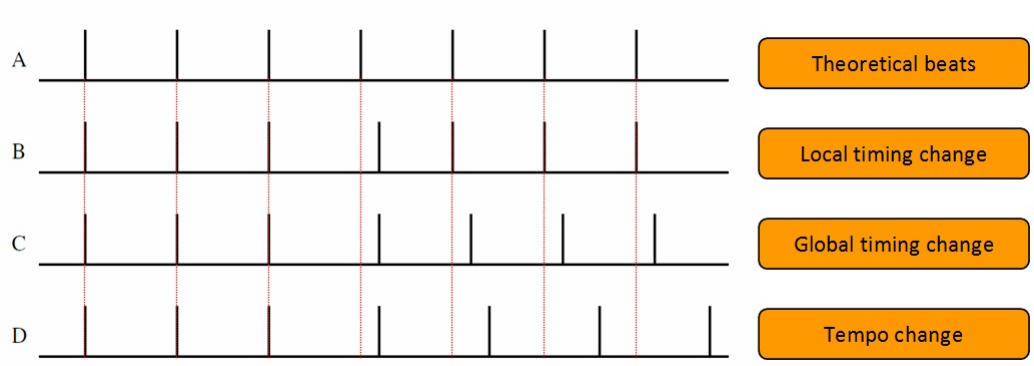

Figure 2.7: Difference between tempo and timing.

The current system has been designed so that it tracks the variations of both parameters; *tempo* and *timing*.

2.1.6 Beginnings of a song

A score can begin in three different ways, depending on the relative position of the first note and the first beat. The most common case, the thetic beginning, corresponds to the situation in which the song begins

on the first beat, which means that the first note carries the first beat. An example of a thetic beginning is shown in Figure 2.8.

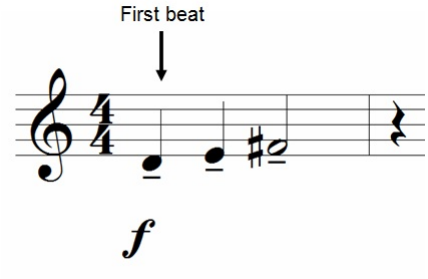

Figure 2.8: Example of a thetic beginning.

Another way to start a song is the anacrusic case, in which the song begins before the first beat. It is normally used to emphasize the first beat. Figure 2.9 illustrates the anacrusic beginning.

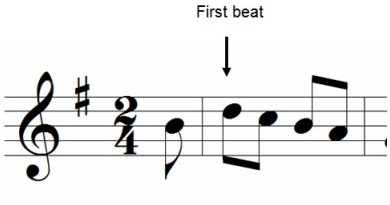

Figure 2.9: Example of an anacrusic beginning.

The last one, and the less used, is the acephalous beginning. It corresponds to the case in which the song starts with a rest and the position of the first beat coincides with this rest (see Figure 2.10).

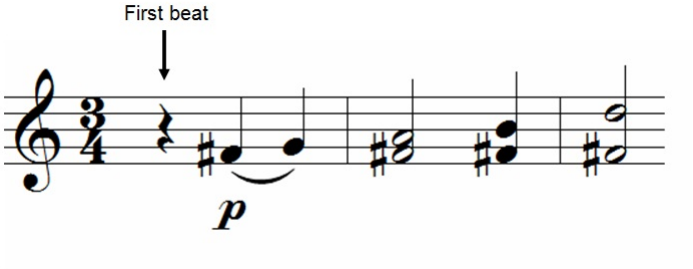

Figure 2.10: Example of an acephalous beginning.

The position of the first beat usually coincides with the first note in a song. However, if this is not the case, the designed algorithm has to take it into account when detecting the position of the first beat.

2.2 About violin

2.2.1 Violin

A violin is a melodic instrument with four strings. The lowest note that it is able to play is a Sol_3 or G_3, sound produced by the vibration of a string at 196 Hz. Under this frequency the violin does not produce any sound. It does not have a top note limit, but violinists usually play frequencies up to 1,4 kHz.

Strings are tuned using the pegs and the fine tuners, adjusting the tension of each string and, consequently, changing the oscillation frequency. The sound produced by the string goes through the *'f'* holes into the sound post, where it is amplified using a resonance cavity (see Figure 2.11).

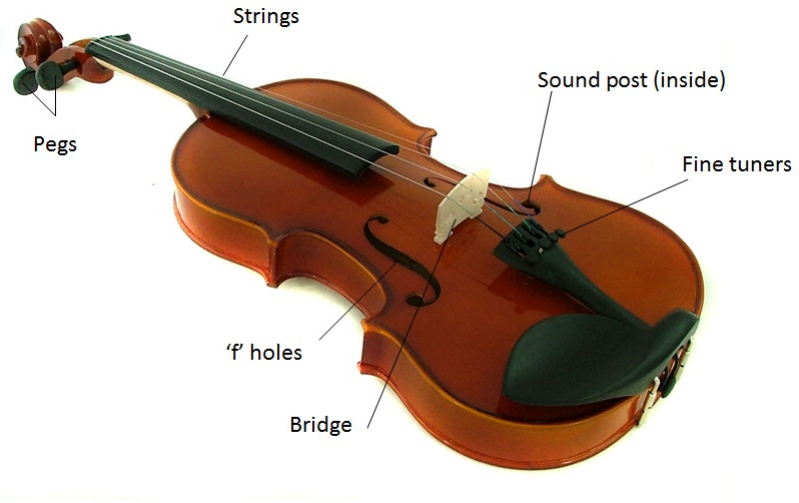

Figure 2.11: Violin.

2.2.2 Bow

The violin is played using a bow. A bow is a stick with ribbon of horsehair strung between the tip and the frog (see Figure 2.12). A typical violin bow may be 75 cm overall, and uses approximately 65 cm of it. At the frog end, a screw adjuster tightens or loosens the hair.

Figure 2.12: Bow.

2.2.3 Music features

Music produced by the violin is normally a soft sound and does not have pronounced attacks. The notes played with a change of the direction of the bow have a little accent, which is produced by the beginning of the friction of the bow with the string. However, this accent is not comparable to the one from the notes played by brass instruments (metal-wind instruments) or by percussion, which is clearly marked and pronounced. This is going to be one of the challenges that the proposed algorithm has to deal with.

Another important matter is the way in which the violinist uses the bow. When the bow is going down, the violinist can apply more force to the string than when it is going up, so normally, when the violinist wants to emphasize a note, he/she changes the direction of the bow from up to down. Musically, this effect is used to give more importance to the beats 1 and 3 (in a $\frac{4}{4}$ time) of the bar, as explained in Section 2.1.4. Moreover, if there is more than one note in an interval of two beats (for instance, the use of eighth notes in a $\frac{4}{4}$ time), the first one (which is the most important one and the one that starts with the beginning of a beat), is usually played with a change of the direction of the bow from up to down to emphasize the note. Thus, the direction of the bow can be also useful for the algorithm to help identifying the location of the beats.

2.3 Audio and gesture data acquisition

This project uses audio and data extracted from the violin, so it is precise to summarize a bit how it works.

The system was created in the Music Technology Group at Universitat Pompeu Fabra. The essential idea of the implemented system is to place a sensor and a microphone on the violin, and another sensor on the bow. This way, the gesture parameters can be extracted from the relative positions of the sensors (see [1] for further information).

- **Audio acquisition:** Is recorded using a piezoelectric pickup placed in the bridge of the violin, so that it captures the mechanical vibrations and converts them into an electrical signal. This way, the environmental noise is avoided as well as the sound of other violins, in the cases in which there is more than one violin playing at the same time.
- **Gesture data acquisition:** One sensor is placed on the back of the violin and another one on the frog of the bow. These sensors are detected using the *Polhemus Liberty*; a commercial device which is based on electromagnetic field sensing, and consists of a transmitting source and two sensors.

A sketch of the position of the sensors is illustrated in the Figure 2.13.

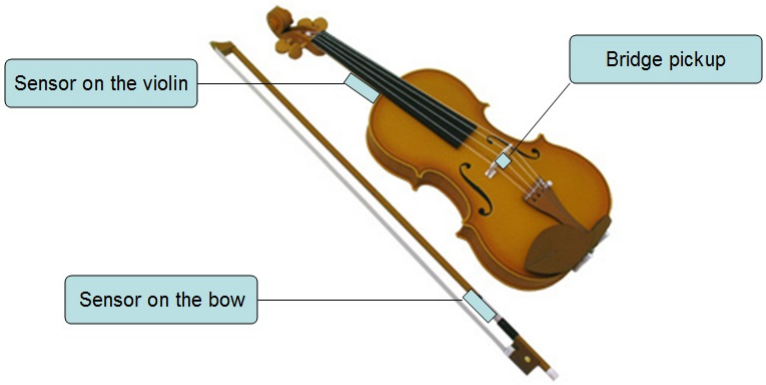

Figure 2.13: Sketch of the sensors.

These sensors allow us to detect their position in the space and calculate the distance between the bow and the violin at any time. The velocity of the bow, the string used, the tilt of the bow and the force applied to the string, among other parameters, can be calculated from the relative position the sensors.

However, the only parameter used in the current implementation is the information about the bow displacement, since the other parameters are not relevant for the algorithm. The information about the bow displacement allows us to find the instants in which the violinist changes the direction of the bow.

Chapter 3

State of the art

This chapter presents an overview of the existing techniques for both the beat detection and the synchronization of a recording with its score. It is important to note that the use of bowing gesture parameters have not been used for beat detection before.

The initial attempts in the field of beat detection were focused on calculating the fluctuation of the tempo during the song. In 1993, J. Bilmes suggested representing timing deviations in percussion as systematic shifts occurring within the span of the fastest beat, while keeping a constant execution speed [2]. Later, in 2002, A. Friberg and J. Sundström, focused their research on the swing ratio, which refers to the mathematical expression of the duration of the first eighth-notes divided by the duration of the second eighth-notes [3]. Among others, these were some ways to analyze the deviation of the timing and tempo, but the main goal was to detect the real tempo or the BPM. This was the purpose of a system designed in 2001 by S. Dixon [4]. However, this and most systems available by that time made many mistakes in the estimation of the real tempo. The ratio between the estimated and the actual tempo was usually a rational integer, such as 70 BPM instead of 140 BPM. This was due to the fact that the multiples of the tempo were sometimes identified as the real tempo of the song and the systems were wrong. This problem could have been avoided if these systems had had a reference of tempo extracted from the real score (see Figure 3.1).

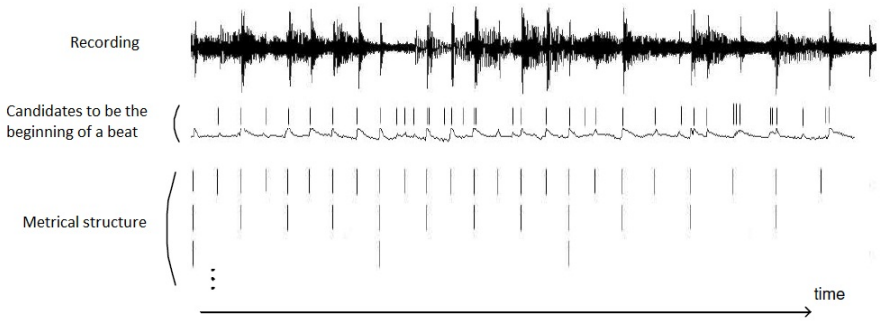

Figure 3.1: Example of an audio recording (top), its possible onset candidates (middle), and the estimated metrical metrical structure (bottom).

Other differences between the algorithms were the inputs used; Y. Wang and M. Vilermo designed an algorithm that used an MP3 audio song as an input [5], while A. Schloss only used acoustic signals as inputs for an automatic transcription system aimed to work with percussive music [6].

J. Brown made the first attempt at using only the score to calculate the correlation of some parts of the song, in order to find relations and patterns [7]. However, the problem of multiples of the tempo was still present.

All the systems mentioned above, regardless of the input data used (score or acoustic signal), follow a specific procedure to estimate the tempo:

1. First, a feature list with all the beat candidates is created.

2. Next, the algorithm decides which candidates are correct and which are wrong.

The features taken into account when detecting the beats depend on whether monophonic (one voice) or polyphonic (more than one voice) music is being analyzed.

Polyphonic music

The percussive instruments present in a symphonic orchestra clearly mark the tempo of the music. Thus, the global tempo in a polyphonic song can be easily estimated by searching only the high energy peaks of the signal (D. Rosenthal, 1992 [8]).

Monophonic music

Monophonic songs do not usually have high energy peaks, making it more complicated to estimate the tempo. Thus, all the possible onsets in the song have to be identified. Moreover, some other features of the recording, such as the fundamental frequency of the audio recording, have to be analyzed in order to help matching the recording with the score.

Some examples of onset detection methods are the automatic system designed in 2000 by A. Cemgil et al. [9] and the system based on a graphical model designed by C. Raphael [10].

There are two methods used to calculate the followed tempo in a song; the pulse selection method and the periodicity function computation, both described below.

Pulse selection

In the pulse selection method, each Inter-Onsets Interval (IOI) defines a possible beat period, and the corresponding events in the instant where the beat starts define the importance or phase of the beat. The main challenge is how to define the length of the beats. For example, C. Longuet-Higgins and C. Lee simply considered the first two events as the first two beats [11], whereas R. Dannenberg and B. Mont-Reynaud took the first two agreeing IOIs as defining the beat [12].

P. Allen and R. Dannenberg used a metrical value given from the score (the theoretical BPM) for the first event. From this value and the first IOI, they derived the length of the beat [13].

Another way to define the length of the beat is to consider the first two events as a potential beats, and the subsequent events considered using the length of this potential beat. If subsequent events do not coincide with the proposed beat, a new potential beat is considered using these events.

In order to define the importance (or the phase) of the beats, some considerations have to be taken:

- **Amplitude**

The relative amplitude is a factor that greatly contributes to perceptual accentuation. Since it can be easily computed from recordings, it is used for weighting onsets derived from audio data. In 1999, M. Gasser et al. developed a system using also this feature [14].

- **Division in frequency bands**

This method consists of dividing the recording in several frequency sub-bands. For each frequency sub-band, a feature list is created. Finally, the algorithm creates a final list with all the possible beat candidates by integrating all the single list for each sub-band. J. Herre et al. created a system using this method [15].

Periodicity function computation

An alternative method to pulse selection is the computation of a periodicity function. It is based on finding different periodic patterns in the audio signal and deciding which one is the real the tempo of the song. Periodicity functions are usually calculated in several frequency sub-bands using the Fourier Transform and the autocorrelation function. An example of this method can be found in the system designed by F. Gouyon and P. Herrera which consists of creating several feature lists, then calculating some periodicity functions for each sub-band and, finally, integrating the results to get the final tempo [16].

An improvement to this system was implemented by R. Parncutt, by multiplying the periodicity function by a tempo preference distribution, taking advantage of the fact that humans consider tempo with some a priori preference [17].

Another example of a system that uses the periodicity function method is the one designed by J. Brown, which computes a sample-by-sample autocorrelation function of a sequence of onsets, and weights them by their durations [7].

These are some different methods used to calculate the variation of the tempo in the song. For the current project, the pulse selection method has been selected, taking into account the amplitude of the onsets and some additional features described below.

Chapter 4

Methodology

4.1 Problem statement

As mentioned above, the main objective of the implemented system is twofold. First of all, it aims to estimate the number of *Beats Per Minute* (BPM) used by the violinist during the performance, then to match the notes played by the violinist with the corresponding score and, therefore, to establish how long each note lasts.

The data that is going to be used is the audio signal of the recording, the position of the bow and the violin and a digital representation of the score. The type of data used is a *.WAV* file for the audio signal, a text file with all the positions of the bow (see Section 4.2.1.3) and another text file for the digital score with all the notes and the theoretical duration of each one (see Section 4.2.4).

