WiiCon: *Acceleration Based Real-Time Conducting Gesture Recognition for Personal Orchestra*

Diploma Thesis at the
Media Computing Group
Prof. Dr. Jan Borchers
Computer Science Department
RWTH Aachen University

by
Anna Koster

Thesis advisor:
Prof. Dr. Jan Borchers

Second examiner:
Prof. Dr.-Ing. Hermann Ney

Registration date: Jul 26th, 2008
Submission date: Dec 12th, 2008

I hereby declare that I have created this work completely on my own and used no other sources or tools than the ones listed, and that I have marked any citations accordingly.

Hiermit versichere ich, dass ich die Arbeit selbstständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt sowie Zitate kenntlich gemacht habe.

Aachen, December 12th 2008

Contents

List of Figures

List of Tables

Abstract

Personal Orchestra is a research project which allows laymen to conduct an audio and video recording of an orchestra. The orchestra adapts to the tempo of the conductor and plays in the given speed. Up to now, a robust gesture recognition of authentic conducting gestures for professional conductors was missing.
In this thesis, a new real-time gesture recognition plug-in for Personal Orchestra is developed, which is able to process three-dimensional acceleration data of the Nintendo Wii remote.

Due to the time-based character of conducting gestures, the hidden Markov model is chosen to solve the task of automatic learning and recognition of 1-beat to 4-beat gestures.
To achieve good recognition results, a large amount of training data is needed. For this reason, two user studies for data acquirement were organized. Over 16000 conducting bars were recorded from 70 differing participants of the studies.
The hidden Markov model was trained with these conducting gestures for each type of bar separately with the help of dynamic programming. After some iterations of time alignment, the attributes in the training phase were optimized for the recognition phase.

The recognition phase of conducting gestures works in real-time with unknown conductors. Some adjustments were made in the recognition algorithm, so that the recognition works while the conducting is executed. It is possible to recognize at which position the conductor is in the bar at each time-point.
The hidden Markov model was trained with half of the conductors, the recognition phase was evaluates with the other half, respectively. The correct position of beats in a given type of bar is recognized correctly up to a rate of 92%.
Additionally a second task was solved: Where to recognize the beats by the system? We identified the beats at the maximum peaks of the acceleration data, especially in the down and forward acceleration of the Wii remote.

Überblick

Personal Orchestra ist ein Forschungsprojekt, welches Laien ermöglicht eine Audio- und Videoaufnahme zu dirigieren. Das Orchester passt sich dem Tempo des Dirigenten an und spielt in der vorgegebenen Geschwindigkeit. Bislang fehlte eine robuste Gestenerkennung für authentische Dirigiergesten von professionellen Dirigenten.
In dieser Diplomarbeit wird eine neue Echtzeit-Gestenerkennungs-Erweiterung für Personal Orchestra entwickelt, welche dreidimensionale Beschleunigungsdaten von der Nintendo Wii Fernbedienung verarbeitet.

Aufgrund der zeitbasierten Beschaffenheit von Dirigiergesten wurde das Hidden Markov Model ausgewählt, um die Aufgabe des automatischen Lernens und Erkennens von 1er-Takt bis 4er-Takt Gesten zu lösen.
Für ein gutes Training und eine gute Erkennung wird eine enorme Menge an Trainingsdaten benötigt. Aus diesem Grund wurden zwei umfangreich angelegte Studien zur Datenakquise durchgeführt. Über 16.000 Dirigiergesten wurden von 70 verschiedenen Personen aufgenommen.
Das Hidden Markov Model wurde mit diesen Dirigiergesten für jeden Takttyp mit Hilfe von dynamischer Programmierung separat trainiert. Nach einigen Iterationen der Zeitanpassung werden die Attribute in der Trainingsphase für die Erkennungsphase optimiert.

Die Erkennungsphase der Dirigiergesten arbeitet in Echtzeit mit unbekannten Dirigenten. Einige Anpassungen wurden im Erkennungsalgorithmus vorgenommen, damit die Erkennung bereits während des Dirigierenvorganges funktioniert. Es ist zu jedem Zeitpunkt möglich zu erkennen an welcher Position sich der Dirigent innerhalb eines Taktes befindet.
Das Hidden Markov Modell wurde mit einer Hälfte der Dirigenten trainiert, mit der jeweils anderen Hälfte wurde die Erkennungsphase evaluiert. Die korrekte Position der Schläge in einer vorgegebenen Taktart wird bis zu 92% richtig erkannt.
Außerdem wird eine zweite Aufgabe gelöst: Wo sollen die Schläge von dem System erkannt werden? Wir haben die Schläge an den maximalen Spitzen der Beschleunigungsdaten der Nintendo Wii identifiziert, besonders in der Beschleunigung nach unten und nach vorne.

Acknowledgements

First, I want to thank Prof. Dr. Jan Borchers, Prof. Dr.-Ing. Hermann Ney, and my advisor Thorsten Karrer, who made this work possible with mentoring me. Especially I want to consider Dr. Ralf Schlüter for the plenty of time he spent for me, answering all my questions in the area of speech and gesture recognition. His valuably ideas guaranteed the quality of my work in the field of hidden Markov models.

Also I want to thank Christian Schmitz, the students of Prof. Reiner Schuhenn, the helpful musicians in the Aachener Students Orchestra and all other people, who participated in my conducting studies.
I am indebted my friends and collegues of i10 making the everyday-life more comfortable and diversified They often gave me useful tips and new ideas how to go on. Additionally a thank to all people, who worked on Personal Orchestra before me and build it up so far, that I could work on this topic. The provided guidances and pictures made it more easy to work on this project. I were lucky to use Dan Greenblatt's framework written in Objective-C to access all Wii remote data.
I appreciate Melanie Kleiner, Elaine Huang and Eric Lee for reading and reviewing my thesis in the final stage and giving me useful tips.

In particular I want to thank my friend Nils-Per Steinmann for exercise patience in the time I worked on this thesis, and for this difficult time without vacations, and the little time I had for him during this work.
Last but not least I want to thank my loved parents Annette and Albert Koster for supporting me psychologically and financially in so many years. Closing, I send my regards to the rest of the "ABC-Bande" to my brothers Benjamin, and Christoph.

Thank you!

Conventions

Throughout this thesis we use the following conventions.

The plural "we" will be used throughout this thesis instead of the singular "I", even when referring to the work that was primarily or solely done by the author.

Text conventions

Definitions of technical terms or short excursus are set off in orange boxes.

EXCURSUS:
Excursus are detailed discussions of a particular point in a book, usually in an appendix, or digressions in a written text.

Definition:
Excursus

Source code and implementation symbols are written in typewriter-style text in yellow boxes.

MYALGORITHM:
```
myClass:  Here you can find the algorithm
```

Algorithm:
myAlgorithm

The whole thesis is written in American English.

Mathematical conventions

"Global" maximum or minimum is meant to be global in one bar of a conducting gesture.

Figure 1: Conductor at work

Chapter 1

Introduction

Interaction in music applications is arousing more and more interest of companies to make producing music available for lay musicians. The gaming area seems to be most promising, also for commercial purposes. New interaction techniques and devices are invented to develop a new feeling closer to the application area apart from mouse and keyboard. In the following, four examples are given:

- A musical interface with new interaction technique is Soundnet[1] . It is a giant web of eleven meters consisting of thick ropes. Stretching and movement is detected at the end of the ropes by eleven sensors. Three people can climb in the net at once and perform the instrument.

- Japanese game developers created collaborative cell phone games ("Chokkan Band" and "Chokkan Classics"[2]), where the players are collective musicians of a band or an orchestra. Instruments like flute, violin or cello can be played by moving something in front of the infrared sensor of the mobile phone or moving the phone itself. All players have to sync their movements for a good sound. If anyone stops moving, his instrument mutes.

[1]http://www.sensorband.com/soundnet/index.html
[2]http://www.taito.co.jp/mob/title/i/classic/index.html

- With a new intuitive interface of YAMAHA you can produce sound and light by painting on a board with a matrix of 16*16 LED switches (see figure 1.1). It contains six different performance and light modes which can be combined for a more complex musical expression. For more demos of Tenori-On visit the site of YAMAHA[3] .

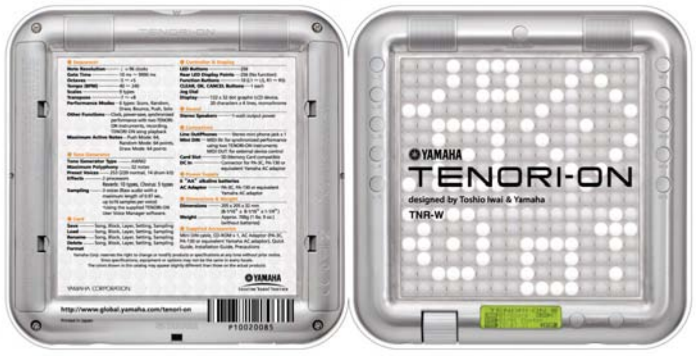

YAMAHA
TENORI-ON
designed by Toshio Iwai & Yamaha
TNR-W
http://www.global.yamaha.com/tenori-on

Figure 1.1: Other possibilities for lay-musicians to play music: TENORI-ON interface by YAMAHA

- A new collaborative electronic music instrument, called Reactable[4] , was integrated in a table was developed by the Music Technology Group at the University Pompeu Fabra. Several performers can influence all instruments simultaneously with controlling physical objects on the multi-touch table. These relating objects can be moved and rotated on the surface of the table and can synthesize sound with these objects which represent generators, filters and modulators.

These types of musical interaction open questions in research areas such as real-time gesture recognition, natural interaction, hardware, reaction times, time stretching etc..

[3]http://www.global.yamaha.com/design/tenori-on
[4]http://mtg.upf.edu/reactable

1.1 Personal Orchestra

The research project Personal Orchestra[5] at RWTH Aachen University is an interactive conducting system. Lay conductors can conduct a virtual orchestra, which reacts to the speed and size of the conducting gesture. Audio and video recordings of orchestras such as the Vienna Philharmonics were recorded and preprocessed, so that interaction got possible (for example marking the beats). An infrared baton system is installed, which transmits its position in two dimensions. The existent project is especially a good experience for kids, because they can move the baton only up and down, that the orchestra plays in this tempo. A "Wiggle" mode was established, so that the orchestra reacts on arbitrary movements. Additionally the orchestra pauses, if the conductor does. The musicians will complain if the conductor moves much too fast or too slow for a long time.

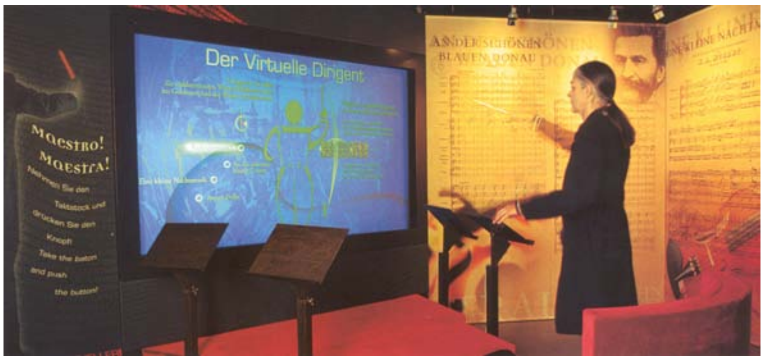


Figure 1.2: Personal Orchestra exhibit in Vienna

The main task of this thesis is to create a completely new gesture recognition system, based on three-dimensional acceleration data. This system will be a plug-in for Personal Orchestra, such that musicians and conductors, who know the basic conducting movements of professionals, can conduct the virtual orchestra. Thus, this kind of users have a more natural experience, which is more close to a real conducting practice. The resulting system could also be a training system for prospective conductors who want to learn these professional gestures.

[5]http://hci.rwth-aachen.de/po

1.2 Pattern recognition and hidden Markov models

Pattern recognition was originally introduced in the early 60s and was not very successful, because of delimited technical possibilities. The hidden Markov model was developed by Baum et al. [1970] in the late 1960s and early 1970s. It is named after the russian mathematics "Andrei Andrejewitsch Markow" and is a statistical model, which can be described by two Markov processes. The model is assumed to be a Markov process with unknown parameters, and the challenge is to determine the hidden parameters from the second process, the observable parameters.

Automatic gesture and speech recognition are two related research topics. Gesture recognition is very close to speech recognition application area: due to the time-based nature of speech and gestures. The hidden Markov model can describe temporal content, because it learns and calculates probabilities of hidden sequences automatically, if observable clues occur in a time-based environment.

Despite good processing power, it is still hard to design speaker independent systems with a big word pool. Furthermore, the original algorithms are not intended to work online while the gesture is executed. The algorithms estimate probabilities for a pattern having occurred, but after the pattern ended before. That is why the original gesture recognition algorithms are not real-time compatible. In this problem, the business for this work is given: How to create a real-time algorithm, which can interpret at which point in the pattern we are at this moment. This problem will be the most challenging part of this thesis. It is important to recognize where the beats of the conducting gestures are. This is the next part, which will be analyzed in this thesis. Where is the beat in the conducting gesture "felt" by the conductor. Is it as described in the literature for conducting students at the lower turning point of the gesture?

1.3 Main challenges

In this work, an adaptive real-time gesture recognition is implemented for Personal Orchestra. The positional gesture recognition is replaced by an acceleration based recognition system. The acceleration data is transmitted via Bluetooth by a robust Nintendo Wii remote. A time-related real-time gesture recognition of conducting gestures is needed, which is able to learn the gestures automatically. Here, the hidden Markov model is chosen, because of its time-based learning methods. The recognition phase of the hidden Markov model has to be modified, because it should be operating while the gesture is performed. This time-based aspect is important, because the position in the gesture has to be recognized instantly. We want to recognize which gesture is performed, but also at which position in the gesture we are at the moment. This is why the hidden Markov model has to be adapted, because it must decide in real-time at which position in which gesture it is at each moment. Also the speed and the size of the gestures arouses our interest, another exercise of this thesis.

Therefore, it has to be examined in many conducting gestures, where the beats are. Many conductors were invited and recorded, so that we could analyze how the acceleration data of a conducting gesture looks like in different speeds of different conductors in different styles. The system is designed in such way, that other people can conduct the system, than those who trained it.

1.4 Thesis structure

In the following chapter we explain the basic knowledge which is needed to understand this thesis, especially the responsibilities of conductors, main shapes of conducting gestures, and an introduction of hidden Markov models.

In chapter 3 ***Related work*** existing work in the field of virtual orchestras and gesture recognition algorithms are presented.

Chapter 4 ***Data acquirement*** includes descriptions of our conducting studies and first results in the area of acceleration-based conducting patterns.

In chapter 5 ***Hidden Markov model and its application*** the applied hidden Markov model (training and recognition phase) and necessary adjustments are explained in detail.

Important implementation details of the training and recognition phases are specified in chapter 6 ***Implementation details***.

In chapter 7 ***Evaluation*** the evaluation process and the results of the recognition system are presented. Here, the size of pruning window and speed of the algorithm is evaluated. Additionally, we analyze at which position the training algorithm assigns the full beat numbers in the trained models.

Concluding, a summary of all results and proposals for future work are given in chapter 8 ***Summary and future work***.

Chapter 2

Basic theory

" Orchester haben keinen eigenen Klang; den macht der Dirigent."
"The conductor forms the sound, not the orchestra."

— Herbert von Karajan (1908-89), Austrian conductor

To understand this thesis completely, some knowledge about conducting and hidden Markov models is required. The reader, who has no knowledge of one of these topics can find a sufficient introduction in this chapter.

2.1 Conducting an orchestra: duties and responsibilities

ORCHESTRA:
To be considered an orchestra, an instrumental ensemble must: (a) have strongly centralized leadership, (b) bow in unison, (c) refrain from adding freely extemporized ornamentation [Zaslaw, 1988].

Definition: *orchestra*

Beating the rhythm is historically handed down by the primitive people [Wendelberger, 1983]. In the basso continuo era in Europe around 1800 musical ensembles were

The occupation of a conductor came up in the early 19th century led by active musicians: the bandmaster playing the cembalo, and the concertmaster playing the violin. Only in particular cases a leader stood in front of the orchestra stomping a stick rhythmically on the floor. The occupation of a conductor in the context as we know today came up in the early 19th century. Chamber music ensembles grew to orchestras which made leading while simultaneously playing an instrument impossible. Until the 20th century conductors mostly were composers, staging only their own works [Wendelberger, 1983].

Definition:
conductor

CONDUCTOR:
A conductor (german: Dirigent, Kapellmeister) is:
1. Leader of the orchestra of a princely private band or other musical institution
2. A conductor working for the general music director with an orchestra or an opera ensemble
[Stanley Sadie, 1986]

The conductor's duties and responsibilities are leading the orchestra in technical and musical questions

Conducting is a medium for communicating real-time information to performers. The conductor is responsible for technical coordination of the musicians and posesses the interpretational sovereignty. His prerequisites are musicality, sense of style, charisma, familiarity with the score, and mastery of the conducting technique [Wendelberger, 1983]. He must work out the musical expression (tempo, dynamics, articulation) with the orchestra. Beneficial is the ability to communicate nuances of phrasing and expression through his gesture. The conductor is particularly responsible for the choice of tempo for all parts of the composition. The Italian terms (Adagio, Allegro, etc.), which are suggestions of the composer, influence the conductor's decision. Metronome markings of the componists are also a relevant indication. The precise meaning of Italian terms is always open to interpretation which is why the conductor's training includes structural analysis, style studies, performance practice, and courses of composition. For a good orchestral work, the conductor should have a good ear to recognize faults such as wrong tones up to small inaccuracies by the musicians in pitch and tempo. To communicate his aims, a conductor should also have founded knowledge about all instruments in the group and he should additionally be able to play at least one instrument. Knowledge about how to work with a group of people also helps to quickly get good results with motivated musicians [Rudolf, 1995].

2.1.1 The conductor's baton and its usage

The conductor's baton works as elongation of the conductor's arm to be visible in to the last rows of the musicians

In the 19th century, the spatial distance in growing orchestras between the conductor and some of his musicians became so big, that the conductor needed a utility to enlarge and clarify his movements, so that musicians in the last row could also see small details of movements. The conductor's baton was manufactured from white beech wood, as it was best viewable in front of the black tailcoat in a dark orchestra pit. Today a baton has become a symbol for a conductor. An example of a modern baton is shown in figure 2.1.

Figure 2.1: A modern baton

Conductor's gesture consists of beat, tempo, dynamic, cueing

A conductor's gesture depends on the composition and its beat structures. The most common bars are the 4-beat, 3-beat, and 2-beat. A conductor's gesture is a form of non-verbal communication moving the hands or the baton in a specified pattern. There are no absolute rules on how to conduct correctly, and a wide variety of different conducting styles exist, especially world-wide known conductors develop their own styles. In this thesis the most common conducting patterns in use will be presented. Figures 2.2, 2.3, and 2.4 illustrate the motion sequences of different beats. The size of the whole gesture assignes the level of dynamics. Depending on the interpretation style, the shape of the pattern is more smooth and flowing for a legato, or more peaked for staccato approach. A sharply angular pattern suggests a marcato style (more information can be found on this page[1]).With the second hand without baton, changes of dynamics or indication of entries of instrument groups can be signaled.

Conductor's education in practice is mostly practicing without any orchestra

The author's experiences have shown that for beginners and students it is very hard to find an orchestra to practice with. Hands-on experience with real orchestras is difficult to aquire. Students can consider themselves lucky if they can practice a musical score once with a lay-orchestra before having an exam or performance. Most often conducting exams are held conducting two pianists instead of 100 musicians. This proceeding is naturally a bad alternative and takes place under unrealistic conditions. Here a virtual orchestra could help for practicing and augment this scene.

2.2 Shape of a conducting gesture

A piece of music consists of smaller components: movements, phrases, bars, and beats

A musical composition can be divided into smaller units, for example, into phrases which in turn are broken down into bars and beats. Every bar represents a repeating pattern in the conductors gesture, the overall shape is the same, but the style of interpretation and speed can be varied in a piece of music. Every bar contains a fixed number of beats. These beats are marked in the gesture, mostly by acceleration and change of the direction of the baton (see, for example, figure 2.2). For most orchestral music the first beat in a bar is the most accentuated one. It corresponds to the lower turning point of the gesture (downbeat), the last beat in a bar is the most unaccented, it is represented in the gesture by an upwards movement (upbeat). The beats between the downbeat and the upbeat are usually performed to the left or to the right. This is a natural combination of

[1]http://www.theconcertband.com

an ergonomic and a musical presentation of every beat and its musical meaning [Rudolf, 1995].

Here, "gesture" describes a characteristic pattern of motions, which correspond to its signal data (eg. position or acceleration). Gesture means an ordered, finite series of signal data with an explicit start and end. In a conducting gesture the same pattern is repeated every bar. Consequently, we define the start of a conducting gesture at the beginning of its confirmation, on count one and the end shortly before the count one of the next bar.

The conducting gesture is defined by a repeating pattern starting at count one and ending shortly before the next count one of the next bar

2.2.1 The 4-beat pattern

The neutral-legato 4-beat pattern is a smooth motion in four different directions on a two-dimensional plane in front of the conductor, generated by an up-down and left-right axis of the writing hand. The following patterns are described for a right-handed person. Left-handed conductors must mirror the left-right movement. This pattern should be neutral in character, hence, it only uses straight lines and a continuous motion. The gesture starts with a down-movement of the baton before the first beat. The first count is "felt" at the bottom turning point before the rebound up. Then the baton is moved to the conductor's left for the second count. Then the third count lies before the right edge of the pattern. On count four, a movement to the upper mid of the gesture closes the loop of one bar. On the schematic figure 2.2 it becomes clear where the beats of the gesture should be placed by the conductor. Also, it is obvious that the distances between the counts are different: the conductor must achieve a smooth motion by accelerating his movement at some points without stopping at the counts or at any other point. That is a bit different in the staccato pattern (figure 2.2): here a stop at each count is essential, and the rebound after the first count is missing. The expressive-legato pattern is similar to the neutral legato pattern, but its shape is more curved. So a rounder shape is recommended by this style which is interpreted with sustained tones. The interpretation of a piece of music with a certain style can be identified by the musician by the shape of the gesture.

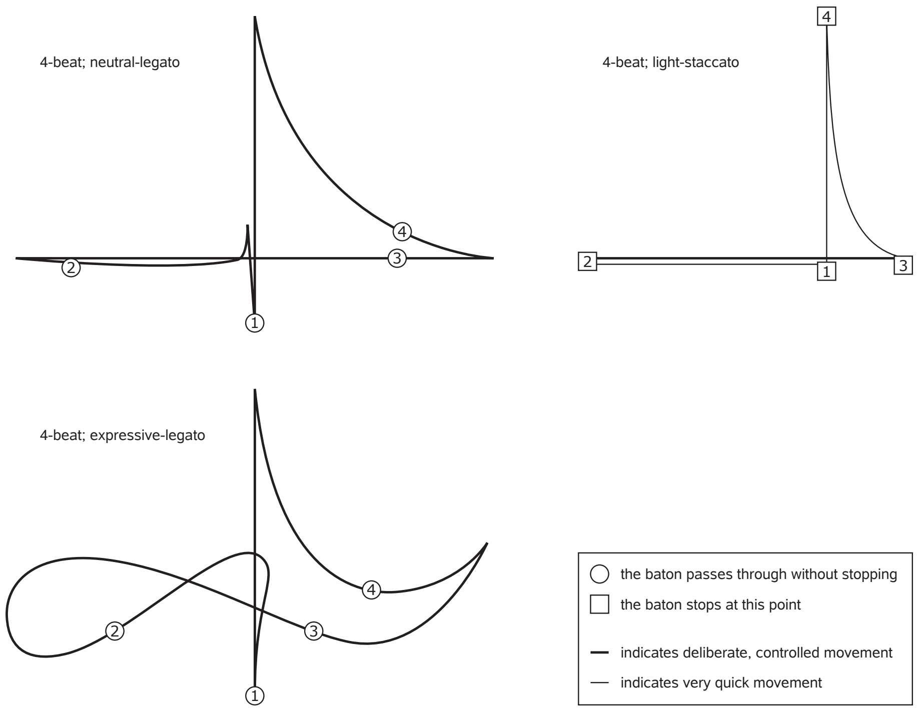

4-beat; neutral-legato
1
2
3
4
4-beat; light-staccato
1
2
3
4
4-beat; expressive-legato
1
2
3
4
○ the baton passes through without stopping
□ the baton stops at this point
— indicates deliberate, controlled movement
— indicates very quick movement

Figure 2.2: 4-beat patterns: neutral-legato, light-staccato, expressive-legato. Image adopted from Rudolf [1995]

2.2.2 The 3-beat pattern

The 3-beat pattern starts with the same movement to the bottom for the first count. Afterwards the movement goes to the right side of the conductor, which is in the opposite direction to the 4-beat. With other words, the movement to the left of the second count of the 4-beat is left out. Obviously, the second and third count of the 3-beat pattern are comparable to the third and fourth count of the 4-beat pattern. The overall shape of neutral-legato, expressive-legato and staccato is equivalent to the 4-beat shapes (for 3-beat gesture see also figure 2.3).

