RWTH AACHEN UNIVERSITY

Saltate!

A Sensor Based System to Support Dance Beginners

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

by
Dieter Drobny

Thesis advisor:
Prof. Dr. Jan Borchers

Second examiner:
Univ.-Prof. Dr. phil. Iring Koch

Registration date: May 14th, 2008
Submission date: Dec 29th, 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 vorliegende Arbeit selbständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt, sowie Zitate kenntlich gemacht habe.

Aachen, December 2008
Dieter Drobny

Contents

List of Figures

List of Tables

Abstract

During the last years, more and more sensor based wearable computing systems have been developed that are used for physical activities in people's leisure time.
While sensor based systems can be used for monitoring activities over time, thus giving their users feedback about performance improvements over time, such systems possibly allow the calculation of complex concurrent feedback during a task's execution as well. In cognitive psychology it is well known that concurrent feedback has a strong positive impact on task execution, but also usually a negative effect on learning, i.e., people who used concurrent feedback during practice are later usually outperformed by people who trained without concurrent feedback.
Human teachers usually give less and less feedback regarding specific movements the better their trainees become in the movements' execution. We developed *Saltate!*, a sensor based system to support dance beginners that avoids constant use of concurrent feedback, providing help only when it is needed.
Saltate! was developed to support Slow Waltz, but the general architecture is easily expandable to other dances as well. Saltate!'s feedback aims at supporting dance beginners' beat recognition capabilities, emphasizing the music's beats if the dancers dance not, or incorrectly.
The system's impact on short term learning, as well as the beginner's acceptance of the system, was evaluated with the help of eight volunteer couples with little or no experience in dancing.
The system has a very high acceptance among beginning dancers, both its hardware components and its supporting function. Sadly, we couldn't acquire enough data to calculate significant results regarding the effect of Saltate!'s feedback on learning, but the data we acquired are looking promising.

Überblick

In den letzten Jahren wurden mehr und mehr sensorbasierte tragbare Computersysteme für den Einsatz bei körperlichen Aktivitäten in der Freizeit entwickelt.
Während sensorgestütze Systeme für die zeitliche Beobachtung von Aktivitäten gebraucht werden können um so ihren Benutzern Feedback über ihre Leistungsverbesserungen zu geben, könnten sie prinzipiell auch zur Berechnung von komplexem Feedback während einer Aktion genutzt werden. Im Bereich der kognitiven Psychology ist bekannt, dass gleichzeitiges Feedback einen starken positiven Einfluß auf die Ausführung einer Tätigkeit hat, aber gleichzeitig einen Effekt auf das Lernen, d.h. Leute die gleichzeitiges Feedback beim Training benutzten zeigen später, ohne Feedback, üblicherweise schlechtere Leistungen als Leute die ohne Feedback trainierten.
Menschliche Trainer geben normalerweise immer weniger Feedback zu einzelnen Bewegungen, je besser ihre Lehrlinge diese beherrschen. Wir entwickelten *Saltate!*, ein sensorgestütztes System zur Unterstützung von Tanzanfängern, das ständigen Einsatz von gleichzeitigem Feedback vermeidet und Hilfe nur dann anbietet, wenn sie gebraucht wird.
Saltate! wurde entwickelt um langsamen Walzer zu unterstützen, aber die Architektur ist leicht erweiterbar auf andere Tänze. Saltate!s Feedback zielt darauf ab die Fähigkeit von Tanzanfängern Taktschläge in der Musik zu erkennen zu unterstützen, indem es diese betont falls die Tänzer nicht, oder falsch, tanzen.
Wir haben mit Hilfe von acht freiwilligen Paaren mit wenig oder keiner Tanzerfahrung den Einfluß des Systems auf das Kurzzeitlernen und die Akzeptanz des Systems bei den Tänzer untersucht.
Das System hat eine hohe Akzeptanz unter Tanzanfängern, sowohl seine Hardwarekomponenten als auch seine unterstützende Funktion. Leider konnten wir nicht genügend Daten sammeln, um signifikante Ergebnisse bezüglich des Einlusses des Systems auf das Lernen zu berechnen; aber die Daten, die wir sammeln konnten, sind vielversprechend.

Acknowledgements

(In order of their assumed influence on this thesis' grade :-))
I want to thank:

Prof. Dr. Jan Borchers for sparking my interest in human-computer interaction with his interesting lectures and the associated exercises.
Univ.-Prof. Dr. phil. Iring Koch for voluntarily examining an information scientists' diploma thesis, and for his advice on setting up the evaluation of this thesis' system.
My supervisor Malte Weiß for enabling me to work on such an interesting topic. Without him, I would have missed a couple of thousands surprised **"You're doing WHAT?"** after I told people about my diploma thesis.
My "i10 roommates" Noriyasu Vontin, Leonhard Lichtschlag, Moritz Wittenhagen, and Florian Heller, for sharing so many delightful hours of studying, talking, and working with me.
Ralf Trimborn, Stefan Kuypers, and my sister Silvia Drobny for proofreading this thesis. (Moritz, I mentioned you already.)
Thorsten-Joachim Baulig: He must have explained much more things about standard dancing and Slow Waltz to me than I could remember.

Without my great dance partner Anja Wätjen and her contagious enthusiasm for Rock'n'Roll dancing I certainly would have worked on a very different topic in my diploma thesis.
The cooking abilities of my flat mates Gabriele Rönneper and Christian von Lechberg always kept me from starving to death.

Last but not least I thank all dancers and non-dancers who willingly put my sensors into their shoes. Following the conventions of experimental studies, your names or any other data that might be used to identify you will not be made public.

Conventions

Throughout this thesis we use the following conventions:

Text conventions

Terms from the domain of dancing, music, and terms that are introduced and described by this thesis in continuous text are written in *italic typeface* at their first appearance.

The whole thesis is written in American English.

For a better readability, the plural form “we” will be used instead of “I” and unidentified third persons are described in male form.

Links to web pages

Links to web pages, project sites, and download links are shown in a footnote at the bottom of the page, and linked to in the pdf-version of this thesis.

Chapter 1

Introduction

"The journey of a thousand leagues begins with a single step."

—Lao Tzu

Dancing is a part of western culture.

Ballroom dancing is an important part of western culture. Though most people only rarely visit a formal dance, the ability to dance still becomes important occasionally, at the latest when you have to open the dances on your own wedding. Thus, most people take part in a dancing course at least once in their life. While taking a dancing course, you normally have to share the instructor's attention with several or many other pairs of dancers - unless you are willing to pay for private lessons. Hence, you do not get individual feedback related to your own mistakes very often. Outside of a course practicing is very hard for beginners as they don't get any instructions at all.

Sensor based training systems are researched in other domains.

In other domains, like martial arts or snowboarding, the development of smaller and cheaper sensors already lead to the research of wearable computing systems with the aim of supporting their users in performing or learning of domain specific tasks. Such systems can increase the frequency and amount of feedback given to its wearer (compared to a human instructor) and offer the possibility of concurrent feedback that human instructors often cannot give. However, such a system does not exist in the domain of dancing yet.

Music forms an important difference between dancing and many other motor skills.

In dancing we can find an important difference to many other domains where motor learning plays a role: music. Since a dance is normally performed to music, dancers not only have to learn specific movements, they also have to adapt their speed to external acoustic stimuli. For most beginners of ballroom dancing the hardest thing is not the execution of a dance's steps, but their synchronization with the beats of the music.

The usefulness of concurrent feedback for motor skill learning is still unclear.

It is still an open question in how far concurrent feedback can be useful for beginners to learn specific motor skills, in dancing as well as in other domains. On the one hand, sensor based training systems can provide feedback a human teacher cannot give, e.g., resulting in a higher feedback frequency or in completely different kinds of feedback. On the other hand, psychological experiments showed that, usually, concurrent feedback improves performance, but also decreases learning, i.e., people are better as long as they get feedback, but they learn to rely on it. Once the feedback is taken away from them, they perform worse than people who trained without feedback. However, the feedback's frequency in those experiments was not adaptive to the people's performance - it was always present. A human teacher adapts his feedback to the students' performance: As they get better, he corrects less and less beginner errors and instead offers advice concerning other, more complicated aspects of the task.

Irrespective of questions regarding the effectiveness of feedback is the matter of feedback acceptance by the system's user. A human teacher usually gives concrete verbal instructions. Possibly, explicit instructions like these are not accepted by trainees if they are given by a computer, even if they were the most effective ones.

For an advanced dance training system, which could be used by the training couple on its own, questions regarding the command input arise:
While dancing, the dancers probably move away from a stationary input device and their hands are not free to hold a remote device. As the dancers are equipped with sensors, these could be used for command input as well. How this could be done is not researched well yet.

Saltate! features performance-adapted support using nondistracting hardware.

Within this thesis' project we developed a system that is capable of tracking data of a couple of beginning dancers good enough to provide performance-adaptable feedback or guidance. The most important subgoal for hardware development was to keep the size and weight small in order to not disturb the dancers in their movements. We decided to spend much time into the hardware development, which is one reason why the developed feedback's complexity is relatively low. The other reason is that there was no experience with dancers' reaction to automatic help during practice. We had to gather these during Saltate!'s development and evaluation.

With the help of 16 volunteers we we studied the influence of Saltate!'s supporting functions on short term learning of beginning dancers. We further evaluated our volunteers' subjective impression of the system with a questionnaire.

In the following chapters we give an overview over related systems and psychological findings about the learning of motor skills, as far as they are of greater interest to us. A short excursus names the terms of dancing and music that are used within this theses and explains these without formal definitions.

The main part of our work describes the development of Saltate!'s hardware, the design of Saltate!'s feedback, and the development of its software. The experimental setup of Saltate!'s evaluation and its discussion, followed by a summary and future work, form the end of this thesis.

Chapter 2

Related work

"When a thing has been said and well, have no scruple. Take it and copy it."

—Anatole France

Related work is organized into three subchapters: Technical systems, feedback effects, and perceived simultaneity.

Prior the beginning of the practical development of Saltate!, and partly during the development process, we accomplished researches into several different directions that seemed to be of interest for a dance training system. The results of these researches are grouped into three main areas, each of which we present in a different subchapter. In the first one we give an overview over related systems. In the second one we expose psychological research results about influences of different variables on the learning of motor skills. Inquiries in these two subchapters have been done by literature research, only in case of one related system we had the additional opportunity to talk to its developers personally. In the last subchapter we present findings about the perceived simultaneity of different events by humans, i.e., for which time differences are events perceived as synchronous, and for which time differences are they not?

2.1 Related systems

Since in dancing movements are to a huge degree determined by the feet, systems that use sensors in or on shoes

are of interest to us.

The GaitShoe: Equipped with a huge amount of sensors.

Paradiso et al. [2000] developed a very interesting sensor shoe: It is equipped with a huge amount of sensors, measuring pressure on three points near the toe, one point at the heel, bending of the sole, height of the foot over conducting strips in the floor, tilt, acceleration, and position. All of these values are transmitted wirelessly at 50 Hz. The shoe was used in several projects involving dancers, but always as a supporting feature for shows: Mapping foot movements to sound, the shoe was used as an interface for experienced or professional dancers to augment their performance.

The huge amount of sensors bring along some complications and restrictions with them: Some sensor values are only usable within a prepared surrounding, like the height measuring or the positioning. The sensors are build into the shoe which makes it hardly reusable for wearers of different shoe sizes.

An automatic detection of waltz and tango steps.

In a small project the shoes have been used to distinguish between tango and waltz steps, initiating the playing of appropriate music after detection. The authors even saw this project as a

> first step in a promising trajectory of applying more sophisticated analysis for extracting higher-level features that may be useful in dance and sports training or podiatric therapy.

Unluckily, this project was not carried forward.

Urs Enke did another study using sensors on dancers [Enke, 2006]. In his work he uses acceleration sensors to determine rhythms that dancers are moving to. The extracted rhythms then could be used to search for appropriate songs within a database or to manipulate currently played music.

Wearable sensors are not only interesting in the domain of dancing. In a more general article, Knight et al. [2007] show a variety of areas in which acceleration sensors can

be used, e.g., for coaching sport activities or human movement research. For coaching activities, accelerometer data might be used to create an individual training schedule. If, e.g., accelerometer data of a soccer player shows that he has a high amount of short sprints, additional training on short sprints might be beneficial to his performance during a game. For coaches of endurance athletes, acquired sensor data might give more detailed information about the actual training performance than the trainee's subjective feeling. In these settings, the benefit to the trainee is very indirect as the sensor system helps the teacher to create a better training schedule but does not influence a training session directly. Knight gives another example in which the trainee would benefit more directly from gathered data: A javelin thrower could benefit from a visual representation of his own acceleration data during a throw if he can compare it to those of a better javelin thrower.

Sound feedback has been used in a karate training system.

In martial arts, Takahata et al. have taken the idea of comparing acceleration data from beginners to those of highly skilled individuals to another level. Their system for karate training [Takahata et al., 2004] produces sound feedback depending on acceleration data of certain body parts. While training Tsuki[1], devices on the wrists, ankles, and the waist measure accelerations and produce sound feedback. This helps trainees to distinguish better trials from worse ones. The authors argue that, for beginners, detailed instructions about the correct movement are hard to convert into practice and that sound feedback is another, possibly better way, to understand how the desired motions have to be executed. Apart from the training on specific movements, the authors also experimented with music and sound to support the natural rhythm of the trainees. Subjective evaluations and questionnaires showed a positive effect of the sound system on motivation and pleasure, but there is no proof that the given feedback improves actual learning as well.

Motion chunks: decomposing human movement into elementary motions.

Kwon and Gross developed another training system in the domain of martial arts [Kwon and Gross, 2005]. They introduce the idea of motion chunks in order to analyze human movement more easily. The idea of motion chunks is

[1] a Karate punch

that complex human movements can be decomposed into elementary motions, similar to human speech, which can be decomposed into elementary phonemes. A similar approach seems to be promising in the domain of ballroom dancing as well, as complex dance steps are normally already described as a series of elementary steps.

Haptic feedback is being researched in a realtime snowboard training system.

Training systems are not restriced to the martial arts. In the domain of snowboarding a system is being developed by Daniel Spelmezan [Spelmezan and Borchers, 2008]. The system works wirelessly via bluetooth and a mobile phone, and uses sensors in the shoes to measure the weight distribution of trainees. Feedback is given by small vibrational devices mounted on the body. As this system is developed at the same chair this thesis was written at, we were lucky to have a system with comparable technical affordances available to learn from.

A dancing robot for human-robotic interaction research.

Kosuge et al. [2003] developed a dancing robot that is capable of taking the female part in a waltz. The work was done to research human-robotic interaction and not to teach dances, however, the fact that the robot can distinguish several different dance figures and react accordingly with a success rate of about 85 percent shows that realtime adaptive feedback in dancing should be possible.

Nakamura's dance training system uses vibrational devices but no sensors.

In the domain of dancing, a training system was developed by Nakamura et al. [2005]. The system supports a trainee of a japanese folk dance in three different ways. A screen shows a model dancer performing the dance. The screen is mounted on a moving robot. Thus, the dancer can move into the desired direction without losing sight of his model, and without wrong estimations about the distance he has to move. The third part involves vibrational devices which are mounted around the trainee's joints. During an evaluation of the system, vibrational cues were given 0.5 seconds in advance to the correct timing of an action. The vibrational cues given by the system are unique compared to the other described system, as their timings do not depend on the dancer, instead they are determined by the timings at which the dancer should execute a specific move. As there are no sensors involved in this system, it cannot adapt to the dancer's performance. The authors claim that the vibrational devices are effective in teaching timings of actions,

but this is arguable. The evaluation was done with only six subjects and a move was considered a success if a correct action was started within one second after a vibrational cue. One second is a time span that lies within a human's reaction time. Eventually the dancers did not learn when they had to move their arms and simply reacted to the given cues.

We can assume that a sensor based system to support dance beginners does not yet exist. A couple of systems exist that make use of sensors to support beginners of a motion skill, but these are not settled in the domain of dancing. In the domain of dancing, existing systems are either not using sensors or they have been used to support advanced dancers in their performance but not beginning dancers in learning.

2.2 Learning motor skills with concurrent feedback

Concurrent feedback increases performance but decreases learning.

In the field of cognitive psychology, experimental studies about the effect of concurrent feedback most often show that concurrent feedback leads to an increased performance but decreased learning. Winstein et al. [1996] showed this with a partial weight bearing task. Subjects had to support 30 percent of their body weight while stepping onto a floor scale. The test persons were divided into three groups. The first group received concurrent feedback during their trials (using a modified analog display of a scale), the second group received feedback after each trial, the third group received feedback after every five trials. In a retention phase two days later, all subjects performed the same task without feedback. During the training phase subjects from the concurrent feedback group showed better results than subjects from groups two and three. Two days later, though, subjects from group one performed worse than subjects from group two and three. Schmidt and Wulf [1997] got similar results for another task: In their experiment subjects had to tilt a plate in a specific pattern over time. Linden et al. [1993] had subjects exert force in five second patterns. They got the same results, and additionally, in their experiment

less frequently given knowledge of results lead to a better learning than more frequently given knowledge of results.

Self-control increases learning.

McNevin, Wulf, and Carlson [McNevin et al., 2000] give an overview over studies concerning three factors that have influence on the learning of motor skills. Self-control of learners about the amount of feedback they receive, the learning schedule, or the use of equipment, showed positive effects on motor skill learning in a throwing task, a barrier knockdown task, and a ski simulator task. Besides, they found out that learners demanded less and less feedback the longer they trained if they were in control about the amount of feedback.