The block diagram of the system is shown in the Figure 4.1.

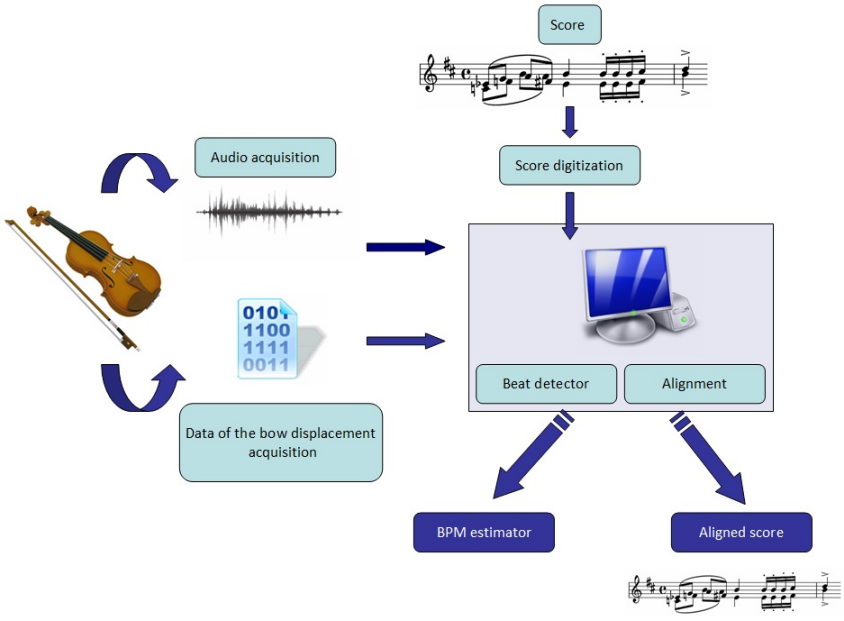

Figure 4.1: Block diagram.

4.1.1 Possible approaches

A possible approach to match the recording with the score would be to detect the fundamental frequency of the recording at each instant of time and matching it with the theoretical frequency obtained from the score. However, in the case that the violinist misses a note, misses the exact tuning or plays a note that is not in the score, the detection of the fundamental frequency might not be correctly detected. In this thesis, the fundamental frequency of the recording is not going to be matched directly with the score, but it is actually going to be used for the detection of the changes of the notes (explained in Section 4.2.1.2).

An improved method to align the recording with the score consists of finding the beats in the recording and changing the theoretical tempo in the score for the estimated one. Figure 4.2 illustrates this idea.

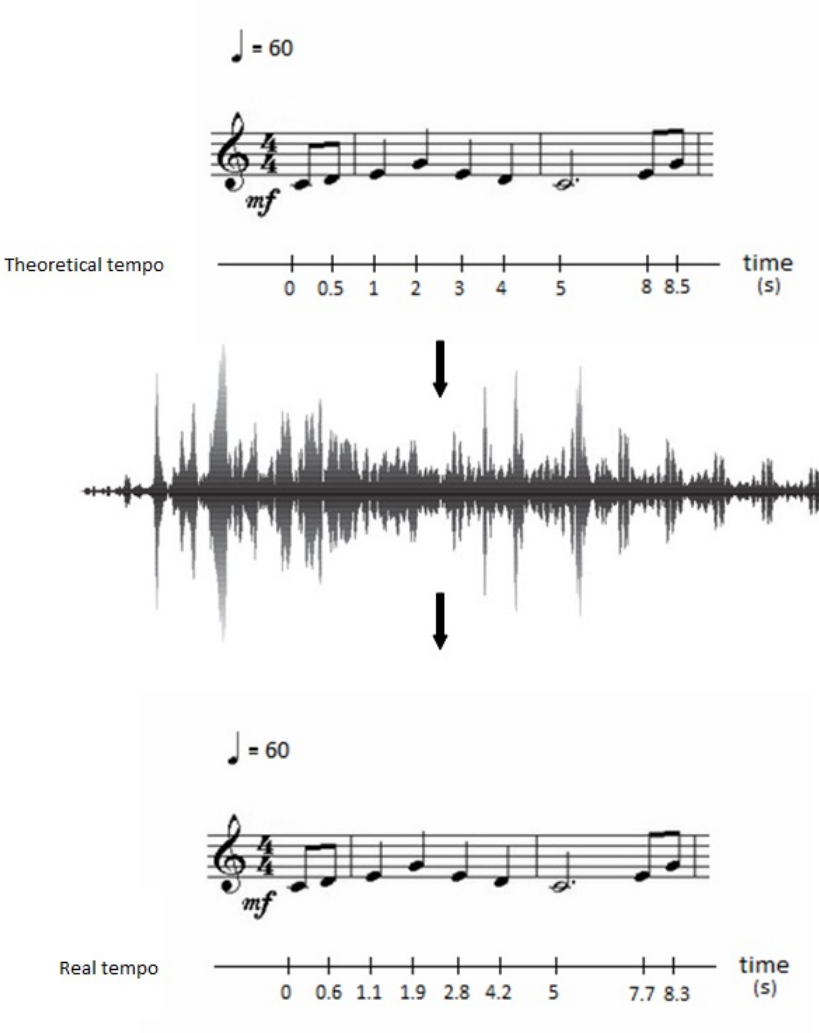

Figure 4.2: General idea of the alignment.

The next step is to calculate how long each note lasts. To do this, we are going to estimate the position of the beats by using the audio recording and the information obtained from the violin performance. As mentioned before, the technique used by the implemented algorithm is the pulse selection, more specifically, the detection of the onsets of the beats. The reason why this method has been chosen is the fact that it is easier to detect the beginning of a beat because the energy at the beginning of a note is higher than at the

middle or at the end. This is consequence of the pressure that the violinist has to do with the bow to start the attack of each note.

It is worth to remark again that the approach of using only the audio recording to detect the real tempo of the song provide results that can be improved. Therefore, the bowing gestures and some additional features extracted from the recording will be incorporated later on the implemented algorithm.

4.1.2 Possible beginnings of a beat

The beats in a song usually start at the beginning of a note. Moreover, the energy of the notes that start with a beat is often higher than the one from the notes that do not start with a beat. This is clearly marked when the songs that are being analyzed were extracted from a symphony orchestra, in which the percussion marks the rhythm and the onsets of the notes are clearly pronounced. However, the audio signals recorded from a violin are very soft in the sense that they do not have strong accents, making it more complicated to detect the beginnings of the notes.

We also want to take advantage of the fact that usually, at every new beat, there is a change of the note. Thus, the algorithm focuses on locating the peaks of the signal (high energy notes) and the moments in which there is a change of the note.

In order to improve the possible results, in the current project is used the information extracted from the performance of the violinist. Actually, the extra-information used is the bow displacement. With this information it is possible to know at every moment the position of the bow and detect all the changes of the direction. We assume that bow direction change coincide with the start of a new beat, because the change of the direction produces a little accent, which is used to mark the onsets of the beats. Moreover, if the direction changes from up to down, the probability of a new beat is higher than if the direction changes from down to up. This is because the accent given by the bow is higher if the change of the direction is from up to down, and it is used to mark the tempo.

4.1.3 Procedure

The first step is to implement the beat detector algorithm, which will detect the start of each beat and obtain the resulting BPM. One goal of the thesis is to find this information, because itself, it is profitable to investigate about the behavior of the musicians and the changes of the tempo and timing during the performance, so with this algorithm we will achieve useful results.

Next, the aligner algorithm is implemented. Its main task is to match the recording with the score. Taking as an input the BPM obtained by the beat detector, it gives as an output a file that contains the real duration of each note.

After processing the data extracted from the violinist, the candidate instants in which the beats can start are stored in a vector. The candidate instants are selected according to several features, such as high energy peaks, changes in the notes and changes in the direction of the bow. After weighting these parameters, is used a tree shape algorithm that calculates all the possible combinations of beats and decides which beat candidates are valid.

4.2 Beat detector

The procedure followed to implement the beat detector algorithm is detailed in this section.

The main features of the beat detector are:

- Detection of the onsets using:
  - The peaks of energy of the signal.
  - The changes of the notes; changes of the fundamental frequency of the audio recording.
  - The changes of the direction of the bow.
- Detection of the beginning and the end of the recording.
- Detection of the estimated average tempo.
- Adaptive tempo; the tempo of the song can vary considerably during the song.
- Division of the song using the rests to improve the results.

4.2.1 Data processing

In order to identify when the beats start, it is necessary to process the data extracted from the violinist. After that, a vector containing the possible beginnings of a beat is created.

4.2.1.1 High energy peaks of the recording

The first data processed is the audio recording. As mentioned before, the audio signal is acquired using a pickup placed on the bridge of the violin. The audio signal is recorded using a sampling frequency of 44.100 Hz. Thus, for a typical song with a duration of 5 minutes, we will have around $5.2{\cdot}10^6$samples, which requires too much computational memory to be processed for a typical computer. Therefore, the vector containing the song has to be downsampled. The downsampling rate that has been used is 200, so the resultant vector has around $2.6{\cdot}10^4$samples, which is more affordable for a basic computer.

Important information used to decide when the beats start, are the peaks of the recorded audio signal. As mentioned before, beats usually start in a note with high intensity. The best way to find these high peaks is by computing the Root Mean Square (RMS) energy of the audio signal (see Equation 4.1), which is a good method to avoid weak peaks. The RMS calculates the root mean square energy of a period of the signal, so that the resulting signal is smoothed. To perform this calculation, the algorithm uses a moving rectangular window, whose length has to be long enough to avoid weak peaks of energy and short enough to not to hide two important consecutive peaks. The length of the window is chosen taking into account the fact that the algorithm must be able to discriminate between two consecutive notes in the case in which the song has a fast tempo (i.e., a short period of time will contain a large number of notes). An example of a typical fast tempo is the *allegro* tempo, which is around 120 - 168 BPM. In this case, assuming that the basic value is the quarter note and taking the worst case, each beat would last $\frac{60\,s}{168\,beats} = 0.36$ s/beat. Assuming also that the fastest musical figure usually used in this tempo is the sixteenth note, each sixteenth note would take around $\frac{0.36\,s/beat}{4} = 0.09$ s/beat'. Finally, taking into account the fact that all the recordings used a sampling frequency of 44.100 Hz and the downsampling rate used was 200, each quarter note has a number of samples around $\frac{44100\,Hz}{200}{\cdot}0.09$s/beat' $\simeq$ 20 samples/beat', where *beat'* is each beat for one sixteenth note.

But the algorithm must be able to calculate a faster tempo, such as *presto*, which is around 200 BPM. In this case it is reasonable to assume that the fastest note played is the eighth note, since it is complicated to play faster. Thus, each eighth note would take $\frac{60\,s}{200\,beats}\frac{1}{2} = 0.15$ s/beat', which corresponds to $\frac{44100\,Hz}{200}{\cdot}0.15$

s/beat'$\simeq$ 35 samples/beat'. In this case the length of the window is higher than in *allegro* tempo, because the fastest musical figure considered has been the eighth note instead of a sixteenth note.

Therefore, the chosen length for the window is 20 samples, which is the minimum resolution to discriminate the fastest musical figures used in most cases. Note that, if a higher resolution was chosen in order to discriminate faster tempos, the RMS would not affect songs in lower tempos and weak peaks would remain in the signal. So, if the used songs are not faster than 170 BPM or around, the chosen length of the window will be useful.

$$RMS[i] = \sqrt{\frac{\sum_{i-\frac{L}{2}}^{i+\frac{L}{2}} x[k]}{L}} \tag{4.1}$$

Figure 4.3 shows an example of a recording and its RMS. Note that in the RMS energy plot, the weak peaks have disappeared.

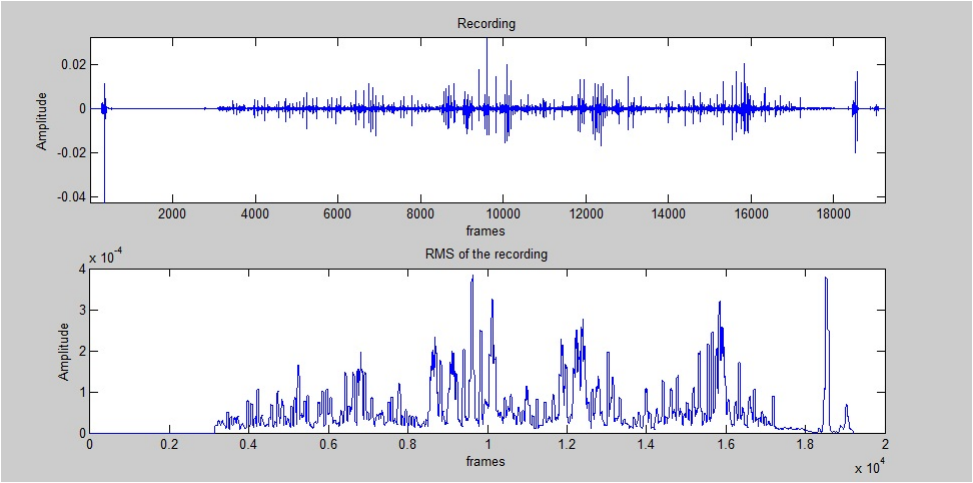

Figure 4.3: Recording and RMS of the recording.

The chosen peaks are the ones that are above a selected threshold. It is also important to take into account that typical songs usually have variable intensities, such as *forte*, *piano,* etc. all in the course of the same song, so the energy of the signal in each part of the song changes. Thus, the selected threshold has to be variable and, consequently, it has to be calculated for each sample of the recording.

Specifically, it is calculated using the Formula 4.2, which calculates the average RMS energy of the signal in a window of 8 times the Inter-Onsets Interval (IOI). The window size can be considered a reasonable value taking into account the fact that a period of time equal to 8 times the IOI contains approximately 10-20 notes. Whit this number of notes, the average energy of the window has a realistic value of the energy level of the part of the piece of the song.

$$Threshold\,[k] = \frac{\sum_{n=-4\cdot IOI}^{4\cdot IOI} RMS[k+n]}{8\cdot IOI} \tag{4.2}$$

4.4 shows an example of the resulting signal after discarding the peaks below the selected threshold. The samples in the resulting signal are called in this thesis "high energy samples".

The next step is to find the onset of each set of high energy samples. Onsets are often located not far from the maximum peak of each set of samples. This is due to the fact that the high intensity of each note

is usually located at the beginning of the note, because of the pressure exercised by the bow on the string. The intensity decreases as the note finishes.

The procedure to locate the onsets consists on looking for the maxima in each set of high energy samples. In the case that there is only one maximum, the first sample of the current set of samples is selected, as we assume that the onset is close to the beginning. Figure 4.5 shows an example in which the red line indicates the value selected as an onset of a note and, consequently, a possible beginning of a beat.