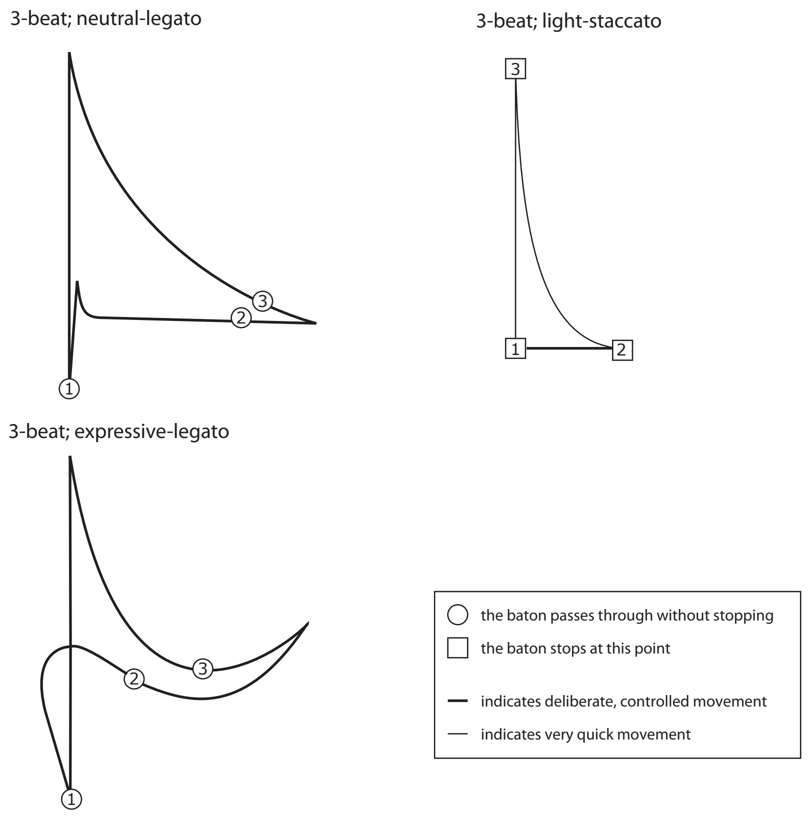

3-beat; neutral-legato
1
2
3
3-beat; light-staccato
3
1
2
3-beat; expressive-legato
1
2
3
○ the baton passes through without stopping
□ the baton stops at this point
— indicates deliberate, controlled movement
— indicates very quick movement

Figure 2.3: 3-beat patterns: neutral-legato, light-staccato, expressive-legato.

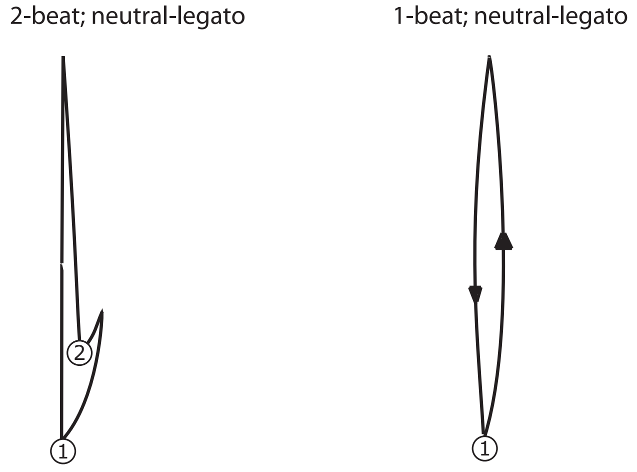


Figure 2.4: 2-beat and 1-beat pattern: neutral-legato. Images adopted from Rudolf [1995]

2.2.3 The 2-beat pattern

The overall shape of the 2-beat pattern (see figure 2.4) is nearly only motion on the up-down axis. The rebound of the first count goes a bit to the right side, so that a small left-movement down is recommended shortly before the second count. However, it must be considered that the second count should be above the first (more important) count.

2.2.4 The 1-beat pattern

The overall shape of the 1-beat pattern is just a movement on the up-down axis, whereas the beat is felt at the bottom of the curve (see also figure 2.4). This gesture is more easy for laymen and it is possible to conduct all beat patterns only with this up-down movement. In Personal Orchestra, the lay-conductor also can exclusively use this gesture to let the musicians play through a whole piece of music.

2.2.5 The preparatory beat

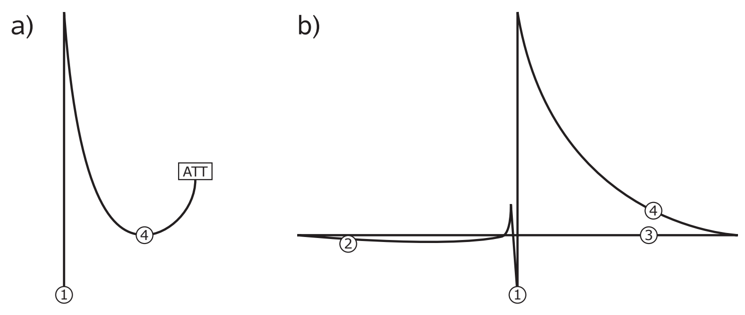


Figure 2.5: The preparatory beat. Image adopted from Rudolf [1995]

The preparatory beat is an aid for a simultaneous and synced start of the musicians

Most often the conductor gives one extra beat before the music starts. He raises his arms and pauses every movement. This is the position of attention and invites the musicians to lift the instruments for breathe in simultaneously. This allows a clear attack at the start. The beat before the first beat is conducted, for example if the music piece is written as a 4-beat and starts with count one, the forth count is given before. This would be a movement described in figure 2.5a. Due to the fact that the preparatory beat is strictly in time of one beat, the musicians anticipate the tempo of the piece before playing the first tone. This allows a perfectly synchronous start for the first bar in the music [Rudolf, 1995].

2.3 Hidden Markov model

The hidden Markov model is a time based learning model

To recognize the position in the conductor's beat pattern, we recommend using a time-based learning framework. An automatic learning hidden Markov model (HMM) seems to be the best choice, because it is able to consider time based processes. The hidden Markov model is also used in speech and gesture recognition frameworks. Identifying an acceleration gesture is assignable to the problem recognizing a spoken word in a speech signal, because of its time-based character. Here, the analogy can be found in a trained word and a trained gesture.

2.3.1 Basic theory of the hidden Markov model

The basic theory of the hidden Markov model was introduced by Baum et al. [1970] in the late 1960s and early 1970s and its application to speech recognition was done by Baker at CMU, and by Jelenik and his colleagues at IBM in the 1970s [Rabiner, 1989]. Speech recognition is still hard due to several factors:

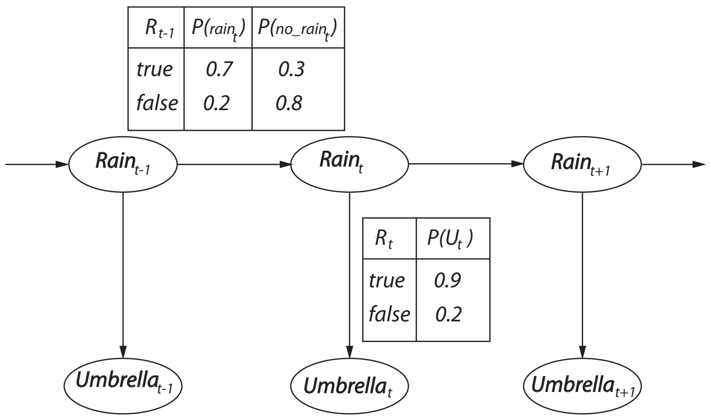
- high variability of the signal
- most decisions are interdependent
  - word and phoneme (sound) boundaries are not visible in the signal
  - high variations in speaking rate
- decision in context: recognize whole sentences rather than single words
- large number of classes to be distinguished.

For our task, the last point is not such a big problem, because at first we only have to detect at maximum four "words" — four different gestures. The first two points are also affecting our problem: Due to varying conductors and their conducting experiences and preferences, the conducting gestures variegate size, shape, clarity, and speed in different phases of one gesture. These factors also influence the quality of the recognition.

HIDDEN MARKOV MODEL:
A hidden Markov model is a doubly stochastic process with an underlying stochastic process that is not observable (it is hidden), but can only be observed through another set of stochastic processes that produce the sequence of the observed symbols [Rabiner and Juang, 1986].

Definition: *Hidden Markov model*

2.3.2 HMM example: the umbrella world



R_{t-1}	$P(rain_t)$	$P(no\_rain_t)$
true	0.7	0.3
false	0.2	0.8

R_t	$P(U_t)$
true	0.9
false	0.2

Figure 2.6: Bayesian network structure and conditional distributions describing the umbrella world. Figure adopted and changed from [Russell and Norvig, 1995].

As an introduction, we want to give an example of a dynamic situation to understand what an HMM can solve in a time-based context. We want to forecast if it rains when we only know the weather the day before and we can see if our colleague is carrying an umbrella. So the weather is not observable (hidden) for us, whereas the umbrella is our observable layer.

Example: Weather forecast with hidden Markov models (The weather is hidden, an umbrella is observable)

To start the prediction we assume that it rained on day $t = 0$ with a probability of 50%.

$$\mathrm{P}(rain_0) = 0.5, \mathrm{P}(no\_rain_0) = 0.5$$

As experts for weather forecast we know that if it rained the day before, the probability that it is raining today is $\mathrm{P}(rain_t|rain_{t-1}) = 0.7$ as shown in figure 2.6. If there was no rain yesterday, the probability that it is raining today is much lower: $\mathrm{P}(rain_t|no\_rain_{t-1}) = 0.2$. Additionally, we know the probability that our colleague is carrying an umbrella if it is raining is $\mathrm{P}(umbrella|rain) = 0.9$. He is carrying an umbrella even though it is not raining with a probability of $\mathrm{P}(umbrella|no\_rain) = 0.2$.

The probability for the first day's weather conditions $\mathrm{P}(R_1)$ can be derived by the probability for rain on day $t = 0$ multiplied with the conditional probability for the weather on day $t = 1$ depending on the weather on day $t = 0$.

$$\begin{aligned}
\mathrm{P}(R_1) &= \sum_{r_0} \mathrm{P}(R_1|r_0) \cdot \mathrm{P}(r_0) \\
&= \mathrm{P}(R_1|rain_0) \cdot \mathrm{P}(rain_0) \\
&\quad + \mathrm{P}(R_1|no\_rain_0) \cdot \mathrm{P}(no\_rain_0) \\
&= \begin{bmatrix} 0.7 \\ 0.3 \end{bmatrix} \cdot 0.5 + \begin{bmatrix} 0.2 \\ 0.8 \end{bmatrix} \cdot 0.5 = \begin{bmatrix} 0.45 \\ 0.55 \end{bmatrix}
\end{aligned}$$

This result shows us: it rains on day $t = 1$ with a probability of 0.55%. On the first day the umbrella is seen ($U_1 = true$), so we can adjust our prediction $\mathrm{P}(R_1|umbrella_1)$ for day $t = 1$ using Bayes' law $\mathrm{P}(A|B) = \frac{\mathrm{P}(B|A)\,\mathrm{P}(A)}{\mathrm{P}(B)}$:

$$\begin{aligned}
\mathrm{P}(R_1|umbrella_1) &= \alpha\, \mathrm{P}(umbrella_1|R_1) \cdot \mathrm{P}(R_1) \\
&= \alpha \begin{bmatrix} 0.9 \\ 0.2 \end{bmatrix} \cdot \begin{bmatrix} 0.45 \\ 0.55 \end{bmatrix} \\
&= \alpha \begin{bmatrix} 0.405 \\ 0.11 \end{bmatrix} \approx \begin{bmatrix} 0.786 \\ 0.214 \end{bmatrix}
\end{aligned}$$

In this case $\mathrm{P}(R_1)$ is again the probability for first day's weather condition, which is adjusted by the probability $\mathrm{P}(umbrella_1|R_1)$ to see an umbrella under certain weather conditions. Here, α is a normalizing constant to make sure that the probability vector sums up to 1, because $\mathrm{P}(umbrella)$ without condition is unknown. The probability for rain on the first day is 78.6%. On the second day the umbrella is seen again, so $U_2 = true$. The prediction from $t = 1$ to $t = 2$ is:

$$\begin{aligned}
\mathrm{P}(R_2|umbrella_1) &= \sum_{r_1} \mathrm{P}(R_2|r_1)\mathrm{P}(r_1|umbrella_1) \\
&= \mathrm{P}(R_2|rain_1)\mathrm{P}(rain_1|umbrella_1) \\
&\quad + \mathrm{P}(R_2|no\_rain_1)\mathrm{P}(no\_rain_1|umbrella_1) \\
&= \begin{bmatrix} 0.7 \\ 0.3 \end{bmatrix} \cdot 0.786 + \begin{bmatrix} 0.2 \\ 0.8 \end{bmatrix} \cdot 0.214 \\
&\approx \begin{bmatrix} 0.593 \\ 0.407 \end{bmatrix}
\end{aligned}$$

So we can expect rain with a chance of 59.3%. This probability can be adjusted further because we observe the umbrella also on day $t = 2$:

$$\begin{aligned}
\mathrm{P}(R_2|umbrella_1, umbrella_2) &= \alpha\, \mathrm{P}(umbrella_2|R_2) \\
&\quad \cdot \mathrm{P}(R_2|umbrella_1) \\
&= \alpha \begin{bmatrix} 0.9 \\ 0.2 \end{bmatrix} \cdot \begin{bmatrix} 0.593 \\ 0.407 \end{bmatrix} \\
&= \alpha \begin{bmatrix} 0.534 \\ 0.081 \end{bmatrix} \approx \begin{bmatrix} 0.868 \\ 0.132 \end{bmatrix}
\end{aligned}$$

With the knowledge that our colleague took an umbrella on both days, the probability for rain on day $t = 2$ is 86.8%. The probability increases intuitively from day $t = 1$ to day $t = 2$ because rain persists.

The prediction can be seen as calculating the probability for state at the next day (rain or no_rain), given the evidences $e_1, ..., e_t$ (e.g. $umbrella_t$) . We can derive a recursive computation of predicting the state at $t+1$ from a prediction for t where X_{t+1} is the set of possible states at $t + 1$ and x_t are the states at t:

$$\mathrm{P}(X_{t+1}|e_1, ..., e_t) = \sum_{x_t} \mathrm{P}(X_{t+1}|x_t) \cdot \mathrm{P}(x_t|e_1, ..., e_t).$$

At day $t+1$ the probability of rain or no rain can be estimated with the knowledge of the last days.

Another possible application of a hidden Markov model is finding the most likely sequence for hidden states. For example we have given an observation sequence of umbrellas for five days $O = [true, true, false, true, true]$. Now we can use the relationships between umbrellas and rainy weather and persistence of rainy weather described in the model to estimate the most probable weather condition for each day of the sequence. (The whole example adopted and changed from [Russell and Norvig, 1995]).

2.3.3 Elements of an HMM

The hidden Markov model can formally be described as follows: There is a finite number S of states $N = \{s_1, ..., s_S\}$ in the model and a finite set of M possible symbol observations per state $V = \{v_1, v_2, ..., v_M\}$. A state transition probability distribution

$$A = \{a_{ij}\},$$

$$a_{ij} = \mathrm{P}(q_{t+1} = s_j | q_t = s_i), \quad 1 \leq i, j \leq S$$

influences the entered new state at every clock time t with a hidden state of

$$H = \{h_0, h_2, ..., h_T\}.$$

The state transition matrix contains the a_{ij}, which contain the probability of a transition from state S_i to state S_j. In our example case the state transition matrix is

$$A = \begin{pmatrix} 0.7 & 0.3 \\ 0.2 & 0.8 \end{pmatrix}.$$

The observation symbol probability distribution B in state j is given by

$$B = \{b_j(k)\},$$

$$b_j(k) = \mathrm{P}(v_k \text{ at } t | q_t = s_j), \quad 1 \leq j \leq N, 1 \leq k \leq M$$

In this example, the umbrella world, we get for B

$$\begin{aligned} B &= \{\mathrm{P}(umbrella|rain), \mathrm{P}(no\_umbrella|rain), \\ &\quad \mathrm{P}(umbrella|no\_rain), \mathrm{P}(no\_umbrella|no\_rain)\} \\ &= \{0.9, 0.1, 0.2, 0.8\}. \end{aligned}$$

Finally there is an observation sequence $O = (O_1, O_2, ..., O_t)$ with initial state distribution $\pi = \{\pi_j\}$, $\pi_j = \mathrm{P}(q_0 = s_j)$. The observation sequence in our example case is $O = (U_1, U_2, ...) = (true, true, ...)$: the umbrella was seen the first two days. The initial state distribution here was $\pi = \{0.5, 0.5\}$.

2.3.4 Hidden Markov model: What can it solve?

A hidden Markov model can solve ([Rabiner and Juang, 1986]):

1. **Evaluation Problem:** the probability $\mathrm{P}(O|\lambda)$ of the observation sequence O, given the model λ,
2. **Decoding Problem:** a corresponding state sequence $Q = (q_1, q_2, ..., q_T)$, which is optimal, given a sequence O,
3. **Learning Problem:** adjust the model parameters $\lambda = (A, B, \pi)$ to maximize $\mathrm{P}(O|\lambda)$ with a given observation sequence O.

How we can adapt the model to conducting gestures? In our case, the hidden layer is the type of the conducting gesture, whereas the observable process is represented by the chain of acceleration values.

Acceleration values = observable;
Position in conducting gesture = hidden

The first case is reflected in the evaluation problem: we want to measure the probability that an observed sequence was produced by a given model. The solution of this problem facilitates deciding among several competing models. If the model was calculated and optimized before, this evaluation lets us recognize the best fitting model. Calculating the probability $\mathrm{P}(O|\lambda)$ measures how close the model is to the observation.

Transfering this to the conducting problem, the evaluation can choose the model which matches the conducting observation data best. The models must be generated before. Therefore, if we decide to learn models for 4-beat, 3-beat, 2-beat, and 1-beat gestures, the models can be distinguished and recognized for example by the highest probability. Our recognition phase is described more detailedly in chapter 5.4 ***Recognition phase***.

The decoding problem is solved in the training phase by the Viterbi algortihm

The second solution uncovers the hidden part of the hidden Markov model. The task is to find the "best" corresponding state sequence. To solve this estimation problem, an optimality criterion is used. The choice of optimality criterion is influenced by many criteria such as the topology of the model, averages, and behavior of individual states. The easiest way to work with dynamic processes like acceleration signals is to cut the signal into small segments, so that it can be represented as a synchronous sequence of symbols or states. Each segment contains given attributes (for example: mean-value,variance-value, weight,...), which can be used to assign observations to this segment. We try to segment each gesture training sequence into states and study the attributes of each state. With this procedure we get defined attributes for each state and can refine the states by repeating this alignment problem, trying more or less states, adding attributes, etc.. The optimized attributes for each segment can be stored in a codebook, which can be used in the recognition phase. The way we solved this decoding problem, is explained in chapter 5.2 ***Time alignment***.

Mainly for this problem the Viterbi or forward-backward algorithm is used. Unfortunately, this approach does not work before the conducting gesture has ended. The learning process of a model with its state sequence is done offline and refined by many iterations. The conclusive state definitions and its calculations must be adapted in the recognition algorithm in the evaluation problem.

The learning problem is solved in the training phase by the expectation-maximization algorithm

The learning problem is one in which we strive to optimize the model parameters to best describe how a given observation sequence for a gesture looks like (for example a 4-beat gesture). To estimate how good a model for the training sequence is, a likelihood function is defined and refined. The Baum-Welch, the Viterbi algorithm or the expectation-maximization algorithm can help to solve this problem. See paragraph 5.3.2 ***Expectation-maximization algorithm for mixture densities*** for more details.

Chapter 3

Related work

"Being deeply learned and skilled, being well trained and using well spoken words; This is good luck."

—Hindu Prince Gautama Siddharta (563-483 B.C.), founder of Buddhism

There has been much effort in virtual orchestras and there exist many different possibilities for input devices. In this chapter we will present current work from similar projects in the field of virtual orchestras and gesture recognition with hidden Markov models. We also show what their limits are and why we have to do more studies in this area.

3.1 Virtual orchestras and their input devices

Beside Personal Orchestra, other virtual orchestras have been implemented with many differing input devices. In the following part, several input devices are introduced and categorized by the kinds of data they transmit.

3.1.1 Position based input devices

Three-dimensional recognition of batons with radio waves

Mathews [1989] built up a system called the Radio Baton which estimates the position of his two batons in three dimensions. This works with radio waves, which are sent from the tips of the two batons. The plate (see figure 3.1) is equipped with antennas, which can estimate the position of the batons. Several scores can be loaded and the tempo can be controlled with the baton.

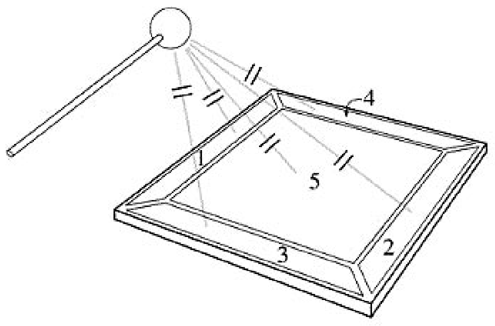


Figure 3.1: Technical drawing of the Radio Baton system. Taken from [Richard Boulanger, 1997]

Existing virtual orchestras are using mouse input, camera input, infrared batons, reflectors, LEDs, or data gloves

Some virtual orchestras work with positional data won by image analysis of videos or simple mouse input, or a baton with a infrared cam like the Buchla system[1] [Garnett and Wessel, 1992]. Personal Orchestra also can process positional data transmitted by the buchla batons ([Borchers et al., 2004]). Reflectors [Garnett and Wessel, 1992] or LEDs on a baton [Bertini and Carosi, 1992] or white points on gloves [Morita et al., 1989] were used as two-dimensional tracking system. Forrest Tobey augmented this 2D space with a second Buchla system to achieve a three-dimensional tracking space, which achieves a better beat pattern recognition [Tobey, 1995] [Tobey and Fujinaga, 1996]. Other systems also work with movement detection like optical flow trackers (also based on frames of a camera).

[1]http://www.buchla.com/lightning/index.html

3.1.2 Acceleration based input devices

The conductor's jacket and new interaction devices are the Wii device and an eWatch

Another possibility is to concentrate on the relative movement in the gesture like it appears in most cases in the field of gesture recognition — the acceleration data. For extracting acceleration data Keane and Gross [1989] mounted a ball on a spring wire in a metal cube. An additional foot pedal was used to start and stop the system. If the tube was accelerated, the ball touched the tube and a pulse was sent.

Figure 3.2: Screenshot of the Pinocchio Project. Taken from the project page[2].

Carnegie Mellon University and Technische Universität München developed a watch [Maurer et al., 2006] with integrated acceleration sensors, which is announced as an alternative input device for conducting the Virtual Symphony Orchestra/ Pinocchio[2] [Bruegge et al., 2007]. They are planning also to make further research using hidden Markov models to detect their gestures with a watch ([Schmidt et al., 2007]) with integrated acceleration sensors.