An external focus of attention is superior to internal focus of attention.

The second important factor described in this paper is discussed in much more detail by Wulf [2007]. Wulf distinguishes between internal and external focus of attention. Roughly speaking, adopting an internal focus of attention means that you are concentrating on your movements, while adopting an external focus of attention means that you are concentrating on an effect of your movement. The author discusses many experiments, conducted by herself as well as other researchers, that were settled in the domain of golfing, balancing, basketball, and many others. Results showed that an external focus of attention is superior to an internal focus of attention both in performance and in learning. A growing distance of an external focus seems to lead to even better results, but this might not be applicable to all subjects equally. While a beginning golfer might be able to concentrate on his club's swing instead of his arm's swing, he will most probably not benefit from concentrating on the area he wants to bat the ball to. A more experienced golfer probably will. This might be explained by the fact that a beginning golfer does not yet have enough experience about the connection between his battings and the flight of the ball.

Though no experiment described in this work was settled in the domain of dancing, and its author was not aware of any experiments in this or closely related domains either[2], these results are interesting when we have to decide between several different kinds of feedback for Saltate!.

[2] Reply to a personal email

Summing up, we can say that the learning of motor skills has been researched well enough to make design decisions based upon these results. The most important facts are:

Psychological research progressed enough to serve as hints for design decisions.

- Concurrent feedback has a strong effect on performance and learning of motor skills: performance increases but learning decreases if concurrent feedback is permanently given to the learner.
- Learners should adopt an external focus of attention rather than an internal focus of attention, as this increases both performance and learning.
- Learners should be given some freedom of choice considering the learning schedule, amount, or kind of feedback they receive.

2.3 Perceived simultaneity

In dancing, people perform steps to specific events within the music. These timings do not necessarily fall together with the timings of the music's beats. Within this project, we want to measure time differences of steps and beats to decide whether the dancers dance correctly, or, eventually, to determine the quality of their timings on a finer scale than just right and wrong. It is therefore interesting to know which time differences between different events are still perceived as simultaneous by humans and which time differences are not. If, e.g., all steps within a dance should be made exactly on the beats of the music but humans perceive the acoustic events of the music and the visual or haptic events of a step as simultaneous within a 50 ms interval, it wouldn't make sense to correct them should all of their steps take place within a 100 ms interval around the music's beats.

Levitin et al. [1999] studied the perception of simultaneity between acoustic and haptic, and acoustic and visual stimuli: In their experimental setup they manipulated the asynchrony between a drum strike and its acoustic response. The actual timing of the drum strike was calculated by the

movement of the stick before it hit the drum. Asynchronies in a range of 200 ms were studied. During the experiment one subject (with earphones, blindfold, and the stick) had to hit the drum and decide afterwards whether he accepted the acoustic response given by the earphones as simultaneous or not. A second subject with earphones watched the first subject while hitting the drum, received acoustic response, and made the same decision. The first subject's decisions were used to determine perceived haptic-acoustic simultaneity thresholds, the second subject's decision to determine perceived visual-acoustic simultaneity thresholds.

Each pair of subjects changed roles after each 90 trials and completed 720 trials over three days. Overall, eight subjects were tested. The results are graphically shown in figure 2.1. What can be seen on these curves is that the time-difference intervals in which a significant amount of trials were considered as synchronous is very big: even at an asynchrony of 100 ms over 20% of the trials were considered as synchronous for haptic-acoustic intermodality, with even higher acceptance rate for visual-acoustic intermodality. The 75% acceptance rate proposed by Levitin would lead to approximated perceived synchrony thresholds of about 40 ms.

Even events with asynchronies of up to 100ms can be perceived as simultaneous.

Saskia Dedenbach [2008] gives a more detailed discussion of several other studies concerning perceived synchrony ([Wright, 2002], [Mäki-Patola and Hämäläinen, 2004], [Dahl and Bresin, 2001]). She identified five factors that influence latency detection in human-music interaction:

- Individual cognitive skill
- Musical experience
- Type of provided feedback
- Characteristics of the music piece
- Nature of instrument sound

We assume that most of these factors are valid in the domain of dancing as well. If so, providing successful help or

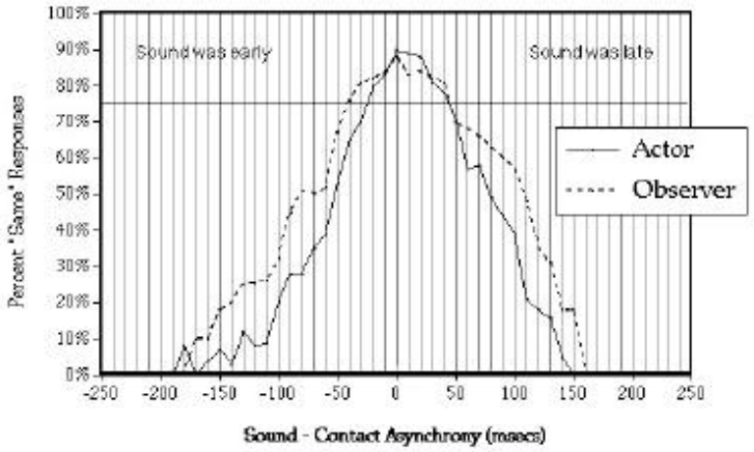

Figure 2.1: Acceptance percentages for different asynchronies between acoustic/haptic (Actor) and acoustic/visual stimuli (Observer). The horizontal bar at 75% acceptance rate marks the (arbitrarily) chosen value for perceived simultaneity.

feedback based on small timing differences might become hard or even impossible as we might have to take into account subjective characteristics of the played music piece, musical or dancing experience of the dancers, and individual cognitive skill.

Chapter 3

Excursus: music and dancing terms, beginner problems

"In the animal kingdom, the rule is, eat or be eaten; in the human kingdom, define or be defined."

—Thomas Szasz

In this chapter we give a short overview over some concepts and terms of music notation and dancing in general, Slow Waltz in particular, and mention the most important beginner problems in dancing and music playing.

3.1 Musical notation

In western music notation, several notes form a *bar* (also called *measure*). The division into bars is not arbitrarily, the kind of bars used - the *time signature* of a music piece - gives a lot of information about the underlying rhythm of the song. The time signature of a song is described by a nominator and a denominator: a song might be written, e.g., in $^2/_2$ time, $^4/_4$ time, or $^3/_4$ time. The nominator tells us how many notes are played per bar, the denominator their relative length when written down. During a bar of

a song written in $^3/_4$ time percussion instruments usually play three notes of $^1/_4$ length. Instruments that are used for the melody and not the rhythm part of this song would play notes of different lengths very often, but for each bar in this song their lengths (and the length of brakes) will sum up to $^3/_4$.

Most pop and rock songs are written in $^4/_4$ time: While listening to such a song, you can repeatedly count from one to four. In songs written in $^4/_4$ time, usually notes on the first and third beats are played by another instrument than notes on the second and fourth beat. For all time signatures usually the first beat is stressed more than the other beats.

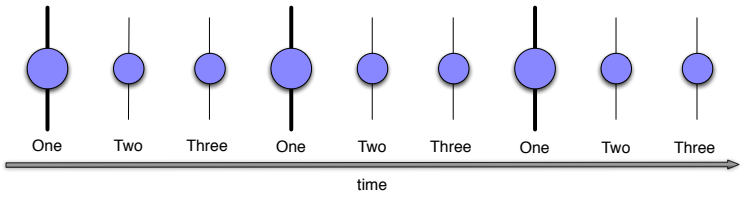

Figure 3.1: A visualization of the typical waltz ($^3/_4$ time) rhythm: The bigger circles indicate the stressed first beat of each bar

We have uploaded examples for music pieces written in $^4/_4$[1] and in $^3/_4$ time signature[2].

The time signature of a song does not tell us how fast a song is played. This information is normally given by the amount of *beats per minute*.

3.2 Dancing terms

Ballroom dances are performed by couples. They consist of a *basic step* and different amounts of *dance moves*. All of these can be described by a sequence of *elementary steps* and further information, for example about the rotations of the dancers. Usually the man decides during the dance

[1] http://media.informatik.rwth-aachen.de/~drobny/saltate!/mmSaltate44timeSig.mp3
[2] http://media.informatik.rwth-aachen.de/~drobny/saltate!/mmSaltate34timeSig.mp3

which dance moves are performed and the woman follows his lead.

In dancing, the speed of the music is normally referred to as *bars per minute*, not beats per minute. Dances are closely linked to the rhythm and speed of the music. A specific dance can be performed to music in the correct time signature within a certain speed range. A Jive is usually performed to music in $^4/_4$ time with a speed of 32 to 44 bars per minute, while a Tango is usually danced to music in $^2/_4$ time at a speed of about 32 bars per minute.

3.3 Slow Waltz

Slow waltz is performed to music in $^3/_4$ time at a speed of 28 to 30 bars per minute. It is danced in *closed dance frame*: The dancers stand in front of each other, each positioned slightly left of the dance partner in such a way that each one's right foot can move between the partner's feet. The man's right arm is on the lower part of the woman's left shoulder blade, the woman's left arm lies on the man's right arm. The other hands are held together in height of the smaller dancer's eyes, the arms are bent and pointing away from the dancers.

The Slow Waltz' basic step spans over two bars of the music: On the first beat of the first bar, the man steps forward with his right foot, followed by a step to the left with his left foot on the second beat and a closing step with his right foot on the third beat: The right foot gets positioned right next to the left foot. The woman's steps mirror the man's steps: A step back with the left foot, a step to the right with the right foot, and a closing step with the left foot. On the second bar, the man's and the woman's steps are swapped: The man starts with a step back with his left foot, a step to the right with the right foot, and closes with his left foot, the woman mirrors these steps. After this, the couple stands on the same position as before. In the basic step, no rotation is involved.

The easiest dance move of Slow Waltz is the *natural turn*.

Figure 3.2: Closed dance frame.

It is used during evaluation of Saltate!. During a natural turn, the couple performs a rotation into clockwise direction and leaves its place. The first step of the clockwise rotation is identical to the first step of the basic step. The man's second step is set forward into direction of the first step. With the last step, the feet are closed again. Rotation takes place beginning with the second step and ending with the closing third step. During the next bar, the same is done with the man walking backward, starting with his left foot. A correct natural turn includes an overall rotation of 270 degrees, but it is easier to turn through 180 degrees. The step sequence in figure 3.4 shows the steps performed with only 180 degrees rotation as they were performed during Saltate!'s evaluation. For a more detailed description, see Moore [1982].

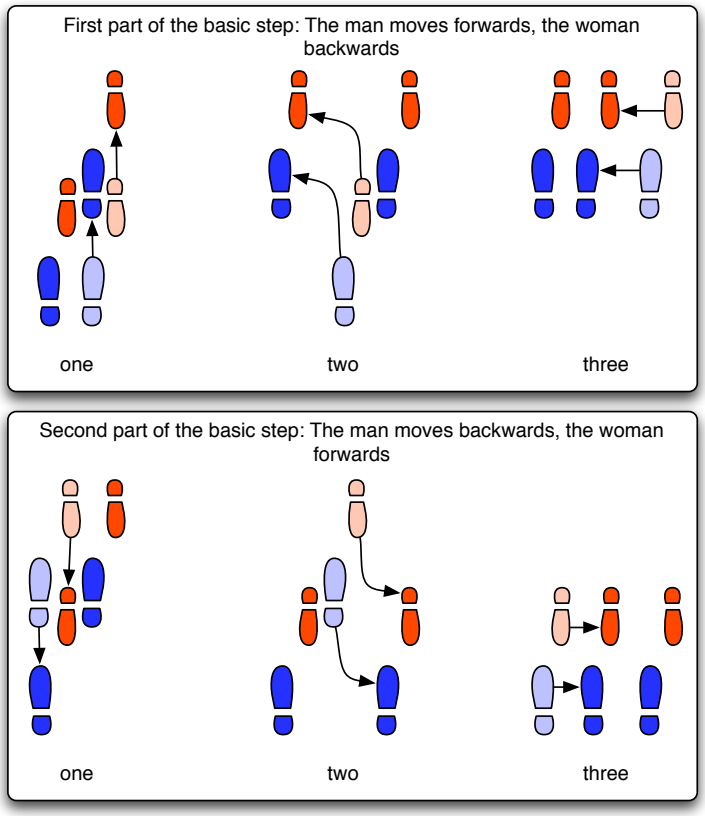

Figure 3.3: The Slow Waltz' basic step.

3.4 Beginner problems

Interviews with dance and music teachers showed that beginning dancers and beginning musicians (at least when playing percussion instruments) face similar problems: The exact timing of actions is harder than the actual execution, even if they are performed at slower speed. While dancers have to execute their actions to the music's rhythm, percussionists have to generate rhythm. In both cases, beginners have to develop their "inner clock".

At the beginning, many dancing teachers count along with the beats of a song in order to help beginners to start dancing or to keep dancing. Alternatively, verbal hints are

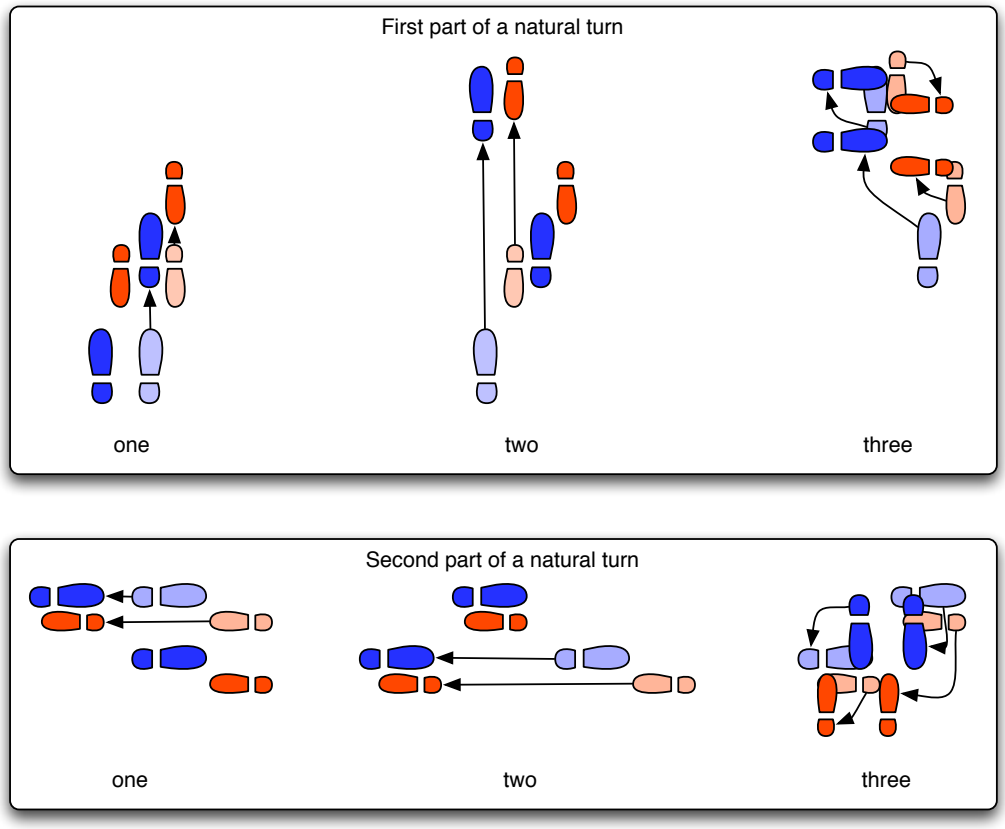

Figure 3.4: A simplified natural turn as it was used in Saltate!'s evaluation: a correct natural turn would include a 270 degree rotation instead of a 180 degree rotation.

given in the rhythm of the music ("quick, quick, slow"). Once dancing, beginners often do not face great problems to dance in step. It seems that following an induced rhythm is easier than to detect it autonomously.

For beginning percussionists, practicing with a metronome is not unusual. The metronome provides a cue for regular time intervals that the musician can practice to. It is important to note, however, that excessive use of metronome exercises has a detrimental effect on the musician's sense of timing, as he begins to rely on the metronome and does not use it as a test for his own timing anymore.

Chapter 4

Hardware development

"Hardware: the parts of a computer that can be kicked."

—Jeff Pesis

In this chapter, we describe the development process of Saltate!'s hardware, from general design decisions over a short description of important development prototypes to a description of the final version.

4.1 General design decisions

The first decision: Which sensors should be used?

As the very first decision we had to choose which sensors to use. As already described in 3.4—"Beginner problems" the most important problems beginning dancers regularly encounter are related to the timing of steps. In order to give support for these problems, we need to use sensors that enable us a precise measuring of step timings. Mounted in or under the shoes, force sensors and acceleration sensors are a promising approach: While a foot is standing on the ground, an acceleration sensor would measure no acceleration; while a step is executed, acceleration should increase and decrease. Force sensors mounted under the feet would measure a high force when the foot is standing on the ground, and less force when the feet is in the air. Both approaches have their benefits and possible drawbacks.

Pros and cons of acceleration and force sensors.

While acceleration values of a standing feet should be within a very close range around zero, acceleration values close to zero are also possible during a step. Force sensor values would not be within such a close range while standing on them, as the force exerted on them depends on the dancer's weight and weight distribution. Thus, we have to expect different values for different dancers. On the other hand, we can assume that force sensor values will always be lower when the foot is in the air than when it stands on the floor. Acceleration sensors could be used to determine the direction of a step, while force sensors cannot.