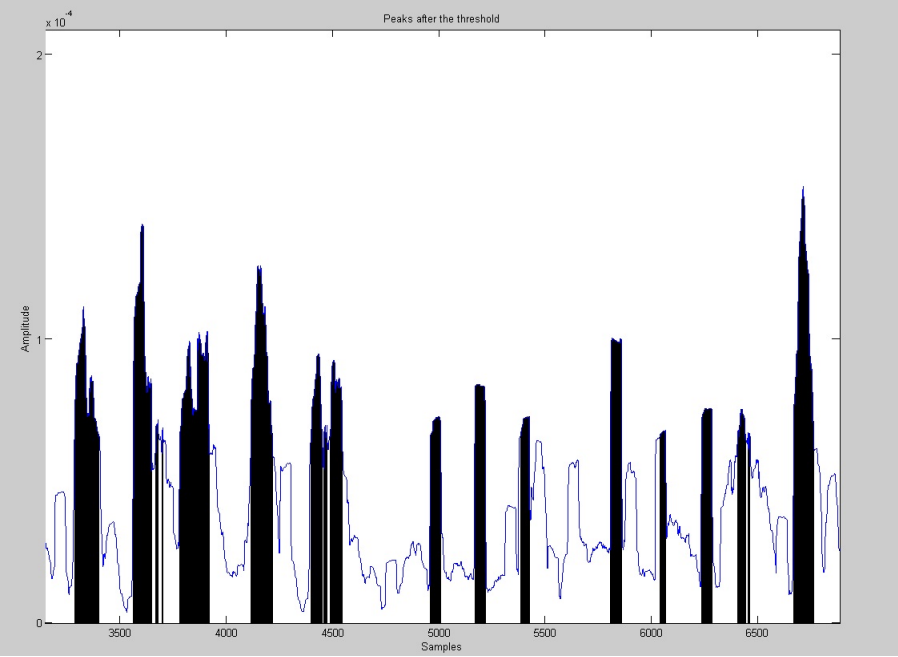

Figure 4.4: Sets of samples with high energy.

If there is more than one maximum peak in a set of high energy samples (see Figure 4.6), the peaks are smoothed in order to avoid false (or not relevant) maximums. To do it, each set of samples with high energy is convolved with a rectangular pulse of 5 samples. After that, if there is still more than one maximum peak, the next step is to find the beginning of each one.

The first sample of the first maximum peak is considered to be the first onset. If there is more than one maximum, the next onsets are selected by finding the minimum values in the signal (see Figure 4.6).

The positions of each selected peak are stored in a vector, altogether with the energy value of the corresponding peak. Later on, the energy will be used to decide the importance of each peak.

4.2.1.2 Fundamental frequency detector

In order to detect the positions in which the notes change, the fundamental frequency of the audio recording is calculated at each instant of time.

There are different methods to estimate the fundamental frequency of a song, either in the time domain or frequency domain.

An example of a simple algorithm that works in the time domain consists of measuring distances between zero crossing points. However, it does not work well with complex waveforms which are composed of multiple sine waves with different periods. More sophisticated algorithms use the autocorrelation function of the audio recording in order to find periodicities in it.

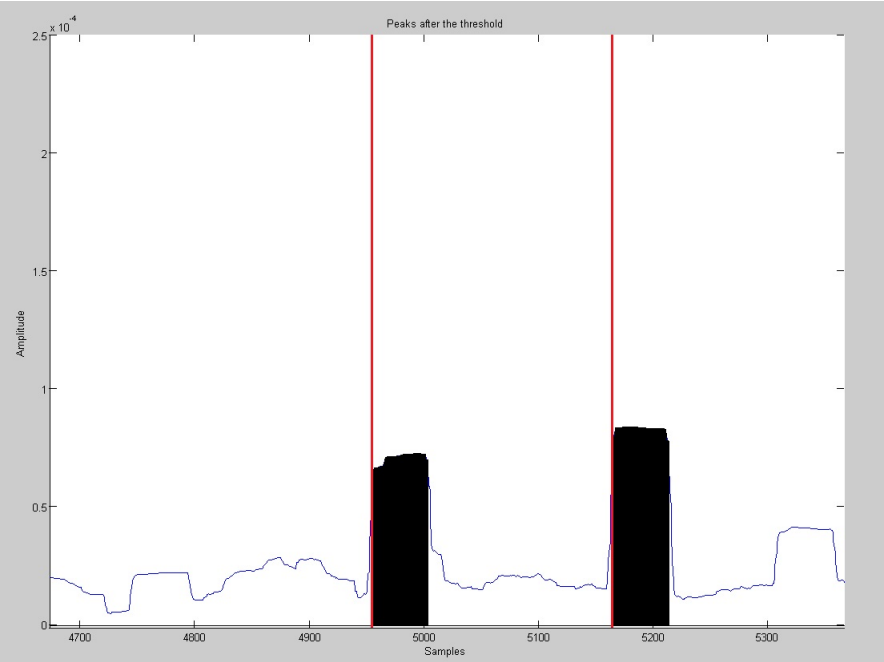

Figure 4.5: Example of two different sets of samples with high energy, with only one maximum peak in each group.

The algorithms that work in the frequency domain usually make use of the Fast Fourier Transform (FFT) to convert the signal into a frequency spectrum.

The algorithm used in the current thesis, named Yin [18], uses the autocorrelation function (ACF) in the time domain, in addition to some extra features that avoid possible errors in the ACF (see Figure 4.7). This method relies on the fact that in response to a periodic signal, the ACF shows peaks at the multiples of the signal period. Therefore, the fundamental frequency can be obtained by calculating the periodicity among these peaks.

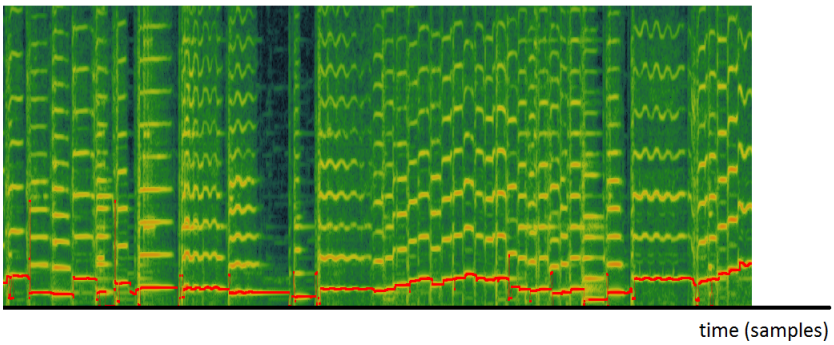

Figure 4.7: Example of a spectrum of a recording. The red line indicates the fundamental frequency, while the rest of the lines correspond to the harmonics (multiples of the fundamental frequency). Note that the color depends on the magnitude of each frequency.

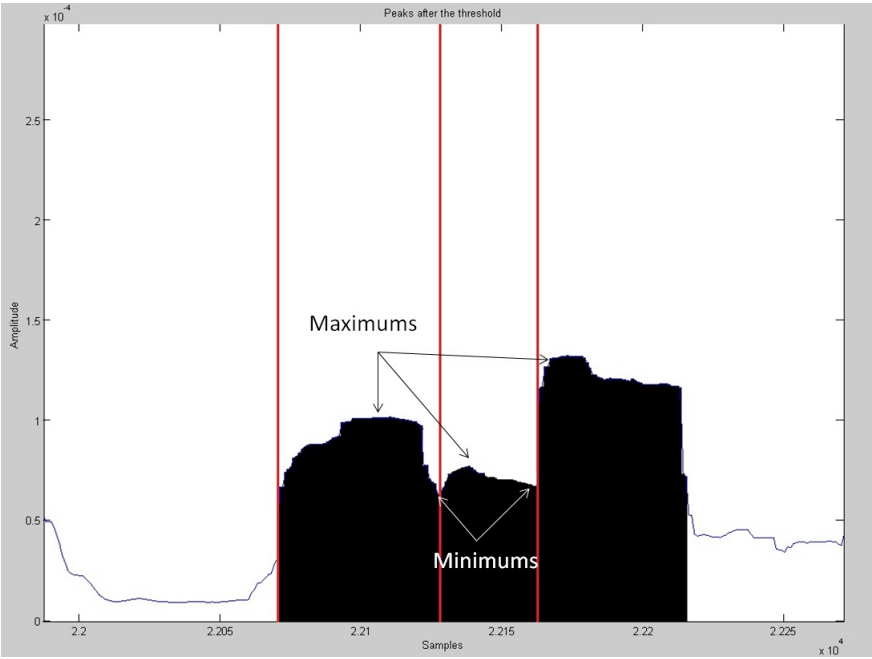

Figure 4.6: Example of a set of peaks with more than one maximum peak.

Taking the audio recording as an input, the algorithm gives as an output a vector that contains the fundamental frequency converted to a linear scale of octaves, denoted by $La_4 = 440$ Hz. The scaling operation performed by the algorithm is indicated in Equation 4.3.

$$n = log_2\left(\frac{b}{a}\right) \tag{4.3}$$

Where a is 440 Hz, b is the note that has to be scaled and n is the fundamental frequency on the scale of octaves.

The next step consists of detecting the instants in which the resulting fundamental frequency changes abruptly, indicating a change of the note. The procedure to find these points is based on the difference in the frequency of two consecutives notes. The fact that the frequencies are on a linear scale implies that the difference between La_4 and $La_5$is 1. Similarly, the difference between a semitone La_4 and $La_4\#$ is $n = log_2\left(\frac{466.16\ Hz}{440\ Hz}\right) = 0.083$ octaves.

In the current implementation, the algorithm assumes that there is a change in the note when the difference between two consecutives notes is larger than 0.083 octaves. However, there is usually a transition period, which implies that the change in the note will not occur within two consecutives samples. It is important that the algorithm takes this margin into account by computing the relative difference among several samples.

This method is also useful to detect the points in which there is a rest (see Section 4.2.5), which are indicated by the presence of noise. Similarly, if the fundamental frequency has a continuous trend in some time interval, the algorithm assumes that there are no rests in this interval.

Another issue to be taken into account is the fact that, if the detected frequency is below 196 Hz -the lower bound frequency that a violin is able to play-, the algorithm assumes that the signal in that interval is noise and gives as an output a frequency of 0 octaves. The same happens if the detected frequency is above the maximum frequency that we assume that the violin is able to play: 1.4 kHz.

Figure 4.8 shows an example of the output of the frequency detector, clearly matching the changes in the notes and the rests in the song with the score. Note that the changes of notes are clearly defined, making it easy to find the exact position in which these changes occur. The rests in the song occur in the points in which the value is 0 octaves.

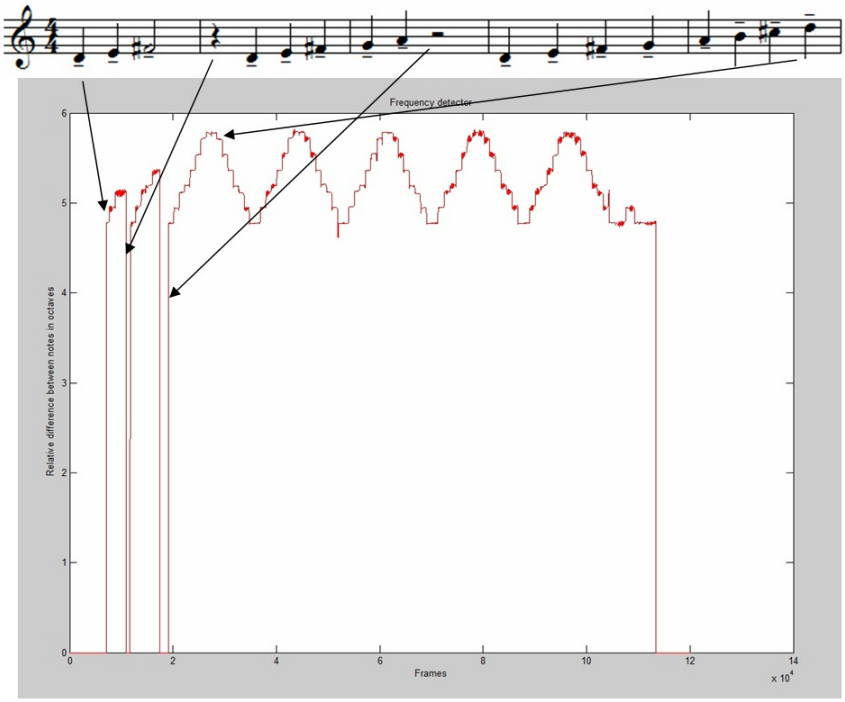

Figure 4.8: Example of the output of the fundamental frequency detector.

4.2.1.3 Bow displacement

As mentioned before, the data extracted from the performance of the musician referred to the bow is obtained using a sensor placed near the bow frog. The electromagnetic field created allows us to calculate the position of this sensor. The position of the bow in relation with the bridge of the violin is finally calculated by inferring the distance between this sensor and the one placed on the violin.

The output is a file that contains the position of the bow, calculated with a sampling frequency of 240 Hz. This information is useful to calculate the changes in the direction of the bow, which will help identifying the beginning of a beat (see Section 4.1.2).

Figure 4.9 shows an example of the bow displacement data. Note that the maximum value of the bow displacement is around 65 cm, due to the fact that the usable length of the bow is approximately 65 cm. It is worth to remark that when the plot reaches 0 cm, the part of the bow touching the string is the frog. Similarly, when it reaches 65, the string is being touched by the tip of the bow.

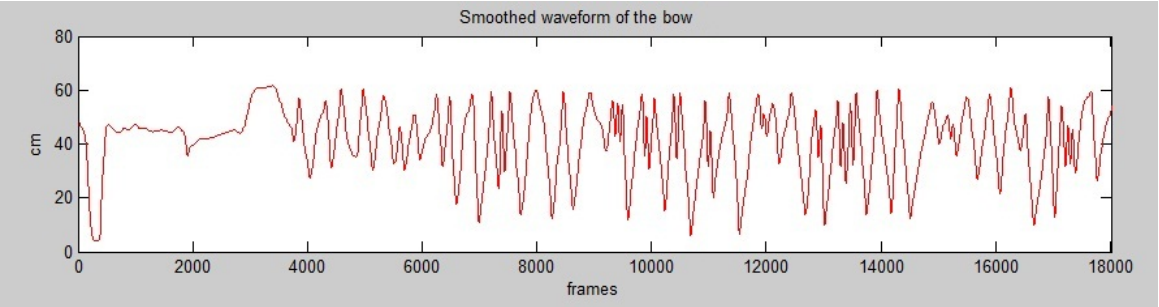

Figure 4.9: Example of bow displacement.

An algorithm is then applied to the resulting data in order to avoid potential errors in the detection of the sensors and reduce the effects of the noise (Equation 4.4). The width of the window used by the algorithm is 7 samples. Figure 4.10 illustrates an example of the bow displacement: the blue signal is the one before using the algorithm. As it is possible to see, there is some noise which might mislead the algorithm that decides the changes of the direction of the bow. After using the algorithm (red signal), the maxima and minima are clearly defined. The result is a signal with a rectangular shape, but the maxima and the minima are placed in the correct position. Then it is easy to detect the peaks of the graph, so it will be useful to detect the exact moments of the changes of the direction of the bow.

$$y[k] = \frac{\sum_{n=-3}^{3} x[k+n]}{7} \tag{4.4}$$

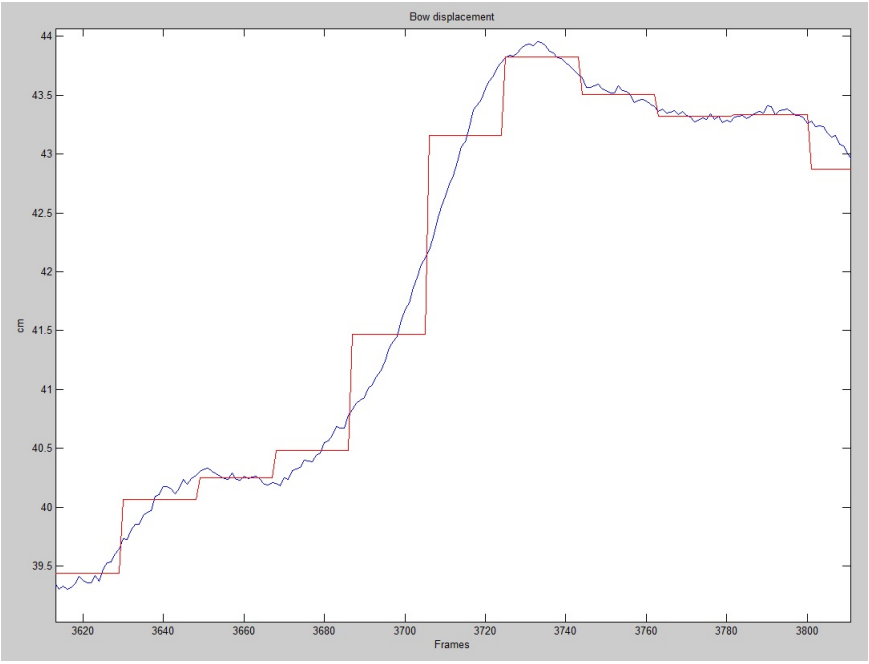

Figure 4.10: Example of the bow signal before and after the algorithm to smooth the maxima and minima.