[2]http://wwwbruegge.in.tum.de/Pinocchio

Pinocchio has similar main goals as Personal Orchestra

Pinocchio will be a further development of the Project Virtual Symphony Orchestra, 2006[3] . Their main goals are like ours; build a virtual orchestra for professional musicians and musical layman, especially children. The Bavarian Radio Symphony Orchestra produced digital audio and video recordings. The conductor should control tempo and volume in different forms of interaction. Personal Orchestra is planned to process acceleration data of the Nintendo Wii device from this work on.

3.2 Personal Orchestra

Personal Orchestra[4] is a research project at the Media Computing Group at RWTH Aachen University. Different releases were developed for the House of Music in Vienna (Austria), the Children's Museum in Boston (USA), the Betty Brinn Children's Museum in Milwaukee (USA) and the RWTH Science Truck. The Vienna Philharmonics,[5] the Boston Youth Symphony Orchestra,[6] and the Aachener Students Orchestra[7] played famous musical pieces for digital audio and video recordings.

In this recordings the beats are marked to make any further interaction possible (to change the speed of a music file). Visitors of the exhibitions are invited conduct the Virtual Orchestra with infrared batons. The orchestra reacts on changing tempo and size of the gesture. The music is fastened and slowed down with a time stretching algorithm implemented by [Karrer et al., 2006]. The main conducting direction lets play some instrument groups louder than others. Additionally, a score is displayed on a second screen which should offer orientation in the music piece.

For this interaction a gesture recognition system (CONGA) [Lee et al., 2006] was developed, which workes with positional data of an infrared baton system. This framework is

[3]http://wwwbruegge.in.tum.de/VSO
[4]http://hci.rwth-aachen.de/po
[5]http://www.wienerphilharmoniker.at/
[6]http://www.bysoweb.org/
[7]http://www.aso.rwth-aachen.de/

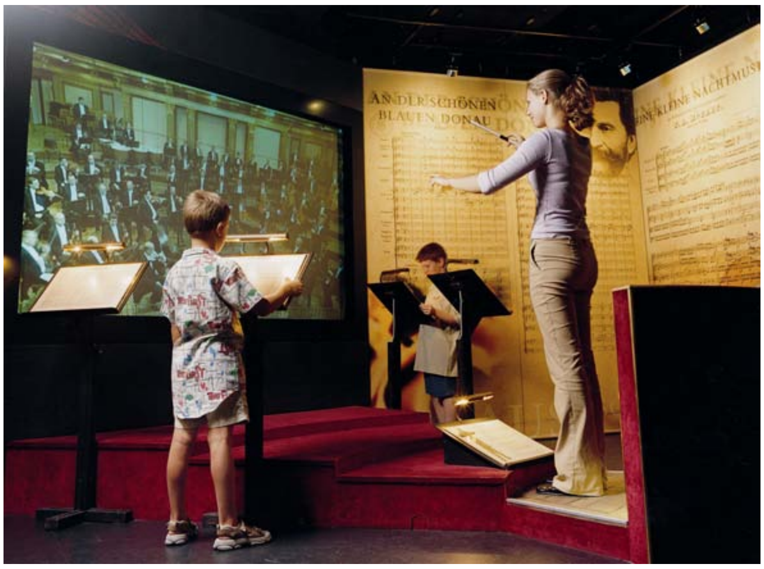


Figure 3.3: Personal Orchestra: Exhibit in Vienna

insufficient for fluid interaction, because of latency caused by the infrared batons. A finite automaton recognizing the 4-beat gesture has five states (up/down/left/right/none) which seems to be much too little for this system to recognize the gesture in a sufficient resolution. With CONGA the directions are tracked, and the beat positions are calculated after they appeared. Some development aspects make it difficult to extent or change it, because the implemented methods for the defined states were very complicated to work with. The framework is not trainable automatically and is adjusted for one ideal person, so it workes only with few trained persons. A new trainable and robust system is needed, because the computer system and *not* the user has to learn how a conducting gesture looks like.

For an advancement in future we will use acceleration data due to the fact the gesture has the character of relative motion which is better to analyze in three relative dimensions than in a two dimensional static angle. The more informations the recognition algorithm has, the better the recognition can work. Choosing the Wii remote for this aims is the fact that it is a widespread low-cost device, so no extra special device has to be bought. Additionally, the Wiimote is very infrangible in contrast to the infrared batons, which is a big advantage for the usage in museum expositions, where it is used especially by kids.

3.3 Live concert with computer orchestra database

The austrian company "Vienna Symphonic Library" (VSL)[8] digitalized sounds, which can be produced by a symphonic orchestra. This library consists of 1,7 million music-snippets of different tones and tone sequences which are played by 32 different instruments in different varieties. More than 150 musicians have worked for more than five years in a special studio to create such a big database. Now everyone can arrange compositions and use this database. Example compositions can be heard at the VSL library[9] . Personal Orchestra does not work with such a library. Real audio and video recordings of several orchestras are used instead.

Paul Henry Smith, Music Director of, wants to present all Beethoven symphonies played by his virtual orchestra, created with these music snippets. The service of putting these code-snippets together can be bought by composers instead of paying a whole symphony orchestra and still is used for film music. Paul Henry Smith is a conductor trained by Leonard Bernstein and Sergiu Celibidache and he wants to "turn digital music-making into a performing art" [Henry Smith][10] . The concerts are being arranged for 2008-2009 in Brooklyn, Boston and in the Washington, D.C. area. Actual details to the Beethoven Digital Symphonies Project can be found at Beethoven Digital Symphonies Project[11] . Paul Henry Smith announced to conduct with the Nintendo Wii remote as a baton to realize a real-time reacting system. It is conceivable for him to use a conductors jacket ([Marrin and Picard, 1998]) which US-musicologist Dr. Teresa Marrin Nakra [2000] started to develop in 1997.

[8]http://vsl.co.at
[9]http://vsl.co.at/en/67/394/253.vsl
[10]http://www.fauxharmonic.com/ [october 30th, 2008]
[11]http://www.fauxharmonic.com/beethoven-symphony-project/

This project seems to be very similar to our aims, but it is optimized only for one or few trained persons. Our project wants anyone, also laymen, to feel like a real conductor — and this is a more complex problem. In our case the system has to learn many diversities of people and variations of styles, whereas Paul Henry Smith has to learn his gestures to get the right reaction of the computer system.

3.4 Nintendo Wii: Interaction possibilities

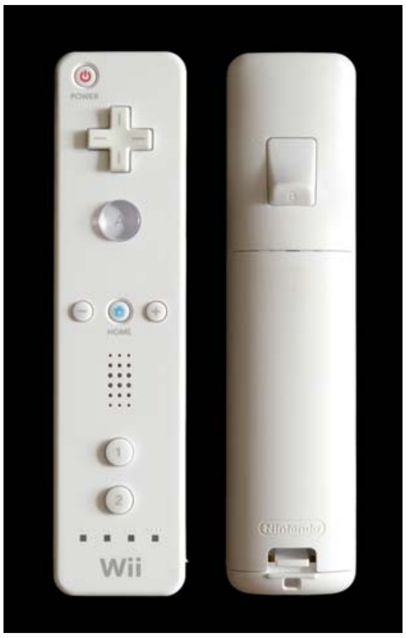


Figure 3.4: The Nintendo Wii remote

The Nintendo Wii was designed to feature a new game experience by using the whole body. For the hand input the Wii remote (see figure 3.4), Wii Nunchuk (if second hand is needed) or alternatively the Classic Controller for video games and for the legs the Wii Balance Board are available. New generations should be won because of the new, easier, and natural feeling of the gameplay. The sales figures and the success argue for this concept and for the development of new interaction devices.

Nintendo Wii and its controllers are designed for new interaction techniques

The Wii remote, short Wiimote, has a build-in 3-axis linear accelerometer (ADXL330)[12] which reports back the instantaneous force imparted on the controller. In free fall, the controller reports back approximately zero force. This device is physically rated to measure accelerations over a range of at least +/- 3g with 10% sensitivity. The Wiimote sends reports to the host with a maximum frequency of 100 reports per second[13] . The Wii remote has a latency of 4.5 milliseconds with Core Audio, which is the smallest delay in the test of possible music controllers by Dedenbach [2008].

In 2006 at Electronic Entertainment Expo (E3) Nintendo gave a first demo of their Wii Orchestra which will be released at the end of 2008 in a package called Wii Music (for a screenshot see 3.5). Players can conduct a virtual orchestra by moving the Wii remote up and down in a self chosen tempo. Also individual instruments can be played concurrently by up to four players. No conducting gestures have to be accomplished, only movement of the Wiimote is sufficient for playing music. More details can be found on the official homepage of Nintendo.[14]

Figure 3.5: Demo of Wii Music. Photo by Arturo J. Paniagua

[12]http://www.analog.com/en/prod/0%2C2877%2CADXL330%2C00.html
[13]http://www.wiili.org
[14]http://e3.nintendo.com/wii/wiimusic/index.html

3.5 Gesture recognition frameworks for the Wii remote

AiLive : First commercial gesture recognition framework

There is a big interest in using the Wii remote for other interactions than for its proper assignation. AiLive Inc [http://www.ailive.net/liveMove.html, 2006] published the first commercial tool LiveMove Pro[15] to record gestures five times, and then recognize them when they are performed afterwards. There is no possibility to adjust it for our aims, yet. It is not considered to recognize gestures in real-time, what is the highest criterion for Personal Orchestra. Especially no positions in a gesture can be defined and recognized.

Wiigee: Open-source project in Java for classifying trained gestures

Wiigee[16] is a platform independent open-source project, with which the user has access to all data of the Wii remote with a built-in gesture recognition tool. Gestures should be trained 10-15 times to get useful results. After filtering and quantizing their data, they use the training phase of the hidden Markov model. In the recognition phase a Bayes classifier is used to classify the gestures, wheras a gesture was marked by pressing a button simultaneously. Testing the system, they train the system with 5 gestures (square, circle, roll, z, and tennis), each one 15 times per participant. Then the system decides which of the five gestures is performed afterwards from the same person! Evaluation results are an average recognition rate from 84,3% for a roll to 94,5% for tennis [Schlömer et al., 2008].

We cannot use these frameworks, because we want to detect positions in the gestures in real-time, and not the gesture when it has completed

Our aims in this case are much higher, the persons doing the training task are never the persons using the system for the recognition task. We want to recognize more complex gestures in real-time and additionally say at which point we are at any time in this gesture. So no gesture recognition framework of these will fulfill our aims completely, but the usage of hidden Markov models seems to be promising, because hidden Markov models can be modified to solve more questions than only detecting a gesture.

[15]http://www.ailive.net/liveMovePro.html
[16]http://wiigee.sourceforge.net/

3.6 Gesture recognition with the hidden Markov model

As a hidden Markov model is the standard algorithm for pattern recognition, we concentrate in the following on gesture recognition with HMMs with acceleration based input. After a more general gesture recognition project, conducting systems are presented, which are working with acceleration based input.

3.6.1 Gesture recognition with acceleration based input

HMM in accerleration-based recognition systems

There also are other existing gesture recognition systems which are working with the HMM. Kim and Chien [2001] and Jani Mäntyjärvi et al. published studies for their online gesture recognition system for mobile interaction [Mäntylä et al., 2000],[Kallio et al., 2003],[Mäntyjärvi et al., 2004], or [Kela et al., 2006] whereas with "online" they mean after the gesture stopped. They achieved a recognition rate of 95% with 16 gestures and 20 training sequences with accerlerometer-data. Main motivation for choosing HMMs for their purposes is that it is a modeling tool "that can be applied for modeling time-series with spatial and temporal varibility" [Mäntyjärvi et al., 2004]. After preprocessing the data (low pass, normalizing using interpolation) they do a vector quantization (using the k-means algorithm) on the training data to make the three-dimensional data one-dimensional. For their recognition task they implemented the Baum-Welch [Baum et al., 1970] and Viterbi algorithms (e.g. see [Rabiner and Juang, 1993]).

Backward pass is not suitable for a real-time recognition without waiting for the end of the gesture

Using the Viterbi algorithm for our recognition phase is not possible, because it causes a big latency, a backward pass through the collected data is needed, so all data must be seen before making assumptions [Narasimhan et al., 2006]. We have a continuous gesture input and we cannot wait until it stops, because we want to give real-time feedback. leaving out the backtracking algorithm we decrease recognition time, but decrease recognition quality, too. Also our problem is a bit different from all other gesture recognition tasks: our premium task is to observe at which position we are in a gesture additionally to what gesture it is. In a 4-beat we know that the conductor will use the 4-beat gesture. If it is not this gesture, then we have to look what the conductor is doing else as second task.

3.6.2 Recognition of conducting gestures using hidden Markov models

First conducting simulator using HMM

Usa and Mochida [1998] designed "A Multi-Modal Conducting Simulator" which uses two acceleration sensors, an eye camera (for tracking the cue with the eyes), a still video image and a breathing sensor to detect the conductors intention. This conducting simulator was the first system using the hidden Markov model to recognize which beat the right hand of the conductor is accomplished. A likelihood is calculated which beat of which measure is conducted with the forward algorithm (needs 20ms). Other information about the score and beat occurrence had to be incorporated and combined with fuzzy production rules to achieve a recognition rate, that allows to conduct a whole music piece without stopping. The system starts with one extra beat (the preparatory beat) combined with a breathing in by the conductor. The autonomy of the orchestra is adjustable: if the autonomous degree is set to 0%, the system is "glued" to the conducting. When the autonomous degree is 100%, the system ignores the conducting and plays in its recorded speed.

While this project is very similar to our main goals, a few differences can be seen. We will work with three-dimensional acceleration data and not with still camera image or breath sensor. The autonomous degree will not be adjustable, all conductors should have the same experience. The recognition of the multi-modal conducting simulator needs a preparatory beat to start the recognition. Contrariwise, we allow a start at every point of the gesture.

Three-dimensional conducting gesture recognition

Wang et al. [2001] use a hidden Markov model for their conducting gesture recognition, because motions contain spatial and temporal variation. The three-dimensional motions are recorded on two-dimensional video and divided in five basic gestures (2-beat to 6-beat). The HMMs are trained with these five gestures and a distance matrix is calculated. Eight words are extracted, because subgestures are also recognized by the automatic clustering algorithm. In the unsupervised recognition phase 64% of the segments are recognized correctly as known patterns, the remains are labeled as unknown patterns. We solve our tasks more supervised on four different beats, especially our classification task is done with a time alignment and no distance matrix.

MAX/MSP prototype for recognizing conducting gestures (not in real-time)

Kolesnik and Wanderley [2004] also want to use hidden Markov models for their conductor's gesture recognition task. They produce a MAX/MSP prototype with an integrated HMM object for learning, finding an optimal sequence of states and recognition. The observation sequence recognition is implemented with a forward-backward algorithm, the Viterbi algorithm calculated the optimal sequence as usual in an HMM, and the model training is done by a Baum-Welch re-estimation. The conducting gestures are captured by two cameras: one positioned in front of the conductor, and the other positioned at the side of the conducting hand, wearing a color glove. Beats can only be recognized in real-time by detecting maxima and minima in the gesture without using the hidden Markov model. To differentiate several conducting gestures, a training set consisting of two different gestures (4-beat staccato and 4-beat legato) is recorded with one doctoral conducting student. Twenty sets are learned, and ten testing sets were recorded by the same person. The recognition rate of 98% of them are aligned to the correct pattern.

Our problem is more difficult than recognize gestures

We have to distinguish more gestures than two, and the main idea is that we have to recognize conducting gestures of arbitrary persons, which did no learning phase with the HMM of our system. Additionally, they could not solve this recognition task in real-time, because of the forward-backward algorithm. This problem we solve, using no backtrace algorithm for the recognition task, which perhaps could degrade the recognition rate.

Chapter 4

Data acquirement

"The great thing about a computer notebook is that no matter how much you stuff into it, it doesn't get bigger or heavier. "

— Bill Gates (1955 -), American Entrepreneur and Founder of Microsoft Co.

In this chapter the work with conductors and future users of the system is described. A high amount of data has been acquired (over 16000 conducted bars). About 4000 bars for each kind of conducting gesture were collected from about 70 different test persons.

4.1 First user study

To recognize a conducting gesture we started to run some studies to get to know how the acceleration data of a Nintendo Wii of conducting gesture looks like, and if there are differences between conductors and non-conductors. Especially, it is useful to analyze the shape and the kind of the acceleration data we get from the Nintendo Wii remote.

Therefore, we ran a study with 50 test persons who were asked to conduct 1-beats to 4-beats in 3 different speeds (76, 100 and 132 beats per minute). In the study we invited 10

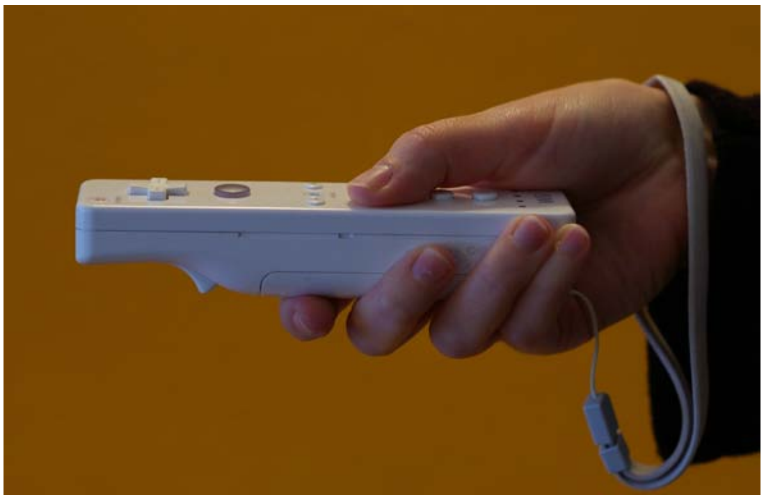


Figure 4.1: Correct hand position on a Wii remote

professional conductors, 12 musicians, and 23 layman. The layman were taught very shortly by showing pictures of the movement and a short training phase before starting the recordings. All participants were asked to conduct exactly on a given beat which was audible from a metronome. The Wii remote must be hold as shown in figure 4.1. To use as much as possible of the available conducting data, all three axes of acceleration data were recorded, though the conducting gesture seems to be only a two-dimensional gesture. This was a good decision, because the third axis (z-axis) contains more information than assumed. In most of the gestures, the beats can be detected in this dimension, because the remote is tilted at the beats (see figure 4.4). The data of left-handed persons was mirrored in the left-to-right axis.

Over 450 recordings of few minutes each were made and analyzed afterwards in a sorting session in a team of eight. Before, all recordings were compressed to one average bar per conductor and kind of beat. After all different bars were printed and glued on sticky notes, every kind of beat was clustered separately (see also figure 4.3). The tables in Appendix A represent the results of this session; the extracted features and their appearances in percent. The first part of each table contains the percentage of the number of appearances per different kind of bar. The second part shows the

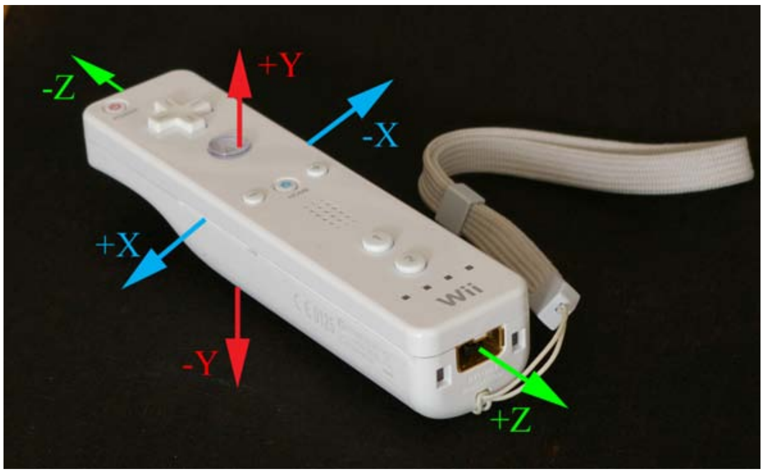


Figure 4.2: The Nintendo Wii remote with its three-dimensional acceleration sensors

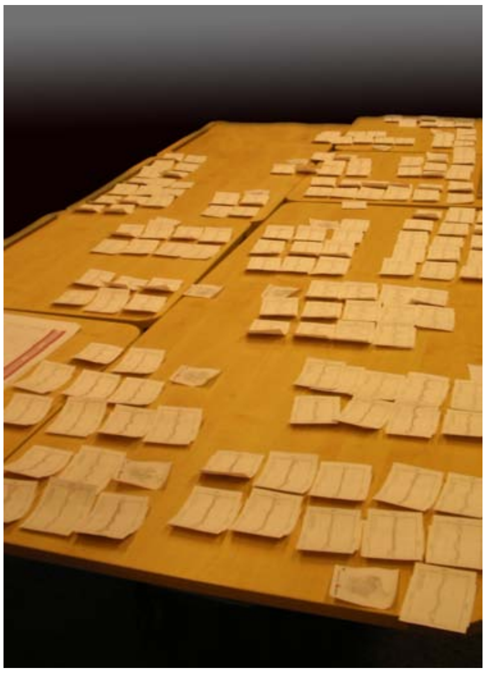


Figure 4.3: The workstation after one clustering session

probability that a certain feature occurs in a certain type of bar. Gray colored fields mark high percentages of features which would be very suitable to recognize a certain beat by hand. Among other things, aspects were found to distinguish between 1-beat, 2-beat, 3-beat, and 4-beat. In the 4-beat in figure 4.4 an example bar for every time signature up to 4-beat measure is given. It is remarkable that on every beat in every bar the curve of the y-axis and the curve of the z-axis has a maximum, which is also represented in 88% of all data for the y-axis, and in 64% for the z-axis (see table A.1).

The first beat of each bar is recognizable by a global maximum in the y-axis in 61% of all data. Shortly before this maximum there is a detectable global minimum in this curve (in 80% of all bars). This information should be sufficient to detect 1-beats. It is possible to distinguish between 2-beat and 1-beat bars by observing the y-axis and the x-axis. If a minimum on the y-axis corresponds to the count 2.5, and at the second count the curve on the x-axis has a minimum and then increases during the count 2.5, it is a good indication for a 2-beat bar (see table A.4).

Table A.3 contains the characteristics of a 3-beat gesture, which were observed at the sorting session and the average curve of all 3-beats which were recorded with the Nintendo Wii remote (see 3-beat in figure 4.5). The most important values, which also can be seen on the 3- beat example in figure 4.4, are at the full beats. We can recognize — aside from the maximum in the y-axis — an increasing curve on the x-axis on the second count which is decreasing again at the third count. A decreasing curve on the x-axis paired with a maximum on the y-axis at the third count was only found in 3-beats. With a occurrence of 100% in all 3-beats, this is a good criterion for a 3-beat gesture is found in this criterion. On count 2.5 we find a global maximum in the x-axis in 87% of all 3-beats. Here, 69% of all gestures with this feature were 3-beats, and 31% were 2-beats.

The 4-beat gesture data contains many features which can be used to detect a 4-beat rhythm and the position in the measure. On the second count, a maximum in the y-axis and a fast decreasing curve on the x-axis can be detected, crossing the zero from a maximum to a global minimum.