A test with sensors from a snowboard system showed good results for force sensors.

We had the opportunity to get a look at sensor values of ourself walking forward, backward, and performing some dancing steps using a part of Spelmezan's snowboard system [Spelmezan and Borchers, 2008]. The sensors used during this test session were force sensors mounted under the heel and ball, and an acceleration sensor mounted on the shoe. Force sensor values changed between high values while standing on them to very low values while the foot was in the air quickly and reliably. The acceleration sensor's values were much harder to read. After this test we were confident that force sensors mounted under the feet would allow for automatic step detection and decided to use these in our first approach, but to keep the ability to add additional sensors to the system. We felt this flexibility might be necessary should force sensors not suffice, or in case there was enough time left for adding other sensors to gather additional data, e.g., acceleration sensors to detect the direction of steps.

In order to disturb dancers as little as possible in their movements and concentration, we decided to spend a lot of effort in keeping the hardware small and light. Technically, the hardware used in the snowboard system would suffice for Saltate!'s purposes as well. There, sensors from both feet are wired up to a belt bag containing batteries and an Arduino[1] Bluetooth module that transmits sensor data wirelessly. However, we feared that the belt bag's size and the cables' wiring over the complete length of the legs would feel strange when worn to normal clothes while dancing.

[1] http://www.arduino.cc Arduino is an open-source electronics prototyping platform based on flexible, easy-to-use hardware and software.

Figure 4.1: Wires and belt bag of the snowboard training system.

This is probably a smaller problem in snowboarding, where you wear a ski suit.

The sampling interval of sensor data sampling should be at most 50ms.

Sensor data must be sampled frequently enough to keep the error for measured step timings small. As shown in 2.3—"Perceived simultaneity", some research results show that people do not necessarily perceive different events as asynchronous if the time difference between them is small enough. We decided to aim at a sampling interval of at most 50 ms. A better accuracy did not seem to be necessary for the calculation of feedback, as many people probably will not detect asynchronies of less than 50 ms regularly anyway.

Based on these thoughts, we set requirements for Saltate!'s hardware:

- The hardware worn should support force sensors under the feet's heel and ball with a sampling interval not longer than 50 ms.
- It should be expandable to further sensors.
- Sensor modules must be able to transmit acquired data wirelessly, at least 20 times per second.

- Sensor modules should be small and light enough to mount them onto a shoe without disturbing the wearer (Unless fulfilling this requirement would take an unaffordable amount of time).

4.2 Engineering

In this subchapter we give an overview over the hardware deployed and explain for what reasons the components were chosen, present prototypes built while developing the system, and discuss the final hardware version in greater detail.

4.2.1 Deployed hardware

The Arduino Mini: Functionality and flexibility as the Arduino Bluetooth and very small.

The Arduino Bluetooth modules used in the snowboard training system in combination with force sensors fulfilled all the requirements introduced in 4.1—"General design decisions" except for the desired size. Their size is 8.2cm x 5.3cm. The Arduino Mini[2] offers the same functionality with a size of only 3.1cm x 1.8cm. It cannot communicate wirelessly on it's own, so in addition a radio module has to be attached to it.

The Arduino Mini communicates through a serial interface using one input and output pin. Additional requirements for radio modules in Saltate! are small size and low energy consumption. Range is not very important, as the system is intended to be used within one room. The maximum data rate of Arduinos is 115.200bps (bits per second). Sensors attached to Arduinos can be sampled with a ten bit voltage resolution, a maximum of eight analog inputs can be attached.

Assuming four sensors per Arduino (two pressure sensors plus acceleration sensors in two directions) the data rate of 115.200bps would allow to send all acquired sensor data theoretically 2.880 times per second:

[2] http://arduino.cc/en/Main/ArduinoBoardMini

$$4 * 10bit * 2880s^{-1} = 115.200bps$$

In reality, this rate will be lower: If we use only one receiver, we have to share the radio channel among four Arduinos for a dance couple, and there is additional overhead in the radio protocol. However, a sampling rate in the range of milliseconds seems possible, which is good enough for our purposes.

Saltate! uses XBee modules for radio transmissions.

XBee modules from Digi[3] are small in size (2.44cm x 2.76cm), known to work with Arduinos, and designed for low power consumption. The standard Xbee has a range of 40m indoors, a data rate of up to 250kbps, and a maximum operating current of 40mA. They operate on a 3.3V basis.[4] Power consumption for other radio modules are generally higher.

For Arduino Diecimilas[5] , which can be connected to a computer via a USB connection, adapter shields for radio transmissions[6] exist that use XBee modules. Since other radio modules could only slightly decrease the power consumption, we decided to develop Saltate! using Arduino Minis and XBee modules for a star-topology radio network.

Deployed sensors: Force sensing resistors.

We use force sensing resistors (FSRs) as force sensors. Technically, these sensors are components that change their electrical resistance depending on the force exerted vertically onto their sensor area. The higher the force exerted, the less the electrical resistance. Figure 4.3 shows our sensors with opened and closed surrounding construction.

To use them with an Arduino, you have to connect one of the two FSR connections to the supply voltage, and the other to an analogue input pin of the Arduino and, with an additional resistor in between, to ground. The Arduino can then measure the voltage on its analogue input which changes when the force exerted on the sensor changes.

[3] http://www.digi.com/products/wireless/zigbee-mesh/xbee-series2-module.jsp

[4] values taken from the XBee documentation

[5] http://arduino.cc/en/Main/ArduinoBoardDiecimila

[6] http://arduino.cc/en/Main/ArduinoXbeeShield

4.2.2 Development prototypes

Radio network

At first, we tested power supply and communication,...

After Xbee communication worked successfully using prebuilt Arduino Diecimilas and XBee shields, we built a first battery powered prototype using an Arduino Mini and a XBee module. The purpose of this prototype was to test the construction that connects Xbee and Arduino Mini to each other (for communication) and to the power supply (the Xbee module operates at 3.3V, the Arduino Mini at 5V).

First sensor readings

... then step recognition, using a first prototype with sensors.

Once the radio networking prototype worked successfully, we soldered it, put it into a box, and attached two sensors to it. With this prototype we were able to sample data of one foot and test step recognition algorithms. Mounting this sensor box onto a shoe was not easy[7], but once mounted, it didn't disturb the wearer much. Therefore we developed the final box physically in such a way that it could be mounted on top of a shoe, but with less effort than this prototype.

4.2.3 Final hardware version

Technically, the prototype used for first sensor readings worked fine. Based on our experience with it, we decided to improve the following aspects:

- Reduce the size of the box
- Reduce the amount of batteries needed
- Make it physically robust
- Add a switch to turn the box on and off

[7] We used an additional shoelace to fix the box on the shoe

- Add a status LED
- Add connectors for the sensors
- Find a way to allow a comfortable and stable attachment to a shoe

Saltate! sensor box internals

Saltate!'s sensor boxes consist of five components:

1. Batteries and power switch:
   Two 1.5V AAA (micro) batteries are connected in series to a power switch, serving as a power source of 3V.

2. DC/DC converter:
   We use a LT1073 converter from Linear Technology[8] to transform the 3V input voltage from the batteries to 5V.

3. Arduino Mini:
   The Arduino Mini is connected to the 5V input, with RX (receiver pin) and TX (transmitter pin) for communication to the radio module, and with four analogue inputs to the sensor connectors. A status LED is connected to one of the Arduino's digital input/output pins.

4. Radio module:
   The Xbee module is mounted onto a XBee Simple Board[9] that serves as a hardware interface to the rest of the system. Using the Simple Board has two benefits: It serves as a physical adapter (the Xbee modules have a different pin distance than the rest of the components) and transforms a variety of input voltages to 3.3V, which is required by the Xbee module. Thus, both the Arduino and the XBee module can be connected to the 5V of the DC/DC converter. The communication pins of the simple board are connected to the communication pins of the Arduino with a 10kΩ

[8] http://www.linear.com
[9] http://www.droids.it/990_001.html

resistor between Arduino TX and Xbee RX as an overvoltage protection. The 3.3V output of the Xbee is detected by the Arduino's input without problems.

5. Sensor connections:
   Externally, the sensors are attached via a D-SUB9 connector. Internally, they are connected to 5V supply voltage, ground, and the analogue input pins of the Arduino Mini.

Figure B.1 and B.2 in appendix B—"Pictures and sketches" show sketch and figure of Saltate!'s internal electronics.

Sensor box

Our final sensor box has about the size of a cigarette packet.

The sensor boxes consist of an upper and a lower part which are screwed together. They are only slightly bigger than their internals. Thus, it was not necessary to glue or screw the battery holder or the electronics part onto the boxes' walls, a blister foil was sufficient to prevent the parts from displacements while the box is in motion. To attach the box to a shoe, we cut a flexible ruler into size and glued it's ending under the box. The ruler can be pushed under a shoelace, providing an easy, yet stable, hold. Overall, Saltate!'s sensor boxes have a size of 8.6cm x 6.5cm x 2.7cm (without ruler and sensors) and weigh 119g (including batteries and ruler).

Sensors

Two sensors are connected to a D-SUB9 connector for an easy de- and attachment to the sensor boxes. While in use, one sensor is placed under the heel, and one under the ball of a foot. This is done by sticking them under shoe insoles which are then placed into the shoe. If the sensors are used without modifications, received sensor data are pretty low, especially for those sensors placed under the balls. To improve sensor readings we added a simple construction to the sensor: On the sensor's upper side, a small cardboard is mounted, with a small bump right over the active sensor

Figure 4.2: One of our sensor boxes mounted on a shoe

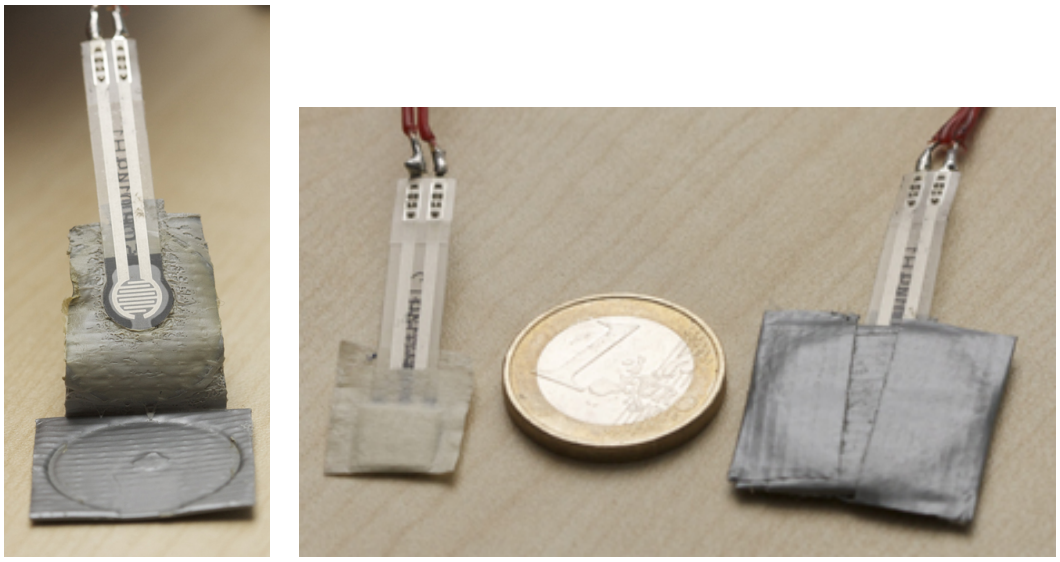

Figure 4.3: Pictures of our sensors: A ball sensor with opened surrounding construction on the left side, closed constructions for heel and ball sensors on the right side.

area. (The sensor area is surrounded by a comparatively hard ring. Thus, a hard plate over the ring would inhibit any forces from reaching the sensor area.) On the lower side, a hard background is placed. In case of the ball sensors we use a 5 cent piece, in case of the heel sensor another cardboard. This makes the distinction of the sensors easier.

Radio receiver

To receive data from the sensor boxes and communication with the computer, an Arduino Diecimila with Xbee shield and demounted microcontroller is attached via USB cable to the computer. (With demounted microcontroller, the computer communicates directly with the XBee module)

Chapter 5

Feedback design

"For every complex problem there is an answer that is clear, simple, and wrong."

—H. L. Mencken

In this chapter we lay down a couple of ideas for different kinds of feedback and discuss their assumed benefits and drawbacks. The most promising feedback was chosen for Saltate!'s implementation.

Feedback depends on a user's actions, supporting functions do not.

Within this theses we speak more often of supporting functions than of feedback. In our understanding there is a difference, as supporting functions could be implemented without sensor equipment, whereas feedback cannot. Supporting functions, however, can become activated or deactivated based on a user's actions. When we speak of supporting functions within this thesis we always mean the latter case: Some kind of system output whose intensiveness is modified by the user's actions, but which in itself is not dependent on user input. The feedback term is used for any system output that is in some way modified by a dancer's actions.

If a system played a sound each time a dancer's foot touches the ground, this would be a feedback function, as it cannot become active if the dancer does not move, whereas a light that sends a light pulse to each beat of the music is a supporting functions that takes over feedback character-

istics if its intensiveness depends on a dancer's steps.

5.1 Feedback ideas

We divided our different feedback ideas based on their sensor modality:

5.1.1 Haptic feedback

Vibrational devices or actuators that are able to exert pressure on body parts, e.g., if they are fastened around a foot link, could be used to indicate the leg or foot that has to be moved.

5.1.2 Visual feedback

We developed a couple of ideas for visual feedback: Modifications of the surrounding illumination could be used to create a relaxed atmosphere when the dancers dance well, or a light source could send pulses to the rhythm of the music. Visual stimuli could be projected to the ground or, theoretically, to the eye, to indicate the next position to which a foot has to be set.

5.1.3 Acoustic feedback

Acoustic feedback could take very different forms. First of all, the system could use recorded speech to talk to the dancers like a dance teacher. Then, the system could modify the music currently played so that it sounds nicer if the couple dances better. Dancers' steps could generate sounds that should match to the music's beats if they dance correctly. Finally, the music's beats could be emphasized.

In addition, we thought of something with a game character: A rating for each dance, or cheering "spectators" if a

couple dances well.

5.2 Feedback discussion

We want to direct the dancers' attention towards the music.

As a starting point for discussion about the assumed effectiveness of our feedback ideas we used the results presented in chapter 2.2—"Learning motor skills with concurrent feedback": Concurrent feedback usually has a negative effect on learning, and an external focus of attention is superior to an internal one. Together with our findings from chapter 3.4—"Beginner problems", that beginners usually have more problems to find the correct timing for their steps than to physically execute the steps, we decided that Saltate!'s feedback should avoid to direct the dancers' attention to their feet while they are dancing, instead it should direct their attention to the music.

Based on this decision, we did not take haptic or visual feedback into serious account. Visual feedback will draw the dancers' attention away from the music, and haptic feedback will draw the dancers' attention towards his feet or other body parts.

We do not believe that haptic or visual feedback is useless for all kinds of dancers: Experienced dancers who have no problems to dance to the beat of the music might benefit from feedback provided in these modalities, but they do not belong to Saltate!'s target group.

We did not use speech output for several reasons: Saltate!'s abilities to detect different kinds of errors are very limited compared to a human teacher. Speech output might easily raise expectancies Saltate! cannot fulfill. If the same mistakes were corrected very often, the impression of a simulated human teacher would also have to break down very quickly: A system's capabilities to slightly modify voice and wording for similar sentences are very limited compared to a human.

We rejected the idea of step generated sounds as well. As soon as these are played, they might direct a dancer's at-

tention to his feet again. Also, we have to take into account that we have dancing couples, not single dancers. Slight differences in step timings might result in strange sound experiences if they are directly transformed into sounds.

The idea of modifying the played music in order to sound nicer has a clear weakness: If a couple performs badly the music has to sound weird as well. We are sure that this would not help a couple to dance better.

Saltate! emphasizes the music's beats.

The last idea of acoustic feedback shows none of the discussed weaknesses: Emphasized music beats will direct a dancer's attention to the music. They also help to identify the rhythm of a music piece. We assume that a feedback that quickly draws a dancer's attention has a higher risk of beginners to become dependent of. Emphasizing beats can be done rather slowly, so that it is perceived as a change in the music instead of an extra feature.

Feedback with a game character might have a positive influence on dancers' motivation. However: Evaluating different kinds of feedback individually would have been impossible within the time available for this thesis. Since we assumed that emphasizing music beats is more promising than a game, we decided to develop this kind of feedback.

5.3 Saltate! feedback

After our decision for a specific kind of feedback - emphasized music beats - we had to decide how this emphasis should be implemented.

Saltate!'s feedback does not disturb the character of a waltz.

The typical waltz rhythm contains of a stressed first beat and two following beats. In order to not disturb this characteristic with our emphasized beats, we decided to use one sound for the first, and another sound file for the second and third beat of each bar. Very quickly, we decided to use a soft base drum sample for the first beat of a bar. We experimented with different base drum samples as well as other instruments for the second and third beat for a while, in the end we used a sample from a rather soft high hi-hat.

Both sound files are easy to distinguish and do not disturb the character of a typical waltz song, unless they are played back very loud.

Chapter 6

Software development

"Software is like entropy. It is difficult to grasp, weighs nothing, and obeys the second law of thermodynamics; i.e., it always increases."

—Norman R. Augustine

In this chapter we identify software requirements for Saltate!, discuss design decisions, and give an overview over the actual implementation.