Once the algorithm has been applied, the maxima and the minima are identified and classified depending on whether they correspond to a change in the direction from up to down o from down to up.

Figure 4.11 illustrates an example of the output of this algorithm. Red lines correspond to a change in the direction from up to down, while green lines represent a change in the direction from down to up.

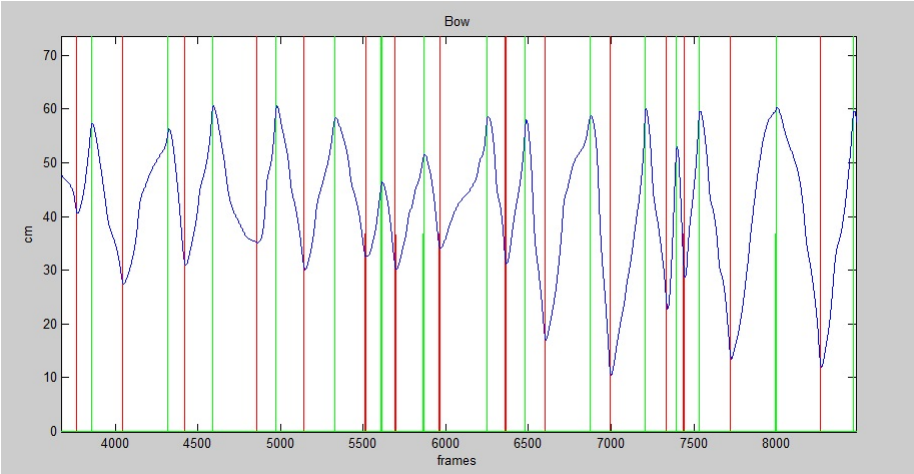

Figure 4.11: Detection of changes in the direction of the bow.

4.2.2 Beginning and end of the recording

All recordings used in the experiments always have at the beginning and at the end a short period of silence. Thus, the system must be able to detect the instant in which the song starts and finishes. This task becomes complicated by the fact that some noise is usually present in those periods of silences, making it difficult to identify which parts are actually noise and which are the beginning and the end of the song. This low intensity noise is produced by the microphone itself and the vibration of the violin.

The most intuitive method to find the beginning of the song consists of looking for the first value in the signal higher than a certain threshold, obtained from the average energy of the whole signal. However, in case that there is a high noise peak before the songs starts, the algorithm would consider it as the beginning of the song and would mislead prediction.

An alternative method consists of, firstly, computing the average energy of the whole recording. This parameter will be taken as a reference, as it is lower than the average energy of the signal without silences and higher that the average energy of the noise. The next step consists of calculating the average energy in a moving window. The point in which the moving window average energy is higher than the reference is considered the beginning of the song. Using a window comparison instead of a sample by sample comparison makes the algorithm more robust against noise peaks. Thus, this algorithm has been chosen for the current thesis.

The end of the song is similarly found by moving the window backwards, starting from the end of the audio recording.

As explained in the Section 4.2.2, the beginnings of a song are classified depending on the relative position between the first beat and the first note. Thus, the beat detector has to take it into account; if the first beat does not match with the first note (anacrusic and acephalous beginnings), the algorithm indicates it using a flag. This flag contains the ratio between the musical figure played before the first beat over to the reference musical figure of the score. As a numerical example, if the reference musical figure is a quarter note and an eighth note is played before the first beat, the value of the flag is 0.5. Note that if the first beat matches

with the first note (thetic beginning), the delay between the first beat and the first note would be 0 and the flag also would take the value 0.

Undoubtedly, the beginning and the end of the song can be considered as the beginning of a beat. Actually, the end of the song is just an end of a beat, but can be considered as a beginning of an hypothetical new beat.

Taking these considerations into account, the time instants in which the song starts and ends are calculated and incorporated into the vector of events which contains the rest of the beat candidates.

4.2.3 Theoretical tempo, estimated tempo and predicted tempo

The *theoretical tempo* is the reference tempo used in the score. The algorithm simply reads this value from the beginning of the score file in order to know the kind of figures present in the score.

The *estimated tempo* is used by the algorithm in order to predict the possible positions of the beats. There are two different methods to calculate it:

- The first one consists on taking into account the time interval between relevant events in the song, like changes of the notes, high energy peaks and changes in the direction of the bow. After calculating the histogram of the tempos, the most probable tempo is selected as the final estimated tempo. However, the main problem of this method is that the histogram will show a high peak not only in the correct tempo (instant t), but also in its multiples (instants $2t$, $4t$, etc.) and in its dividers (instants $t/2$, $t/4$, etc.), making it difficult to determine which is the valid tempo. Thus, this method only works well when the majority of the notes are quarters notes (or the reference musical figure), which corresponds to the case in which the time interval between notes is actually the real tempo.

- The second method it is only possible if the digital representation of the score is available and is the one used in the current thesis. It is calculated the total number of beats of the song from the score, taking the duration of the whole score and dividing it by the reference musical figure. Later, the estimated tempo is calculated dividing the total duration of the audio recording by the total number of beats.

Finally, the predicted tempo is the output of the beat detector, which is the real tempo followed by the violinist during the performance.

4.2.4 Score

The score file contains all the notes with their theoretical duration. It is worth to remark that the first note (or rest) starts always at 0 seconds, although the real audio recording will contain a period of silence at the beginning and at the end of the song (Section 4.2.2). The theoretical tempo is also indicated in the score (see 4.2.3).

Figure 4.12 illustrates an example of one of the scores used in the experiments. The first and second column indicate the instants at which each note starts and ends, respectively, while the third column indicates the name of the note.

The total number of beats in the song is calculated using the score file. It is important to remark that this parameter must coincide with the total number of predicted beats calculated by the algorithm. With the score information, it is possible to calculate an important parameter that is the total number of beats of the song.

The score file will be also used in the aligner algorithm (Section 4.3).

```
0.0000000 0.7500000 D4
0.7500000 1.5000000 E4
1.5000000 3.0000000 F#4
3.0000000 3.7500000 rest
3.7500000 4.5000000 D4
4.5000000 5.2500000 E4
5.2500000 6.0000000 F#4
6.0000000 6.7500000 G4
6.7500000 7.5000000 A4
7.5000000 9.0000000 rest
9.0000000 9.7500000 D4
9.7500000 10.5000000 E4
10.5000000 11.2500000 F#4
11.2500000 12.0000000 G4
12.0000000 12.7500000 A4
12.7500000 13.5000000 B4
13.5000000 14.2500000 C#5
14.2500000 15.0000000 D5
15.0000000 15.7500000 D5
15.7500000 16.5000000 C#5
16.5000000 17.2500000 B4
```

Figure 4.12: Example of a score file.

4.2.5 Division of the recording using the rests

One of the main troubles that the algorithm needs to deal with is the fact that, when the song is too large, there is a higher probability to miss some correct beats or take the wrong ones as valid, thus causing desynchronization problems.

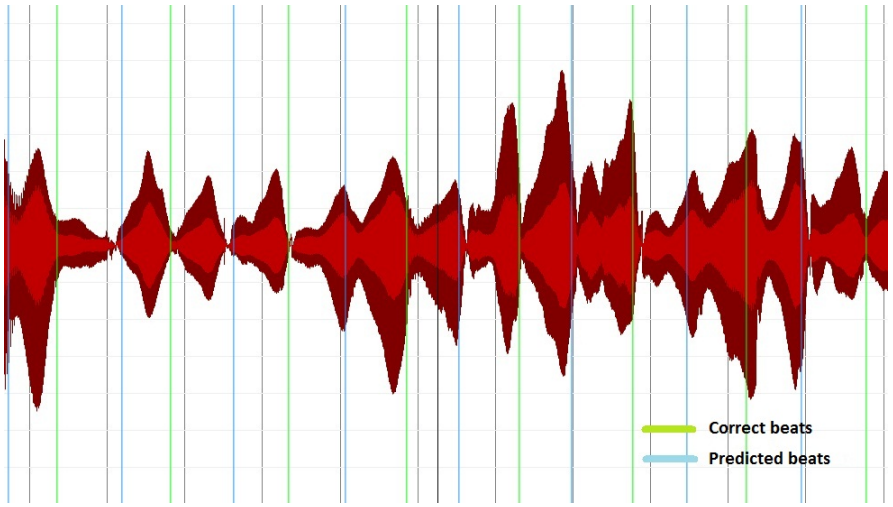

Figure 4.13: Example of desynchronization.

Figure 4.13 shows an example of desynchronization. In this case, the algorithm accepts false positives. Taking as a reference the correct beginnings of the beats, in this example, the wrong detected points are delayed 50% out of the total beat duration. This is due to the fact that, in case that there are many consecutive eighth notes, the algorithm takes the second note as the valid beginning of the beat (instead of taking the first note). This leaves the algorithm in a permanent desynchronization state, which is difficult to leave.

After several experiments, this error has been proved to be solved by dividing the audio recording into several pieces, taking the rests as the dividing points, since they are easier to find than some specific note. This shortens the duration of the song, so that the desynchronization errors are not propagated.

In order to locate the rests in the song, the algorithm searches both in the digitalized representation of the score and in the audio recording. On the one hand, the rests in the score are clearly marked with their name (*"rest"*). On the other hand, the rests in the audio recording are identified when the output of the

fundamental frequency detector is 0 octaves, which corresponds to an absence of periodicity (Section 4.2.1.2)

However, the instants at which the fundamental frequency detector detects a rest may not coincide with the instants marked in the score. One of the reasons for this error is the fact that the violinist sometimes extends some notes that are shorter in the score, changing the real tempo. Another important reason is that when the violinist stops playing and starts a short rest, the string keeps vibrating. Since the microphone detects the vibration as a real note, the frequency detector also detects certain frequency instead of the real rest.

The algorithm takes as a reference the theoretical rests in the score in order to match them with the rests in the audio recording. Several parameters are taken into account when finding the rests in the audio recording: the instant at which the recording starts, the estimated tempo of the recording, the theoretical tempo used by the score file and the theoretical instant of time when the rest should start. From these parameters, the instant at which the rest should be located is calculated using Equation 4.5. After that, the algorithm searches the rest at the calculated position, always within a certain tolerance margin. If the rest is correctly found, its position is saved and the song is divided into two parts, taking the position of the rest as the dividing point. Then, each part is treated as a separate song.

If the algorithm does not find any rest within the tolerance margin, the rest is discarded and the algorithm divides the song using the next rest.

If the algorithm finds two or more rests inside the tolerance (which is possible if the rests are very short and consecutive), the algorithm reduces the tolerance margin until it finds only one rest.

$$Rest\,pred[k] = \frac{Estimated\,tempo}{Theoretical\,tempo} \cdot Rest\,position\,in\,the\,score + Beginning\,of\,the\,recording \tag{4.5}$$

It is worth to mention that at the beginning of any rest, the string may keep vibrating even though the violinist is not playing, making it difficult to determinate when the rest starts. On the contrary, the end of the rests is usually easier to identify, since the beginning of the note after the rest is clearly well defined. Thus, the implemented algorithm looks for the end of the rests instead of their beginning.

Figure 4.14 illustrates the procedure to find the rests in the recording.

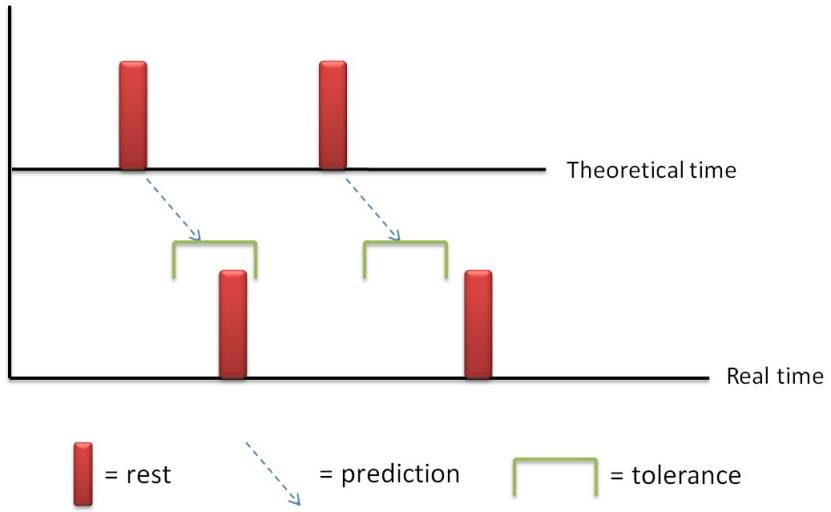

Figure 4.14: Example of the method used to find the rests.

4.2.6 Decision

Once the information is processed, the following step consists of deciding which of the beat candidates stored in the vector of events can be considered as valid onsets of a beat. To do it, several features at each time instant are weighted. A relevant instant of time is an instant that there is a change of a note, a change of the direction of the bow or a high peak of energy of the recording. For instance, if the relevant instant contains a change of the note, a change in the direction of the bow and a high peak of energy, there is a high probability that there is a beginning of a beat at that instant. On the contrary, if the instant only contains a change of a note, the probability is lower than in the previous case.

The algorithm must also take into account the cases in which two or more relevant events are separated by only few samples, although they belong to the same beat. This may occur, for instance, when a high peak is located immediately after a change in the direction of the vow. The algorithm, in these cases, must consider both events as the same beat and not as different beats.

Figure 4.15 illustrates this concept. Note that the set of relevant events for the same beat are grouped in small blocks. The blue lines correspond to changes in the direction of the bow; the green lines correspond to high energy peaks and the red lines correspond to changes in the note.

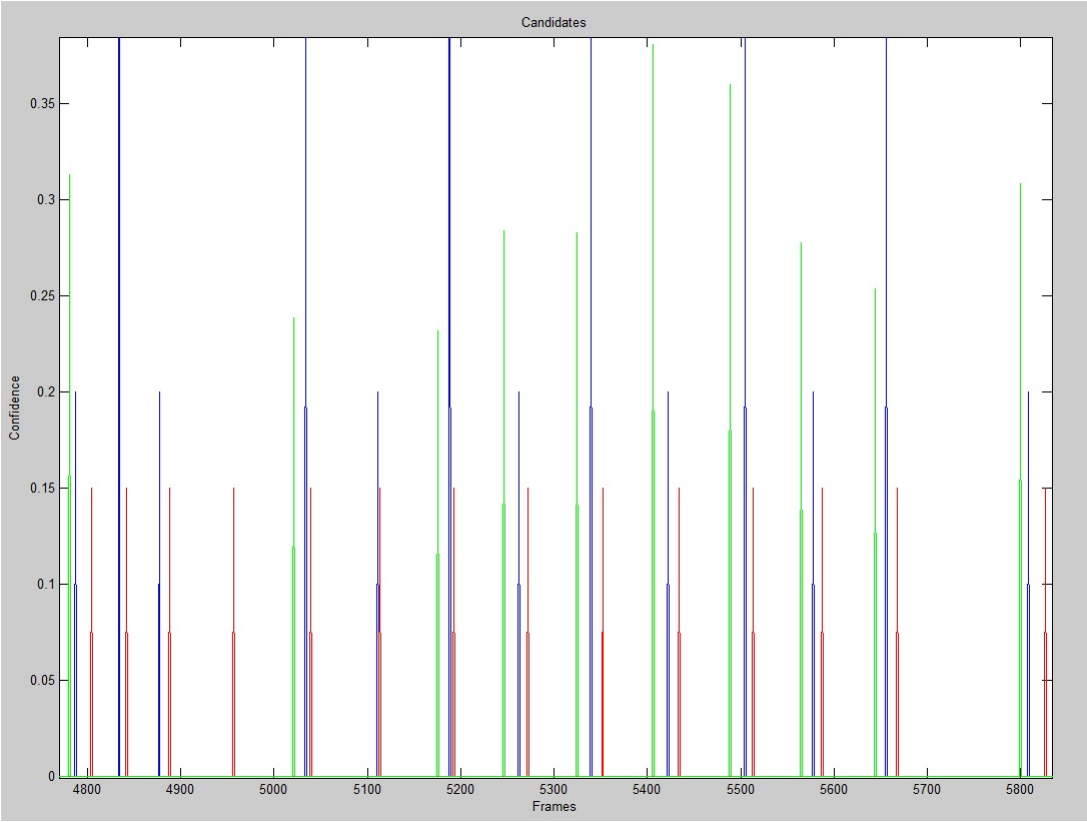

Figure 4.15: Example of different candidates for the same beat.