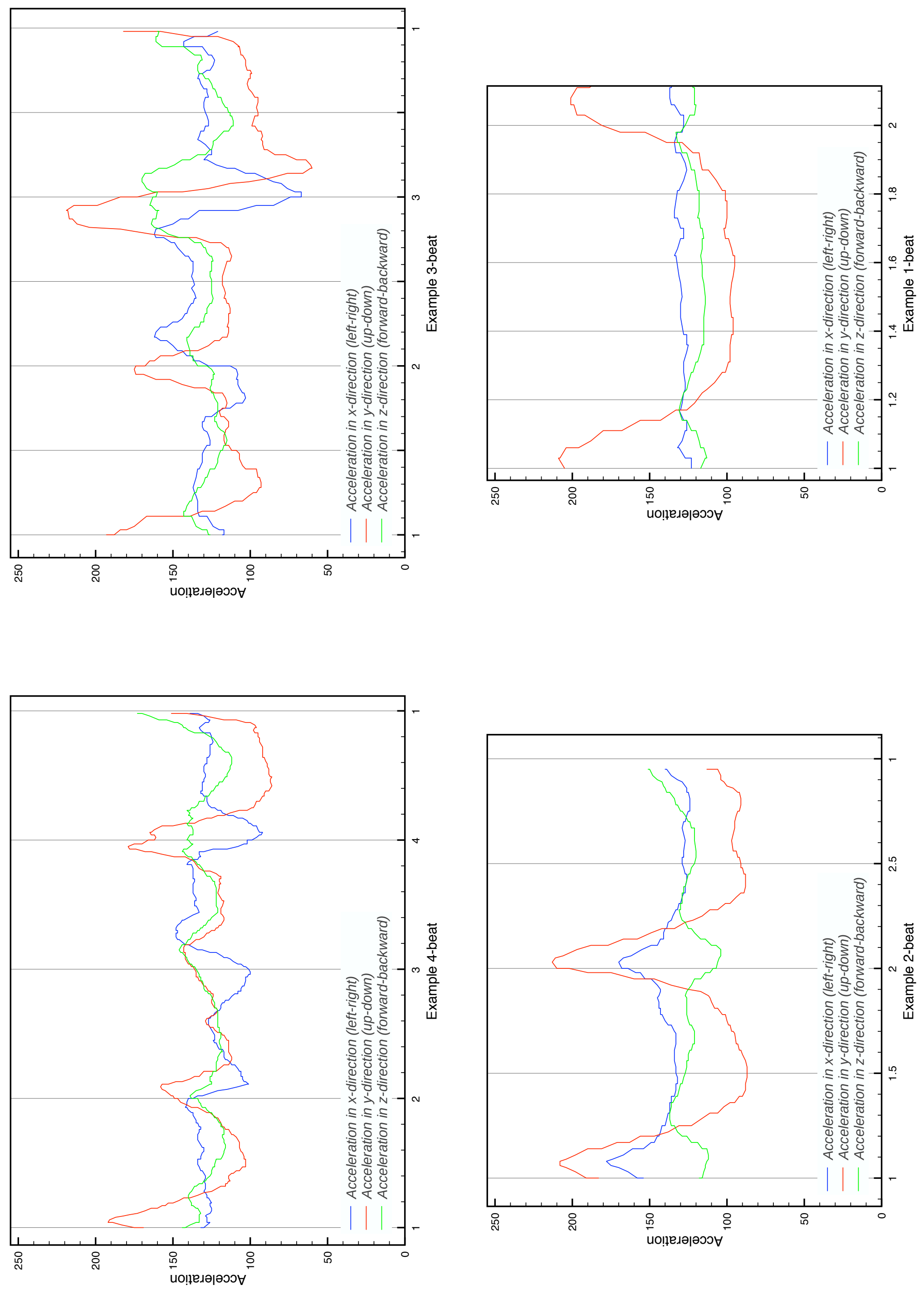


Figure 4.4: Example bars of the acceleration data transmitted by the Wii remote

The same criterion is observable at the third count, but here, at the maximum on the y-axis is a decreasing curve on the x-axis crossing the zero from a minimum to a global maximum. This phenomenon is due to the fast movement to the right before the second count, and the fast movement afterwards to the left until the third count (as explained in section 2.2.1 ***The 4-beat pattern*** and depicted in figure 2.2). The fourth count looks similar to the second count, whereas the minimum on the x-axis is not global in the bar. More features between the beats can be looked up in table A.2 or regarded in the average curves in figure 4.5 over all recordings.

4.2 Second user study

The more data the hidden Markov model learns, is similar to the data it shall recognize in the future, the better the recognition is. Therefore a second user study was led to gain conducting gestures similar to the application area. The conductor can influence the orchestra by changing the size and speed of the conducting gesture. Many users want to test the Personal Orchestra functions and tend to conduct gestures with varying tempo and speed. 21 test persons were asked to conduct with fast varying tempo and size of 1-beat to 4-beat gestures. They were also requested to start with a preparatory beat at the beginning of their conducting, to train this gesture, too. The beats were marked by the conductor by pressing a button on the Nintendo Nunchuk device, holding in the second hand. The second hand were used, because pressing a button on the Wii remote, could influence the acceleration sensors, which would be bad for the learning data.

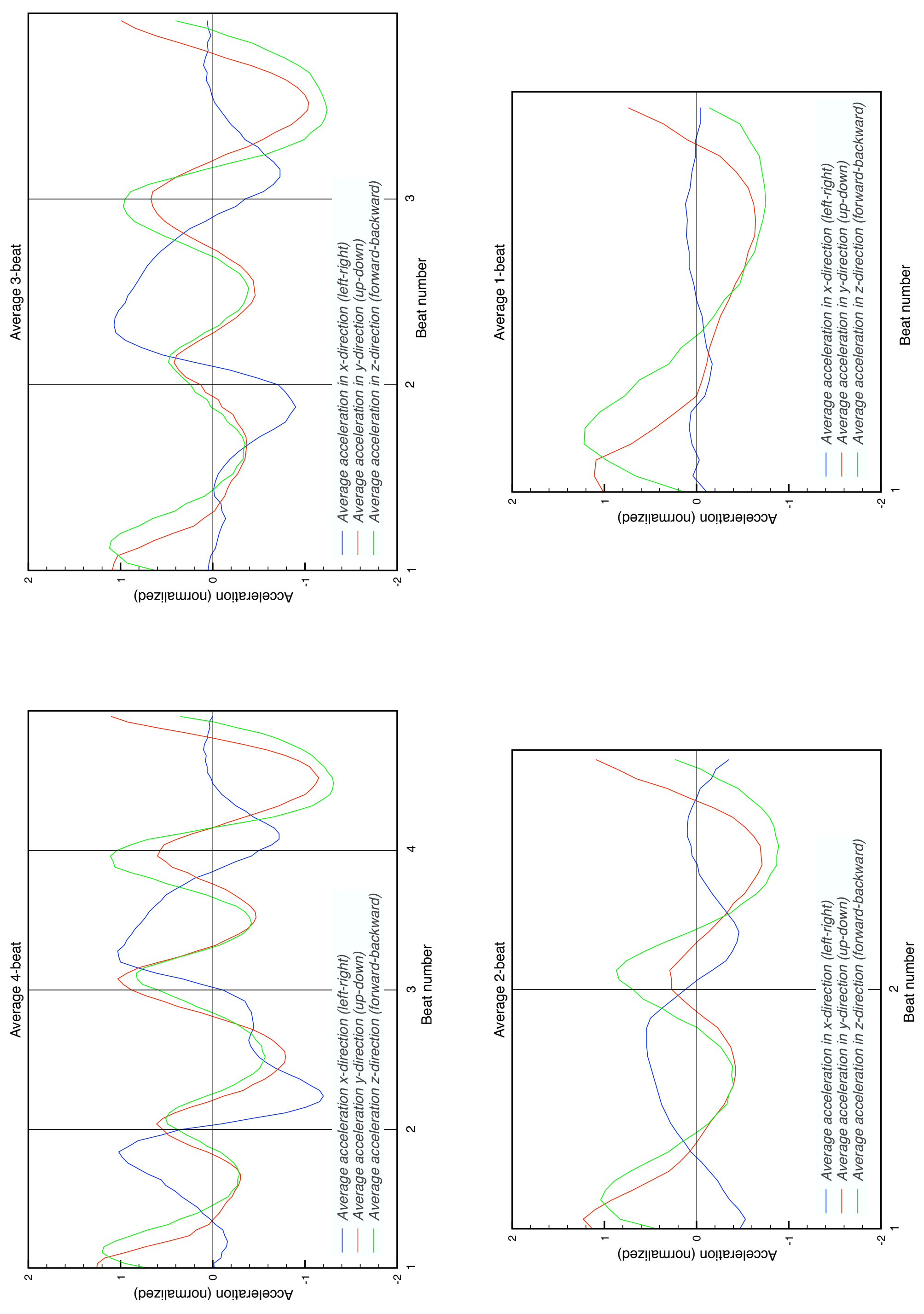


Figure 4.5: Average acceleration curves for all conducting gesture types

Chapter 5

Hidden Markov model and its application

"Causa latet, vis est notissima"
"The cause is hidden. The effect is visible to all."

— Ovid (43 BC - 17 AD), Roman poet

The recognition system consists of four subtasks, which are necessary to solve:

- signal analysis and preprocessing: normalization of observation values
- hidden Markov model structures: definition of models and structures
- training phase: learning the parameters for the model structures
- recognition phase: calculation the most likely sequence.

5.1 Preprocessing

To achieve better recognition results, the acceleration data must be normalized before learning and recognition phases. Here the data is normalized over mean (μ) and variance (σ^2) values. Means and variances are calculated within a time window W with the size of one loop, which is equivalent to a bar in musical meaning. The following equations show the mean and variance at time t with regard to the observed values o_n in the time window:

$$\mu = \frac{1}{W+1} \cdot \sum_{n=t-W}^{t} o_n \tag{5.1}$$

$$\sigma^2 = \frac{1}{W+1} \cdot \sum_{n=t-W}^{t} (o_n - \mu)^2 \tag{5.2}$$

The normalized acceleration curve at time t then is computed by the term:

$$\tilde{x}_t = \frac{o_t - \mu_t}{\sigma_t} \tag{5.3}$$

This normalization is done separately for all three dimensions. The result is a feature vector for each point in time. By taking into account the first derivative of the measured values, it becomes possible to include more features in the learning and recognition phase. So the feature vector is elongated to $x_t = [\tilde{x}_t, \tilde{y}_t, \tilde{z}_t, f'(\tilde{x}_t), f'(\tilde{y}_t), f'(\tilde{z}_t)]$. This means that our feature vector has the size $D = 6$.

5.2 Time alignment

After the three dimensions of the data are normalized, for the training we also need curves which are corresponding on the time axis. We recommend a normalized time base, represented by $S = 100$ states. This number was evaluated with the time constraints and our first order topology model as shown in figure 5.2. The recognition algorithm can recognize up to a speed of 240 beats per minute: the recognition rate is theoretically $100\ values/sec$ and depends on the transmission rate of the Nintendo Wii. One 4-beat bar has 100 $states$, which depends on the model ($\Rightarrow$ one 4-beat $bar \mathrel{\widehat{=}} 1\ sec$). Following the maximum conducting tempo which can be recognized is 240 $beats/minute$. Here, time alignment can help us to align the curves in a time-based context, for example moving the position of maxima or minima one onto the other (for an example, see figure 5.1). The values are non-linearly shifted and assigned to S states.

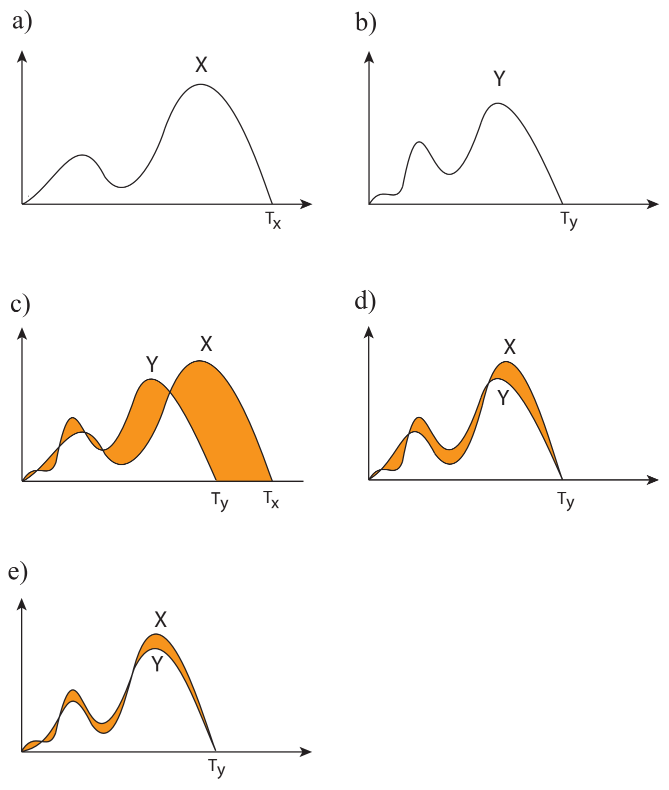


Figure 5.1: Example for time alignment
(a)(b) two different sequences X and Y
(c) comparison without time alignment
(d) linear time alignment
(e) non-linear time alignment

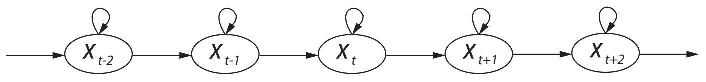


Figure 5.2: First order topology model: possibilities of searching a path through the states are: staying in the state or going to the next state

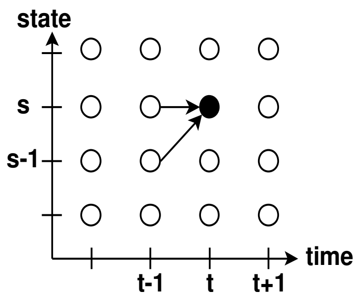


Figure 5.3: Part of the trellis of a first order topology model: possibilities of searching a path through the states are: staying in the state or going to the next state

The solution of time alignment must be found within the given model constraints (as introduced in chapter 2.3.3 ***Elements of an HMM***). In this case the first order topology is chosen which is represented in figure 5.3. We allow to change from state s to state $(s+1)$ mod S, or to stay in the state at the next time-step. So, time alignment can be regarded as an optimization problem: The sequence of states with the minimum overall distance from observation sequence to this sequence is searched for.

In the $initial$ time alignment a feature vector x_t is assigned to a state s by its position in time or — in our case — by the position in the conducting gesture. For these states the time alignment calculates new means and variances iteratively by ordering the acceleration values to the "nearest" state of the reference pattern from the time alignment before. The nearest state is defined by the minimal possible distance $dist(x_t, s)$ (see equation 5.4) to the measured values x_t.

To express the dependence of the distance of the assigned state s and the measured acceleration values x_t, the distance dimension is written as: $dist(x_t, s)$. In this case the squared Euclidean distance with variance weights is chosen, where D is the dimension of the feature vector $x_t = [x_1, \ldots, x_d, \ldots, x_D]$ at time t:

$$dist(x_t, s) = \sum_{d=1}^{D} \left(\frac{x_{td} - \mu_{sd}}{\sigma_{sd}} \right)^2 \tag{5.4}$$

The recursive equation

$$\tilde{Q}(t, s) = dist(x_t, s) + \min(\tilde{Q}(t-1, s), \tilde{Q}(t-1, (s-1) \bmod S)) \tag{5.5}$$

can be solved by dynamic programming (Bellman [2003]). It describes the minimum distance of the best sequence of hidden states s to x_t at time t by regarding the topology parameters. In this case with repeating bars the constraints are represented by an ascending circular sequence, where every state must be traversed in the correct order. The best path then can be found backwards by the optimal state at the actual time event as exemplified in figure 5.4. In our topology model this can be one of two possibilities: staying in state s or coming from state $(s-1) \bmod S$. The best predecessor can be found with the term:

$$v_t(s) = \arg\min(\tilde{Q}(t-1, s), \tilde{Q}(t-1, (s-1) \bmod S)) \tag{5.6}$$

As a by-product, time based information about the transitions of the states can be saved. From this data the speed of the conducting gesture can be approximated later, because $C_t(s, \kappa)$ contains for each state s the time t when the state was reached once before:

$$C_t(s, \kappa) = \begin{cases} t & s = \kappa \wedge v_t(s) = (\kappa - 1) \mod S \\ C_{t-1}(v_t(s), \kappa) & \text{else} \end{cases} \tag{5.7}$$

If the first case in this formula occurs, we are able to measure the time to the last occurence of the same state number: $\Delta t : t - C_{t-1}(v_t(s), \kappa)$.

5.2.1 Gaussian distribution

The observation for a state s of a gesture w can have statistical variations. Mostly these fluctuations are described by a Gaussian distribution, regarding a state s of a gesture w and the component x:

$$p(x|s) = N(x|\mu_s, \sigma_s^2) = \frac{1}{\sqrt{2\pi\sigma_s^2}} e^{-\frac{1}{2}\left(\frac{x-\mu_s}{\sigma_s}\right)^2} \tag{5.8}$$

The distribution of the multivariate feature vector $x_t = [x_1, \ldots, x_d, \ldots, x_D]$ is generated by multiplication of all dimensions $d = 1, \ldots, D$ with the assumption of statistical independence of the vector components:

$$\begin{aligned} p(x_t|s_t) &= \prod_{d=1}^{D} p(x_{td}|s_t) \\ &= \frac{1}{\prod\limits_{d=1}^{D} \sqrt{2\pi\sigma_{s_t d}^2}} e^{-\sum\limits_{d=1}^{D} \frac{1}{2}\left(\frac{x_{td}-\mu_{s_t d}}{\sigma_{s_t d}}\right)^2} \end{aligned} \tag{5.9}$$

To this formula, we apply the negative logarithm. The negative logarithm can be interpreted as the distance between a feature vector at time t and the model attributes for state s at time t, because $p(x_t|s_t)$ is positive and the logarithm is a monotonous function for positive values.

$$-\ log\, p(x_t|s_t) = \frac{1}{2}\sum_{d=1}^{D} \log\left(2\pi\sigma_{s_t d}^2\right) + \frac{1}{2}\sum_{d=1}^{D}\left(\frac{x_{td} - \mu_{s_t d}}{\sigma_{s_t d}}\right)^2 \tag{5.10}$$

5.3 Training phase

In the training phase for every word (= 4-beat gesture) the following values are determined:

- state count S
- prototype vector $\mu_s \in \mathbb{R}$
- variance vector σ_s^2

The initial time alignment supplies the state sequence s_1^T, by assigning the observation sequence to the states sequence of the model, so that the distances between observation and the model are minimized for the whole sequence:

$$\arg\min \sum_{t=1}^{T} \log p(x_t, s_t)$$

with model parameters $\theta = \{\mu_s, \sigma_s^2\}$.

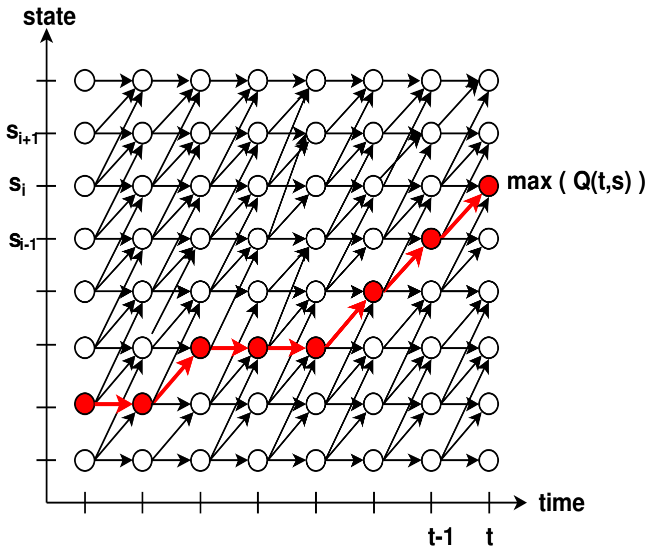


Figure 5.4: Processing of $Q(t, s)$: After a forward pass, maximum of $Q(t, s)$ is estimated at the last time-point t and the best path is chosen backwards

With this knowledge a new dynamic programming equation can be received with this new probability distance at each point of time t with the assumption of state transition probabilities:

$$\begin{aligned} Q(t,s) &= \max_{s_1^t : s_t = s} p(x_1^t, s_1^t) \\ &= p(x_t, s_t) \\ &\quad + \max\left(Q(t-1,s), Q(t-1,(s-1) \mod S)\right) \end{aligned} \tag{5.11}$$

Each iteration consists of two alternating operations:

- **Non-linear time alignment:** this is performed by dynamic programming and provides a new state labeling of the training data, e.g., a one-to-one correspondence between observation vector and state index.
- **Maximum likelihood parameter re-estimation:** using this labeling, the parameters are updated by means of a decision-directed estimation technique.

5.3.1 Mixture densities and cluster analysis

To maintain the diversity (staccato, legato,...) of several conducting gestures, they are not represented in one mean and variance value, but split up into more than one if necessary. So we do not guess that the values in one state are Gaussian distributed, but they can be better split up as mixture densities such as in figure 5.5.

For one state s with L_s mixture densities can be written:

$$c_{sl} = p(l|s) \geq 0, \sum_{l}^{L_s} c_{sl} = 1. \tag{5.12}$$

So these unknown parameters c_{sl} and θ_l of all L_s single densities can be combined under the notation λ:

$$\lambda = \{\theta, c_{sl}\} = \{\mu_{sld}, \sigma_{sd}^2, c_{sl}\} \tag{5.13}$$

For each state s we count the number of values in each density is assigned to

$$N_{sl} = \sum_{t=s_t=s, l_t=l} 1, \tag{5.14}$$

given the best state sequence

$$s_t : s_1^T = \arg\max_{\sigma_1^T} p(x_1^T, \sigma_1^T|\lambda) \tag{5.15}$$

and the mixture density at time t

$$l_t = \arg\max_{l} c_{s_t l} \cdot \mathrm{N}(x_t|\mu_{s_t l}, \sigma_s^2). \tag{5.16}$$

For mixture densities we must also calculate parameters as follows:

$$\mu_{s_t l} = \frac{1}{N_{sl}} \cdot \sum_{t=s_t=s, l_t=l} x_t. \tag{5.17}$$

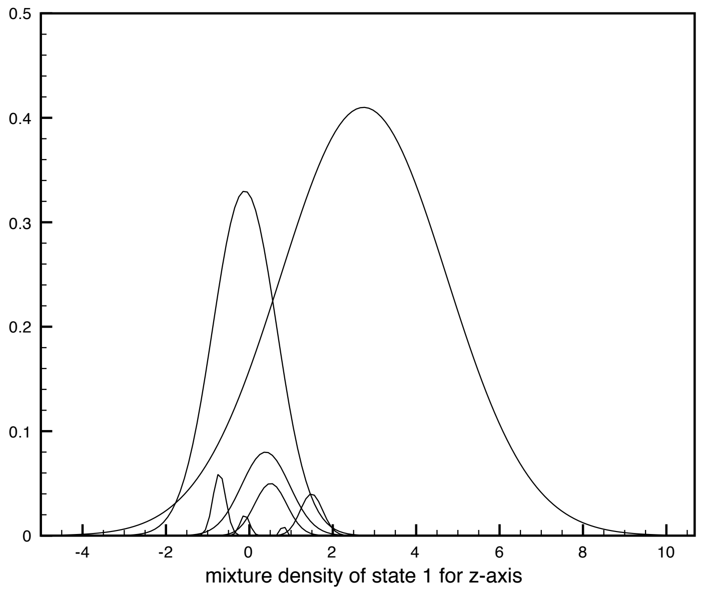


Figure 5.5: Example for a mixture distribution with eight densities

From now on the pooled variance is used to avoid too small variance values, which would influence the learning and recognition phase negatively:

$$\sigma^2 = \frac{1}{N} \cdot \sum_t (x_t - \mu_{s_t, l_t})^2. \tag{5.18}$$

5.3.2 Expectation-maximization algorithm for mixture densities

For finding the maximum likelihood for mixture densities we examine the model $p(x_t|\lambda)$. We start with initial values for λ, estimates by the initial time alignment.

Algorithm: *expectation-maximization*

```
EXPECTATION-MAXIMIZATION:
Iterations:

  1. Expectation step: expectation over
     the hidden variables: calculate
     $R(\lambda, \tilde{\lambda})$

  2. Maximization step: maximization over
     $\tilde{\lambda}$: find $\arg\max_{\tilde{\lambda}}\{R(\lambda, \tilde{\lambda})\}$

next iteration with $\lambda := \tilde{\lambda}$
```

The maximum likelihood function is defined $R(\lambda, \tilde{\lambda})$ by

$$
\begin{aligned}
R(\lambda, \tilde{\lambda}) &= \sum_{s_1^T} p(s_1^T|x_1^T, \lambda) && \cdot \log p(x_1^T, s_1^T|\tilde{\lambda}) \\
&= \frac{1}{p(x_1^T|\lambda)} \sum_{s_1{}^T} p(x_1^T, s_1^T|\lambda) && \cdot \log p(x_1^T, s_1^T|\tilde{\lambda}) \\
&= \frac{1}{p(x_1^T|\lambda)} \sum_{s_1{}^T} \prod_{t=1}^{T} p(x_1^T|s_1^T, \lambda) && \cdot \sum_{\tau=1}^{T} \log p(x_1^T|s_1^T, \tilde{\lambda}) \\
&= \sum_{t=1}^{T} \sum_{s} \gamma_t(s|x_1^T, \lambda) && \cdot \log p(x_t|s, \tilde{\lambda}) \\
&= \sum_{t=1}^{T} \log p(x_t|s_t, \tilde{\lambda})
\end{aligned}
\tag{5.19}
$$

with s_1^T as defined in equation 5.15. $R(\lambda, \tilde{\lambda})$ is a measure of how good the assigned states and their definitions fit to the training sequence. So, the expectation-maximization algorithm can be iterated until $R(\lambda, \tilde{\lambda})$ converges.