6.1 Software requirements

Since, at the beginning of this project, it was unclear how much time would be available to develop different kinds of feedback, we identified requirements that are necessary for all kinds of feedback and guidance we imagined, and tried to build the software flexible enough to allow the implementation of different kinds of feedback. There are three main requirements that the Saltate! software has to fulfill:

- Saltate! must be able to play music files.
- Additional information about the songs played, like the timings of the beats within the music, must be available.

- Timings of sensor data have to be comparable to timings within the music.

In addition of fulfilling these requirements, several other questions influencing software design had to be answered:

- Which minimum accuracy should sensor data have?
- Which latency is acceptable from the performing of steps to their recognition and a possible reaction of the system?
- How is dance move identification implemented: Does Saltate! use files that can be changed to modify detection behavior or to add new moves to the system, or is it necessary to change program code to do so?
- Which values are used to determine how well or badly dancers perform?

We discuss the answers to these questions in the following paragraphs.

The accuracy of sensor data should be equal or better than 50ms.

The acquired accuracy of sensor data timings depends on the sampling rate and on further methods of forwarding data. Based on the findings about perceived simultaneity (2.3—"Perceived simultaneity") we wanted to achieve an overall accuracy of 50ms: therefore, we defined in 4.1—"General design decisions" that the sampling interval must not be longer than 50ms. Related to the question of sensor data accuracy is the question of latency: If we forward data as soon as it gets sampled, we would get a minimum latency. However, this approach might decrease accuracy: In radio networks, devices cannot necessarily send data when they want to; if other devices send data at the same time, data transmission can take place only after the other device has finished its transmission. That means, the time needed from sampling of data until it is received by the software fluctuates. In order to maximize accuracy, we decided to send data including their time stamps, and not to use the arrival time in the Saltate! software as the basis for further calculations.

Our chosen feedback, which principles we explained in chapter 5—"Feedback design", does not require a very fast step recognition, as the feedback's volume should change rather slowly. A latency of 100ms between an event and a possible system reaction would be acceptable.

The latency of sensor data reception should not be bigger than 100ms.

For a fully developed dance training system, i.e., a system that is not restricted to beginners, functionality to teach the system new dance figures would be very desirable.

Though it was not developed with the aim of being used in a dance training system, we suppose that Exemplar, developed by [Hartmann et al., 2007], could be used to teach a system new dance moves: Exemplar was developed to allow rapid physical prototyping. It is possible to demonstrate movements of sensor based devices and Exemplar identifies the same movements afterwards. A human developer does not have to manually analyze sensor data any more. The same should be possible for dance moves if sensor data used as input are taken from dancers. Exemplar has already been used in one project by Zhang and Hartmann [2007] to allow users to teach the system new motion gestures. Within this thesis such a feature would be nice, but is not necessary, i.e., it has a very low priority.

Since Saltate!'s target group are beginning dancers, it focuses on step timings; interesting derived values are: average time differences of the dancers' steps to the beats of the music or the dancers' standard deviation of their average timing difference. We want to mention that an average timing difference of zero is not necessarily the best dancing style, "correct" step timings have to be acquired from experienced dancers. Besides the timing of the steps, Saltate! should of course decide whether steps are correct at all. Even a perfectly timed step is wrong if it was done with the wrong foot.

We identified step timings, average time differences and standard deviations as possible variables for a performance evaluation.

6.2 System design

In this subchapter, we describe the tasks of Saltate!'s subsystems in detail.

6.2.1 Overall system structure

Saltate! consists of two main parts: Sensor boxes and Saltate! software. The sensor boxes are used to acquire sensor data and forward it wirelessly to the Saltate! software, which plays the music and decides according to incoming sensor data if, and how, feedback is given to the dancers.

6.2.2 Radio network

The radio network has a star-topology: A central station connected to the computer communicates with all four sensor boxes, the sensor boxes do not communicate with each other.

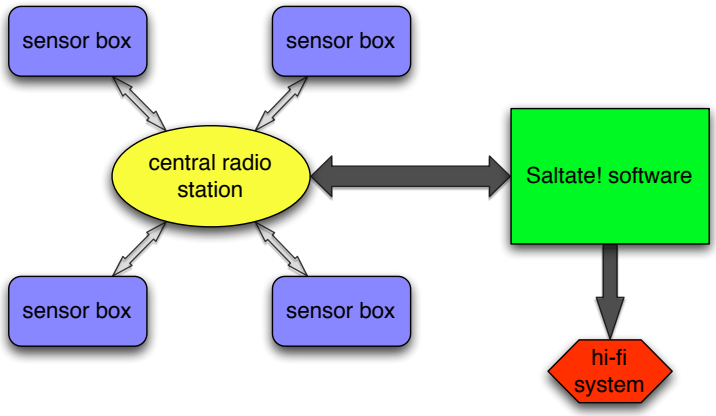

Figure 6.1: Saltate!'s radio network topology and its connection to the software and hi-fi system

6.2.3 Song information retrieval

We used recorded music and marked beats manually.

Saltate! has to retrieve information about beat timings of played music. If recorded music is played, these timings have to be marked either manually or automatically. If midi files are used, determining beat timings will not suffer from objective inaccuracies, as the exact timings of all notes are

known. However, as Saskia Dedenbach [2008] found out, characteristics of a music piece and the nature of instrument sound have an influence on human delay detection. Eventually humans would, on average, not mark the exact timing of a beat as defined in a midi file as the timing of this beat. We do not claim that humans will perceive automatically marked beat timings from midi files as wrong, but we are not sure that they will accept them as correct either.

A disadvantage of midi songs is that, in most cases, they do not reach the natural sound experience of recorded files - at least not without much work, e.g., on setting up synthesizers.

Automatic beat detection in recorded music is still researched: Had we used automatic beat detection, we would have felt the necessity to control the detected beat timings manually anyway. Therefore, we decided to manually mark the beats in recorded files in the first place.

We used BeatTapper[1] , a software tool for MacOS X that enables beat marking in a very comfortable way: While listening to a song, mouse click timings are recognized and stored, afterwards it is possible to change the exact timings by dragging the marked beats' symbols. Visualization of the audio data can be zoomed in or out, and music can be played faster and slower.

This method is not error free, as it depends on the user's musical abilities. In order to minimize the error, BeatTapper's features of zoomable visual representation, its ability to play songs at a slower speed, and to play a sound at the timings of already marked beats were used iteratively a couple of times on different days. That means, we marked the beats and fine adjusted the timings on a first day, and controlled the results on several other days, again adjusting the timings of beats if we perceived them as incorrect.

We always marked the first beat of each song's bars, timings of other beats are calculated later by the time difference to the next bar's first beat.

[1] http://stf.sourceforge.net/apps.html

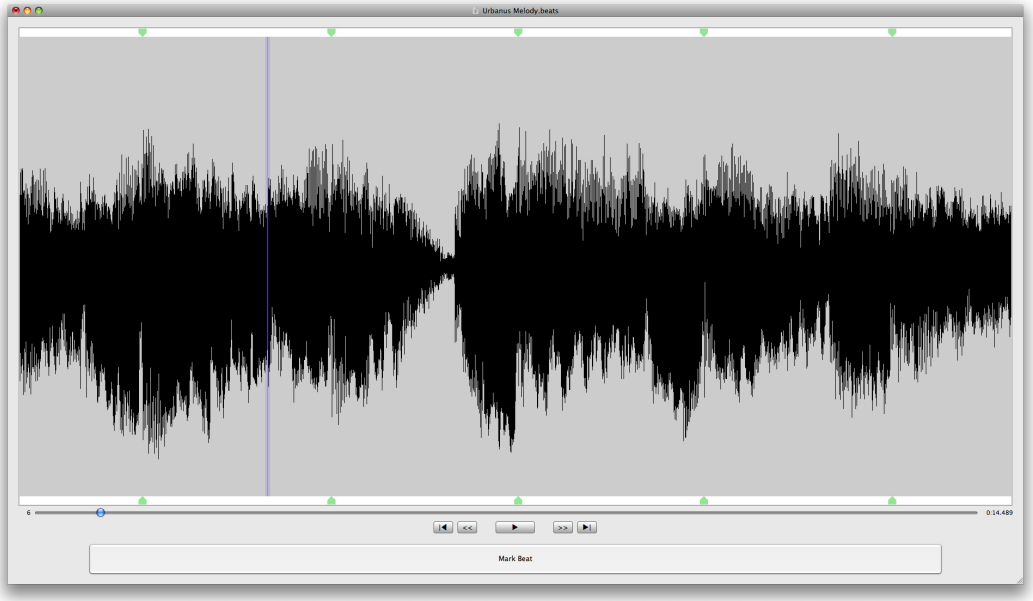

Figure 6.2: A screenshot of BeatTapper. The vertical blue line indicates the current position in the song, the green markers at the top and bottom indicate positions of already marked beats.

6.2.4 Saltate! core structure

Saltate!'s software is written in Java.

Saltate!'s core software is written in Java. In the beginning of the project, using Exemplar for data analysis (and to teach the system new dance moves) was an option we wanted to keep. Exemplar is an Eclipse[2] [3] plugin written in Java. A disadvantage of Java is its possibly slower program execution compared to other programing languages like Objective C, which do not use an additional byte code interpreter. However, since Saltate! does not need to deal with very complex calculations this is not a serious disadvantage.

The Saltate! software is conceptually divided into six groups:

- A graphical user interface (GUI).

[2] http://www.eclipse.org/
[3] Eclipse is an integrated development environment (IDE)

- An interface class responsible for communication with the sensor boxes.
- Several classes that analyze sensor data and search for certain events like a foot's step.
- An evaluator class that analyzes detected events and compares these to the music currently played. Depending on this analysis, the evaluator class changes the feedback given by the system.
- A central player class that plays music and sends information to the other classes about the current status of the system, e.g., which song is currently playing, and when it started.
- A protocol class. This class stores all data that might be of interest for later analysis.

6.3 Implementation

We describe the underlying methods and algorithms of Saltate!'s subsystems in this subchapter.

6.3.1 Radio network

XBee networks can be set up very differently.

Networks of Xbee modules, which we described in 4.2.1—"Deployed hardware", are structured hierarchically. One module, the coordinator, creates a network and assigns network addresses to joining modules, which become end devices. For bigger networks a third type of module, a router, is possible. These can allow other modules to join the network as well. They are not needed in a network of only five nodes, however. Modules have to be flashed with a different firmware version to become a coordinator or an end device. XBee networks can be run in transparent mode, or in API [4] mode (This depends on the firmware version as well).

[4] application programming interface

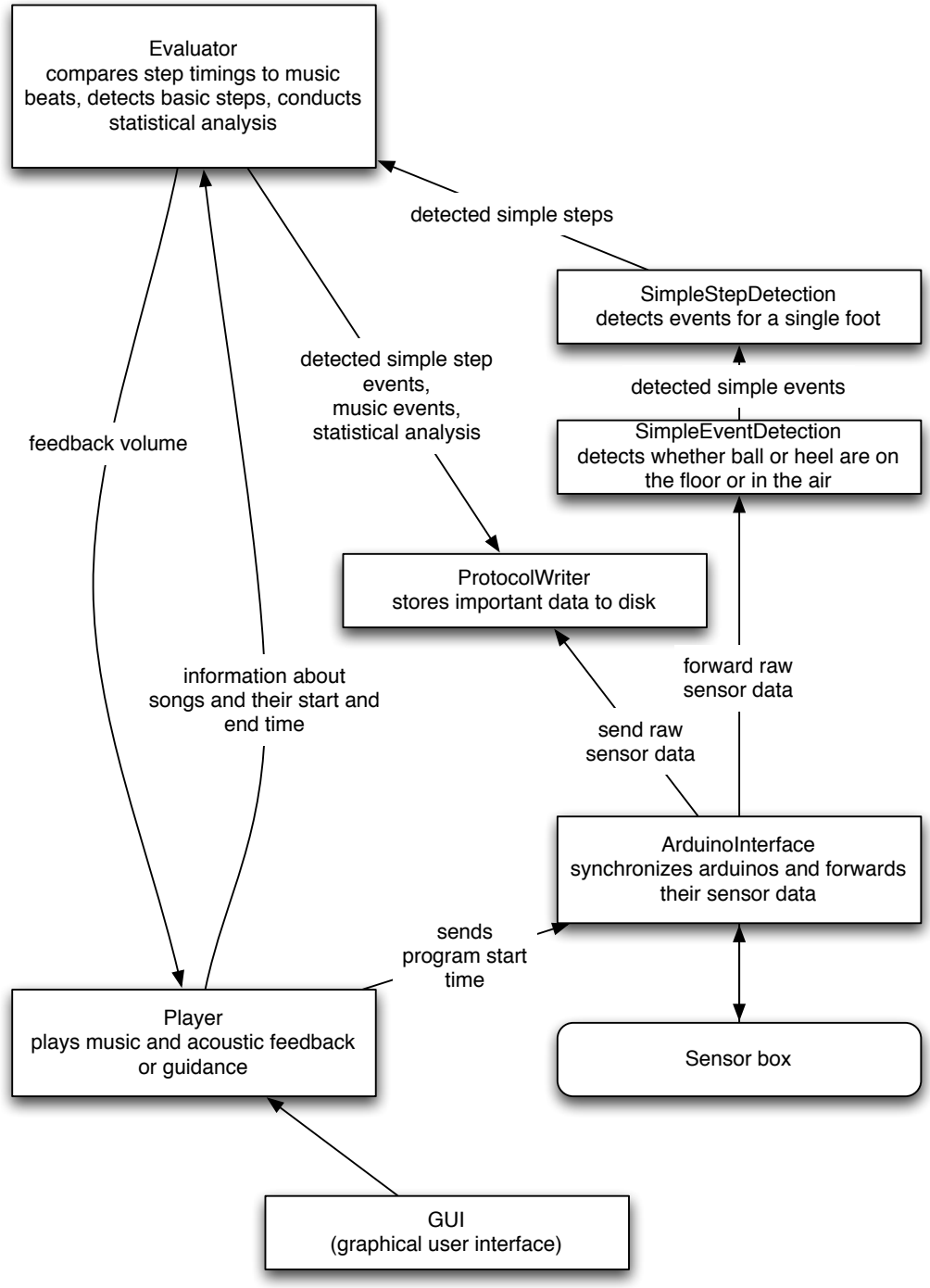

Figure 6.3: Important information flow in a running Saltate! program.

XBee's can operate in transparent mode...

In transparent mode, each XBee module has a predetermined target node, and all data sent to the module is directly forwarded to the target node. Target nodes, as well as other parameters, can still be changed during runtime if the module is set to command mode, but this cannot be done very fast: To set a module into command mode, a specific string of symbols has to be sent to the module, and no other symbols must be sent for a specific amount of time (guard time) before and after these symbols. While in command mode, data sent to a XBee module are interpreted as setup commands and not forwarded to the target node.

...or in API mode.

In API mode, data sent to XBee modules is not directly forwarded, it has to be packed into frames that contain additional information like the length of a frame and a checksum. In API mode, XBee setups can be changed more quickly by sending command frames instead of data frames to the module.

In a simple network with static addresses, transparent mode offers advantages: It affords less programming work since no frames have to be programmed, and without the overhead of frame data, the amount of data that can be sent is slightly increased.

API mode is more flexible and should be preferred when the network changes dynamically or messages have to be sent to different nodes frequently.

In our network, we have four end devices (one for each foot of a dancing couple) that send data to a fifth, central module, which is controlled by Saltate!'s software. The purpose of the network is to send sensor data from the end devices to the central device. Communication between the end devices is not necessary, and communication from the central device to the end devices is rarely taking place.

In Saltate! XBees operate in transparent mode.

Thus, the XBee network in Saltate! is working in transparent mode: The end devices send their data only to the central device, the central device broadcasts to all end devices. This setup could lead to problems in case a message of one end device is sent, e.g., in two parts, and another end device sends its message during the two parts of the first end device. Technically, the central device only receives a stream

of bytes, and could not determine which byte has come from which end device. To avoid this, the XBee's packetization timeout has been set to ten: The XBee modules start transmitting data only after a time in which ten additional characters could have been sent to the module. Thus, we make sure that every message is sent in one packet, and messages of different modules cannot intermix with each other.

6.3.2 Arduino programming

Programs loaded into Arduinos are automatically started when the Arduino is started or reset. Every program running on Arduinos consists of at least a setup method, which is called once, and a loop method, which is repeatedly called afterwards. For complex programs like ours, additional methods can be programmed which are called by the setup or loop method.

Arduinos are initialized first...

We have given each Arduino a device number (1,2,3, or 4). The rest of the program code is identical for all Arduinos. While executing the setup method, the status LED of the sensor box blinks, afterwards it is turned on. During development, the setup method was used to setup the XBees' parameters, including the node identifier string, which is set to the device number of the Arduino. Once the program ran stable, we stored these into the XBees' non-volatile memory.

...and wait for instructions or send sensor data afterwards.

While running, each Arduino repeatedly checks if a message has arrived and whether it has to send sensor data or not. Arriving messages cause the Arduino to start or stop sensor reading, calibrate sensors, synchronize its clock with the central clock, or to stop sending data until the synchronization of another module has finished.

Clock synchronization

Saltate! software and Arduinos have different clocks available.

We want to attach time stamps to the acquired sensor data, therefore we need to synchronize the different Arduino

clocks with the central Saltate! clock. Each Arduino has its own internal clock that is started with the Arduino. Saltate! uses a time measured at its own startup as a starting point for its own time. All time values are measured in milliseconds: Theoretically, time values of one clock could be translated into time values of another clock by adding the difference of their start times. In reality, it is not quite that simple. We explain why, and how we solved this problem, later. First we explain a solution for the basic synchronization problem of calculating the time differences of two clocks:

The synchronization process is started by the Saltate! software.