In order to put all the candidates together, the system uses an algorithm that finds the first beat candidate of the song and applies a window of 20 samples forward. Afterwards, it creates a unique beat candidate equal to the sum of all the candidates contained in these 20 samples. This process is repeated for each beat candidate. The length of the window has been set to 20 samples because this is the maximum resolution of

the algorithm, as mentioned in section 4.2.1.1).

As mentioned before, a certain importance or confidence has to be given to each candidate. More accurately, the confidence function used in this thesis is given by Equation 4.6. In this formula, *bow down* takes the value 1 when there is a change in the direction of the bow from up to down and *bow up* takes the value 1 in the opposite case. Thus, these variables will never take the same value at the same time. If there is a *change of a note*, this parameter takes the value 1 and the *normalized energy of the peak* parameter takes a value between 0 and 1, depending on the energy of that instant. Thus, the maximum confidence that can be applied to a candidate is 1.

The weights for each parameter have been chosen after doing around 2000 simulations, using 17 different songs in each simulation. These values were the ones that provided better results, as explained in Chapter 5.

$$Confidence = 0.45 \cdot bow\,down + 0.25 \cdot bow\,up + 0.4 \cdot change\,of\,note + 0.15 \cdot normalized\,energy\,of\,the\,peak \longrightarrow$$

$$Confidence = 0.45 \cdot bow\,down + 0.25 \cdot (1 - bow\,down) + 0.4 \cdot change\,of\,note + 0.15 \cdot normalized\,energy\,of\,the\,peak \tag{4.6}$$

Since the most important two relevant instants of time are the beginning and the end of the song, they are calculated and added to the vector of candidates, as mentioned in Section 4.2.2. The confidences given to these instants have to be high enough to indicate that there is no doubt they are the beginning of a beat. For the current implementation, they have been assigned a confidence of 100.

Figure 4.16 shows a portion of an audio recording with all the relevant instants of time. The correct positions of the beginnings of the beats are indicated with arrows. Note that there are relevant events (green lines) that do not match with the correct position of the onsets of the beats. Therefore, the algorithm has to decide which ones are correct and which ones are not. It is also important to highlight that there are some beats that do not match with any relevant event. Thus, these beats will have to be predicted from the adjacent ones, as explained later on.

Note that a song that takes around 2 minutes usually contains around 200 beats. As demonstrated in the experiments, the typical amount of beat candidates in a song with this duration is around 500. This implies that the algorithm will have to discard around 60% of beat candidates.

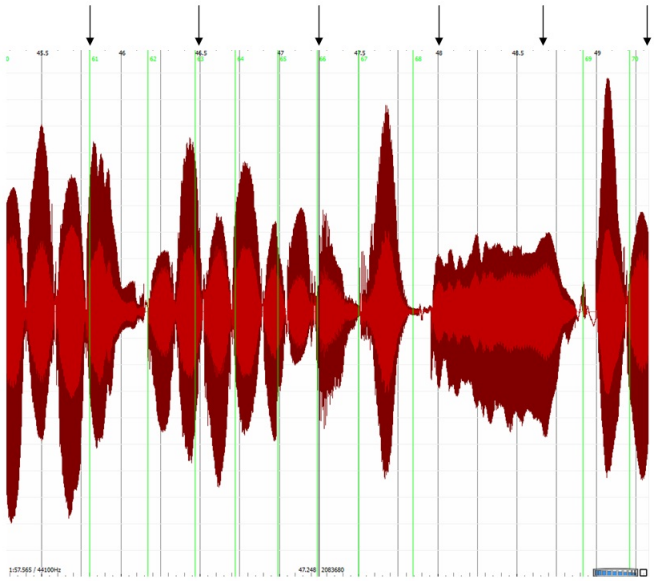

Figure 4.16: Example of a portion of a recording with the relevant instants of time.

4.2.6.1 Procedure

This section details the procedure to decide which beat candidates are valid and which are not.

The first relevant instant of time chosen is the beginning of the first beat, which has to match with the beginning of the song of the audio recording. Afterwards, the algorithm looks for the next candidate and checks if the interval of time between these two instants (the first relevant instant and the next one) is approximately a multiple of the estimated Inter-Onsets Interval, calculated from the estimated tempo. If the comparison is positive, it takes the second relevant instant as a beginning of a new beat. It is necessary to take into account that it is likely that the interval of time between two relevant instants of time is not exactly a multiple of the IOI, but it will take a value very close to it. For this reason, the algorithm has to give a certain tolerance margin to accept or deny the relevant instant.

Some considerations must be made:

- It is necessary to know the duration of the interval between two relevant and accepted instants, in terms of multiples of the IOI. Considering a multiplicity n, $n-1$ beats have to be added equidistantly between the two relevant instants. Figure 4.17 shows an example in which a beat is added between two accepted candidates.

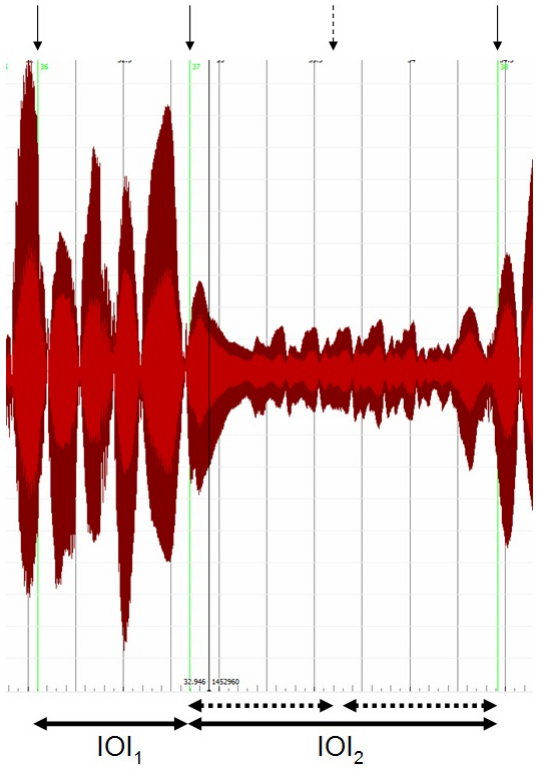

Figure 4.17: Example of an interval between candidates, multiple of the IOI.

- The tolerance margin used to accept or reject a candidate depends on the confidence given to that instant and on how high is the error between the estimated IOI and the real one. The criteria used to decide the tolerance margin is given by Equation 4.7. The maximum possible error is 0.5, which would imply that the candidate is just in the middle between the previous accepted beat and the estimated position of the next one. Note that the confidence value is multiplied by 0.3 in order to discard the beat candidates that give a variation of the tempo higher than 0.3 times the tempo because, musically, a variation of the tempo higher than 0.3 times the tempo is not usual, so all the candidates over this range are automatically discarded. I.e, only if the variation is 0.3 times the tempo and the candidate has a confidence of 1, it will be accepted as a valid beat. In that sense, it is also important to highlight that the fastest musical figure used in a song is usually the sixteenth note, which is 0.25 times the quarter note. Since the latter is often used to determine the BPM, a tolerance of 0.3 times the confidence (which in most cases is much lower than 1) avoids accepting many incorrect beats as false positives.

  Figure 4.18 shows the relative distance between the point in which the beats should start (marked with arrows) and the possible events that the algorithm has to deal with. In this case (and also in most cases), the lowest relative distance is 0.25 times the tempo, so the algorithm will probably find a candidate at around 1 IOI, 0.75 IOI and 1.25 IOI. The tolerance margin will be used to discard the false beats and accept the unique valid one. Figure 4.19 illustrates an example of the tolerance margin procedure.

$$\text{if} \quad \frac{Pos.\,candidate - Pos.\,previous\,pulse}{IOI} - \left\lfloor \frac{Pos.\,candidate - Pos.\,previous\,pulse}{IOI} \right\rfloor \leq 0.3 \cdot Confidence$$

$$\text{or} \quad abs\left(\frac{Pos.\,candidate - Pos.\,previous\,pulse}{IOI} - \left\lceil \frac{Pos.\,candidate - Pos.\,previous\,pulse}{IOI} \right\rceil \right) \leq 0.3 \cdot Confidence \tag{4.7}$$

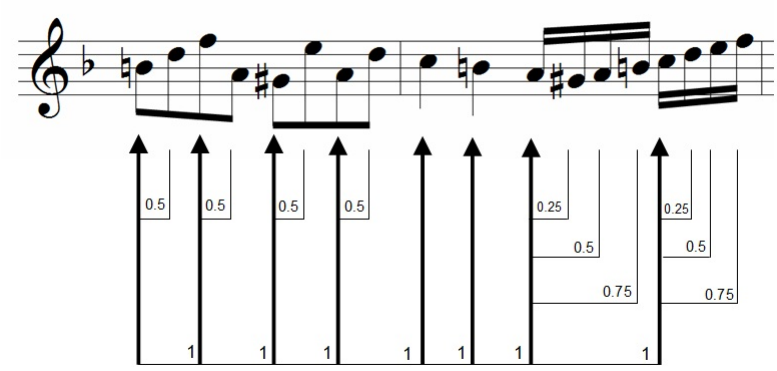

Figure 4.18: Example of relative distances between the theoretical beats and possible candidates.

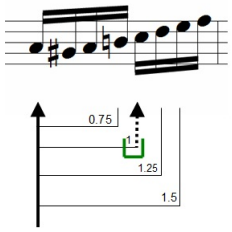

Figure 4.19: Example of the proceed of the tolerance. The first arrow points the last accepted beat and the second arrow the theoretical position of the following beat with its corresponding tolerance.

- The tolerance margin must change its size according to the distance between the last accepted beat and the following candidate. If the distance between them is a multiple of the IOI higher than one, the tolerance margin has to increase proportionately. The reason for this is that we assume that tempo changes linearly, so the tolerance has to follow the same trend.

In the cases in which the tolerance margin is too large and there are two consecutive beat candidates that might belong to the same beat, both beat candidates must be considered. In order to do this, the system uses an algorithm that divides the original path and creates as many paths as needed using a tree shape. In our case, the algorithm takes the original path (a vector with the IOIs calculated before the bifurcation is found) and creates two paths with the two possible new IOIs. Figure 4.20 shows an example of the use of the tree shape algorithm.

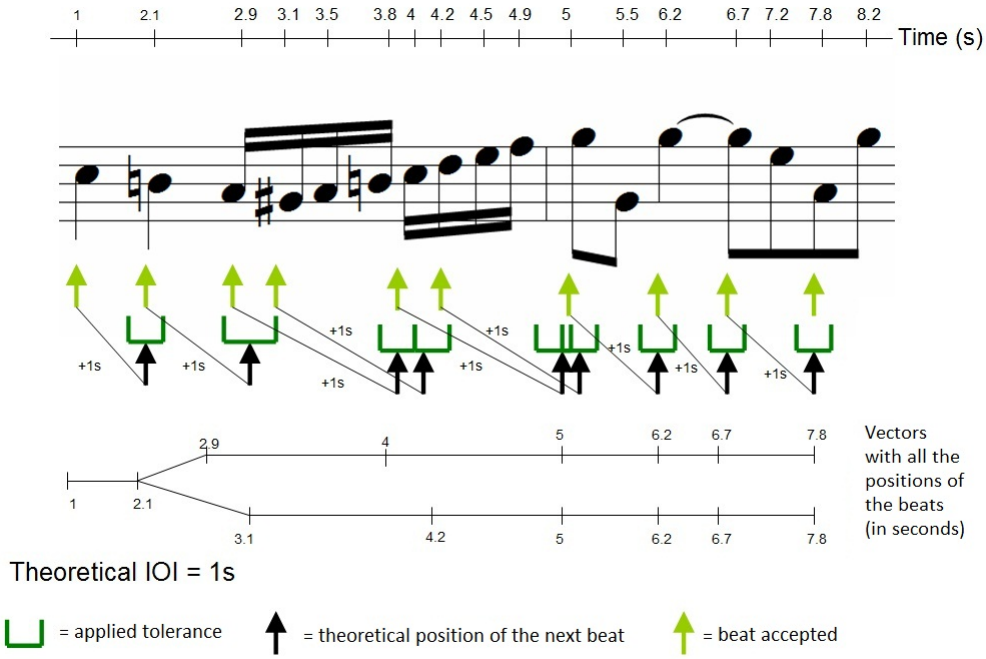

Figure 4.20: Use of the tree shape algorithm.

Moreover, in order to adjust the precision in the estimation of the *estimated tempo*, if the confidence of a new beat is equal or higher than 0.8 (note that the maximum is 1), which means that is highly probable that the election of this candidate is correct, is recalculated the estimated tempo of the song and consequently the estimated IOI. The latter is calculated by dividing the remaining time of the song by the number of remaining beats in the score. By doing this, the estimated tempo changes properly and the algorithm adapts its predictions to the variations of the tempo.

4.2.6.2 Tree shape algorithm

Once the beat prediction algorithm has finished, the system decides which path created by the tree shape algorithm is the correct one.

The criteria to decide the correct path is detailed below:

1. The number of predicted beats has to be the same as the number of beats calculated using the score. After several experiments, we can conclude that the number of predicted beats varies considerably in comparison with the ones calculated using the score because of the tolerance margin used to accept or refuse the candidates. If any of the paths has the same number of beats as the calculated using the score, the whole decision process starts again, giving a shorter tolerance margin to the decision maker. At the limit, the minimum tolerance is 0, which would imply that all the BPM would have the same value and the number of predicted beats would match with the beats calculated using the score.

2. Afterwards, if there is still more than one path, the decision is taken according to the confidence of each path. This confidence is calculated by adding all the confidences of the predicted beats. The selected path is the one with higher confidence.

4.2.7 Result

The output of the tree shape algorithm is a vector with the position in seconds of the onsets of the predicted beats. From this vector, it is easy to obtain the BPM followed by the violinist, using Equation 4.8.

$$Estimated\,BPM[k] = \frac{60\,s}{Predicted\,beat\,onset[k+1]\,s - Predicted\,beat\,onset[k]\,s} = \frac{60}{IOI[k]} \tag{4.8}$$

Figure 4.21 shows an example of the result.