For S classes, the training can be done for each class s separately. With S classes and mixture densities our maximum approximation model can be defined by:

$$p(x|s) = \max_{l}(c_{ls} \cdot p(x|l, \theta)) \tag{5.20}$$

Expectation step

In the expectation step we use the Viterbi algorithm of the hidden Markov model to asses the most probable state sequence s_1^T for the observation sequence x_1^T

$$p(x_1^T, s_1^T) = \prod_{t=1}^{T} p(x_t, s_t | x_t^{t-1}, s_1^{t-1}) \tag{5.21}$$

Application of the first order Markov equation:

$$p(x_1^T, s_1^T) = \prod_{t=1}^{T} p(x_t | s_t) \cdot p(s_t | s_{t-1}) \tag{5.22}$$

To solve this formula dynamic programming can be used. We use the help function $Q(t, s)$ (see equation 5.11) to calculate the best path of hidden states s_1^T with a given observation path x_1^T. Here we must expand the above formula (see 5.9) without mixture densities by adding the mixture weight $c_{s_t l}$:

$$\begin{aligned} p(x_t|s_t) &= \max_l p(x_t, l|\lambda) \\ &= \max_l p(l, \lambda) \cdot p(x_t|l, \lambda) \\ &= \max_l c_{s_t l} \cdot p(x_t|l, \theta) \\ &= \max_l c_{s_t l} \cdot N(x_t|\mu_{s_t l}, \sigma^2). \end{aligned} \tag{5.23}$$

Maximization step

The maximization step should find a maximization of parameters λ (mean μ, variance σ^2, and mixture weights c_{ls}) while using the most probable state sequence from the expectation step. First this method works as updating function which calculates mean and variance for every state. After the splitting step the training method maximizes equation 5.23 for each state (shown in section 6.2.1 ***Maximum approximation***). For each observation value x_n the best single density is chosen in the aligned path and counted. So, each single density is assigned with its own attributes, calculated as our mean and variance for the single Gaussian distribution before.

Splitting step

To improve of our results, we double the parameters before each expectation-maximization iteration by splitting the parameters of the distribution. For splitting the Gaussian distribution in every step in two mixture densities we add and subtract the factor

$$\epsilon = 0,1 \cdot \sqrt{(\sigma_{ls}^2)}$$

on the calculated mean of every state.

$$\tilde{\mu}_{(2 \cdot l)s} = \mu_{ls} + \epsilon, \quad \tilde{\mu}_{(2 \cdot l+1)s} = \mu_{ls} - \epsilon$$

The variance and mixture weight is adapted from the calculation before

$$\tilde{\sigma}_{(2 \cdot l)s} = \sigma_{ls}, \quad \tilde{\sigma}_{(2 \cdot l+1)s} = \sigma_{ls}$$

$$\tilde{c}_{(2 \cdot l)s} = c_{ls}, \quad \tilde{c}_{(2 \cdot l+1)s} = c_{ls}$$

and does not change in the maximum approximation.

5.4 Recognition phase

Changes in Viterbi Algorithm are required: no backtrace for online estimation of most probable state

For the recognition algorithm, some changes must be undertaken, so that real-time recognition becomes possible. Normally, the procedure of the time alignment can be copied from the training phase to the recognition phase one by one, containing the Viterbi algorithm. This algorithm can only recognize which gesture it is, when it is performed entirely. So the gesture must be completed just before, so that the backtracking part of this algorithm can optimize the path of states through the gesture. Though we want an online estimation of the most probable state. In the forward pass, the model constraints are just considered, we can be sure to find a good path, but not the best global one. That means, we get a worse recognition rate, but an online algorithm.

Also recognizing percental progress inside a gesture is desired, so we added a percentaged measure of times, which the recognition algorithm gets from the training phase. Consequently, the recognition phase can transfer the recognized state numbers to the percental process in a bar along with the time points of the beats in a bar. We also add some constraints to the algorithm to adapt to the specialties of Personal Orchestra. So we do not allow to jump to a slower path, which would mean that the orchestra would also has to jump back in its performance. This would not be a natural behavior and irritates the interacting conductor. In this case we let the orchestra "wait" for the conductor.

5.4.1 Pruning

In the recognition phase, big jumps in time are not desirable by the orchestra, because this will cause unreal behavior. Thus, we only allow to recognize the position in the conducting gesture in a window around the last recognized state (pruning).

Hence, the maximum $Q(t, s)$ (see equation 5.11) is just only estimated in this pruned window. The maximum value of $Q(t, s)$ is chosen as end point of the best path (red arrows). Figure 5.6 shows that not only one path is followed. If the maximum of $Q(t, s)$ is at the end of another path through the trellis (blue), the recognition can decide to change on this path if it is in the pruning window (dark circles). If it changes to a "slower" path, the orchestra has to wait until it arrives again the last maximum state.

The size of the pruning window has to be evaluated empirically. Results show that is should have the size between $\frac{1}{2}$ bar and $\frac{4}{5}$ bar around the last found state (for results see chapter 7 ***Evaluation***), so that no bar can be jumped over.

5.4.2 Smoothing function

Another possibility to find the best path in the pruning window is to search for a mean value of all $Q(t, s)$ inside the pruning window. With this method the decisions are smoothed, which state to assign to the observation values.

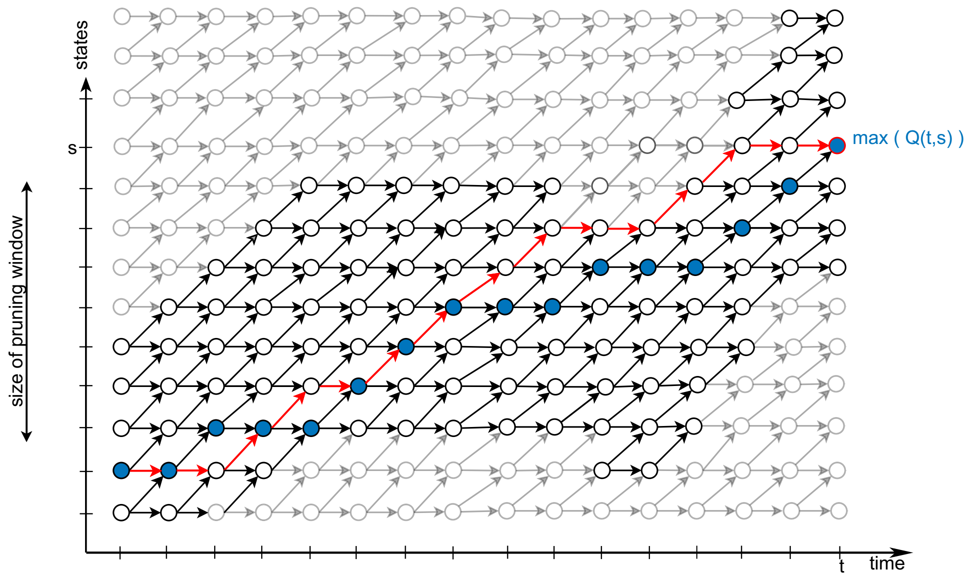


Figure 5.6: Trellis with pruned window (dark circles), the decision in the pruning window at every time point for the best state (blue), and the best path at time point t (red) estimated by backtrace

So the decision is not focussed to the maximum value, and gives a second maximum the chance to influence the decision in its direction. We assume, $\hat{s}_t$ is the state which maximizes $Q(t,s)$:

$$\hat{s}_t = \arg\max_s Q(t,s).$$

Then our decision is to assign state `round(`$\bar{s}_t$`)` to the observation at time t, the mean state number.

$$\bar{s}_t = \sum_{s=min_s}^{max_s} \frac{\left(e^{Q(t,s)-Q(t,\hat{s})}\right) \cdot s}{\sum\limits_{s=min_s}^{max_s} \left(e^{Q(t,s)-Q(t,\hat{s})}\right)} \tag{5.24}$$

Chapter 6

Implementation details

"In general, any form of exercise, if pursued continuously, will help train us in perseverance. Long-distance running is particularly good training in perseverance."

—Mao Tse-Tung (1893-1976), Chinese stateman

Personal Orchestra is implemented in Objective-C, an object-oriented programming language for developing on Mac OS X systems. To integrate the gesture recognition plug-in best in Personal Orchestra, the whole work is implemented in three separate projects:

1. Acquirement of acceleration data of the conducting gestures,
2. Training phase for learning the state attributes from the training data,
3. Recognition phase for recognition of gesture and time-point in the gesture.

6.1 Acquirement of conducting gestures

The acceleration data of the Nintendo Wii remote is processed and additional data such as the type of bar and position in gesture are added. For recording the conducting data, two versions are designed to store:

- static gestures [V 1.0]
- variable gestures [V 2.0]

In version 1.0 a certain tempo is given by a metronome and has to be followed by the conductor; the gesture shape (neutral, legato, staccato) and tempo (76 bpm, 100 bpm, 132 bpm) can be chosen and is written in the filename.

In version 2.0 the tempo is defined by the conductor by pressing a button on the Nintendo Wii Nunchuk at each beat number. Variable speed and size of conducting gestures are recommended.

Both versions have a similar output file. A comma-seperated file, where the acceleration data (0-255) in three dimensions at every time-point, and the position in every bar is written. In version 2.0 only full counts can be written, at the time when the button is pressed. Afterwards, a more detailed position inside a bar (linear speed between two appointed beat counts) can be calculated.

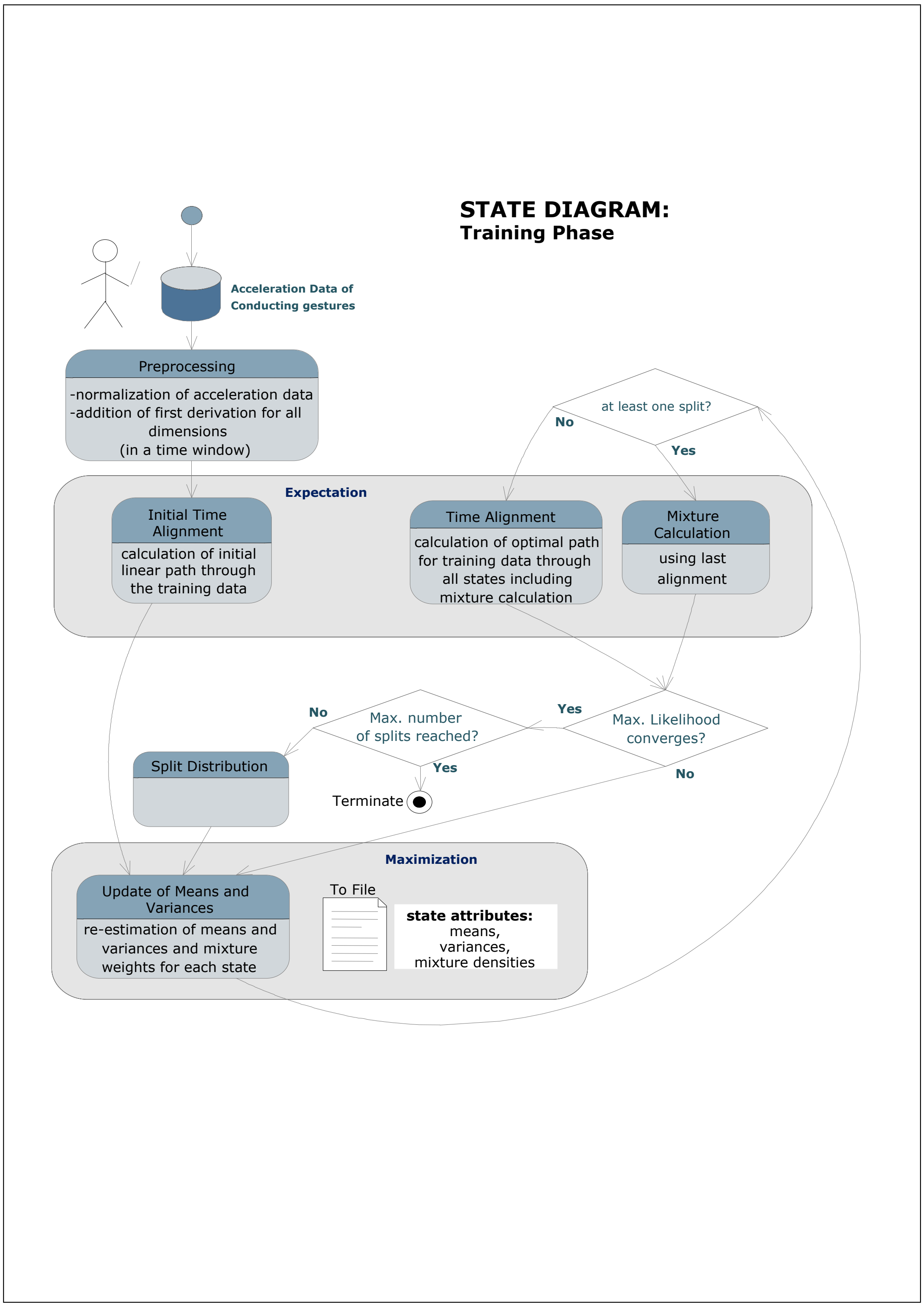

STATE DIAGRAM:
Training Phase
Acceleration Data of Conducting gestures
Preprocessing
-normalization of acceleration data
-addition of first derivation for all dimensions
(in a time window)
Expectation
Initial Time Alignment
calculation of initial linear path through the training data
Time Alignment
calculation of optimal path for training data through all states including mixture calculation
Mixture Calculation
using last alignment
at least one split?
No
Yes
Max. number of splits reached?
No
Yes
Max. Likelihood converges?
Yes
No
Split Distribution
Terminate
Maximization
Update of Means and Variances
re-estimation of means and variances and mixture weights for each state
To File
state attributes:
means,
variances,
mixture densities

Figure 6.1: State diagram of the implementation of the training phase

6.2 Implementation of the training phase

In this section, the flow of processes and their implementation details in the training phase are presented. In the preprocessing step the acceleration values on the left to right axis of left-handed persons are mirrored. After normalizing the conducting data as described in chapter 5.1 ***Preprocessing***, the first derivation for each dimension is added and the initial time alignment is calculated. Therefore, the data from version 2.0 of the recording program, must be specified in such way, that the time between each beat is estimated linearly. The 4-beat gesture patterns are trained with 100 initial states, 3-beats with 75 initial states. Analog, the 2-beats are learned with 50 states, and 1-beats with 25 states. So every beat has 25 states, which can be compared between the four models.

6.2.1 Maximum approximation

The initial time alignment assigns every point of time linearly to its corresponding state. With this assignment initial attributes for each state are defined. With these attributes the iteration of expectation-maximization is started with all training data of one kind (for example all training bars of 4-beat gestures). Different kinds shall be differed later in the recognition with these trained and differing models. The expectation part is done by dynamic programming (Bellman [2003]) of time alignment. After the split distribution, the time alignment and five times mixture calculation are alternated to save time. In the mixture calculation, the last estimated path from time alignment is used to assign better densities to it. For further details see the state diagram 6.1 for training phase. Now, the states and their attributes are assigned non-linearly, dependent on the most probable path through the training data. The new mapping of states to observation values is saved and the attributes for each state can be re-estimated. The maximization step is accomplished by re-estimating means, variances, and mixture densities for the most probable path calculated in the expectation-step just before. The training method for maximizing

$$p(x) = \max_l \left(c_l \cdot p(x|l, \mu_l, \sigma^2)\right)$$

for each state is shown in the following algorithm:

Algorithm: *Maximum Approximation*

```
MAXIMUM APPROXIMATION:
For each state Loop {
    determine the best single density for
      each observation $x_n$:
            $\arg\max_l(c_l \cdot p(x_n|l, \mu_l, \sigma^2))$
    The observation $x_n$ only contributes
      to the training of one single
      density, namely the best one.

    Parameter estimation for
    $c_l$:  relative frequency of choosing $l$ as
      the best single density
    $\mu_l$:  mean of the observations $x_n$
      assigned to each single density $l$
    $\sigma^2$:  variance of the observations $x_n$
      assigned to each single density $l$
}
```

In this phase we also estimate the average state length, which is needed in the recognition, by counting the assigned values to each state. The relative length of a state can be defined by:

$$len(s) = \frac{|\text{observation values assigned to state } s|}{|\text{all existing observation values}|} \tag{6.1}$$

The maximum likelihood function as defined in equation 5.19 gives a measure of how well the state attributes and their assignment to the training data are, and can notify convergence of the iterating procedure. If this likelihood value converges, we split each state into a mixture density distribution with double count of parameters for each state (as described in section 5.3.2) and begin again with time alignment. The whole training phase is shown in the following algorithm:

Algorithm:
Workflow in training phase

WORKFLOW IN TRAINING PHASE:

```
The last version of learning steps is
implemented as follows:

0: initial time alignment

1 - 6: time alignment

7,9,11: split parameters, 5 * mixture
    calculation, time alignment, 5 *
    mixture calculation

8,10,12-15: time alignment, 5 * mixture
    calculation
```

6.2.2 Numerical robustness

For numerical robustness we avoided the calculation of all exponential functions, and used the logarithm function instead. Consequently, we did no calculation by maximizing probabilities, but by minimizing their distances (logarithm of probability). Calculation time can be saved by storing pieces of the calculation for later iterations. Details about how to compute the logarithm of the model presented in equation 5.23 at each time point are shown in the following:

$$\begin{aligned} D(x|s) &:= \log p(x|s) = \max_l c_{sl} \cdot N(x|\mu_{sl}, \sigma_{sl}^2) \\ &= \max_l \log c_{sl} \cdot N(x|\mu_{sl}, \sigma_{sl}^2) \\ &= \max_l \left(\log c_{sl} - \tfrac{1}{2} \cdot [\log(2 \cdot \pi) + \sum_{d=1}^{D} \log(\sigma_{sld}^2)] \right. \\ &\quad \left. - \sum_{d=1}^{D} \frac{x_d - \mu_{sld}^2}{\sigma_{sld}^2} \right) \\ &= \max_l \left(\tilde{w}_{sl} - \sum_{d=1}^{D} \frac{x_d - \mu_{sld}^2}{\sigma_{sld}^2} \right) \end{aligned} \tag{6.2}$$

whereas $\tilde{w}_{sl}$ can be pre-computed and saved for later iterations.

$$\tilde{w}_{sl} := \log c_{sl} - \frac{1}{2} \cdot \left(\log(2 \cdot \pi) + \sum_{d=1}^{D} \log(\sigma_{sld}^2) \right) \tag{6.3}$$

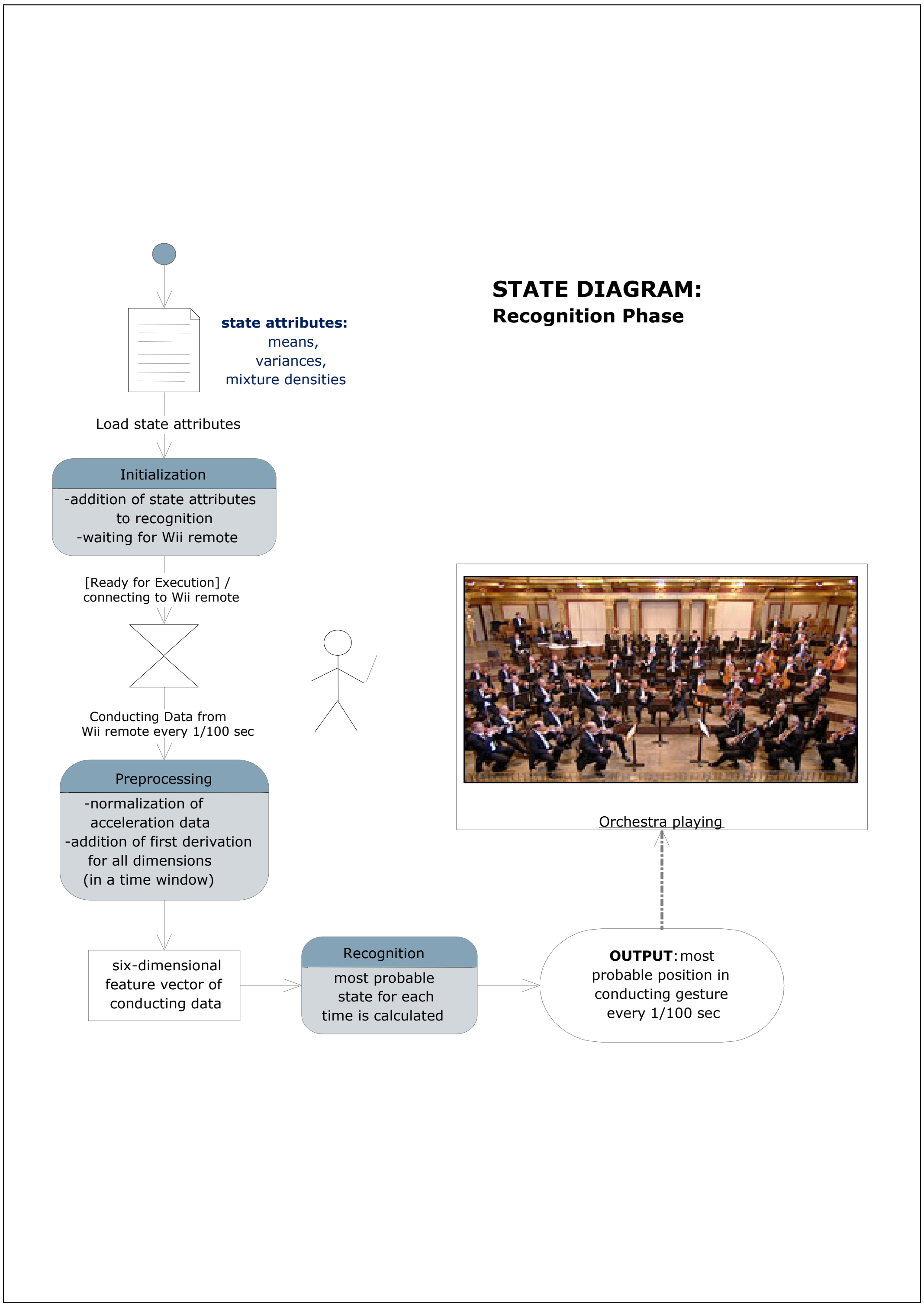

STATE DIAGRAM:
Recognition Phase
state attributes:
means,
variances,
mixture densities
Load state attributes
Initialization
-addition of state attributes to recognition
-waiting for Wii remote
[Ready for Execution] / connecting to Wii remote
Conducting Data from Wii remote every 1/100 sec
Preprocessing
-normalization of acceleration data
-addition of first derivation for all dimensions (in a time window)
six-dimensional feature vector of conducting data
Recognition
most probable state for each time is calculated
OUTPUT: most probable position in conducting gesture every 1/100 sec
Orchestra playing

Figure 6.2: State diagram of the implementation of the recognition phase

6.3 Implementation of the recognition phase

In this section the special features of the recognition phase are described. In the recognition phase, the same assumptions as in the training phase must be worked on.

First, the learned state attributes (means, variances, mixture densities, state length) are loaded into the program, which achieved the highest maximum likelihood in the training phase. The procedure waits for the Nintendo Wii remote to connect for sending the acceleration data to the framework every $1/100$ seconds. The conducting data is then preprocessed as in the training phase: normalization of acceleration values in a run-time window with size of 100 values and addition of first derivative for all dimensions.