The synchronization process is triggered by the main Saltate! software, the Arduinos reply to these synchronization attempts. The basic synchronization process is pretty simple: The Saltate! software sends a synchronization request to sensor box b at a time t_0. This message arrives at b at time t_1. As soon as b receives the synchronization request, it replies by sending its own time. This message is received by the Saltate! software at a time t_2. We make the assumption that t_1 is exactly between t_0 and t_2 : $t_1 = \frac{t_0+t_2}{2}$.

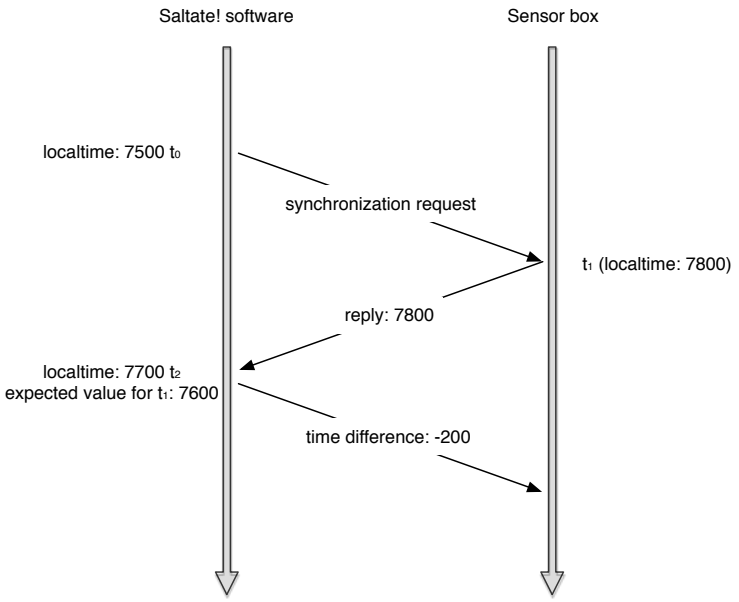

Figure 6.4: Example for a basic clock synchronization.

If the two clocks are already synchronized, the reply message send by b would be identical to the expected time t_1. The Saltate! software calculates the time difference between the received time of sensor box b and the expected time t_1. This time difference is then sent to b, which then adjust its internal clock by adding the given difference.

Arduino clocks are not very exact.

Testing this first approach showed that the Arduino's internal clocks are not very accurate, they drift about 6 to 10 ms per minute away from the actual time. This drift is not randomly distributed, but very constant for each Arduino, i.e., one Arduino might always drift about 6 ms, another might always drift 10 ms. The reason for this behavior are the quartz crystals within the Arduinos' microcontroller: They oscillate constantly, but the frequency is not constant among all microcontrollers. For Saltate! a growing error in the range of 10 ms per minute is not acceptable.

There were three possibilities to deal with this problem:

1. Connect a more exact external quartz to the Arduino.
2. Synchronize frequently to keep the error small.
3. Use data from several synchronizations to calculate the correct time.

The first solution would allow Saltate! to operate with a single synchronization, but needs additional hardware which would have caused additional time to order, set up, and test. The second solution is the simplest one, but more a workaround than a solution: it would not eliminate the error, but only keep it small by resetting the clocks before the error becomes too large. The last solution is more complicated, as it uses information from several synchronizations instead of only one, but requires much less synchronizations over time to achieve the same accuracy. This solution is used in Saltate!, we explain it in detail in the following paragraph.

Saltate! derives its program run time from Unix time,[5] which is given with an accuracy of one millisecond. This

[5] Unix time is the number of seconds passed since January 1, 1970.

time is used as a basis for all synchronizations and calculations. On the Arduinos, we use a function that gives us the amount of milliseconds passed since the Arduino started the last time. As mentioned above, this time value has an error of 6 to 10 ms per minute.

Saltate!'s advanced synchronization algorithm uses several synchronizations to increase accuracy.

In our advanced synchronization algorithm, we synchronize at several points of time to get a better calculation of Saltate! time on the Arduinos. The first time a synchronization takes place (measured in Arduino time) is stored in an Arduino as t_0, the error received after the first synchronization time is stored as *initialTimeCorrection* and serves as an offset to calculate between local Arduino time and Saltate! time. From then on, each Arduino always stores the time of the last synchronization as t_n, and additionally, the accumulated errors received after the first synchronization attempt as $delta_{t_n}$.

Now, an Arduino can calculate the approximated drift of its own clock and minimize its time error: At an arbitrary time t_{n+1} (measured in Arduino time) Saltate! time can be estimated to:

$$\text{Saltate! time} = t_{n+1} + \textit{initialTimeCorrection} + (delta_{t_n} \frac{t_{n+1}}{t_n})$$

For the second overall synchronization, an Arduino calculates its time by subtracting *initialTimeCorrection* from its current local time. We demonstrate this algorithm with the example in figure 6.5.

The estimated time error of sensor data is smaller than 4ms.

The bigger the time difference between the first and the last synchronization, the smaller the error of timings calculated by the Arduinos. A single synchronization, as described in the beginning of this chapter, takes about 32 ms (time difference between t_0 and t_2). When tested without radio transmission (using several Arduino Diecimilas connected to the same computer) this time window was about 16 ms. The error of a single synchronization can thus be assumed to be smaller than 8 ms, which is a worst case estimation, we assume that the actual error is smaller than 4ms. To guarantee a small error, synchronizations that take longer than 35 ms are discarded.

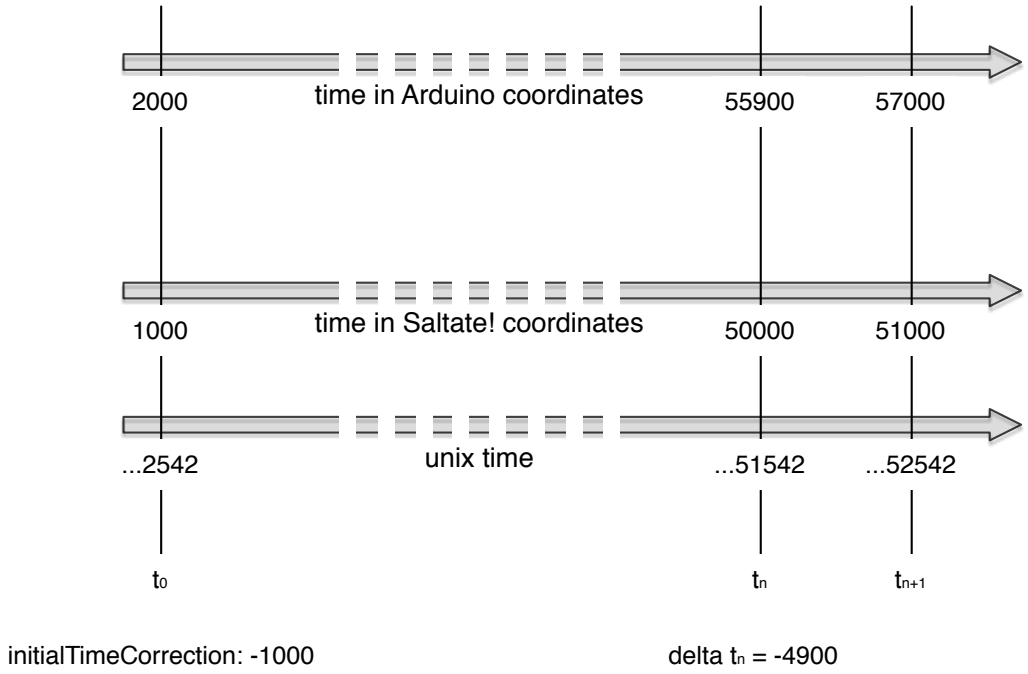

Figure 6.5: Example of several synchronizations:
The first synchronization takes place at time t_0, which is 1000 ms after Saltate! program start and 2000 ms after the Arduino's start. In this example, the Arduino's clock runs 1.1 times faster than the correct clock. Saltate! time coordinates are defined by a constant offset to unix time. At time t_0, the Arduino's *initialTimeCorrection* is set to -1000. At time t_n the Arduino calculates a Saltate! time of 55900-1000 = 54900 and receives a time correction of -4900, which in this example is already the accumulated error $delta_{t_n}$. At time t_{n+1}, the Arduino calculates a Saltate! time of $57000 - 1000 - 4900 * \frac{57000}{55900}$ = 51003.6. Which results in an error of (rounded) -4. After this is sent to the Arduino, the accumulated error $delta_{t_n+1}$ would be -4904.

In practice, the advanced synchronization algorithm proves to be very accurate: If the second synchronization has taken place one minute after the first synchronization, typical errors after another minute are about 4ms. If the second synchronization has taken place two minutes after the first synchronization, typical errors after already five more minutes are about 5ms.

Calibration

Calibration is needed to adapt to different weights or standing habits.

Sensor calibration is necessary to adapt the sensors to different characteristics for each program start: Different people have different weights and different standing habits, resulting in different pressures exerted onto the sensors.

Calibration in Saltate!: Averaging of sensor values over four seconds.

Sensor calibration is done by averaging sensor values for each sensor over a period of four seconds. The Saltate! software sends a calibration message, then all Arduinos calculate their average sensor values over the next four seconds and send these values back. This procedure is initiated manually. In the beginning of Saltate!'s development we asked dancers to stand still during the calibration time. At the end, we instructed them to stand with both feet on the ground, but to move their knees back and forth. Apparently, the first procedure was not showing very constant values for all persons. Probably, many persons depart from their normal standing habits when they are asked to stand still[6]. Moving the knees back and forth results in dynamically changing sensor values, but as these are averaged over a period of four seconds, the resulting average value seems to be more stable. Whether for this or for another reason, step recognition improved after we switched to the "moving knees" method.

Sensor data transmission

A sensor data packet consists of box identifier, time stamp, and sensor values.

The Arduinos start to send sensor data after they have received a start message. A sensor message contains the sensor box's device number, the time stamp of the sensor data in Saltate! time (calculated as described in 6.3.2—"Clock synchronization"), and the sensor value of the ball and the heel sensor. Each value is seperated by a semicolon, the message is terminated by an exclamation mark:

2;34068;678;245!

[6] You might try this in a small self-experiment: If you concentrate to stand upright and still, you will probably recognize that you tilt more than when you do not concentrate on standing still.

This string could be sent by sensor box number two, if it has measured a ball sensor value of 678 and a heel sensor value of 245, 34.068 milliseconds after Saltate!'s start.

While the step recognition algorithms were still under development, we sampled and sent sensor data after each 40 ms with all sensor boxes. At higher sampling rates, the radio network became unstable. After the step recognition algorithms were tested and it turned out that a simple thresholding algorithm in the first analysis was sufficient, this analysis was transfered to the Arduinos. In the final version, sensor data are sampled without delay (except for the delay caused by the Arduinos' loop method), but messages are only sent when a step event has been detected. We describe step recognition algorithms in 6.3.4—"Step recognition".

6.3.3 Song and feedback information retrieval

Saltate!'s music's beat timing information are stored in BeatTapper files.

Saltate! retrieves information about the name and dance type of all supported songs from a text file (*songinfos.info*). A song's dance type is used to switch to the according feedback settings. Song names are used to load the correct mp3 file and to read the corresponding BeatTapper file. BeatTapper files contain information about the timings of the marked beats. In our case, the first beat of each bar is marked. Timing information from the BeatTapper files are needed for step analysis and feedback calculation. This information retrieval allows an expansion with further songs without changing Saltate!'s code: You have to mark all of the song's bars' first beats with BeatTapper, copy the BeatTapper file and the song file into the according directory, and add information about song name and type to the songinfo file.

6.3.4 Music playback

Feedback is played from different lines that run parallel to the music line.

Saltate! uses the Javax.sound.sampled package to play back all of its sound data. In addition, we use javazoom[7] in

[7] http://www.javazoom.net/mp3spi/mp3spi.html

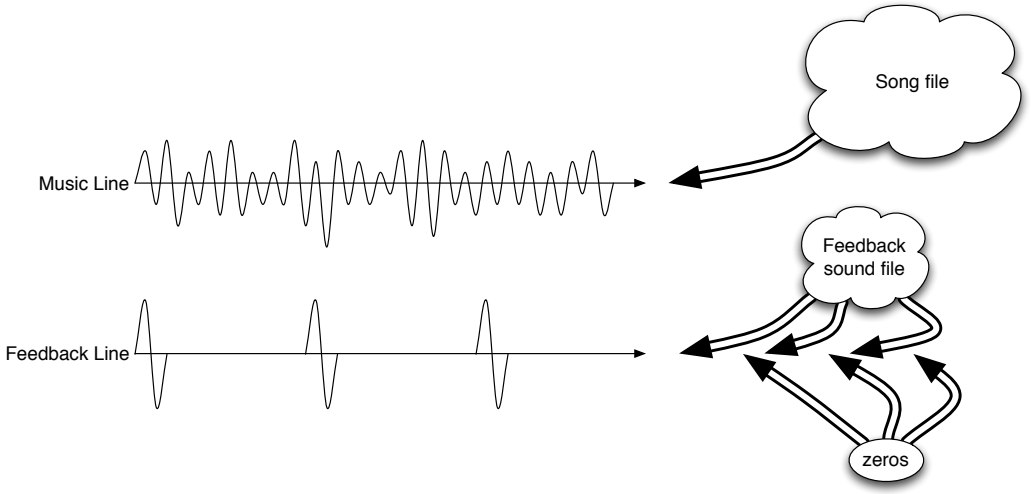

Figure 6.6: This figures shows how sound data of a played song and feedback data are mixed: A music line is constantly filled with sound data of the currently played song, feedback lines are filled with sound data of a feedback file and zeros in between.

order to use mp3 as input format. Sound data from song and feedback sounds are played over different music lines which run in parallel. Song data is copied to its buffer as a stream, whereas feedback sound data files are repeatedly copied to their buffer at predetermined times. (In waltz, for each beat of a bar one feedback sound is being played) When no feedback sound has to be played, the feedback's buffer is filled with zeros.

The amount of frames per second that are played on each music or feedback line depends on the sampling rate of the song and feedback files used. The amount of zeros that have to be copied into the feedback's buffer is calculated depending on the timing of the next feedback sound, the amount of data copied into the music's buffer, and the difference in the lines' playback speed (measured in frames per second).

Step recognition

First part of step recognition: From raw sensor data to simple events.

Step recognition in Saltate! is realized in different layers. On the lowest layer, raw sensor data is acquired and forwarded to the next layer, which detects *simple events*. There are four simple events in Saltate!:

- ball touch events
- heel touch events
- ball leave events
- heel leave events

Simple events are detected by thresholding.

These events are used to determine the timings at which the balls or heel of a feet touches or leaves the ground. Technically, this is implemented via thresholding: If a sensor is considered to be in the air, a touch event is generated as soon as the sensor value of this sensor becomes higher than 0.87 times the calibration value for this sensor, and is then considered to be on the ground. If the same sensor's value becomes lower than 0.82 times its calibration value, a leave event is generated and the sensor is considered to be in the air again. We use two different values to avoid rapidly and incorrectly generated events, which could happen with only one threshold if the sensor's value is alternating around this threshold quickly. This could happen, e.g., for a left foot's sensor, if a dancer stands with both feet on the ground with his weight slightly shifted to the right. Then his left foot would exert less pressure on the sensor than it did while in calibration.

We chose the values of 0.87 and 0.82 after some tests. Initially, we experimented with upper values that were close to the calibration value: When a step is performed, the whole weight is on one foot, and the pressure should be higher than while standing on both feet. This did work fine for some persons, but not for everyone. Probably some persons stood on other parts of their feet while performing steps than other, and sometimes these parts where too far away from the sensors' positions. However, while a foot was in the air, its sensor's values were reliably much

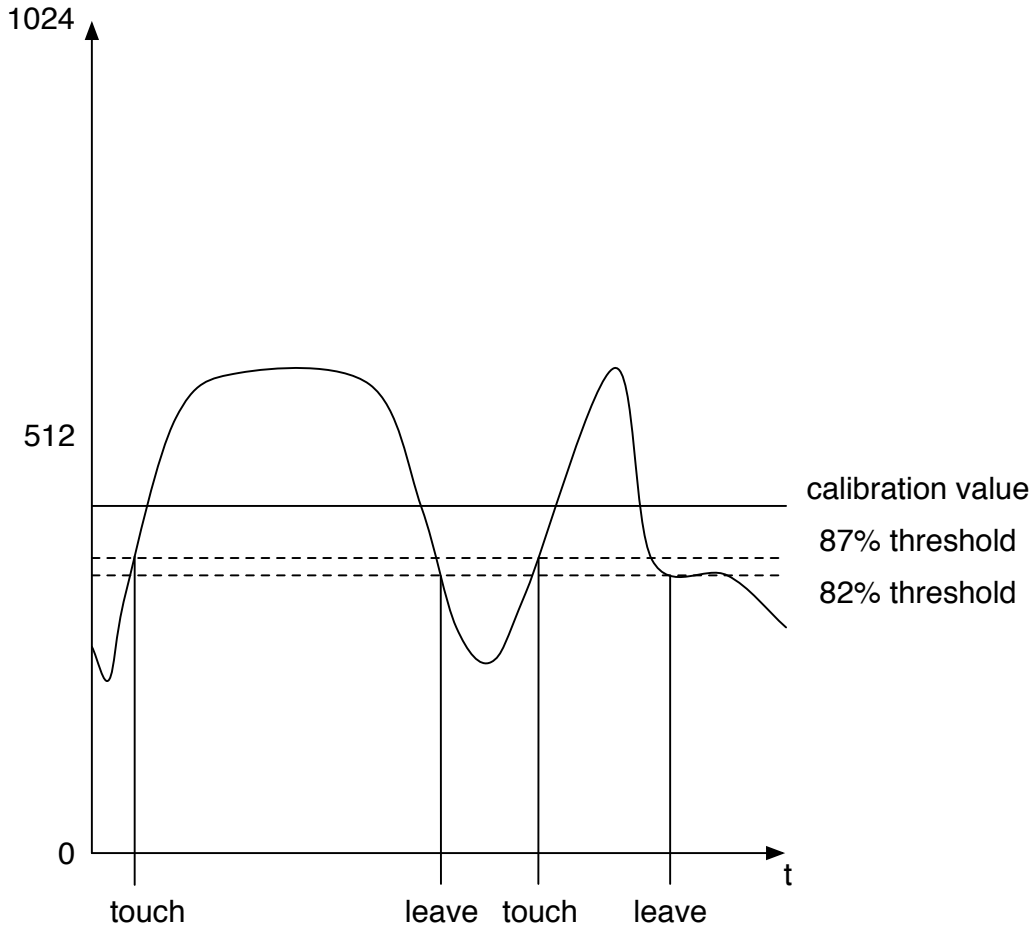

Figure 6.7: An example of simple event detection: The ball or heel sensor in this example starts in the air and is then set to the ground twice, finally leaving it again.

lower than while the foot was standing on the ground. The chosen thresholds of 87 and 82 percent of the calibration value trigger slightly before the dancer's complete weight is placed on it, and slightly before it leaves the ground entirely. Since this error is systematical, it is not a problem for determining the step's timings, as long as the system is using the same thresholds.