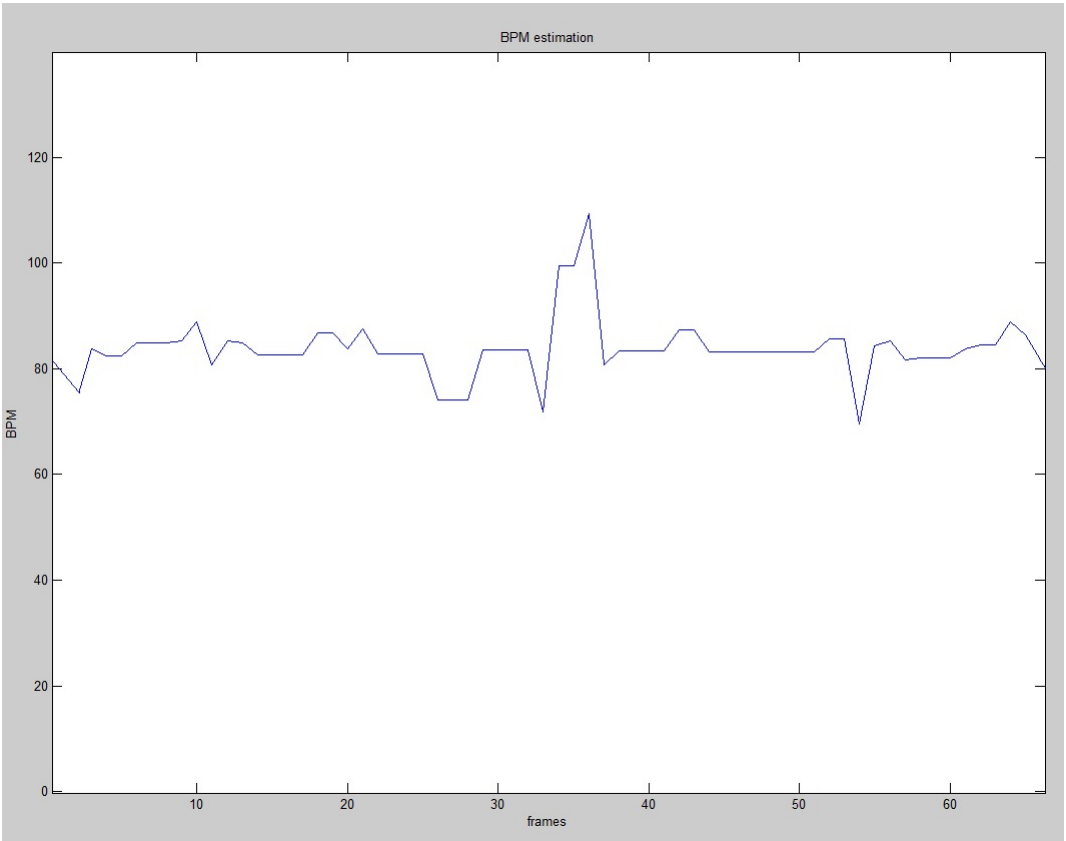

Figure 4.21: Example of the BPM followed by the violinist.

As mentioned before, obtaining the BPM followed by the violinist is one of the goals of this project. This parameter allows us to study the changes in the tempo performed by the violinist. It is important to highlight that these variations are more pronounced if the violinist plays without using a metronome, which the recordings used in this project were taken.

4.3 Matching

The other important goal of this project consists of matching the recording with the score. With the BPM estimation achieved, the next step is to create a new file with the real tempo of each note using the score file with the theoretical duration of each note. In order to do it, the theoretical duration of each note has to be divided by the theoretical BPM and multiplied by the estimated BPM (Equation 4.9). Note that the estimated BPM depends on the time because it varies continuously along the song.

$$Real\,duration(t) = \frac{Theoretical\,duration}{Theoretical\,BPM} \cdot Estimated\,BPM(t) \tag{4.9}$$

Two cases must be differentiated:

- Assuming a quarter note as a reference tempo, if the score has a quarter note that coincides with the beginning of a beat, the note will last the whole interval between beats. Thus, the quarter note will have to be compressed or stretched depending on the BPM followed at that instant of time.
- If the quarter note starts at the middle of the interval between beats, half of the note has to be modified using one BPM and the other half has to be modified using the following BPM, because the latter does not belong to the previous interval.

Using Equation 4.9, it is possible to create the file that contains the real duration of each note. Later, it can be converted into a MIDI file to be used in some music programs or in MATLAB.

4.4 Methodology for more instruments

The methodology explained above is useful when the data comes from only one violin. However, the algorithm may be modified if there are two or more violins playing together, such as a duo, trio, quartet, etc., so that the results can be improved. Before going into more detail, it is necessary to clarify that:

- A duo, trio or quartet correspond to the situation in which two, three or four violins are playing different voices of a song at the same time. This is a completely different situation from the one in which several violins are playing exactly the same score at the same time.
- In a duo, trio or quartet all the violins are sharing the same tempo and, consequently, the same beats.

One of the weakness of the algorithm for only one violin is that, if during some seconds of the song there is no instant of time in which the onset of a beat is clearly marked, the algorithm could easily mislead and accept a false beat as a valid one, causing the desynchronization of the algorithm for several seconds or even the whole song. Usually, there are some parts of the song in which the beats are clearly marked and some other parts in which they are not, especially when the music is going *off-beat*.

In the case of more than one violin playing together, at least one of the violins is usually marking the beats. Therefore, the way to improve the algorithm consists of deciding which one of the violins is marking the beats at every instant and taking its tempo as the general tempo of the song. Thus, the existing algorithm has to be modified.

In order to do it, the BPM detector is computed for each one of the violins. Then, using the confidence given to each beat, another algorithm mixes all the possible onsets of beats and decides the general tempo of the song.

The method used is similar to the beat detector for only one violin and has several steps:

- All the predicted beats belonging to all violins are grouped and treated as candidates (see Figure 4.22).
- If two consecutive candidates are separated by only a few samples and both belong to the same beat, the algorithm groups these candidates to the same beat. The algorithm is the same as the one used for one violin and it uses a rectangular window of 20 samples that adds the confidences of all the candidates.

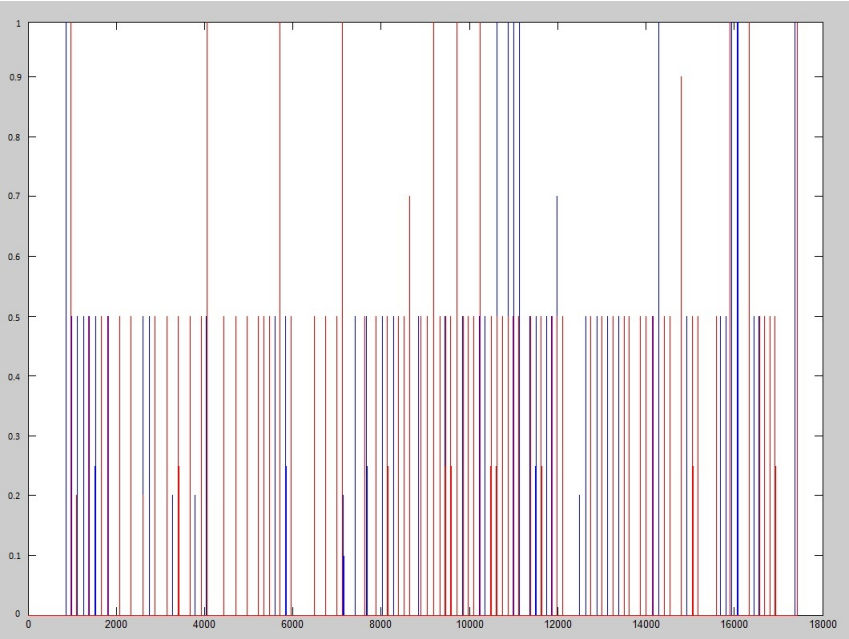

Figure 4.22: Example of different beat predictions of two different violins, grouped and weighted using the condifence of each beat. The blue beats are the ones predicted using the first voice of the song and the red ones are predicted using the second voice. Note that in some cases the beat prediction is not regular (there is a gap between two beats).

- Once the candidates have been grouped, the algorithm starts with the first candidate (which has to match with the beginning of the first beat of the song) and looks for the next candidate. If the distance between them is a multiple (or approximately, giving a tolerance margin) of the estimated IOI, it takes the candidate as a beginning of a new beat.
- If there are two candidates within the tolerance margin, the tree shape algorithm is used again and both candidates are taken into account.
- The best path created by the tree shape algorithm is chosen, using the same method than in the BPM detector. The confidence of the beats is added and the path that has the highest total confidence is chosen.
- Using the selected path, the global BPM is computed by using the same procedure as the one used with one violin.

The result is the BPM followed by the two (or more) violinists, which has to be the same one for each them. It is worth to mention that the BPM obtained when analyzing two or more violins is more accurate than in the case in which we only have one violin. Section 5.2.3 shows the results of this method and compares them with the ones obtained using only one violin.

Chapter 5

Experiments and results

This chapter contains the results of the simulations conducted to prove the correct performance of the developed system. The results show that the use of the bow displacement as an extra input improves considerably the efficiency of the BPM detector. Moreover, some other experiments have been conducted in order to demonstrate the correct performance of some parts of the algorithm, such as the detection of the rests and the prediction of the beginning and the end of the songs.

5.1 Basis of tests

The algorithms have been implemented in MATLAB code. Some of the 17 songs that we have been tested have most of the beats clearly defined and, thus, the beat prediction is very accurate. However, there are some songs in which the beat prediction becomes a challenge for the implemented algorithm because the position of the beats is not so clear.

In order to evaluate the results, two different parameters have been taken into account:

1. The accuracy in the beat prediction, which is the difference in time between the predicted onset of the beat and its real onset.

2. The BPM estimation. Although the accuracy in the prediction is not correct (in most of cases due to a desynchronization), the BPM estimation can be correct. Since the BPM is computed by calculating the difference in time between two consecutive beats, this value can be the same as the theoretical one.

To decide the correct position of the beats, the positions of the correct beats in the songs have been established manually, song by song. This has been performed using *Sonic Visualiser*; a program that allows annotating manually some instants of time in a song and saving them in a *.txt* file. Then, it is possible to compare them with the predicted beats computed in MATLAB.

The efficiency of the algorithm has been compared to one of the algorithms presented in the Music Information Retrieval Evaluation eXchange (MIREX), in the category of *Beat tracking algorithms*. The method used to compare both algorithms is explained below (see [19] for further information).

The results have been also compared with *Essentia* [20]; a set of several programming libraries and algorithms used for audio feature extraction, developed in the Music Technology Group at Universitat Pompeu Frabra by E. Aylon and N. Wack. *Essentia* is an audio analysis and music matching tool, that provides an

extensible collection of algorithms mainly used to extract features from audio files and to compare music pieces.

5.1.1 F-measure

F-measure is a generic evaluation method often used in information retrieval. For beat tracking, *F-measure* is calculated in terms of three parameters: the number of correct predictions (c), the number of false negatives or missed detections (f^-) and the number of false positives, which are the extra detections in a wrong place (f^+).

In this method, a correct detection is defined as a beat falling within a tolerance window around the annotated beat, with a tolerance margin of 70 ms.

The F-measure is calculated using two intermediate quantities: precision p (Equation 5.1), which indicates the proportion of the predicted correct beats, and recall r (Equation 5.2), which indicates the proportion of the total number of correct predicted beats.

$$p = \frac{c}{c + f^+} \tag{5.1}$$

$$r = \frac{c}{c + f^-} \tag{5.2}$$

The accuracy value of the F-measure is shown in Equation 5.3.

$$F\,(\%) = \frac{2pr}{p + r} \cdot 100\% \tag{5.3}$$

This method punishes the algorithms that predict more beats than the real amount of beats. Also, it does not take into account the error between beats and the annotations; if a beat is inside a tolerance window, the algorithm considers that it is correct although it is detected at the limit of the tolerance margin.

5.1.2 Cemgil et al.

The *Cemgil et al.* evaluation method uses a Gaussian error function that penalizes the accuracy of an estimated beat location, based on how far it is from the closest annotation. This method is similar to a tolerance window, because the further the prediction is from the annotation, the smaller the value is:

$$W(x) = exp\,(-x^2/2\sigma^2) \tag{5.4}$$

where $x = \gamma_b - a_j$, (γ_b is the predicted beat and a_j is the annotated value). The standard deviation is defined as $\sigma = 40ms$.

Then, the overall accuracy is:

$$Cemgil\,(\%) = \frac{\Sigma_j max_b W(\gamma_b - a_j)}{(B + J)/2} \cdot 100\% \tag{5.5}$$

where B is the number of the total predicted beats and J is the number of the total annotated beats.

In practice, if the predicted beats are on the off-beat (50% of delay respect to the correct position), the result is $Cemgil \sim 0$.

Note that B can be higher than J because some beat trackers do not use the score and do not take into account the total number of beats of the song, resulting more predicted beats than the theoretical ones.

5.1.3 PScore

PScore is a measure of beat tracking performance defined by McKinney et al. [21]. Beat accuracy is determined by taking the predicted beat and the closest annotation, giving a tolerance of a 20% of the Inter-Onsets annotated Interval (Equation 5.6).

$$T(n) = \begin{cases} 1 & \gamma_b = a_j \pm 20\% \cdot \Delta_j \\ 0 & otherwise \end{cases} \tag{5.6}$$

The method relies on the sum of the correct predicted beats divided by the total number of predicted beats (Equation 5.7).

To remove a possible initial uncertain period of beat predictions, this method removes all the events occurring in the first five seconds.

$$PScore\,(\%) = \frac{\Sigma_n T(n)}{max(B, J)} \cdot 100\% \tag{5.7}$$

where B is the total number of predicted pulses and J the total number of annotated pulses.

This method does not take into account the accuracy in the prediction; if the beat is within the tolerance margin, it is considered as a good prediction. However, it takes into account the IOI in order to define the tolerance margin. It also penalizes the excess of beat predictions.

5.1.4 Continuity-based evaluation

Another evaluation method uses the concept of continuity in the beat prediction, which is based on the idea of consecutive well predicted beats. It uses a tolerance window of ±17.5% of the IOI. A good prediction is defined as the case in which the predicted beat γ_b falls within this tolerance window of the annotated beat a_j and the previous beat γ_{b-1} is also within the tolerance window of its correspondent annotation a_{j-1}. A further condition is taken into account and requires consistency between Inter-Onsets annotated Interval Δ_j and Inter-Onsets predicted Interval Δ_b. Equations are shown in 5.8.

$$a_j - 0.175\Delta_j < \gamma_b < a_j + 0.175\Delta_j$$

$$a_{j-1} - 0.175\Delta_{j-1} < \gamma_{b-1} < a_{j-1} + 0.175\Delta_{j-1}$$

$$(1 - 0.175)\Delta_j < \Delta_b < (1 + 0.175)\Delta_j \tag{5.8}$$

Then, comparing each beat γ_b to each annotation a_b, the algorithm finds the number of consecutive beats that satisfy all the conditions at the same time. Thus, the song can be divided in m different segments Γ_m.

The ratio used to determine the maximum continuity is the longest continuously correct segment divided by the total number of annotated pulses (J) (Equation 5.9).

$$CML_e\,(\%) = \frac{max\,(\Gamma m)}{J} \cdot 100\% \tag{5.9}$$

This ratio only reflects information about the longest segment of the correct beat tracking. Therefore, if a single bad beat prediction occurs in the middle of a good segment, the ratio would fall to the 50%.

A more relaxed ratio, detailed in 5.10, is used in order to avoid this problem. This method adds all the segments that have some beats satisfying the conditions mentioned above (5.8).

$$CML_t\,(\%) = \frac{\Sigma_{m=1}^{M}\,\Gamma m}{J}\cdot 100\% \tag{5.10}$$

5.2 Simulations

5.2.1 Results

In this section the results of the simulations are shown. First, the results are indicated for each analyzed song and are compared with the ones obtained using *Essentia*. The methods used to evaluate the algorithms have been the ones described above.