In the forward pass with pruning window, the path can be changed, if another path becomes better, though it is a path which were outside the pruning window before time point t

Every $1/100$ seconds, the program must decide, which state of each model is most probable. This is the reason why the time alignment procedure from the training phase cannot be used without changes. The whole backward pass of the Viterbi algorithm had to be omitted and the forward pass had to be adjusted. Now the most probable value has to be estimated in the forward pass, so a path is searched through the trellis, which ends with the maximum value for $Q(t, s)$ (see equation 5.11) at the actual time point t. Therefore, a path can be changed, if it seems to be better at time point t. To restrict the number of possible paths and to decrease the complexity of calculations, we establish pruning (as described in 5.4.1 ***Pruning***) to compute only the most probable states in a given window size around the last detected state. For the first 100 values, all states can be assigned before the window is scaled down. The pruning window is only important at the decision time-point t. In the recognition, a path can be chosen, which ends in the window at time point t. All states which are not inside the pruned window, are not calculated to reduce latency of the recognition. Additionally with the best window size we want to maximize the quality of decisions for the most probable state. This is why we evaluated the best size for the pruning window empirically in chapter 7 ***Evaluation***.

The pruning window is calculated as described in detail in this pseudo-algorithm:

CALCULATE PRUNING WINDOW:

```
//just look at values for a window of
//   maxwindow-size around laststate
int $\min_s$, $\max_s$;
//mark the borders in which $Q(t,s)$
//   will be calculated
$\min_s$ = laststate - maxwindow;
if($\min_s < 0$) $\min_s = 0$;
$\max_s$ = laststate + maxwindow;

while($\max_s - \min_s$ >= maxwindow) {
   if($Q(t,\max_s) > Q(t,\min_s)$) {
      $\min_s$++;
   } else {
      $\max_s$--;
   }
}
```

Algorithm: *Calculate pruning window*

Additionally, some restrictions are inserted, which shall avoid unrealistic behavior of the orchestra. An orchestra never would jump back in time; when a tone was just played, it would not be repeated if the musician was too fast. We allow to pass on slower paths through the trellis, but we wait for them at the last recognized state, and do not jump to a state smaller than the last recognized.

Unrealistic behavior of the orchestra is avoided, because it is not allowed to jump back in time

The last version of Personal Orchestra includes some specific extra features, which had to be reinvented and reimplemented for the usage with acceleration data:

Dynamics The virtual orchestra was able to adapt conducted dynamics, big gestures made the orchestra playing loud, minimal gestures made the orchestra to play more quite. We now recognize the size of the gesture by the conductor applied on the acceleration sensors in the last window of one second. Here, we could re-use the standard deviation, which we need as normalization factor, anyway.

Pause mode Personal Orchestra pauses if the conductor stops its movement. The musicians can go on at the stopped position inside the bar. For this function, we keep in mind the last five acceleration values in each dimension of the feature vector and calculates its average difference of acceleration in all dimensions. If the difference is smaller than the given threshold, the pause mode gets active. The orchestra waits for the gesture being continued at this position of the bar.

Wiggle mode The "Wiggle mode" for children allows arbitrary movements to let the orchestra play. We solved that by defining a second threshold for average acceleration. If the conductor conducts over this threshold, this means with force, the Wiggle mode gets active and the orchestra plays in its recorded speed.

1-beat mode We maintained the 1-beat mode for lay-conductors as it was known in the positional based versions before. It is meant, that every kind of bar-type can also be conducted with 1-beat bars. This mode can be chosen by hand, can be run in parallel, but also works with the other modes, because of similarities in the y- and z-axes of the patterns.

6.3.1 Important data structures

Working with an object-oriented language, it is usual to manage data in data structures. In this case, some data structures are created, which are used in training and recognition phase.

- format for acceleration data in three dimensions (normalized), derived acceleration values in three dimensions, assigned state numbers from time alignment, and time points of beats.
- format for calculated attributes for each state (mean, variance, pooled variance, mixture weight, state length). This data structure is saved in data files in the training phase, which can be loaded again in the recognition phase.

Chapter 7

Evaluation

" To achieve great things, two things are needed; a plan, and not quite enough time. "

— Leonard Bernstein (1918-1990), American Conductor, Composer and Pianist

7.1 Evaluation of training phase

The training is done with half of the conducting data, and the recognition is evaluated with the other half of recorded conducting gestures, respectively. The recognition results are compared with the recorded reference beats and with the reference beats assigned in the training phase.

A measure how good the time alignment works is the maximum likelihood function which is calculated after each time alignment. After the fifteenth time alignment we decided to stop the training phase with eight mixture densities for each state. The maximum likelihood converges as figured out in 7.3 for all conducting data. After learning the first half and the second half, the maximum likelihood calclates as described in the formula 5.19 behaved like shown in figures 7.1 rather 7.2.

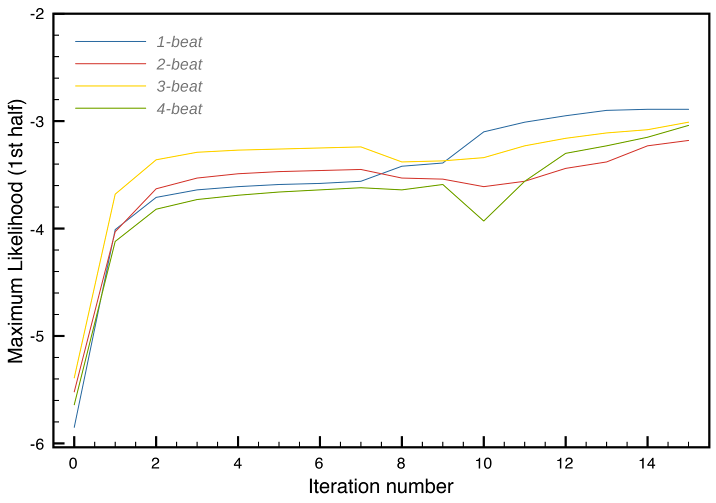


Figure 7.1: Maximum likelihood in training phase after each time alignment with first half of data (for iteration numbers see second box in 6.2.1 ***Maximum approximation***)

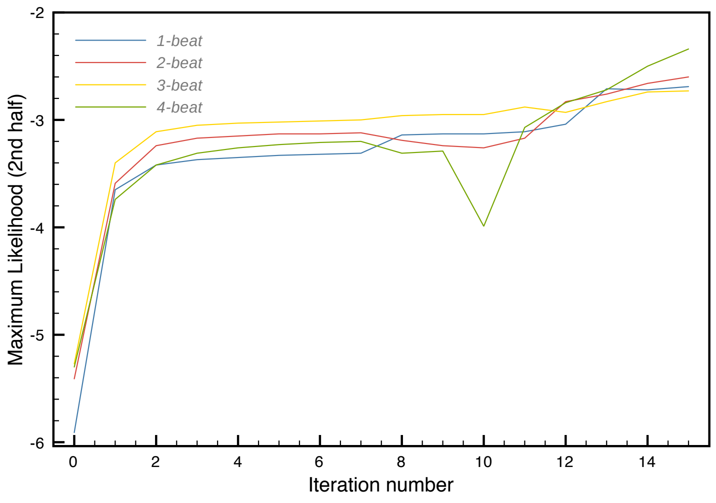


Figure 7.2: Maximum likelihood in training phase after each time alignment with second half of data

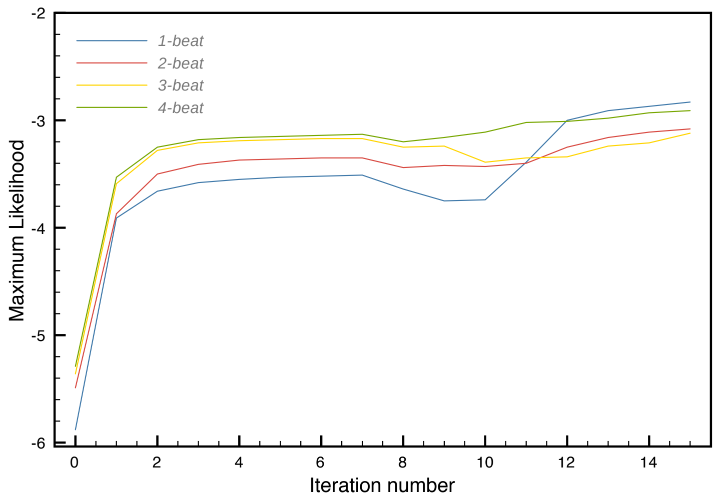


Figure 7.3: Maximum likelihood in training phase after each time alignment with whole data

7.1.1 Where to recognize the beats?

Time alignment re-orderes recorded time points. This is the case, why the recorded beats cannot be taken as reference in the recognition phase. As solution we counted the number of assigned values in each state and interpret it as a progress in one bar (see equation 6.1). Subsequent in a 4-beat, we found in the output files the beat one assigned at 0% ($\widehat{=}$ state 0), beat two at 25% ($\widehat{=}$ state 24), beat three at 50% ($\widehat{=}$ state 50), and beat four at 75% ($\widehat{=}$ state 75) (compare with figure 7.4). The recognition phase adapts these learned values as the progress in a bar.

For demonstration we marked the trained beats in figures in Appendix B.1 to B.4, where the assigned mean curve and its calculated standard deviation are shown after six iterations of time alignment. It is remarkable that the beats are assigned in the maxima of the y-axis (up-down) acceleration curve, which was before stated by Usa and Mochida [1998]. On the first sight this is contrasting to the thesis of Rudolf [1995]. He teaches the conductors, the beat is positioned at the lower turning point, which is after the maximum of acceleration. However, we found in the user stud-

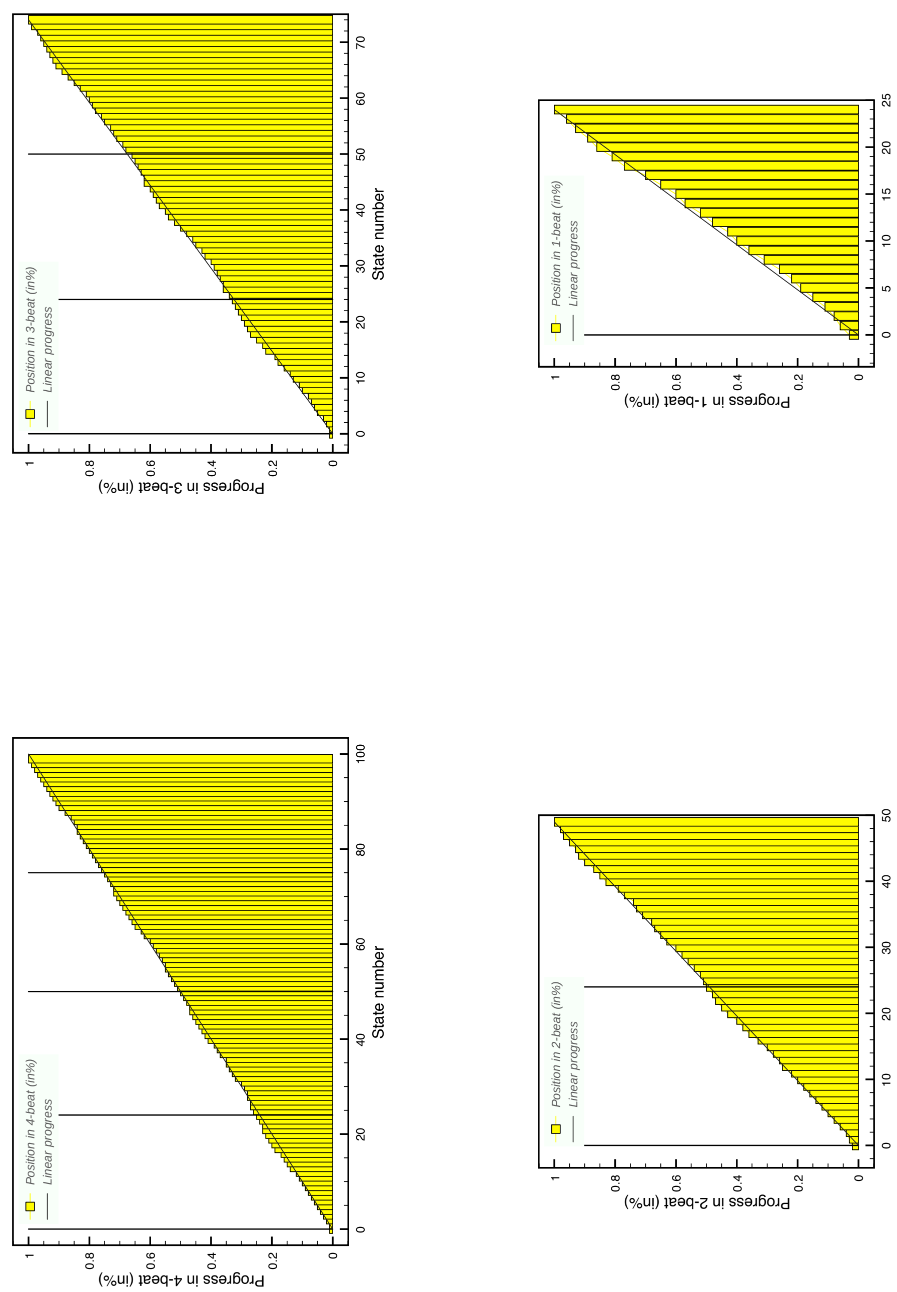


Figure 7.4: Assigned length for each state versus linear process in a 4-beat bar (100 states), 3-beat bar (75 states), 2-beat bar (50 states), and 1-beat bar (25 states)

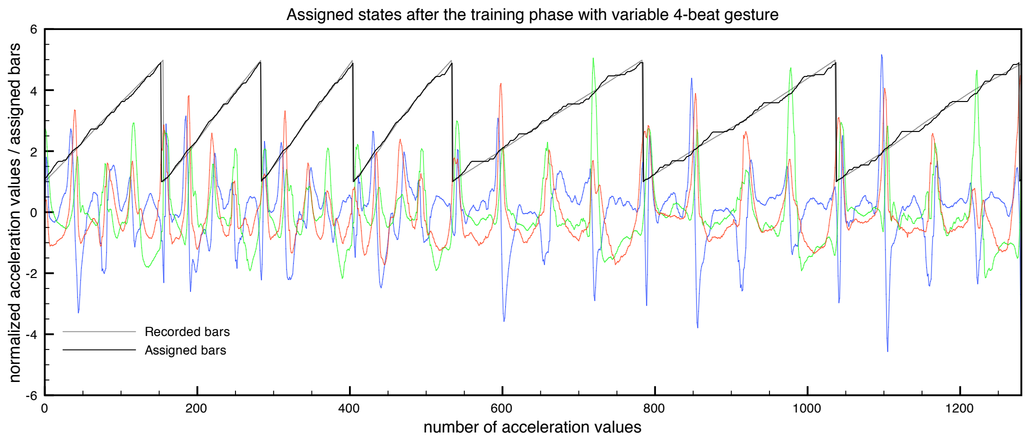


Figure 7.5: Variable 4-beat gesture after 6 iterations of time alignment in the training phase

ies and the average curves, that the beat is *felt* at the point which needs the biggest force. This is the position shortly before the lower turning point of the gesture at the position of maximum acceleration.

Most of the variable bars were assigned correctly such as the un-variable bars (as shown in figure 7.5). All attributes for four different kinds of beats were trained separately and printed after the sixth iteration time alignment (see Appendix B). The process of bars seem to be assigned correctly, also when the speed changes. The assigned beats are very close to the recorded beats. The whole process in the bars and count one is nearly on the recorded ones.

7.2 Evaluation of recognition phase

In this section, the recognition phase and its adjustable parameter — the size of pruning window — is evaluated. Also few recognition problems and the speed of the algorithm are discussed.

7.2.1 Size of pruning window

As mentioned in chapter 5.4.1 ***Pruning***, the size of the pruning window is co-responsible for the quality and speed of recognition. If the pruning window is chosen too big, whole bars can be missed (see curve in 7.6c), if the window is chosen too small. In fast tempi, the recognition is not able to follow the conductor (see figure 7.6a). That is why we suppose, the pruning window must have the size of less than one bar.

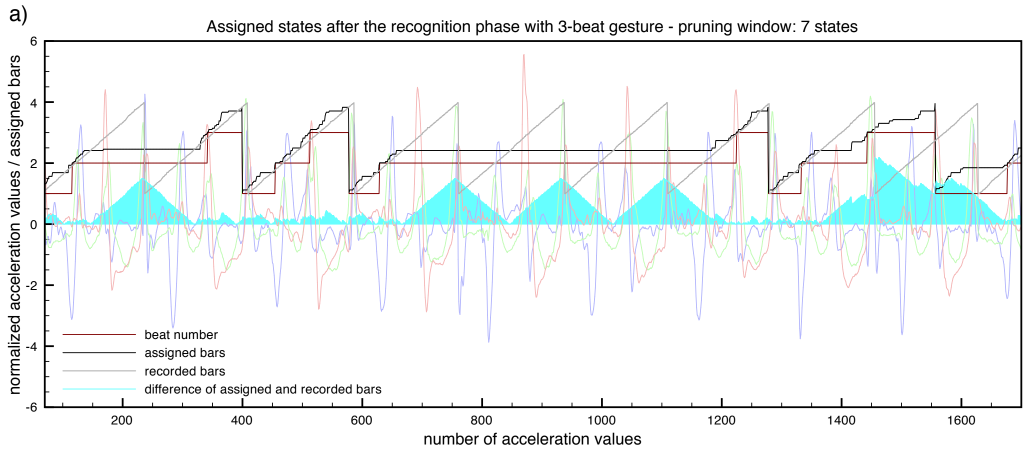
As we can see in figure 7.6a, big differences arise when the correct path of states was lost. Also the distances between the full counts seem to be a bit bigger than on the full counts (see figure 7.6b) without the smoothing function. So we did the evaluation with the integrated smoothing function like in figure 7.8.

When the pruning window is chosen too small, sometimes big distances occur due to the fact, that the correct beats can not be found by the recognition. Some curves look like the curve in figure 7.7: the speed of the gesture is recognized correctly for the whole conducting, but no beat is correctly assigned. The overall difference of this special curve is about 25 states difference. The size of pruning window is too small to jump on the correct beat. With a pruning window of bigger size, the whole beats are assigned correctly and the distances are smaller than in curve 7.7.

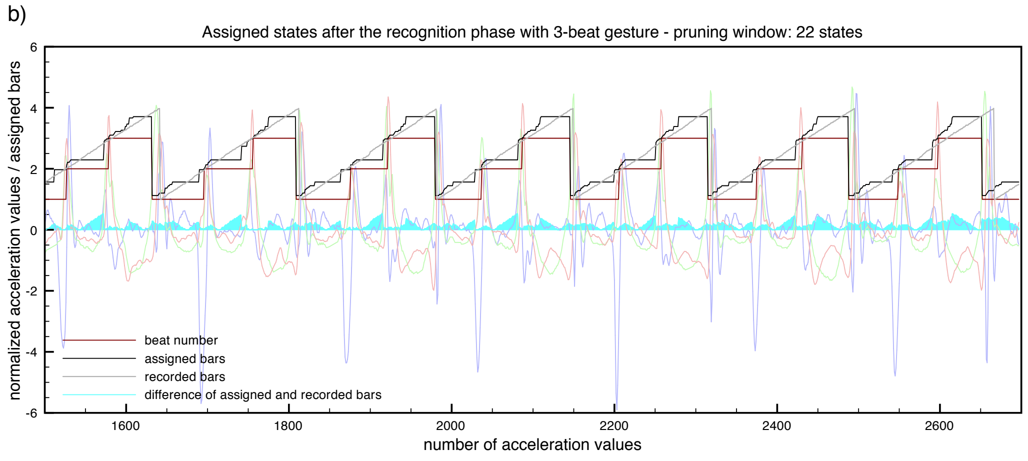
To measure how good the positions in the conducting gesture are recognized, the following criterion was established:

$$E = \frac{1}{T}\sum_{t=1}^{T}(s_r(t) - s_h(t))^2 \qquad (7.1)$$

a)



b)



c)

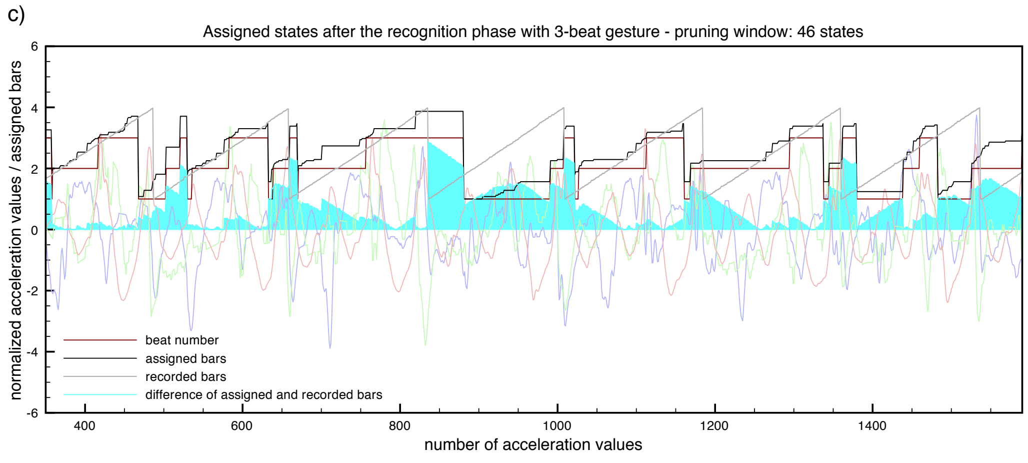


Figure 7.6: Examples of recognition phase with different sizes of pruning window without smoothing

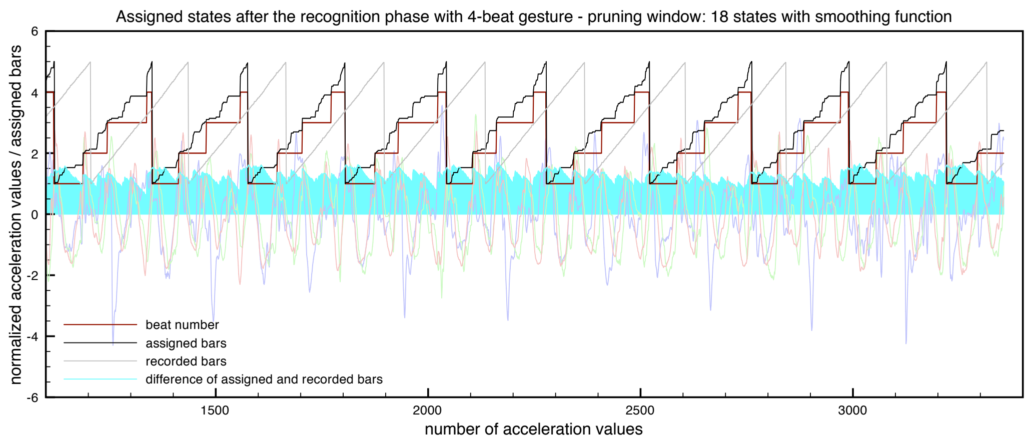


Figure 7.7: Example of recognition phase with size of pruning window 18 with smoothing

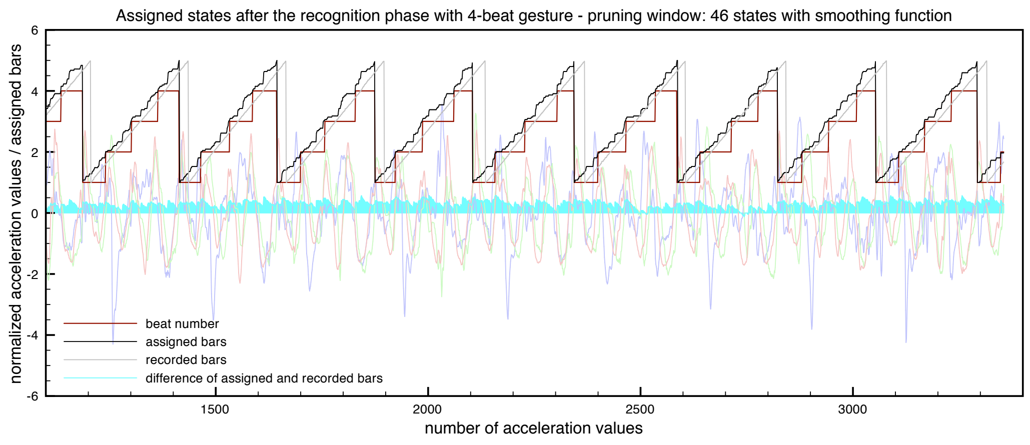


Figure 7.8: Example of recognition phase with same data as in 7.7, but with size of pruning window 46 with smoothing

It gives the squared distance of reference state of recorded gesture s_r to hypothesized state in recognition s_h at every time point t. Because there were many conductors not precisely conducting to the metronome, this measure is not only depending on the recognition, but also on the conductor. That is why we also insert the measure of squared distance from the assigned states s_g in the learning phase to the hypothesized states in recognition for every time point. This represents the difference of the training phase, which uses backtracking, and the real-time recognition without backtracking and is independent of latencies of the conductors.

$$E = \frac{1}{T} \sum_{t=1}^{T} (s_g(t) - s_h(t))^2 \tag{7.2}$$

Now we want to find the minima of this squared distance for all bars, to extract the best size of pruning window for each bar. As we can see in figure 7.9, there is a big area of good sizes for a good recognition. That is why we established a second criterion for the quality of recognition, the correct hypothesized position in a bar, which is described in the following section.