Second part of step recognition: From simple events to simple steps.

Simple events are passed to the next layer, which detects *simple step events*. Saltate! distinguishes four simple step events:

- A forward step touch
- A backward step touch
- A heel tap
- A ball tap

Both forward steps and backward steps are triggered by a combination of a ball touch and a heel touch event, which have to occur within a certain amount of time: If a person performs a normal step forward, his foot touches the ground with its heel first and with the ball shortly afterwards.

Saltate! taps include ball or heel steps and taps.

Heel taps and ball taps in Saltate! slightly differ from taps as they are known in dancing, where a tap describes a foot's ground touch that is not supporting body weight. Working with simple events, Saltate! cannot distinguish whether, e.g., a ball tap or a ball step (a step during which only the ball touches the ground) is taking place.

Taps and steps are distinguishable only after they are completely executed.

A heel touch event might become a heel tap event, but it could also be part of a forward step touch event if a ball touch event follows shortly afterwards. The same applies for ball touch events and ball tap or backward step touch events.

Ball taps are detected if:

- Both ball and heel of a foot are in the air
- A ball touch event is received
- a) - either the next event received is a ball leave event or
- b) - no heel touch event is received for 450 ms

The determined timing of a ball tap is the timing of the ball touch event.

Heel taps are detected accordingly. The parameter of 450 ms was determined experimentally: While performing forward and backward steps, the heel touch and ball touch events were detected with a time difference of up to this value.

Thus, a forward step touch event is detected if:

- Both ball and heel of a foot are in the air
- A heel touch event is received
- The next event received is a ball touch, which has to happen within 450 ms after the initial heel touch event

Backward step touch events are detected accordingly, the timings of these events are determined by the timing of the first involved event (heel touch for a forward step or a ball touch for a backward step).

Steps and taps are detected by a finite state machine.

The state of each foot is modeled by a finite state machine with clocks. At transitions within this machine, simple step events might occur. Figure 6.8 is a graphical representation of this finite state machine.

Additionally, *forward step leave* events and *backward step leave* events are analyzed. However, we did not use them for feedback activation.

Simple step events are passed to the performance evaluating part of Saltate!.

Dancing couple performance evaluation

Saltate! analyzes a dancing couple's performance based on the dancers' step timings relative to the song's beat timings,

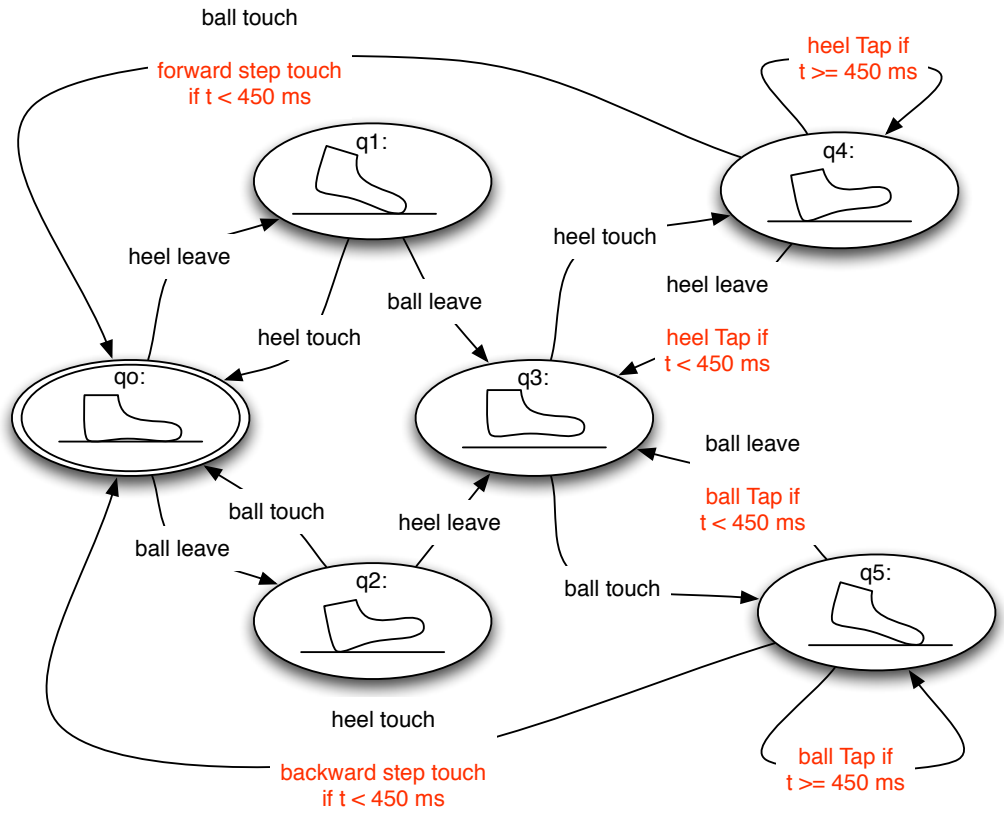

Figure 6.8: Finite state machine for simple step detection of one foot:
The oval states of the machine are graphically showing the current state of a foot. Simple events that lead to a transition are printed in black. Simple steps are detected at transitions, they are printed in red.
The detection of a forward step event could happen like this:
1) The foot is standing on the ground: The machine is in state q0.
2) The heel leaves the ground: A heel leave event is received and the machine's state changes to q1.
3) The ball leaves the ground: A ball leave event is received and the machine's state changes to q3.
4) The heel touches the ground: A heel touch event is received and the machine's state changes to q4.
5) After 300 ms, the ball touches the ground: The machine's state changes to q0 due to the ball touch event, and a forward step touch event is detected, as the ball touch event occured within 450 ms after the heel touch event.

kinds of steps made, and the feet with which steps are performed. First, we explain which algorithms are used for online analyses of performance during song playback. These analyses are used for feedback activation. We explain analyses that are applied after a song has finished later.

A step analysis is executed for each beat.

Performance analysis is done for each beat of the music: Each recognized simple step event is mapped to the beat of the music with the smallest time difference. As soon as no simple step can be matched to a particular beat anymore (because the beat is over), the couple's performance for this beat is analyzed. This does not happen between the analyzed beat and the following beat but 450 ms later, as forward or backward step touch events as well as ball and heel taps are recognized up to 450 ms after they have taken place. This delay does not have a negative impact on the support currently given by Saltate!, as the volume of the supporting functions is designed to change rather slowly over a pretty long time interval anyway.

For Slow Waltz, basic steps and natural turns last over two bars of the music and steps are done alternating (the man starting with the right foot, the woman with the left foot). If Saltate! recognizes a step or tap with the right foot of the man and the left foot of the woman it decides, depending on the last beat's steps and the beat of the music, how to mark these steps:

1. If the beat was the first beat of a bar, the steps could only belong to the first step of a basic step or natural turn
2. If the beat was the second beat of a bar, the steps could only belong to the fifth step of a basic step or natural turn
3. If it was the third beat of a bar, the steps could only belong to the third step of a basic step or natural turn
4. The current steps are marked as belonging to the first, fifth, or third step of a basic step, if:
    - the last beat's steps were mapped to the preceding step of a basic step, or

- the last beat's steps were marked as being wrong, or
- there were no steps recognized at all.

Otherwise, they are marked as wrong.

Steps with the man's left and the woman's right foot are analyzed accordingly.

If a beat's steps are marked as belonging to the sixth step of a basic step, and the last five beats were marked as belonging to steps one to five of a basic step or natural turn, a basic step is detected.

The following scenario is an example of how this analysis works in practice:

> The dancing couple stands ready and waits for the music to start. At the beginning of the song, the couple might still stand for the first two bars. Therefore, no steps are recognized for the first six beats of the music. Then, the couple starts to dance: The man starts with his right foot, the woman with her left foot. Saltate! recognizes a forward step touch for the right foot of the man, and a backward step touch for the left foot of the woman. Both are matched to the seventh beat of the song, which is the first beat of its bar. Both step events are matched to the first step of a basic step.
>
> Five beats later the couple finishes its first basic step: Saltate! detects a ball tap for the man's left foot and the woman's right foot, both are matched to the twelfth beat of this song, which is the third beat of its bar. These taps are matched to the sixth step of a basic step. Now, the last six beats' steps have been matched to the *elementary steps* one to six of a basic step and Saltate! detects a complete basic step.

Saltate! starts all of these analyses when the first marked beat of the music is being played and finishes after the song is stopped manually, or the last marked beat of the music has been played.

Data of all steps are stored for offline analyses.

For each step Saltate! stores to which beat it was matched or if it couldn't be matched. These data are used for statistical analyses after the song has finished. Additionally, the amount of beats to which steps were successfully matched are stored, as well as the amount of identified basic steps.

Saltate! calculates the percentage value of correctly danced steps...

After each song, the percentage value of correctly danced elementary steps and basic steps is calculated. The percentage value of correctly danced elementary steps is calculated using the amount of correctly danced elementary steps and the amount of beats that were played until the song either finished or was manually stopped. For the percentage value of basic steps, the amount of recognized basic steps and the highest multiple of six that is smaller or equal to the amount of played beats are used.

...and average time differences of steps and beats.

In addition, Saltate! calculates mean time differences and standard deviations of step events to their closest beat. This is done sorted by:

- Feet, matched part of a basic step, kind of step, and
- Feet and matched part of a basic step

Thus, it is possible to see, e.g., which average time differences the woman's detected backward step touch events or ball taps of her first elementary steps of a basic step have to the according beats of the music. Additionally, the same is possible for the combination of all tap and step events of this foot which were matched to the first (or any other) part of a basic step.

The following table shows parts of a statistical analyses:

	backward step	ball tap	combined
amount	13	7	20
mean difference to beat (in ms)	-23.6923	-90.8571	-47.2000
standard deviation (in ms)	91.7237	78.7346	91.3867

Table 6.1: This table shows a part of the statistical analysis for one song: The data shown are taken from steps of the man's left foot which were performed as the second elementary step of a basic step. In this example, only backward steps and ball taps have been recognized.

An improved analysis of correctly danced steps compensates for not recognized steps.

Tests during Saltate!'s development with experienced dancers showed that not all steps are recognized. Therefore we slightly changed the analyses described above if only one step is being recognized during a single beat: If the preceding beat was recognized correctly, and the step of the current beat would fit to the next step of a basic step, the current beat and step are recognized as correct. In other words, if only one step is not recognized, but everything else is correct, we treat the detected step and its beat as being correct.

> Imagine a couple dances correctly but from time to time steps from the woman's right foot are not recognized correctly. This could result in the following analyses: On the first beat of a bar, the couple starts dancing and Saltate! detects steps with the man's right and the woman's left foot, marking the steps as belonging to the first elementary step of a basic step. On the next beat, Saltate! only recognizes a step with the man's left foot and misses the woman's step with the left foot. In the original version, Saltate! would have marked this step as incorrect, in the modified version, this step is marked as belonging to elementary step two of a basic step.

This changed detection improved the correct detection of danced elementary steps noticeably. There are two other cases of missed or wrongly detected steps for which we changed analysis, both of which occurred only very rarely:

At some times no steps were recognized at all during a single beat. In case Saltate! detects two correct steps during the next beat and detected correct steps during the beat before, it considers the beat as correctly danced, i.e., the amount of correctly danced elementary steps is increased by one. The second case for which we changed analysis concerns wrongly detected steps: In case Saltate! detects more than two steps during a single beat, it treats the beat as if no steps were detected at all. That means, the steps performed are not used for statistical analyses of mean time differences, but the amount of correctly danced elementary steps might be increased for this beat as well.

Feedback control

Saltate! increases and decreases feedback volume linearly.

Based on online analyses, Saltate! adjusts the feedback volume: If a beat's steps were analyzed as wrong, or there were no steps, feedback volume is increased. If they are analyzed as correct, feedback volume is decreased. Both increase and decrease are performed linearly to values between zero and one hundred: Feedback volume would reach a value of 100 (starting with zero) after 10 beats in the music, volume decrease is being done much slower: from a volume value of 100, a value of zero would be reached after 18 bars. The volume control values between zero and one hundred are intended to simulate a felt volume percentage, they are mapped to sound pressure values logarithmically (a ten dB increase of sound pressure is generally perceived as about twice as loud). Music volume is set to 80, thus feedback can become dominant if the dancers perform badly.

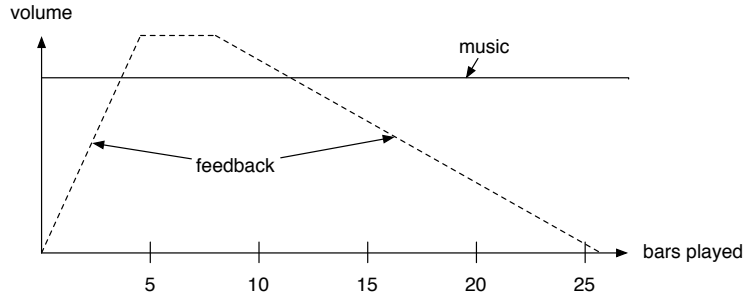

Figure 6.9: A dancing couple that starts to dance with the eighth bar of a song would perceive music and feedback volume as shown in this graph.

We uploaded a sound file[8] for this example.[9]

[8] http://media.informatik.rwth-aachen.de/~drobny/saltate!/mmSaltateFeedbackExample.mp3

[9] The emphasis of the beats is much better recognizable on a hi-fi system than on ususally small PC speakers.

Data storage for later analyses

After each song, Saltate! stores the following data about the dancing session:

- Raw sensor values (for each foot in a different file).
- Timings of detected simple steps (for each foot and event in a different file, and all events of one foot combined in one file).
- Timing of music events, i.e., the beats of each bar.
- Statistical analysis: For each foot and step event one file: All step events are matched to the closest beat counting from one to six. Mean difference, variance, and standard deviations are calculated.
- Statistical analyses of step events sorted by: feet, step events, and their matching to a part of the basic step. Additionally, all step and tap events for one foot and basic step part are combined.
- The amount and percentage value of correct elementary and basic steps that were danced.

Chapter 7

Evaluation

"Advice is judged by results, not by intentions"

—Marcus Tullius Cicero

We evaluated Saltate! with eight couples of dancers, measuring the changing performance of the couples during the dancing session. Our goal was to find out in how far Saltate!'s supporting function helps beginning dancers and to get beginner opinions about the system and its supporting function.

7.1 Experimental setup and execution

In this subchapter we explain which values we measured, how we dealt with possibly interfering variables, and how we executed the experiment.

7.1.1 Independent, dependent, and interfering variables

The goal of the conducted experiment was to find out in how far Saltate!'s feedback helps beginning dancers or not. Therefore, we activated and deactivated Saltate!'s supporting function, which was the independent variable in our

experiment. We used two dependent variables to measure a couple's performance:

First of all, we measured the amount of correctly performed steps each couple performed. If couples that perceive feedback have a higher amount of correct steps, this would be a strong hint that Saltate!'s feedback helps couples to start or to continue dancing.

We use standard deviations as a measure for the timing accuracy of the dancers.

Secondly, we measured the dancers' standard deviation of their step timings. A lower standard deviation shows that the couple dances more regularly. We expected this value to be higher than for experienced couples, even if the amount of steps a couple danced was high from the beginning of the session. Furthermore, we supposed that this value would become lower during a session.

Average timing differences fluctuate too much.

We did not use the average time difference of the dancers' steps to the beats of the music as an additional value to measure performance, since we could not identify a "correct" value for step timings. Tests with experienced dancers showed huge differences, depending possibly on skill, music, or step size. Eventually, it is possible to determine a "correct" timing, but we could not acquire enough data to seriously try to.