Note that the format of the name of the song is the name of the compositor followed by the number of the song and the number of the voice of the song. See that in all cases except one, all the songs have two different voices, although they are treated as individual songs.

Song	**F-Measure (%)**		**Cemgil et al. (%)**		**PScore (%)**	
	Our algorithm	**Essentia**	**Our algorithm**	**Essentia**	**Our algorithm**	**Essentia**
Berio 11 i0	72.6	46.7	54.6	33.7	68.3	42.3
Berio 11 i1	70.5	60.5	52.5	41.0	58.7	47.3
Berio 12 i0	55.3	33.0	46.7	33.2	66.5	43.6
Berio 12 i1	84.1	29.6	67.7	24.9	93.4	44.4
Berio 17 i0	54.0	14.9	42.2	15.0	74.2	38.9
Berio 17 i1	78.8	53.1	62.2	38.4	100	81.6
Berio 19 i0	78.4	40.3	51.3	18.4	67.0	20.0
Berio 19 i1	98.1	71.0	80.1	51.3	99.8	71.0
Berio 24 i0	57.0	26.9	43.8	18.3	87.6	46.8
Berio 24 i1	62.4	23.9	45.6	10.4	90.7	16.8
Mozart 1 i0	64.7	29.0	54.3	30.1	72.7	43.5
Mozart 1 i1	90.4	31.9	71.0	31.8	98.1	42.5
Mozart 2 i0	86.2	60.0	66.6	46.2	81.4	56.6
Bach 1 i0	54.6	34.7	44.6	34.4	72.8	44.7
Bach 1 i1	50.3	30.3	40.8	27.7	98.8	44.4
Bach 2 i0	57.4	26.7	42.6	18.7	82.5	51.9
Bach 2 i1	55.1	21.1	42.8	20.3	83.3	40.4
Average	**68.8**	**37.3**	**53.5**	**29.1**	**82.1**	**45.7**

Song	CMLc (%)		CMLt (%)	
	Our algorithm	Essentia	Our algorithm	Essentia
Berio 11 i0	8.2	10.0	62.8	41.6
Berio 11 i1	17.9	12.5	58.9	5.3
Berio 12 i0	22.6	0	67.8	0
Berio 12 i1	57.1	2.5	98.9	2.5
Berio 17 i0	24.4	0	87.9	0
Berio 17 i1	72.3	29.1	99.8	97.1
Berio 19 i0	27.9	0	86.0	0
Berio 19 i1	31.9	22.4	99.8	95.4
Berio 24 i0	29.6	8.0	97.8	45.1
Berio 24 i1	36.7	0	99.8	0
Mozart 1 i0	20.6	0	94.6	0
Mozart 1 i1	44.4	0	97.9	0
Mozart 2 i0	11.0	9.3	87.7	80
Bach 1 i0	41.9	0	97.8	0
Bach 1 i1	73.8	0	86.2	0
Bach 2 i0	10.5	5.7	80.5	61
Bach 2 i1	30.3	0	98.7	0
AVERAGE	**33.0**	**5.9**	**88.4**	**25.2**

Table 5.1: Comparison between the implemented algorithm and *Essentia*, detailing all the songs used.

Figure 5.1 shows a graphical comparison in which the red lines represent the ranges of the results of our algorithm and the blue lines represent the results of *Essentia*. Note that the dispersion of the results is high due to the diversity of songs used. In all cases, the average results (marked with a circle in the graphic) of our algorithm are higher than the average results of *Essentia*.

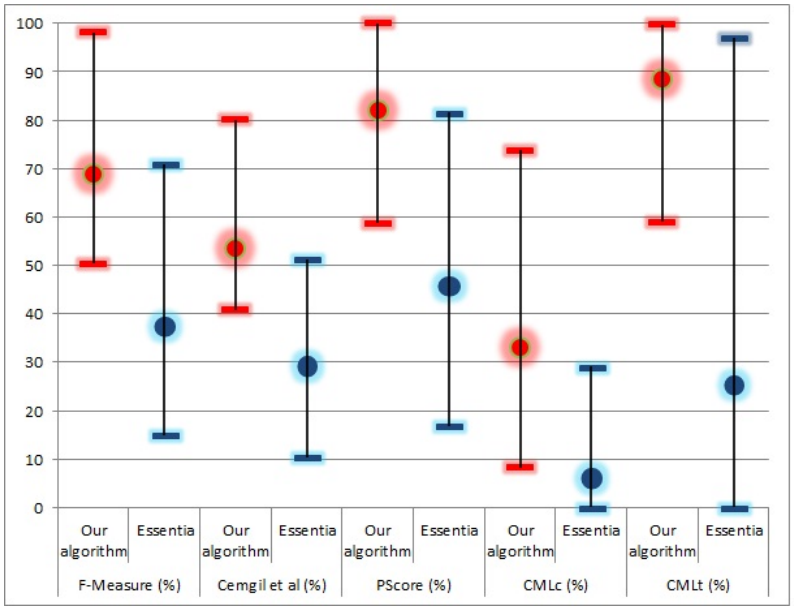

Figure 5.1: Graphical comparison of the results of our algorithm and *Essentia*. The axis y represents the percentage of hits depending on the method used. The axis x has all the evaluation methods.

Table 5.2 compares our algorithm with the algorithms presented in the MIREX 2010. The results are also compared with the ones obtained using *Essentia*.

Algorithm	F-Measure (%)	Cemgil et al. (%)	PScore (%)	CMLc (%)	CMLt (%)
BES3	47.3	38.2	45.9	4.1	26.1
BES4	58.7	51.8	57.9	4.8	32.4
GP3	47.0	35.7	48.7	3.0	21.2
GP4	48.2	36.7	50.0	3.1	23.4
GP5	40.1	29.9	42.8	2.2	16.9
GP6	40.9	30.6	43.8	2.4	18.1
LGG1	29.6	21.3	34.6	1.8	10.5
LGG2	41.4	30.4	43.5	2.7	17.7
MRVCC1	49.1	41.6	51.0	4.2	28.1
MRVCC2	49.2	39.5	51.2	4.0	28.4
NW1	27.5	19.8	31.3	1.6	9.6
TL2	68.4	40.4	72.2	6.1	50.8
ZTC1	1.1	0.9	0.9	0.1	0.1
Essentia	37.3	29.1	45.7	5.9	25.2
Our algorithm	**68.8**	**53.5**	**82.1**	**33.0**	**88.4**

Table 5.2: Comparison of results.

Figure 5.2 also illustrates a graphical comparison of the results of all the algorithms.

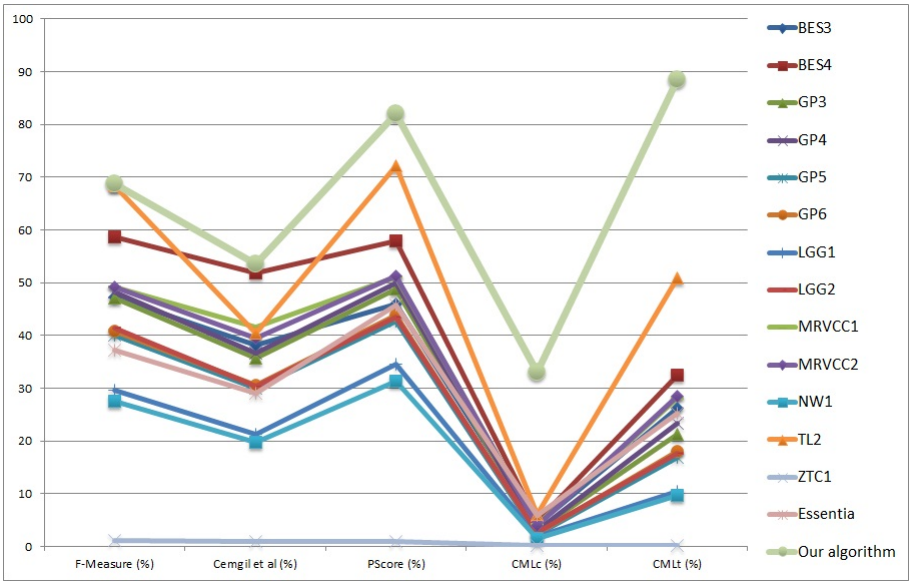

Figure 5.2: Graphical comparison of all the algorithms submitted in the MIREX 2010, *Essentia* and our algorithm.

Note that the results obtained using our algorithm are better compared with the ones obtained by the algorithms submitted to MIREX. This improvement is due to the use of multimodal information such as the bow displacement and the score.

5.2.2 Simulations without the bow displacement input

In this section, the results of our algorithm without using data of the bow displacement are shown and compared with the results using this data. To do that, the weights used in the algorithm have been recalculated. The previous weights were:

- Bow down = 0.45
- Bow up = 0.25
- Change of note = 0.4
- Peak of the energy of the signal = 0.15

After conducting around 2000 experiments, the weights that give better results without using the bow displacement data are:

- Bow down = 0
- Bow up = 0
- Change of note = 0.8
- Peak of the energy of the signal = 0.2

The results are summarized in Table 5.3 and graphically illustrated in Figure 5.3.

Song	**F-Measure (%)**		**Cemgil et al. (%)**		**PScore (%)**	
	Without bow displacement	**With bow displacement**	**Without bow displacement**	**With bow displacement**	**Without bow displacement**	**With bow displacement**
Berio 11 i0	63.4	72.6	42.9	54.6	52.4	68.3
Berio 11 i1	47.5	70.5	36.1	52.5	42.9	58.7
Berio 12 i0	25.0	55.3	20.0	46.7	60.3	66.5
Berio 12 i1	46.6	84.1	31.8	67.7	72.2	93.4
Berio 17 i0	14.3	54.0	11.3	42.2	33.4	74.2
Berio 17 i1	53.2	78.8	41.6	62.2	88.1	100
Berio 19 i0	33.6	78.4	24.5	51.3	30.4	67.0
Berio 19 i1	38.6	98.1	25.82	80.1	34.1	99.8
Berio 24 i0	35.1	57.0	21.8	43.8	59.6	87.6
Berio 24 i1	55.2	62.4	38.2	45.6	77.0	90.7
Mozart 1 i0	28.3	64.7	23.4	54.3	39.5	72.7
Mozart 1 i1	32.8	90.4	24.2	71.0	58.1	98.1
Mozart 2 i0	63.5	86.2	44.4	66.6	57.8	81.4
Bach 1 i0	21.8	54.6	15.8	44.6	41.9	72.8
Bach 1 i1	46.3	50.3	34.2	40.8	93.5	98.8
Bach 2 i0	26.5	57.4	20.0	42.6	52.1	82.5
Bach 2 i1	37.8	55.1	25.9	42.8	58.7	83.3
Average	**39.4**	**68.8**	**28.3**	**53.5**	**56.0**	**82.1**

Song	CMLc (%)		CMLt (%)	
	Without bow displacement	**With bow displacement**	**Without bow displacement**	**With bow displacement**
Berio 11 i0	13.3	8.2	57.8	62.8
Berio 11 i1	17.8	17.9	42.1	58.9
Berio 12 i0	35.5	22.6	74.6	67.8
Berio 12 i1	21.5	57.1	82.2	98.9
Berio 17 i0	7.3	24.4	26.8	87.9
Berio 17 i1	34.1	72.3	97.8	99.8
Berio 19 i0	25.8	27.9	36.8	86.0
Berio 19 i1	8.5	31.9	26.1	99.8
Berio 24 i0	10.6	29.6	69.5	97.8
Berio 24 i1	17.2	36.7	98.8	99.8
Mozart 1 i0	19.1	20.6	54.0	94.6
Mozart 1 i1	25.4	44.4	76.2	97.9
Mozart 2 i0	8.0	11.0	59.7	87.7
Bach 1 i0	17.3	41.9	46.9	97.8
Bach 1 i1	70.8	73.8	98.9	86.2
Bach 2 i0	8.9	10.5	68.1	80.5
Bach 2 i1	17.8	30.3	84.3	98.7
AVERAGE	**21.1**	**33.0**	**64.7**	**88.4**

Table 5.3: Comparison between the use of the bow displacement and without it.

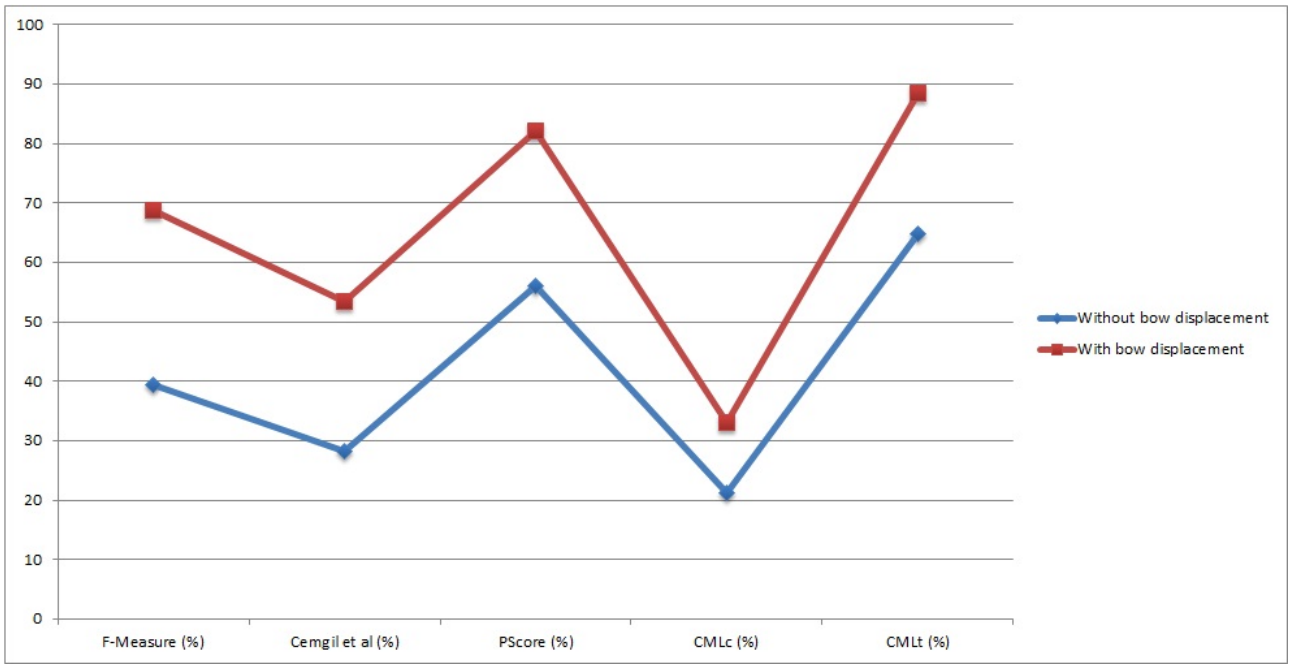

Figure 5.3: Graphical comparison of the results with the use of the bow displacement and without it.

The results obtained using bow displacement data are better than the ones without using this data. The improvement achieved is around 25% (see Figure 5.3). Without the bow displacement, the results are similar to those achieved by the algorithms presented in the MIREX. Thus, we can conclude that the use of gesture data greatly contributes in the efficiency of the algorithm.

5.2.3 Simulations with two violins

Some additional simulations have been conducted in order to analyze if the use of the algorithm for two violins improves the results over the ones obtained with only one violin. The comparison is shown in Table 5.4. Note that in this case, the songs do not have two voices because these are used together. To compare the results with the algorithm that uses only one voice each time, the average value of the results obtained from the two voices analyzed individually has been calculated.