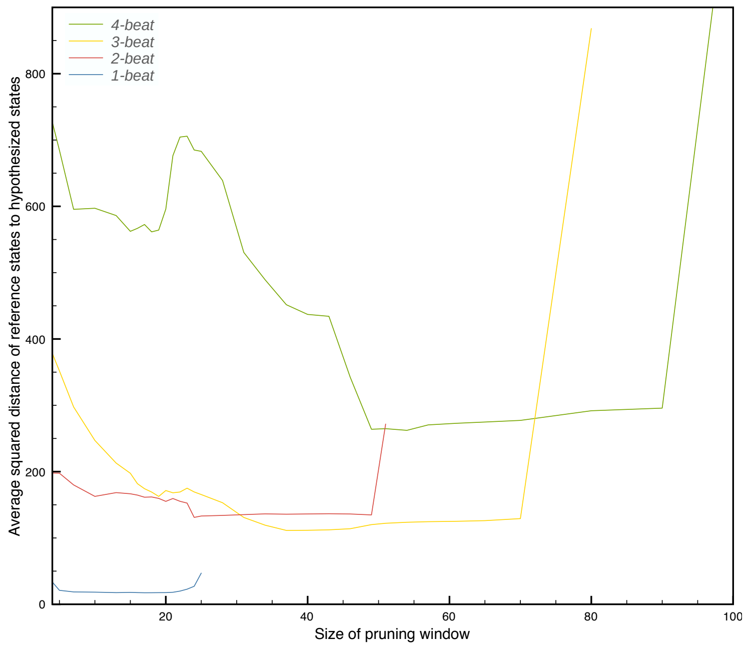


Figure 7.9: Average squared distance of reference state from training to hypothesized state from recognition

The minima for all beats lie between the half number of states of each type of bar and shortly before the full number of possible state for a type of bar. For example, the 4-beat has 100 possible states to assign, and the minimum area of sizes for the pruning window can be found between 50 and 90 states. The 4-beat and the 3-beat type also have a local minimum between 18 and 23 states. With this size it seems to be possible to control the speed of recognition, so that no beat is left out. But the window size seems to be too small to jump inside the bars to the correct beat number, if the beat number before was wrong.

7.2.2 Recognition of the position in beat

To evaluate, if a beat was assigned correctly in the recognition phase, we take the criterion of a window around the recognized beat count, and we search for the same beat count in the assigned reference curve from the learning phase.

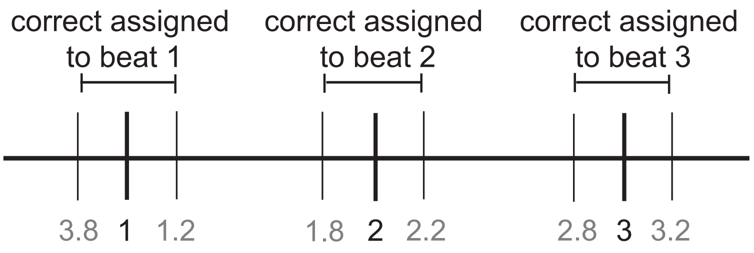

correct assigned to beat 1
correct assigned to beat 2
correct assigned to beat 3
3.8
1
1.2
1.8
2
2.2
2.8
3
3.2

Figure 7.10: Criterion for a correct assigned beat for a 3-beat with 20% window

We decided, that a beat is called correct, if it occurs within a tolerance window of 20% beat forward and backward around the reference beat (see figure 7.10). The actual window size is given by the number of recognition values in the current beat and therefore depends on the current local speed of the gesture.

The percent values, here are not comparable to the referred projects in chapter 3 ***Related work***, because they are not for recognizing which gesture it is, but for recognizing the correct position in a defined gesture. Here, also problems of the pruning window are arising. When the recognition makes a mistake, for example the recognition was wrong, it has to wait for the correct bar, or it has to jump to the next correct bar, but it is only possible to jump to the next states inside the pruning window. So the recognition needs about one bar to reach the correct position in the gesture. In our evaluation it means, for one fault in the recognition of a 4-beat, we count about four wrong beats.

The results in figure 7.11 show more distinct maxima. For the 1-beat bars a recognition rate of 90% are achieved in the 20% window for a pruning window of size 17. The 4-beats are recognized for 81% at the correct position with a pruning window of 80 states. The 3-beat achieves recognition rate up to 79% at the size of pruning window 49. The 2-beat reaches a recognition rate of 78% correct hypothesized beats in the 20% window, the worst rate. This could be explained, that the two beats in a 2-beat bar are very similar in their movements and are the most difficult ones to distinguish. Generally we state, that a pruning window between half and $\frac{2}{3}$ of the number of states, which are learned for a bar, promise good results.

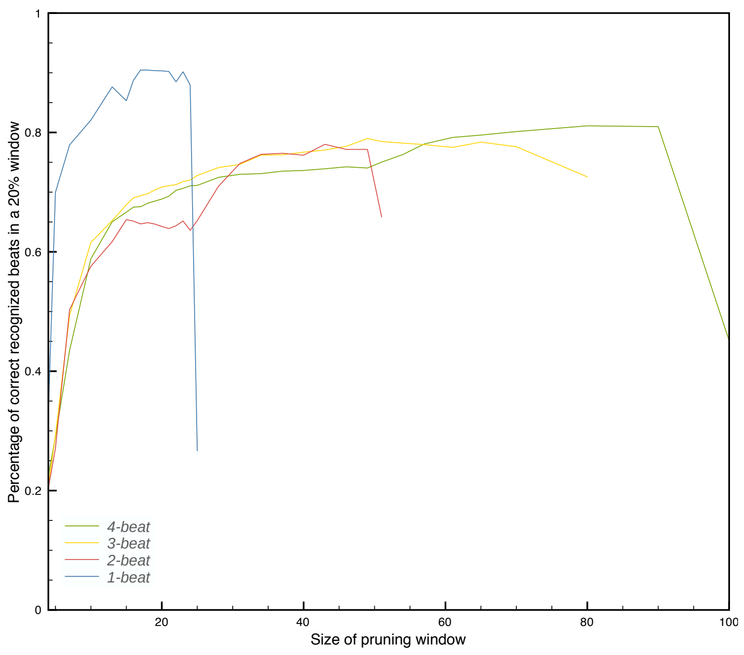


Figure 7.11: Percentage of correct recognized bars in a 20 % window

7.2.3 Training phase versus real-time recognition

Due to the changes in the dynamic programming algorithm for the usage in real-time, we loose quality and precision of recognition of the positions in a bar. This is recognizable in figures 7.12 and 7.13 which contain the same conducting data: the time alignment is more smoothed when using the backtrace part, though we use the smoothing function in the recognition algorithm instead.

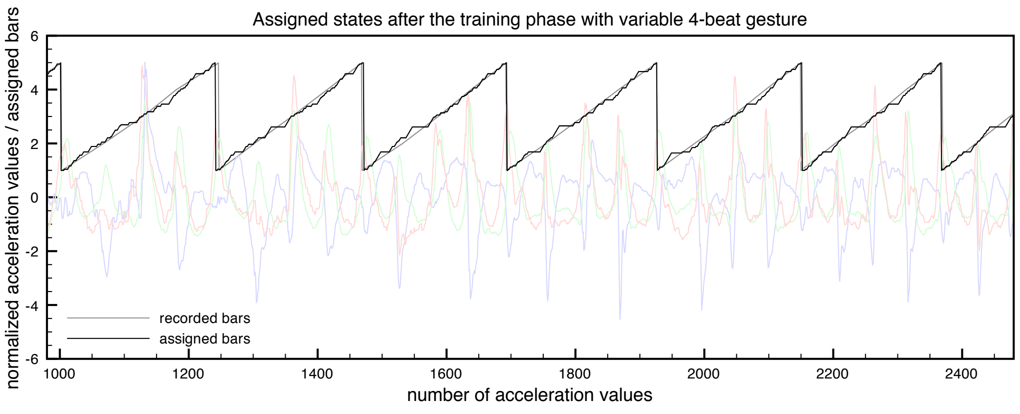


Figure 7.12: Training phase with backtrace

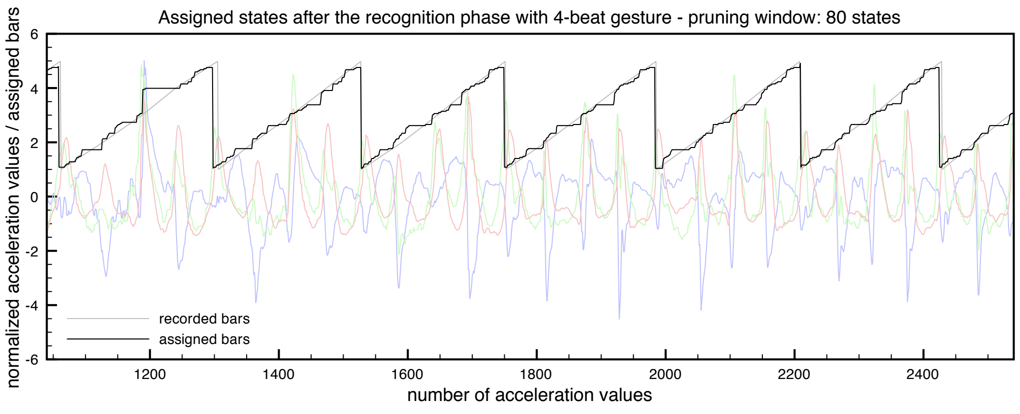


Figure 7.13: Real-time recognition phase of same data without backtrace

7.2.4 Speed of algorithm

Speed of recognition fast enough for Wii remote input and for Semantic Time Framework of Personal Orchestra

The presented algorithm shall need at maximum ten milliseconds from the arriving values to output a position in beat. So we tested our code on a MacPro 3.1 with two 2.8 GHz Quad core processors: on one core probabilities for 25 states with each eight mixture densities can be calculated in ten milliseconds. On a MacBookPro 2.2 with one 2.33 GHz Intel Core 2 Duo processor the estimation of position in beat is done for 15 states with eight mixture densities.

For speed issues small pruning windows are preferred, because the calculation of probabilities is the most expensive calculation in the algorithm. Thus, when the computer is to slow for this recognition, the lower sizes of pruning windows, which achieved good recognition (approximately half of number trained states), should be chosen. Also it is possible to artificially reduce the number of states by changing the topology model and allowing state transitions over several states (and not just one). If the algorithm needs for example 20 milliseconds to calculate the current state, it can just be called 50 times a second and can only process every second acceleration value. Allowing a transition to the second next state makes it possible to replace the missing calculation by a 2-states-transition.

Altogether, estimating the position in beat is fast enough to work with acceleration values arriving at every ten milliseconds from the Nintendo Wii remote. Also the second interface, the Semantic Time Framework (Lee [2007]) works with a communication rate of ten milliseconds. Hence, this algorithm communicates fast enough with this framework and passes positions to it with a sufficient resolution, when a fitting model is used, or multiple cores are used.

7.3 Further aspects in real-time recognition of conducting gestures

Problems because of repeating conducting pattern

In our evaluation of the real-time recognition phase, some problems arose, which are not known from speech recognition. In speech recognition, we often have no circular repetition of a word. That is why the pruning window always is good, if it is bigger than an evaluated border. Due to our circular repeating conducting patterns, the size of pruning window cannot be chosen arbitrary big. In speech recognition the best size of pruning window is can be resized in the recognition phase, if it is not real-time. In our case that is not possible, because our recognition phase cannot decide in real-time, if it is correct or another window size would cause a better recognition.

Restriction of big jumps in time

There are "good" conducting patterns, which are recognized correctly with nearly all sizes of pruning window, others are only recognized correctly with a certain size of pruning window. If we constrain the recognition with a pruning window, we want to restrict jumps and confounds with similar positions in a bar or the same position in the next bar. While doing this, we also constrict jumps to correct beats, when the recognition were too slow or too fast, before. So with the pruning window, we save calculation time and beware the algorithm of big jumps in time, but we also take chances to correct the recognition by a big jump to the correct position in the gesture.

Varying shapes of conducting gestures

Conducting gestures are very heterogeneous between all persons, even though they are professionals or lay-conductors, especially the 2-beat conducting pattern in figure 2.4 was interpreted in very differening ways. Additionally, some lay-conductors sometimes conducted in a delay over half a beat after the metronome. While the recognition noticed the beat correctly, the evaluation would have counted these beats as wrong. That is why we compared the assigned times in the training and the difference times in recognition phase.

For a more fluid interaction with the orchestra, we try to get decisions which are more smoothed in time. Therefore, we established a smoothing function inside the pruning window. This smoothing function constricts far jumps to the next beats, so the recognition for the position inside the bar is not prone to jump to a similar position in beat.

Smoothing function allows more fluid interaction

The more conducting gestures are trained, the better the recognition is. Due to the fact that we trained only half of the conducting gestures for evaluating the recognition phase with the other half of conducting gestures, the final training is done with all available conducting gestures. That means, the final recognition will have better recognition rates, than in this evaluation, cause of the doubled database of learned conducting gestures.

Final training with all recorded conducting gestures

Chapter 8

Summary and future work

This chapter summarizes the whole work done in this thesis. The most important results are represented here again.

8.1 Summary and contributions

In this work we built up an online working recognition algorithm to recognize conducting gestures which is designed to be a plug-in for Personal Orchestra. The resolution with 100 states per bar is high enough to work with the existing semantic time framework for Personal Orchestra and recognizes conducting gestures up to 240 beats per minute. A big data acquirement was done and over 16000 conducted bars were recorded, which also were variable in speed and size and count of beats per bar. Laymen, musicians up to professional conductors took part in this study.

With these recorded conducting gestures, the training of the hidden Markov model was done, for each type of bar separately (1-beat to 4-beat). Preprocessing the data, all values were normalized over the mean and standard deviation in a window of size 100 input values (equates to one second). Then the first derivation was added for each dimension. Time alignment was solved with dynamic programming, whereas all attributes for the topology model were calculated and upgraded in each iteration. The expectation-maximization algorithm was used to optimize all parameters of the model until the maximum likelihood converged. Therefore furthermore the count of parameters were split after two time alignments and optimizing steps to get again better results with mixture densities.

The presented recognition algorithm processes three dimensional acceleration data from the Wii remote and gives the most probable position in every bar every ten milliseconds after preprocessing the data. Therefore, the existing algorithm of the recognition phase the Viterbi algorithm had to be adjusted so that it works in real-time. The recognition phase achieved a recognition rate up to 92% correct beats in a 20% window for all bars with unknown conductors, when the number of beats per bar were clear before. In Personal Orchestra, the music pieces have a confirmed beat-type, so it is clear which type of beat to recognize. If it is not clear, the models have to be calculated in parallel, and the best is chosen. Positive is, that the conductor can start everywhere in the conducting gesture. This was a partial result by the user study, because some conductors preferred starting by the one and not at the preparatory beat.

The functionalities of Personal Orchestra with position-based input are obtained. The dynamics of the musicians is assessable further on, the pause and Wiggle mode are re-implemented for acceleration-based input. Also it should be possible to conduct 1-beats in Personal Orchestra, whether it should be another type of bar. Until now, it has to be chosen before the conducting starts, but it can be adjusted to change the conducting bars in the same conducting.

The training phase automatically assigned the beats on the maxima of the down acceleration, not at the position of the lower turning point. Also left-handed persons can be recognized, if the system is informed before (check-button).

In this thesis Objective-C projects were implemented to record conducting gestures, train a hidden Markov model with these gestures, and recognize them and their progress in time afterwards. Other application areas are imaginable for this project, for example distinguishing gestures of a bendable device.

Additionally, the gesture recognition can also be trained with other signal data than acceleration. To train the system with positional or movement data is also thinkable. However, the Nintendo Wii remote is a more solid hardware, than the Buchla batons, especially if the conductors are kids.

The speed of algorithm should be fast enough to estimate to most possible state in every 10 ms and give it to Personal Orchestra. It not clear how much calculation capacities are needed for the algorithms for Personal Orchestra, so that more calculation power would be needed to run the recognition and the algorithms integrated in Personal Orchestra in parallel.

8.2 Future work

The recognition algorithm is finished to use it in practice as plug-in for Personal Orchestra. It only has to be integrated in the existing project at its designated place to work with the semantic time framework. We tried to eliminate unnatural behavior of the orchestra, but did not test it with the semantic time framework of Personal Orchestra.

A further evaluation with the integrated plug-in in Personal Orchestra should be done. How does the orchestra reacts on the smoothed time informations every 10 milliseconds? Also the interacting conductor should be observed: how does the conductor react on faults of the recognition and the orchestra.

To upgrade the recognition rate, positional or rotational information can be included in the recognition additionally. Taking into account the positional data of the Wii remote would be a small change in the program (elongate the feature vector in the training and recognition phase), but would improve the recognition again. Additionally Nintendo released the Wii MotionPlus, a new expansion device for the Nintendo Wii, which can accurately capture complex motion. Adding this data to the feature vector of training and recognition algorithm, the recognition would be be further improved. The main work would be to acquire new training data for these new features.

With the written Objective-C projects, also more conducting gestures, for example the 6-beat, can be recorded and can be added to the now existing classification library. The code for recording and storing the gestures in the correct format for automatic learning was implemented in this work.

Also the autonomy of the orchestra could be regulated. With autonomy is meant how precise the orchestra reacts on the conducting. If the orchestras autonomy degree is 100% the system plays the video in the tempo it was recorded. If the autonomy degree is 0% it reacts precisely on the conductors speed.

An additional main goal would be to run all trained models in parallel to recognize the most probable gesture types and its appropriate position at the same time. This is normally the standard, what is done with hidden Markov models in gesture recognition and will work without much further work. Only the processing power has to be found and allocated for these time-consuming calculations.

Further, the implemented work can be transfered to other repeating patterns with time-based background. The learning algorithm can be used not only for acceleration data, but also for positional and all other signal data from any hardware. The size of the learned features can be adjusted. Also the model, which were here a circular first order model can be adjusted. So using this training and recognition phase is thinkable for other time-based applications. Another application with a repeating pattern would be to learn and recognize positions of dancing steps.

Figure 8.1: Future conductor at work

Appendix A

Tables with clustering results

	Occurence in All	Occurence in 4-beats	Occurence in 3-beats	Occurence in 2-beats	Occurence in 1-beats	P(4-beat|A)	P(3-beat|A)	P(2-beat|A)	P(1-beat|A)
red max indicates beats	87.94%	77.16%	89.15%	98.32%	95.65%	21.42%	24.74%	27.29%	26.55%
green max indicates beats	64.47%	82.10%	68.99%	44.54%	41.30%	34.65%	29.12%	18.80%	17.43%
before beat 1 global min	80.04%	78.40%	73.64%	84.87%	91.30%	23.89%	22.44%	25.86%	27.82%
before Beat 1 two global min	16.45%	17.90%	21.71%	13.45%	4.35%	31.19%	37.81%	23.42%	7.57%
no global min before beat 1	2.63%	3.70%	3.10%	1.68%	0.00%	43.65%	36.54%	19.81%	0.00%
Beat 1 red max	99.12%	98.77%	99.22%	99.16%	100.00%	24.87%	24.98%	24.97%	25.18%
Beat 1 red global max	60.96%	77.16%	60.47%	26.05%	95.65%	29.75%	23.32%	10.05%	36.88%
Beat 1 green global max	61.40%	67.28%	50.39%	55.46%	86.96%	25.87%	19.37%	21.32%	33.43%
Beat 1 red + green global max	43.86%	59.88%	33.33%	18.49%	82.61%	30.82%	17.16%	9.51%	42.51%

Table A.1: Overall recognizable features of conducting accerleration data

	Occurence in All	Occurence in 4-beats	Occurence in 3-beats	Occurence in 2-beats	Occurence in 1-beats	P(4-beat|A)	P(3-beat|A)	P(2-beat|A)	P(1-beat|A)
Beat 1,5 red min + blue max	23.90%	17.28%	27.13%	28.57%	26.09%	17.45%	27.39%	28.84%	26.33%
Beat 1,5 red min + blue hatmin	16.01%	24.07%	8.53%	16.81%	6.52%	43.04%	15.25%	30.05%	11.66%
Beat 1,5 red min + blue increasing	25.22%	48.15%	4.65%	21.01%	13.04%	55.44%	5.36%	24.19%	15.02%
Beat 1,5 red min + blue max or hatmin or increasing	65.13%	89.51%	40.31%	66.39%	45.65%	37.01%	16.67%	27.45%	18.88%
Beat 2 blue max -> red max -> blue min	51.32%	91.36%	1.55%	70.59%	-	55.88%	0.95%	43.17%	-
Beat 2,5 red min + blue min	11.84%	15.43%	5.43%	18.49%	-	39.22%	13.79%	46.99%	-
Beat 2,5 red min + blue hatmax	25.44%	67.28%	1.55%	4.20%	-	92.12%	2.12%	5.75%	-
Beat 2,5 red min + blue min or hatmax	37.28%	82.72%	6.98%	22.69%	-	73.60%	6.21%	20.19%	-
Beat 3 blue min -> red max (+ green max if exists)	32.24%	90.12%	0.78%	-	-	99.15%	0.85%	-	-
Beat 3,5 red min + blue max	14.69%	29.01%	15.50%	-	-	65.17%	34.83%	-	-
Beat 3,5 red min + blue hatmin	4.82%	0.00%	17.05%	-	-	0.00%	100.00%	-	-
Beat 3,5 red min + blue decreasing	13.38%	37.04%	0.78%	-	-	97.95%	2.05%	-	-
Beat 3,5 red min + blue max or hatmin or decreasing	32.89%	66.05%	33.33%	-	-	66.46%	33.54%	-	-
Beat 4 blue max->red max->blue min (+green max if	32.89%	92.59%	-	-	-	100.00%	-	-	-

Table A.2: Features of conducting accerleration data to recognize 4-beat gestures

	Occurence in All	Occurence in 4-beats	Occurence in 3-beats	Occurence in 2-beats	Occurence in 1-beats	P(4-beat|A)	P(3-beat|A)	P(2-beat|A)	P(1-beat|A)
Beat 1,5 red min + blue min	11.62%	1.85%	17.83%	16.81%	15.22%	3.58%	34.48%	32.50%	29.43%
Beat 1,5 red min + blue hatmax	9.87%	1.85%	20.93%	5.04%	19.57%	3.91%	44.17%	10.64%	41.29%
Beat 1,5 red min + blue decreasing	9.87%	3.09%	17.05%	10.92%	10.87%	7.36%	40.67%	26.05%	25.92%
Beat 1,5 red min + blue min or hatmax or decreasing	31.36%	6.79%	55.81%	32.77%	45.65%	4.81%	39.58%	23.24%	32.37%
Beat 2 blue min -> red max -> blue max	34.87%	0.00%	95.35%	30.25%	-	0.00%	75.91%	24.09%	-
Beat 2,5 red min + blue max	17.98%	0.00%	36.43%	29.41%	-	0.00%	55.33%	44.67%	-
Beat 2,5 red min + blue hatmin	16.67%	0.00%	50.39%	9.24%	-	0.00%	84.50%	15.50%	-
Beat 2,5 red min + blue max or hatmin	34.65%	0.00%	86.82%	38.66%	-	0.00%	69.19%	30.81%	-
Beat 3 blue max -> red max -> blue min	26.54%	0.00%	93.80%	-	-	0.00%	100.00%	-	-
Beat 3,5 red min + blue min	0.88%	0.00%	3.10%	-	-	0.00%	100.00%	-	-
Beat 3,5 red min + blue hatmax	2.19%	0.62%	6.98%	-	-	8.13%	91.87%	-	-
Beat 3,5 red min + blue increasing	14.25%	2.47%	47.29%	-	-	4.96%	95.04%	-	-
Beat 3,5 red min + blue min or hatmax or decreasing	17.32%	3.09%	57.36%	-	-	5.11%	94.89%	-	-