We identified several variables that might have an interfering influence on the dancing couples' performance:

1. The music played (different songs might be easier or harder to dance to).
2. The dancers' experience with dancing or rhythm.
3. Individual aptitude.
4. Familiarity of dancers to each other.
5. Learning effects during the experiment.

We use an in-group design for our experimental setup.

In order to minimize the effects of individual aptitude we chose an in-group design for our experiment: Every couple should dance parts of the experiment with feedback and

other parts without. To minimize learning and music effects we decided to balance these variables:

We balanced the sequence of feedback conditions,... ...played music blocks,...

The first half of the couples danced with feedback in a first training block and without feedback in a second training block, the other half of the couples without feedback in the first and with feedback in the second training block. We divided the music into two groups of songs, using one during the first training block and the other during the second training block for one half of the couples, with the other sequence for the other half of the couples. Balancing these two variables (order of feedback/no feedback and music played) already leads to four subgroups. As we couldn't get only couples that never danced before, we decided to try to balance out the effects of the couple's experience with dancing (or rhythm) as well:

...and dancing experience.

We asked every participant when (and for how long) he has taken a dancing course, or whether he played or plays an instrument. Based on their answers, we tried to divide the couples into one group with little experience, and one group with very little or no experience. To each of the four experimental subgroups described above we assigned one couple with little and one with very little experience.

7.1.2 Experimental setup

Overall ten songs, each played for 45 or 60 bars.

The experiment was divided into two blocks. During the first block the dancing couples performed basic steps, during the second block they performed natural turns. In each block five songs were played, the first and last one for 45 bars, the second, third, and fourth song for 60 bars. The first and the last song were used for performance measurement only, i.e., during these songs feedback was deactivated. During the second, third, and fourth song, feedback was activated or not, depending on the couple's subgroup. These three songs were considered the training phase.

Before starting the first and second block, the couples got a short explanation of the waltz' basic step and natural turn:

1. The instructor demonstrated and explained the steps.
2. The instructor performed the steps together with the dancers, while he counted from one to three.
3. The instructor explained the closed dance frame (only in the first block).
4. The couple performed the steps together while the instructor counted from one to three.

We did not set a strict time limit for these explanations but kept them as short as possible, being long enough for the couple to start to dance to the following songs without further explanations. With a strict time limit, we would have risked that a couple that understands the steps very quickly can use the explanation time to train beyond a stadium of understanding the steps, whereas a couple that understands the steps slowly might have problems to perform them with music without renewed advice.

7.1.3 Example setup for one couple

Following, we give a detailed task list for the execution of the experiment with one couple:

1. Welcome.
2. Explanation of the experiment.
3. Attachment of sensors and sensor boxes.
4. Filling in of the questionnaire's first page; initial clock synchronization.
5. Demonstration of basic step as described above.
6. Second clock synchronization.
7. Sensor calibration.
8. Question to the dancing couple whether they have any questions left, or if they are ready to start.

9. Information of dancing couple whether Saltate!'s feedback functions are activated in the second to fourth song or whether only music is being played.
10. Playing of the first five songs.
11. Short break, if desired by dancing couple.
12. Demonstration of natural turn as described above.
13. Further clock synchronization.
14. Question to the dancing couple whether they have any questions left, or if they are ready to start.
15. Information of dancing couple whether Saltate!'s feedback functions are activated in the second to fourth song of the next block or whether only music is being played.
16. Playing of the second five songs.
17. Detachment of sensors and sensor boxes.
18. Filling in the rest of the questionnaire.
19. Leave-taking and answering of further questions if desired.

Before each song there was a short break, as the instructor had to select the next song and type in a file identifier. During a song the instructor counted the bars played, manually stopped the song, and monitored whether Saltate! detects all steps. If Saltate! missed steps, the instructor initiated another sensor calibration before the next song. In order to not influence the couple in their dance, the instructor did not look into the couple's direction while a song was playing. However, monitoring of the couple's performance by peripheral vision was still possible.

We offered candy to all participants during the experiment.

7.2 Questionnaire

The questions in the questionnaire were chosen to get different kinds of information:

- Possible reasons for unusually good or bad performances.
- Self-assessment of the dancers' performance at the beginning and end of the experiment.
- The system's acceptance by users.
- The dancers' opinion about relative strengths and weaknesses of Saltate!.

You can find the complete questionnaire in appendix A—"Saltate! questionnaire".

7.3 Experimental results

Our experimental setup was too easy for the more experienced dancers.

During experimental sessions with couples from the more inexperienced group of dancers no problems occurred. In two sessions with more experienced couples Saltate! missed too many single steps to correctly identify the amount of correctly danced steps in several songs. Additionally, it turned out that dancing only basic steps or natural turns was too easy for the more experienced group of dancers. Therefore, we weren't able to use data from the more experienced dancing couples for quantitative analysis.

The intended approach for our statistical analysis of the changing amount of correctly performed dance steps was a Wilcoxon signed-rank test as described by Fahrmeir et al. [2007]. The null hypothesis to test against would have been that couples with activated feedback have the same increased amount of correctly danced steps than the average of all couples. Since the values we were interested in (the amount or percentage of correctly danced steps) have an upper bound that was not that far away from measured values, we could not expect these values to be normally distributed. Therefore a parametric test like Student's t-test was not applicable.

For analyses of the changes in the dancers' standard deviation from their step timings a t-test might have been ap-

plicable, but with usable data of only four couples a statistical analysis will not come to a significant result - except for extraordinary strong dependencies of the dancers' performance on the independent variable which we could not see, and which were not to expect either.

Following, we present some of the more inexperienced dancing couples' data graphically and in tabular form, and describe additional observations made by the instructor that were not recognizable by automatically taken data.

7.3.1 Amount of correctly performed steps

The amount of correctly danced steps increased.

As expected, the average amount of correctly danced steps during each song increased in each block of the experiment, both in the feedback and the no feedback group. In either block (basic step and natural turn), the feedback groups' increase was slightly higher than the no-feedback groups'. This can be seen in figures 7.1 and 7.2. Due to the small amount of couples we cannot claim that feedback activation helps dancers to dance more, but we assume at least that it does not disturb short term learning: Though the last song of each block was played with deactivated feedback, the amount of correctly performed steps did not decrease in the feedback group.

	song 1	song 2	song 3	song 4	song 5
all beginners (in percent)	73.00	87.25	92.75	93.50	94.00
activated feedback (in percent)	68.00	85.50	94.50	95.50	95.00
deactivated feedback	78.00	89.00	91.00	91.50	93.00

Table 7.1: Amount of correct steps while dancing basic steps. Underlying data for figure 7.1.

	song 1	song 2	song 3	song 4	song 5
all beginners (in percent)	67.50	90.00	85.00	86.25	93.00
activated feedback (in percent)	59.00	89.00	87.50	91.00	94.00
deactivated feedback	76.00	91.00	82.50	81.50	92.00

Table 7.2: Amount of correct steps while dancing natural turns. Underlying data for figure 7.2.

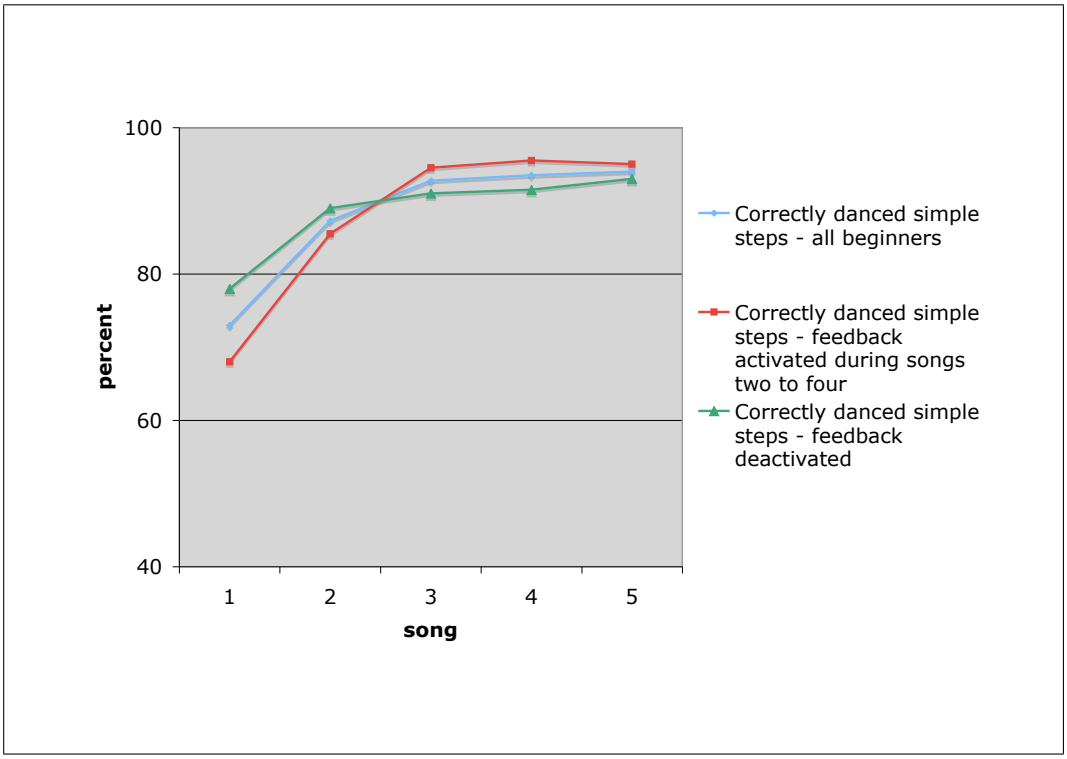

Figure 7.1: Development of correctly danced simple steps during the first part of the experiment, when only basic steps were performed.

7.3.2 Development of dancers' standard deviation of step timings over time

The standard deviation of dancers' step timings decreased.

We analyzed steps that were executed on a bar's first beat, thus belonging to the first or fourth step of a basic step or natural turn. We did not take into account data from the more experienced dancers, and used only data from standard deviations if the standard deviation was calculated from at least 7 basic steps. We show a graphical analysis in figures 7.3, 7.4, 7.5, 7.6, and 7.7. The underlying data shown in table 7.3, 7.4, 7.5, 7.6, and 7.7 are mean values from two dancing couples.

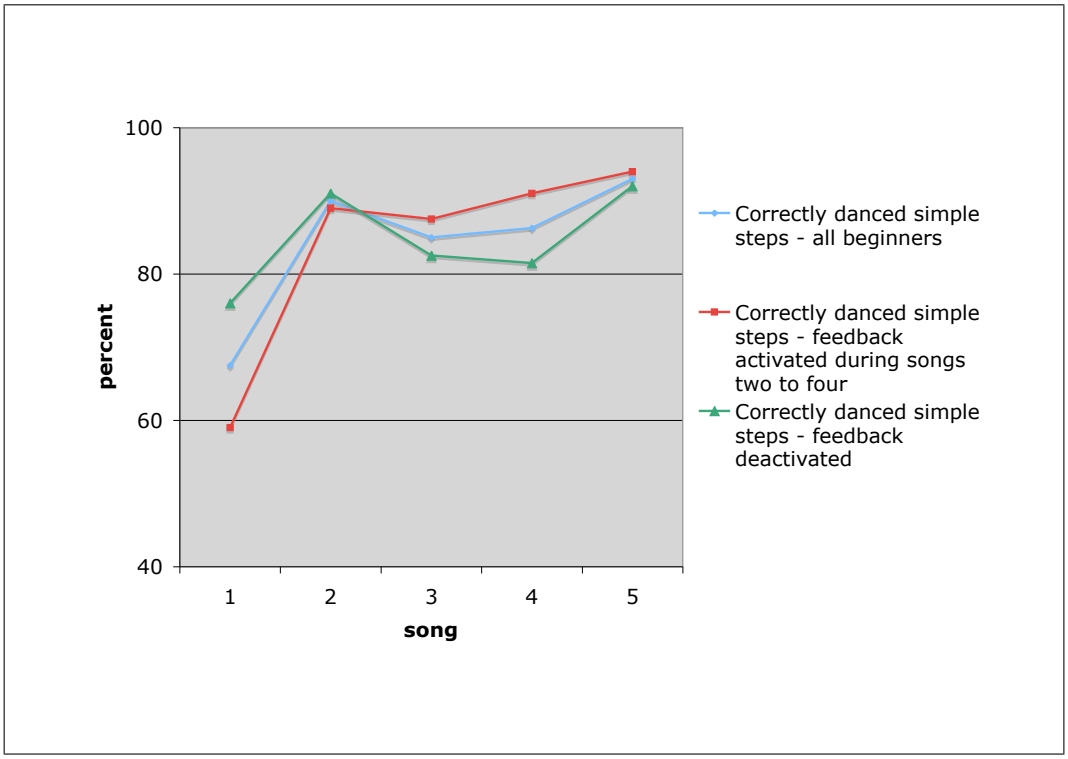

Figure 7.2: Development of correctly danced simple steps during the second part of the experiment, when only natural turns were performed.

7.3.3 Manual observations

During execution of the experiment we noticed, at three out of eight couples, two dancing errors that Saltate! cannot detect automatically or react to successfully:

Though the couples danced to the rhythm of the music, they did so in a wrong way. They executed the first step of a basic step or natural turn to the second or third instead of to the first beat of a bar. This mistake was not only done by the less experienced couples. In fact, two out of the three couples who made this mistake belonged to the more experienced group. One of them even danced to six out of ten songs in this way.

Some couples performed steps to the wrong beats...

The second error lead to the first error a couple of times: While dancing, the couple lost the time mapping of its steps to the timings of the music's beats at some point. Though it

...or occasionally ignored the speed of the music.

step	foot	standard deviation before training (in ms)	standard deviation after training (in ms)	improvement (in ms)
first	man right	126.91	102.59	24.32
first	woman left	150.49	83.62	66.87
fourth	man left	126.20	63.76	62.43
fourth	woman right	124.61	70.77	53.87
	combined	132.05	80.19	51.87

Table 7.3: Development of dancer's standard deviations of step timings for basic step phase with activated feedback. Underlying data for figure 7.3.

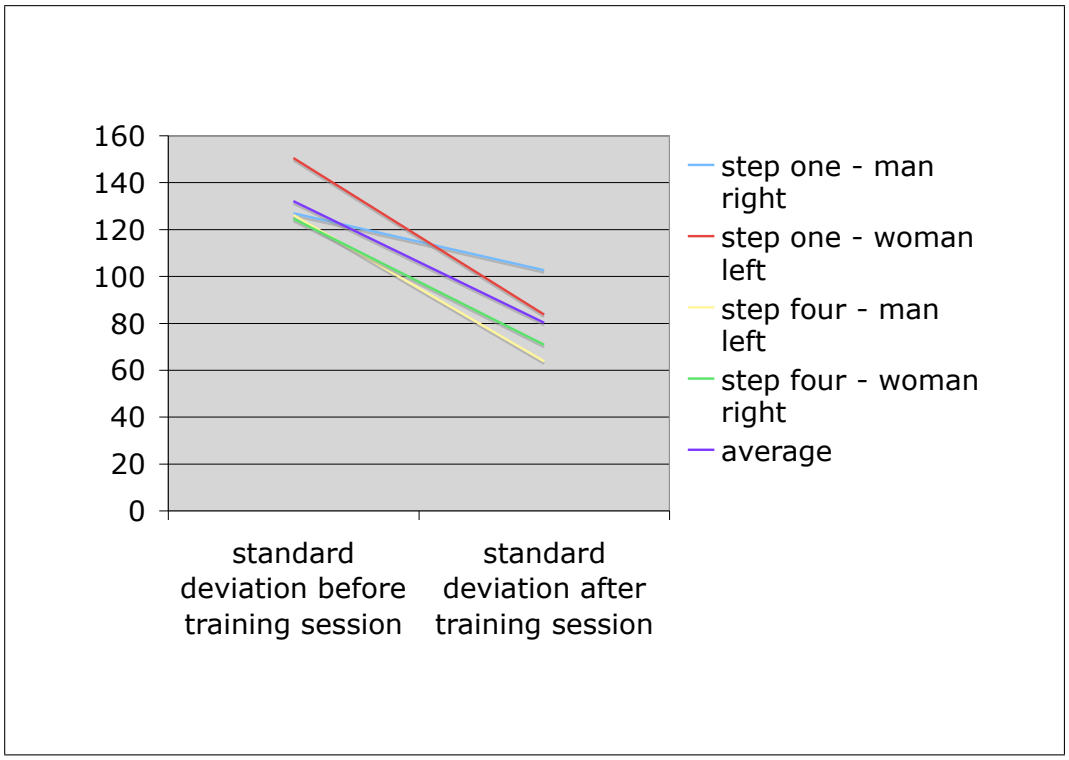

Figure 7.3: Standard deviations before and after a basic step training session with activated feedback.

continued to perform basic steps or natural turns, it did so slightly faster or slower than to the music's rhythm. When it continued to dance to the rhythm of the music, it did so one beat behind or ahead of it.

In the first session, while performing basic steps, even the more inexperienced couples had no problems to keep danc-

step	foot	standard deviation before training (in ms)	standard deviation after training (in ms)	improvement (in ms)
first	man right	137,14	94,11	43,04
first	woman left	140,17	95,55	44,62
fourth	man left	120,44	116,73	3,71
fourth	woman right	109,50	60,59	48,91
	combined	126,81	91,74	35,07

Table 7.4: Development of dancer's standard deviations of step timings for basic step phase without feedback. Underlying data for figure 7.4.