Song	**F-Measure (%)**		**Cemgil et al. (%)**		**PScore (%)**	
	Duet BPM algorithm	**Individual BPM algorithm**	**Duet BPM algorithm**	**Individual BPM algorithm**	**Duet BPM algorithm**	**Individual BPM algorithm**
Berio 11	74.2	71.6	56.1	53.6	66.1	63.5
Berio 12	70.0	69.7	57.5	57.2	80.2	80.0
Berio 17	78.8	664	62.2	52.2	88.1	87.1
Berio 19	98.0	88.3	80.1	65.7	99.8	83.4
Berio 24	60.0	59.7	45.0	44.7	90.0	89.2
Mozart 1	91.4	77.6	64.2	62.7	87.1	85.4
Bach 1	53.1	52.5	42.9	42.7	86.3	85.8
Bach 2	58.3	56.3	44.9	42.7	85.3	82.9
Average	**73.0**	**67.7**	**56.6**	**52.7**	**85.4**	**82.2**

Song	**CMLc (%)**		**CMLt (%)**	
	Duet BPM algorithm	**Individual BPM algorithm**	**Duet BPM algorithm**	**Individual BPM algorithm**
Berio 11	16.0	13.1	63.1	60.9
Berio 12	40.2	39.9	82.5	83.4
Berio 17	72.3	48.4	97.8	93.9
Berio 19	31.9	29.9	99.8	92.9
Berio 24	33.6	33.2	99.0	98.8
Mozart 1	33.6	32.5	97.0	96.3
Bach 1	58.3	57.9	92.4	92.0
Bach 2	23.5	20.4	92.1	89.6
Average	**38.7**	**34.4**	**90.5**	**88.5**

Table 5.4: Comparison between the results obtained using an algorithm that uses two voices of a song at the same time with an algorithm that only uses one song each time.

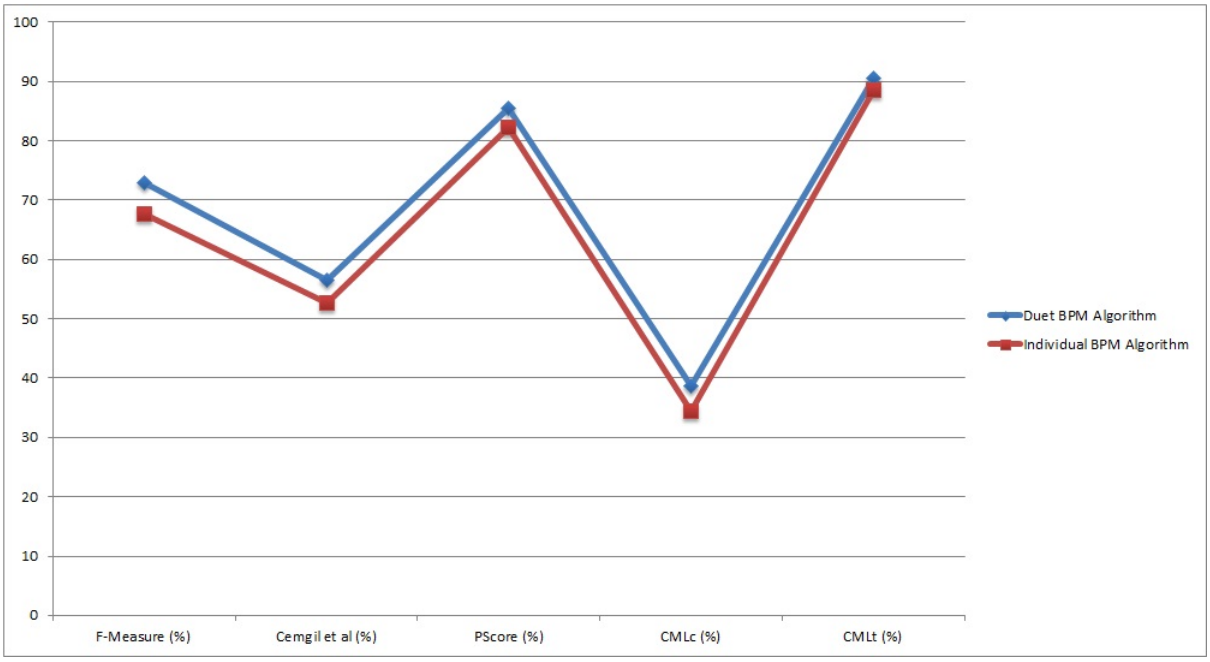

Figure 5.4: Graphical comparison of the results of the Duet BPM algorithm and the Individual BPM Algorithm.

The improvement of the results is around 4% (see Figure 5.4), depending on the method used to evaluate the algorithms. There is a big improvement in the songs in which only one of the voices has the beats clearly defined, which is the case of *Mozart 1*. The results are also improved in the case in which the voices of the song have the beats defined alternately, in the sense that while one of the voices is marking the beats, the other one is not, and then, after some seconds the positions are switched. Finally, the bests beat predictions of both voices is used.

However, in other cases the improvement is insignificant because both voices have the same relevance in the beat prediction.

It is worth to mention that the algorithm could not be tested with more than two violins because we did not have recordings of more than two violins playing together.

5.2.4 Other simulations

Some other simulations have been conducted in order to calculate the efficiency of the algorithm. Specifically, some simulations have been performed in order to estimate the effectiveness of some parts of the algorithm, such as the automatic module that estimates the beginning and the end of the songs and the module that finds the rests in the recording.

5.2.4.1 Beginning and end of the song

Tables 5.5 and 5.6 show the results of the predictions of the beginnings and the ends of the songs used in the simulations. The estimations performed by the algorithm have been compared with manual annotations. The results show in most cases that the estimation of the beginning is some miliseconds delayed from the

real beginning. This is due to the fact that the prediction is estimated by computing the average power of the samples within a window and the result is compared with a threshold related with the average power of the whole recording. The beginning of the first note usually has low intensity, so the average power of the window is lower than the average value of the song. Some miliseconds later, the first note has enough power and the average power within the window is higher than the threshold, thus the beginning is estimated at that point instead of the correct one.

The process is the same with the ending prediction. However, since the window is moving backwards, the end of the song is some miliseconds before the real end due to the same reason.

Lowering the threshold value is an option to obtain precise results. However, there might appear some problems if there is any noise peak before the beginning of the song because the algorithm might detect it as the real beginning of the song.

Song	**Estimated beginning (s)**	**Beginning (s)**	**Relative error (s)**	**Average IOI (s)**	**Error about the corresponding IOI (%)**
Berio 11 i0	4.15	4.05	0.10	0.40	25.0
Berio 11 i1	7.05	7.01	0.04	0.40	10.0
Berio 12 i0	4.25	4.22	0.03	0.45	6.7
Berio 12 i1	7.90	7.75	0.15	0.45	33.3
Berio 17 i0	14.30	14.12	0.18	0.65	27.7
Berio 17 i1	4.80	4.73	0.07	0.65	10.7
Berio 19 i0	6.02	6.15	0.13	0.35	37.1
Berio 19 i1	5.50	5.41	0.09	0.35	25.7
Berio 24 i0	5.85	5.77	0.08	0.67	11.9
Berio 24 i1	12.15	12.10	0.05	0.67	7.5
Mozart 1 i0	4.23	4.18	0.05	0.65	7.7
Mozart 1 i1	4.80	4.63	0.17	0.65	26.1
Mozart 2 i0	5.76	5.65	0.11	0.80	13.7
Bach 1 i0	5.50	5.35	0.15	0.70	21.4
Bach 1 i1	17.26	17.12	0.14	0.71	20.0
Bach 2 i0	6.77	6.51	0.26	0.80	32.5
Bach 2 i1	25.40	25.15	0.25	0.80	31.3
AVERAGE	-	-	**0.12**	-	**20.5**

Table 5.5: Beginning estimation.

Song	**Estimated end (s)**	**End (s)**	**Relative error (s)**	**Average IOI (s)**	**Error about the corresponding IOI (%)**
Berio 11 i0	41.15	41.23	0.08	0.40	20.0
Berio 11 i1	41.15	41.18	0.03	0.40	7.5
Berio 12 i0	40.90	41.22	0.32	0.45	71.1
Berio 12 i1	40.90	41.12	0.22	0.45	48.9
Berio 17 i0	80.53	80.61	0.08	0.65	12.3
Berio 17 i1	80.53	80.70	0.17	0.65	26.2
Berio 19 i0	36.75	36.82	0.07	0.35	20.0
Berio 19 i1	36.75	36.84	0.09	0.35	25.7
Berio 24 i0	86.22	86.35	0.13	0.67	19.4
Berio 24 i1	86.22	86.39	0.17	0.67	25.4
Mozart 1 i0	79.45	79.55	0.10	0.65	15.4
Mozart 1 i1	79.45	79.61	0.16	0.65	24.6
Mozart 2 i0	104.88	104.98	0.10	0.80	12.5
Bach 1 i0	65.25	65.49	0.24	0.70	34.3
Bach 1 i1	65.25	65.53	0.28	0.71	40.0
Bach 2 i0	273.50	273.72	0.22	0.80	27.5
Bach 2 i1	273.50	273.68	0.18	0.80	22.5
AVERAGE	-	-	**0.16**	-	**26.7**

Table 5.6: End estimation.

5.2.4.2 Rests

Table 5.7 shows the detections of the rests. The rest detection is performed taking into account the position of the rest in the score and the difference between the theoretical tempo and the actual one. If the rest is found within a given tolerance margin, the position of the rest is stored and the song divided in two different parts. The main problem is that if the tolerance margin is too large and there are two different short rests, they both may fit inside the tolerance margin. Also, if the real tempo is very different from the theoretical one, the tolerance window can contain only one rest but a wrong one and take it as correct. For these reasons, the given tolerance margin can not be too large and, consequently, lots of predictions do not find any rest within the tolerance margin.

Song	**Number of detected rests**	**Number of rests in the score**	**Relative error**	**Error (%)**
Berio 11 i0	1	2	1	50
Berio 11 i1	0	0	-	-
Berio 12 i0	0	0	-	-
Berio 12 i1	1	1	0	0
Berio 17 i0	2	4	2	50
Berio 17 i1	2	2	0	0
Berio 19 i0	0	0	-	-
Berio 19 i1	0	0	-	-
Berio 24 i0	0	1	1	100
Berio 24 i1	1	3	2	67
Mozart 1 i0	7	21	14	66
Mozart 1 i1	7	25	18	72
Mozart 2 i0	23	23	0	0
Bach 1 i0	2	2	0	0
Bach 1 i1	1	1	0	0
Bach 2 i0	4	10	6	60
Bach 2 i1	4	8	4	50
AVERAGE	**55**	**103**	**48**	**46**

Table 5.7: Rests estimation.

Chapter 6

Conclusions and future work

Taking into account the experimental results, some conclusions may be drawn from the designed system.

The use of multimodal data, such as the bow displacement, is a useful resource in order to improve the already existing methods that only use the recording as an input.

Around 2000 simulations have been performed in order to find the bests weights for the four variables that the algorithm uses to predict the beats (bow down, bow up, change of a note and peak of the signal). The weight of the *bow down* variable is higher (almost the double) than the *bow up* one. This is because as supposed, changing the direction of the bow from up to down, the impulse given by the bow is more intense than doing it in the other direction, and normally it is used at the beginning of the beats.

Otherwise, before doing the simulations, it was expected that the energy of the signal and, consequently the *peaks* of this, was an important parameter to take into account, but after doing the simulations the importance attached to this parameter has been lower than expected. The weight used has been 0.15 and was expected that it was similar to the weight attached to the *bow down* parameter (around 0.45), because we thought that both could have the same relevance.

The parameter *change of note* has more relevance than expected. Initially, it was introduced in the decision module of the algorithm just to add some extra information to predict the beats, without assigning much importance to it. However, it has proved to be almost as important as the *bow down* parameter.

The good performance of the algorithm is only if the changes of tempo of the song are not drastic. The algorithm uses an *average tempo* to predict when the beats should start. If the variation of tempo is oversized, the computed average tempo would not be reliable and the predictions would not be correct.

The algorithm has been developed for the purpose of using it after doing the whole recording of the song, so it can not work in real time. Future work may be addressed to change the algorithm in order to be able to use it in real time. This involves several modifications:

- The algorithm should be taking data continuously from the microphone and the sensors.
- A buffer should be used to store samples in order to calculate the autocorrelation function of the frequency detector, because it can not be computed sample by sample.
- The average tempo could not be estimated using the same method, because in real time the duration of the whole song is unknown.

Some improvements, both in a real time algorithm and in the current one, can be performed:

- Implementation of a module that counts the notes, using the spectrum of the audio recording, by estimating the number of the note that is playing the violinist and deducing the note that he/she should be playing.
- Implementation of a module that estimates the note using its fundamental frequency. This is a difficult challenge because the violinist can miss the tune, miss a note or add an extra note wrongly. But this module could contribute in some part to the decision of the beat predictions.
- Implementation of a module that uses the force applied by the bow to the string in order to achieve a better estimation of the intensity of the notes.

Bibliography

[1] E. Maestre. *Modeling instrumental gestures: An analysis/synthesis framework for violing bowing.* PhD thesis, Universitat Pompeu Fabra and Stanford University, 2009.

[2] A. Bilmes. Timing is of the essence: Perceptual and computational techniques for representing, learning, and reproducing expressive timing in percussive rhythm. Master's thesis, Massachussets Institute of Technology, 1993.

[3] A. Friberg and J. Sundström. *Swing ratios and ensemble timing in jazz performances: Evidence for a common rhythmic pattern.* 2002.

[4] S. Dixon. Automatic extraction of tempo and beat from expressive performances. *Journal of New Music Research*, pages 39–58, 2001.

[5] Y. Wang and M. Vilermo. A compressed domain beat detector using mp3 audio bitstreams. 2001.

[6] A. Schloss. *On the automatic transcription of percussive music - From acoustic signal to high-level analysis.* PhD thesis, Stanford University, 1985.

[7] J. Brown. Determination of the meter of musical scores by autocorrelation. *Journal of the Acoustical Society of America*, 1993.

[8] D. Rosental. *Machine rhythm: Computer emulation of human ryhthm perception.* Thesis / dissertation, Massachusets Institute of Technology, 1992.

[9] Desain P. Cemgil, A. and B. Kappen. Rhythm quantization for transcription. *Computer Music Journal*, 2000.

[10] C. Raphael. A hybrid graphical model for rhythmic parsing. *Artificial Intelligence*, 2002.

[11] C. Longuet-Higgins and C. Lee. Perception of musical ryhthms. *Perception*, 1982.

[12] R. Dannenberg and B. Mont-Reynaud. Following an improvisation in real-time. *International Computer Music Conference*, 1987.

[13] P. Allen and R. Dannenberg. Tracking musical beats in real time. *International Computer Music Conference.*

[14] Eck D. Gasser, M. and R. Port. Meter as a mechanism: a neural network model that learns metrical patterns. *Connection Science*, 1999.

[15] Cremer M. Uhle C. Herre, J. and J. Rohden. Proposal for a core experiment on audiotempo. 2002.

[16] F. Gouyon and P. Herrera. Determination of the meter of musical signals: Seeking recurrences in descriptor of beat segment descriptors. *Audio Engineering Society*, 2003.

[17] R. Parncutt. A perceptual model of pulse salience and metrical accent in musical rhythms. *Music Perception*, 1994.

[18] Alain de Cheveigne. Yin, a fundamental frequency estimator for speech and music.

[19] Degara N. Davies, M. and M. Plumbley. Evaluation methods for musical audio beat tracking algorithms. Queen Mary, University of London.

[20] P. Grosche and M. Müller. A mid-level representation for capturing dominant tempo and pulse information in music recording.

[21] Moelants D. Davies M. E. P. McKinney, M.F. and A. Klapuri. Evaluation of audio beat tracking and music tempo extraction algorithms. *Journal of New Music Research*, 36:1–16.