Table A.3: Features of conducting accerleration data to recognize 3-beat gestures

	Occurence in All	Occurence in 4-beats	Occurence in 3-beats	Occurence in 2-beats	Occurence in 1-beats	P(4-beat|A)	P(3-beat|A)	P(2-beat|A)	P(1-beat|A)
Beat 2 blue global max -> red max -> blue min	10.96%	1.85%	0.00%	39.50%	-	4.48%	0.00%	95.52%	-
Beat 2,5 red min with blue increasing	9.65%	6.17%	0.00%	28.57%	-	17.77%	0.00%	82.23%	-

Table A.4: Features of conducting accerleration data to recognize 2-beat gestures

Appendix B

Trained means and variances for all conducting gestures

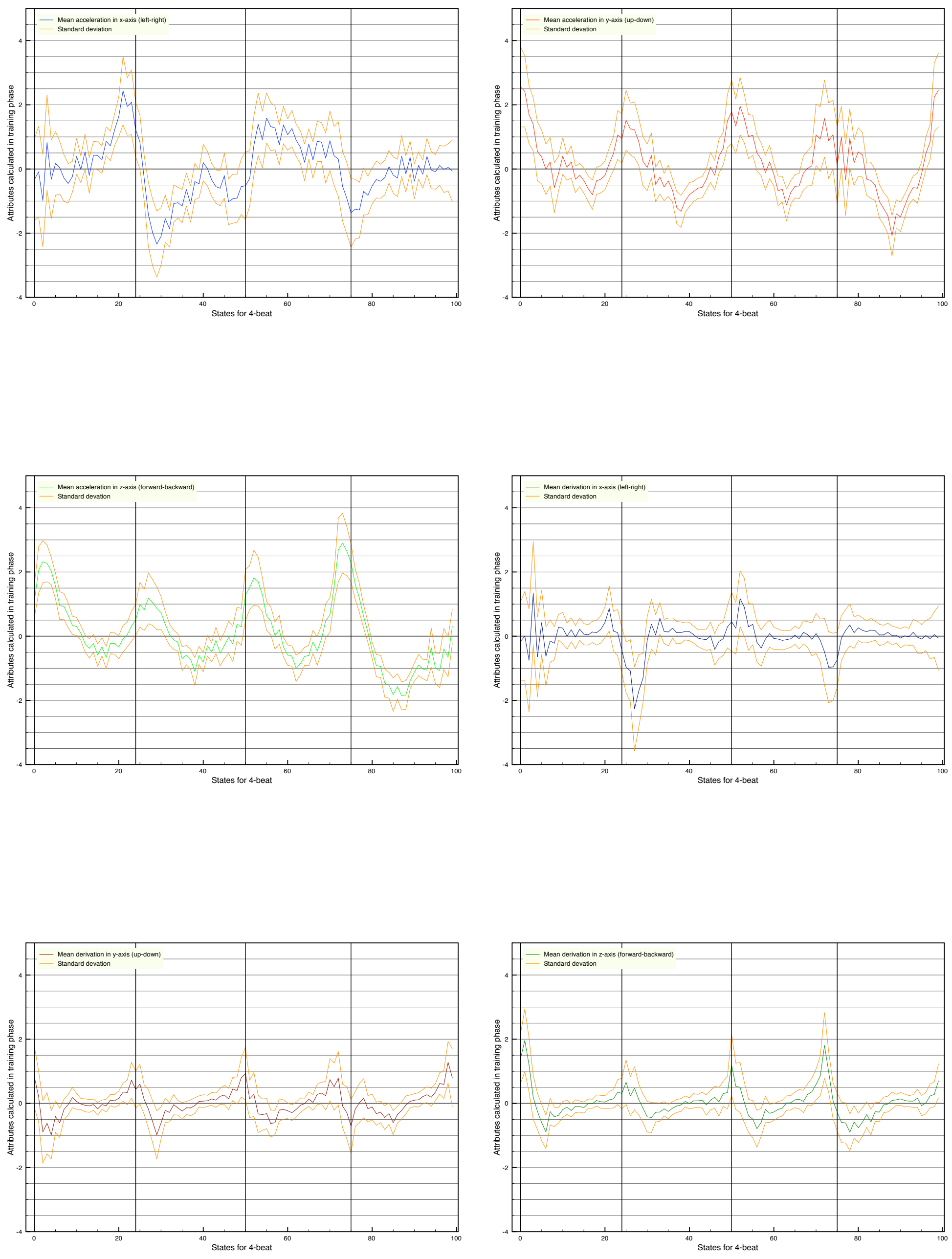


Figure B.1: Trained attributes for a 4-beat conducting gesture after 6 iterations time alignment for all 100 states

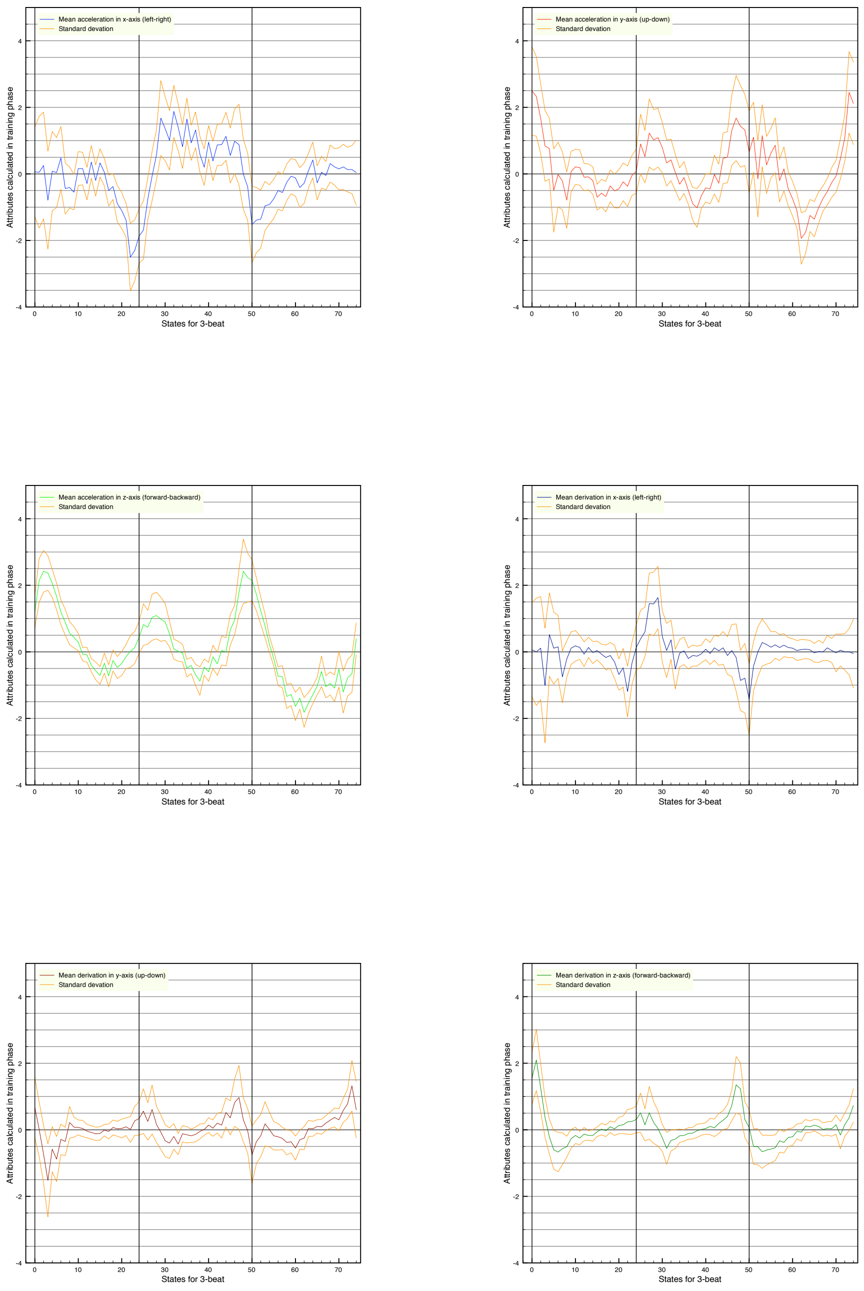


Figure B.2: Trained attributes for a 3-beat conducting gesture after 6 iterations time alignment for all 75 states

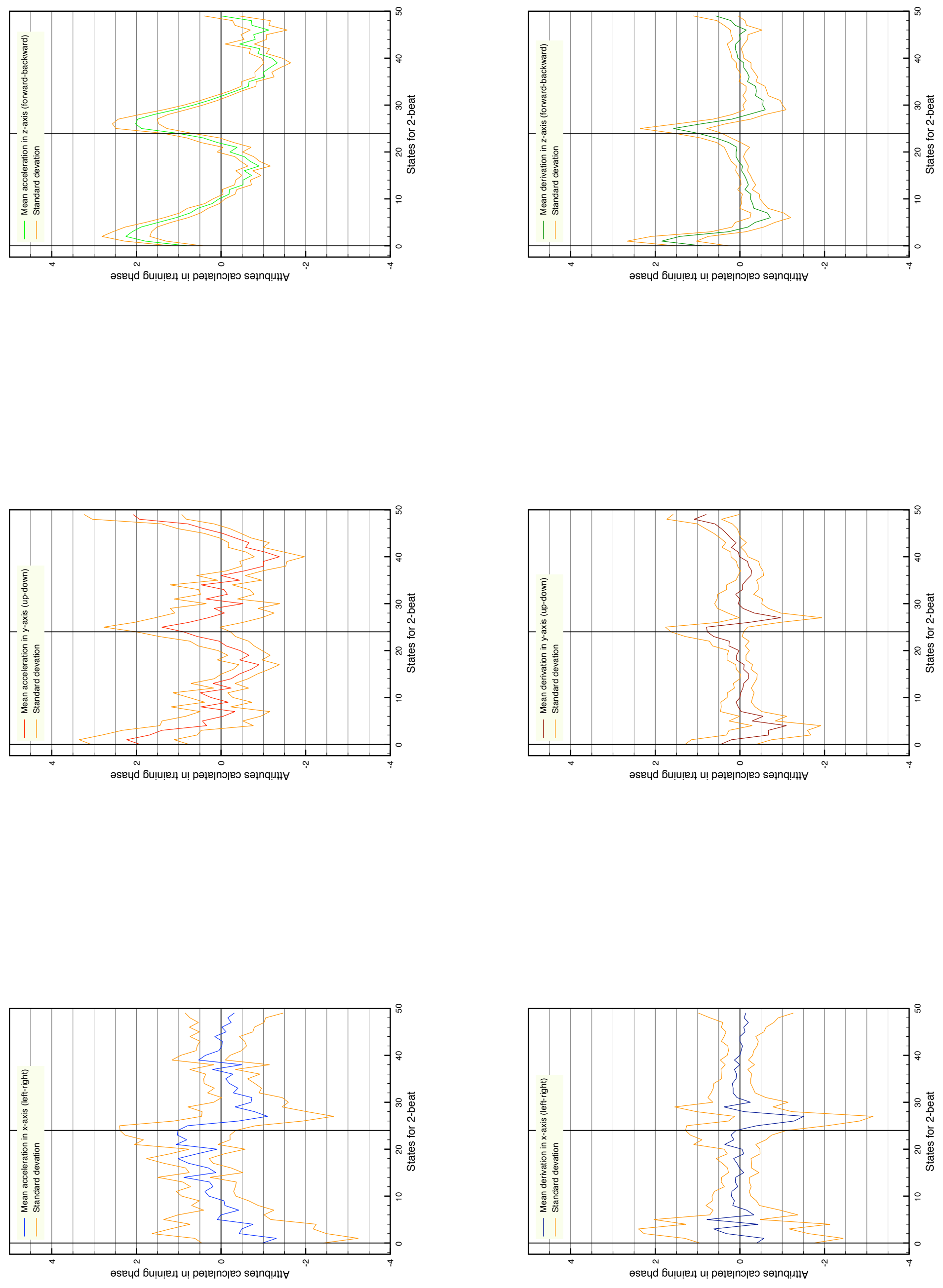


Figure B.3: Trained attributes for a 2-beat conducting gesture after 6 iterations time alignment for all 50 states

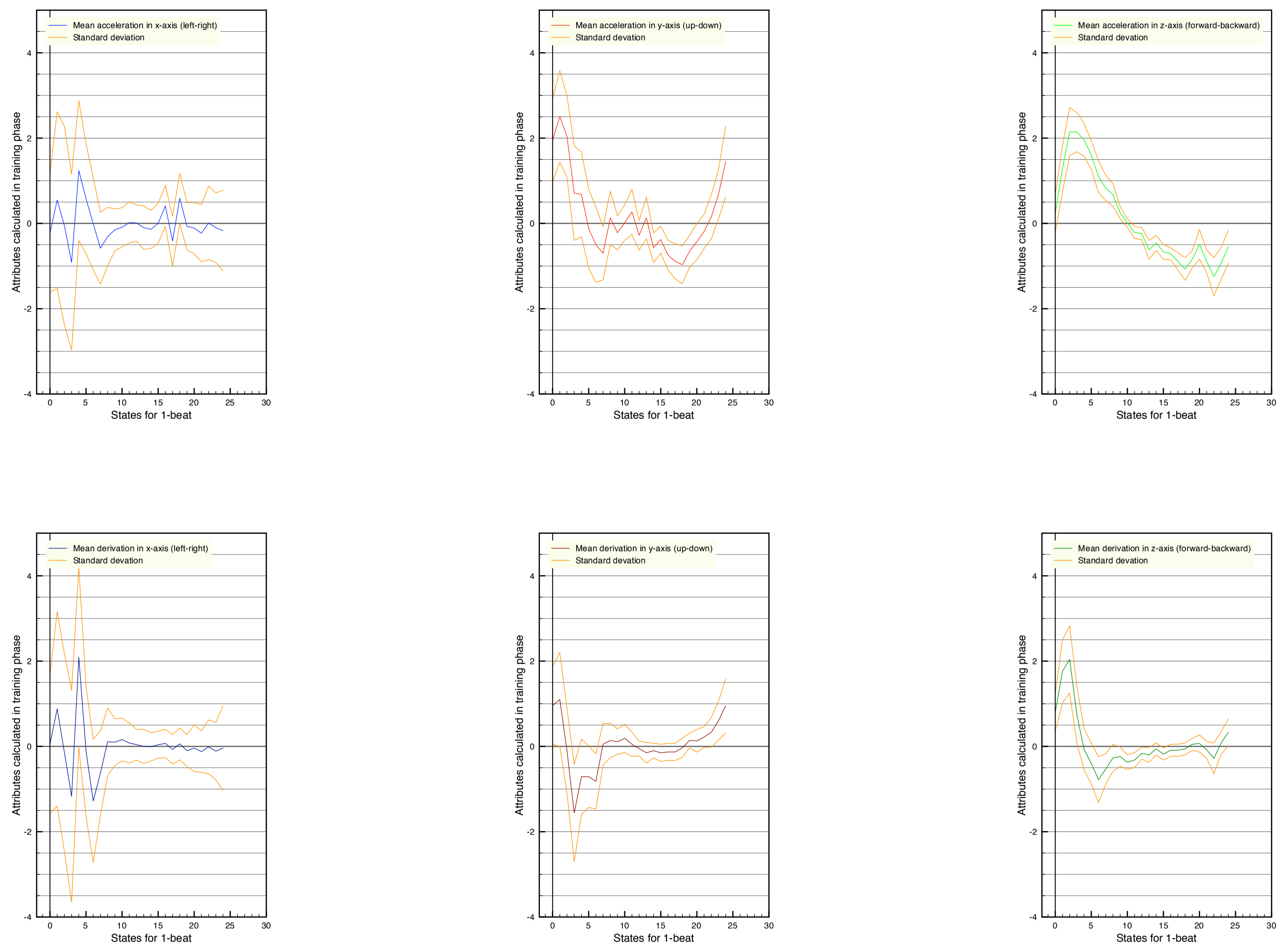


Figure B.4: Trained attributes for a 1-beat conducting gesture after 6 iterations time alignment for all 25 states

Bibliography

L. E. Baum, T.Petrie, G. soules, and N. Weiss. A maximization technique occuring in the statistical analysis of probalistic functions of marcov chains. In *Annals of Mathematical Statistics*, volume Vol. 41, pages 165–171, 1970.

Richard Ernest Bellman. *Dynamic Programming*. Dover Publications, Incorporated, 2003.

Graziano Bertini and Paolo Carosi. Light baton: A system for conduct- ing computer music performance. In *ICMA*, pages 73–76. ICMC92, 1992.

Jan Borchers, Eric Lee, Wolfgang Samminger, and Max Mühlhäuser. Personal orchestra: A real-time audio/video system for interactive conducting. *ACM Multimedia Systems Journal Special Issue on Multimedia Software Engineering*, 9(5):458–465, March 2004.

Bernd Bruegge, Christoph Teschner, Peter Lachenmaier, Eva Fenzl, Dominik Schmidt, and Simon Bierbaum. Pinocchio: conducting a virtual symphony orchestra. In Masa Inakage, Newton Lee, Manfred Tscheligi, Regina Bernhaupt, and Stephane Natkin, editors, *Advances in Computer Entertainment Technology*, pages 294–295. ACM, 2007.

Saskia Dedenbach. Analyzing latency and sampling rates in music controllers. Master's thesis, RWTH Aachen University, 2008.

M. Lee; G. Garnett and D. Wessel. An adaptive conductor follower. In *Proc. ICMC 1992*, pages 454–455. ICMA, 1992.

A. Inc. Livemove white paper. Technical report, AiLive Inc., http://www.ailive.net/liveMove.html, 2006.

Sanna Kallio, Juha Kela, and Jani Mäntyjärvi. Online gesture recognition system for mobile interaction. In *International Conference on Systems, Manand Cybernetics*, volume Vol.3, pages 2070 – 2076, Washington, D.C., USA, Oct. 2003. IEEE.

Thorsten Karrer, Eric Lee, and Jan Borchers. Phavorit: A phase vocoder for real-time interactive time-stretching. In *Proceedings of the ICMC 2006 International Computer Music Conference*, pages 708–715, New Orleans, USA, November 2006.

David Keane and Peter Gross. The midi baton. In *Proceedings ICMC89*, pages pages 151–154. ICMA, 1989.

Juha Kela, Panu Korpipää, Jani Mäntyjärvi, Sanna Kallio, Giuseppe Savino, Luca Jozzo, and Sergio Di Marca. Accelerometer-based gesture control for a design environment. *Personal and Ubiquitous Computing*, 10(5):285–299, 2006.

In-Cheol Kim and Sung-Il Chien. Analysis of 3d hand trajectory gestures using stroke-based composite hidden markov models. *Appl. Intell.*, 15(2):131–143, 2001.

Paul Kolesnik and Marcelo Wanderley. Recognition, analysis and performance with expressive conducting gestures. In *ICMA*, 2004.

Eric Lee. *A Semantic Time Framework for Interactive Media Systems*. PhD thesis, RWTH Aachen University, 2007.

Eric Lee, Ingo Grüll, Henning Kiel, and Jan Borchers. conga: A framework for adaptive conducting gesture analysis. In *NIME 2006 International Conference on New Interfaces for Musical Expression*, pages 260–265, Paris, France, June 2006.

Jani Mäntyjärvi, Juha Kela, Panu Korpipää, and Sanna Kallio. Enabling fast and effortless customisation in accelerometer based gesture interaction. In David S. Doermann and Ramani Duraiswami, editors, *MUM*, volume 83 of *ACM International Conference Proceeding Series*, pages 25–31. ACM, 2004.

Vesa-Matti Mäntylä, Jani Mäntyjärvi, Tapio Seppänen, and Esa Tuulari. Hand gesture recognition of a mobile device user. In *IEEE International Conference on Multimedia and Expo (I)*, pages 281–284, 2000.

Teresa Marrin and Rosalind Picard. The "conductor's jacket": A device for recording expressive musical gestures. In *In Proc. Intnl. Computer Music Conf*, 1998.

Max V. Mathews. The conductor program and mechanical baton. *Current directions in computer music research*, pages 263–281, 1989.

Uwe Maurer, Anthony Rowe, Asim Smailagic, and Daniel P. Siewiorek. ewatch: A wearable sensor and notification platform. In *BSN*, pages 142–145. IEEE Computer Society, 2006.

H. Morita, S. Ohteru, and S. Hashimoto. Computer music system which fol- lows a human conductor. In *ICMC89*, pages 207–210. ICMA, 1989.

Teresa Marrin Nakra. *Inside the "conductor's jacket": analysis, interpretation and musical synthesis of expressive gesture*. PhD thesis, Massachusetts Institute of Technology, 2000.

Mukund Narasimhan, Paul A. Viola, and Michael Shilman. Online decoding of markov models under latency constraints. In William W. Cohen and Andrew Moore, editors, *ICML*, volume 148 of *ACM International Conference Proceeding Series*, pages 657–664. ACM, 2006.

L. Rabiner and B. Juang. An introduction to hidden markov models. *IEEE Acoutics, Speech and Signal Processing Magazine*, 3:4–16, 1986.

L. R. Rabiner and B.-H. Juang. *Fundamentals of Speech Recognition*. Prentice Hall, 1993.

Lawrence R. Rabiner. A tutorial on hidden markov models and selected applications in speech recognition. In *proceedings of the IEEE, vol 77, no. 2*, February 1989.

Max Mathews Richard Boulanger. The 1997 mathews radio-baton and improvisation modes. In *ICMC97*, 1997.

Max Rudolf. *The Grammar of Conducting: A Comprehensive Guide to Baton Technique and Interpretation*. Wadsworth Inc Fulfillment, 3rd edition edition, 1995.

Stuart J. Russell and Peter Norvig. *Artificial Intelligence: A Modern Approach*. Prentice-Hall, Inc., second edition edition, 1995.

Thomas Schlömer, Benjamin Poppinga, Niels Henze, and Susanne Boll. Gesture recognition with a wii controller. In Albrecht Schmidt, Hans Gellersen, Elise van den Hoven, Ali Mazalek, Paul Holleis, and Nicolas Villar, editors, *Tangible and Embedded Interaction*, pages 11–14. ACM, 2008.

D. Schmidt, R.B. Dannenberg, A. Smailagic, D.P. Siewiorek, and B. Biigge. Learning an orchestra conductor's technique using a wearable sensor platform. In *Wearable Computers*, pages 113–114. IEEE, 2007.

Alison Latham Stanley Sadie. *Das Cambridge Buch der Musik*. Calmann and King Ltd, 1986.

Forrest Tobey. The ensemble member and the conducted computer. In *ICMC95*, pages 529–530. ICMA, 1995.

Forrest Tobey and Ichiro Fujinaga. Extraction of conducting gestures in 3d space. In *ICMC96*, pages 305–307. ICMA, 1996.

S. Usa and Y. Mochida. A multi-modal conducting simulator. In *ICMA*, pages 25 – 32. ICMA, 1998.

Tian-Shu Wang, Heung-Yeung Shum, Ying-Qing Xu, and Nan-Ning Zheng. Unsupervised analysis of human gestures. *Lecture Notes in Computer Science*, 2195:174–ff, 2001.

Gerd Seibert; Erhard Wendelberger. *Lexikon 2000*. Zweiburgen Verlag Weinheim, 1983.

Neal Zaslaw. When is an orchestra not an orchestra? *Early Music*, XVI(4):483–495, 1988.

Index

Typeset December 15, 2008