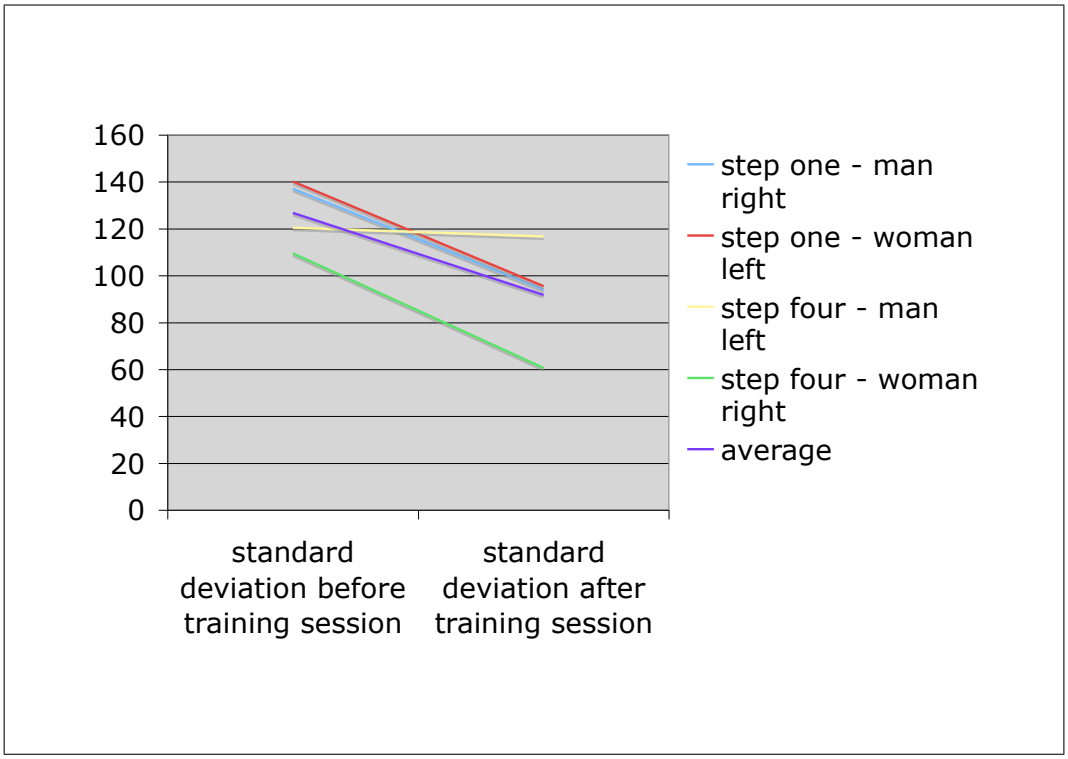

Figure 7.4: Standard deviations before and after a basic step training session without feedback.

ing after the first song, i.e., they had to stop and restart dancing only very rarely.

step	foot	standard deviation before training (in ms)	standard deviation after training (in ms)	improvement (in ms)
first	man right	198,79	119,87	78,92
first	woman left	220,12	105,39	114,73
fourth	man left	161,54	90,68	70,86
fourth	woman right	(207,40)	-	
	combined	193,48	105,31	91,65

Table 7.5: Development of dancer's standard deviations of step timings for natural turn phase with activated feedback. Underlying data for figure 7.5. Saltate! detected too few steps in one case to calculate a reasonable standard deviation. The value in brackets was not used for calculation of the average value.

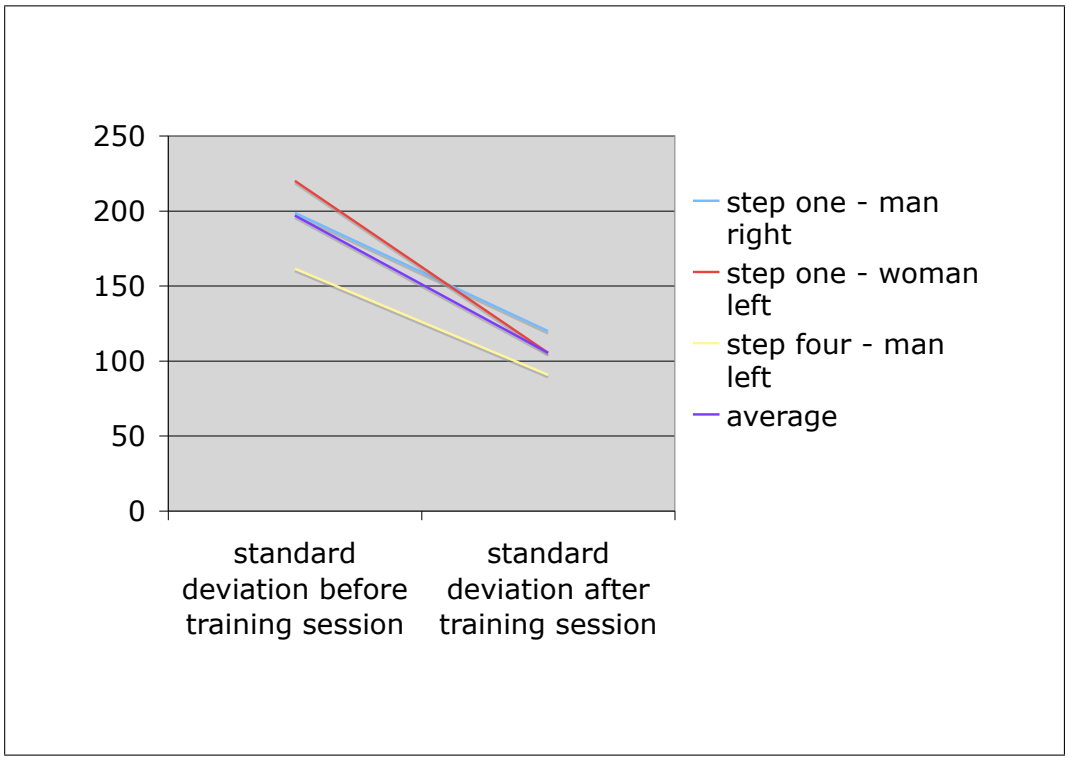

Figure 7.5: Standard deviations before and after a natural turn training session with activated feedback.

step	foot	standard deviation before training (in ms)	standard deviation after training (in ms)	improvement (in ms)
first	man right	-	-	
first	woman left	73,88	85,97	-12,10
fourth	man left	112,81	89,18	23,63
fourth	woman right	99,54	93,73	5,82
	combined	95,41	89,63	5,78

Table 7.6: Development of dancer's standard deviations of step timings for natural turn phase without feedback. Underlying data for figure 7.6. Saltate! detected too few steps for the men's right feet to calculate a reasonable standard deviation.

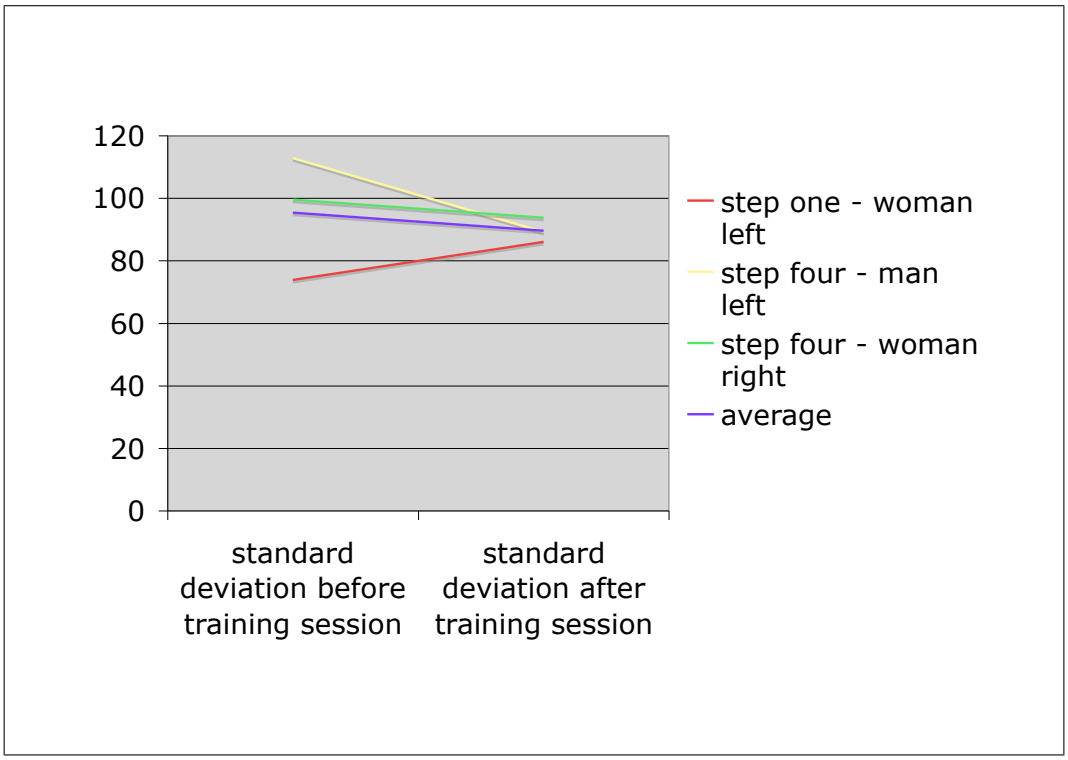

Figure 7.6: Standard deviations before and after a natural turn training session without feedback.

	with feedback	without feedback
basic step	51.87	35.07
natural turn	91.65	5.78

Table 7.7: The improvements of dancer's standard deviations of step timings under different conditions. Underlying data for figure 7.7.

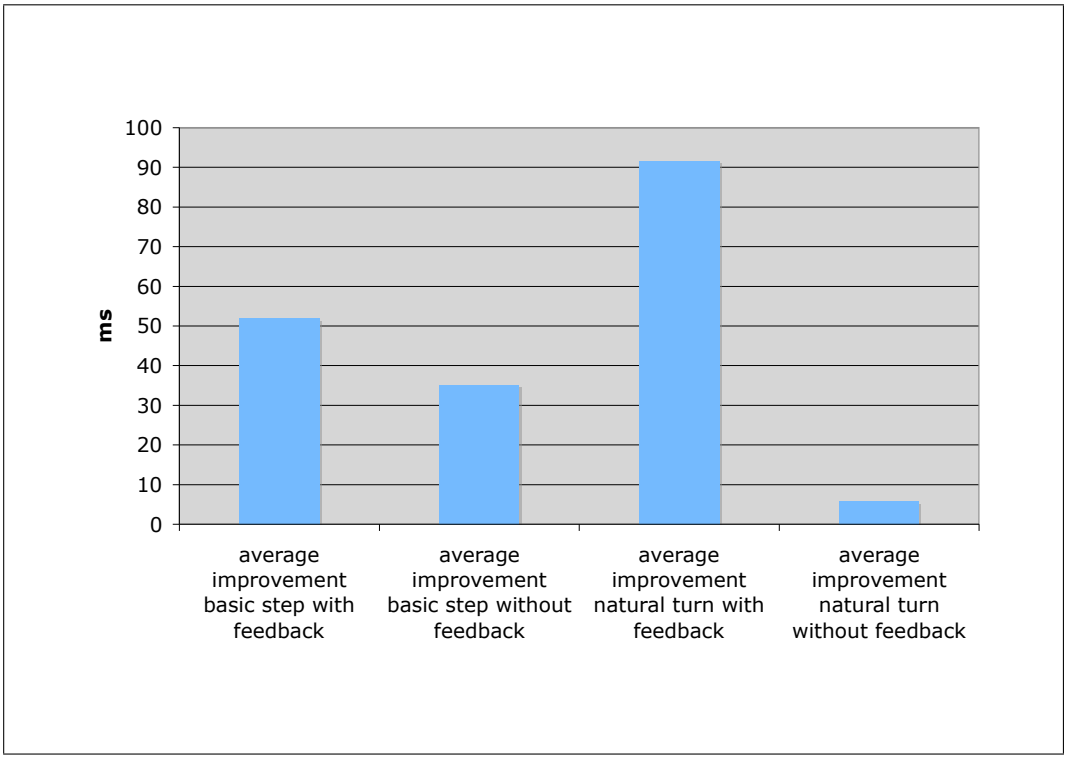

Figure 7.7: Different improvements of standard deviations under different conditions. Note that the big difference under the natural turn condition might be caused by partly missing data.

7.4 Questionnaire results

7.4.1 Overview of participants

Most of our volunteers were students.

Our 16 participants' average age was 26.43 years with a standard deviation of 4.84 years. We had 12 students and one PhD student, accountant, physiotherapist, and IT-professional. Five of our eight couples knew each other before. We used a Likert scale ranging from one to five to calculate average assessments of our participants (1 - strongly disagree, 2 - disagree, 3 - neither agree nor disagree, 4 - agree, 5 - strongly agree).

On average, our participants disagreed to have experience in dancing and to dance without problems (average 2.19, standard deviation 0.98). This self-assessment was lower

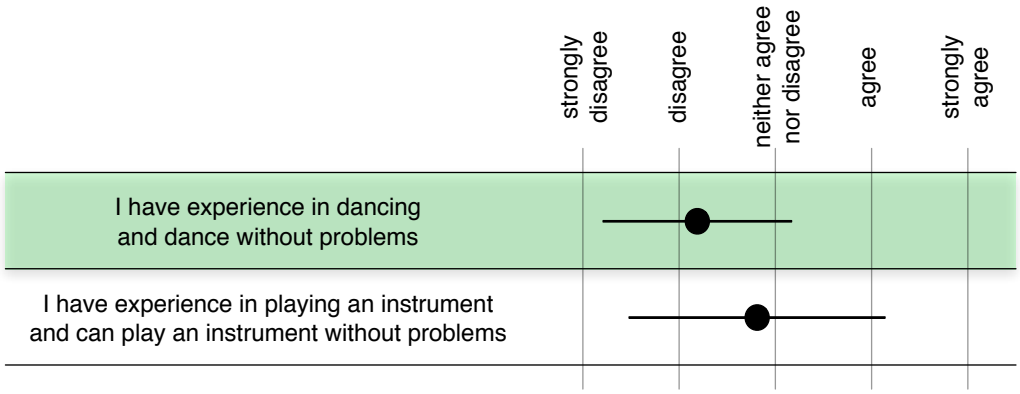

Figure 7.8: Participants' experience in dancing and music playing.

for the four couples we put into the less-experienced group: We calculated an average value of 1.88 (std.dev. 0.99) for the less experienced couples and an average value of 2.5 (st.dev. 0.93) for the more experienced couples. Our participants' experience in playing an instrument was very diverse (avg. 2.81, std.dev. 1.33).

7.4.2 Participants' self-assessment

At the beginning of the experiment the more experienced dancers were able to dance without greater problems (avg. 3.25, std.dev. 0.89) whereas the inexperienced dancers faced smaller problems to dance in step (avg. 2.25, std.dev. 0.71). At the end of the session, both groups were able to dance without problems (avg. 3.75, std.dev. 0.45). All of our participants pretty much enjoyed to take part in the experiment (avg. 4.63, std.dev 0.5), though less of them felt relaxed (avg. 3.38, std.dev. 0.96).

7.4.3 Participants' assessment of Saltate!

Our participants agreed that the sensor modules were comfortable to wear (avg. 4.31, std.dev. 0.70). Saltate!'s supporting functions were considered helpful (avg. 4.38, std.dev. 0.62) and to come at the appropriate times (avg. 4.06,

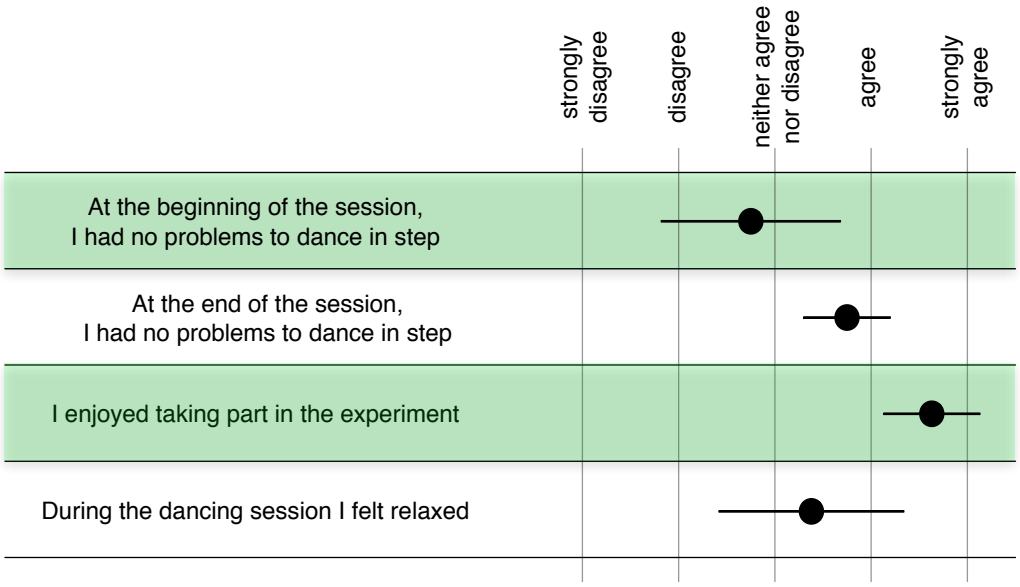

Figure 7.9: Participants' self-assessment during the experiment.

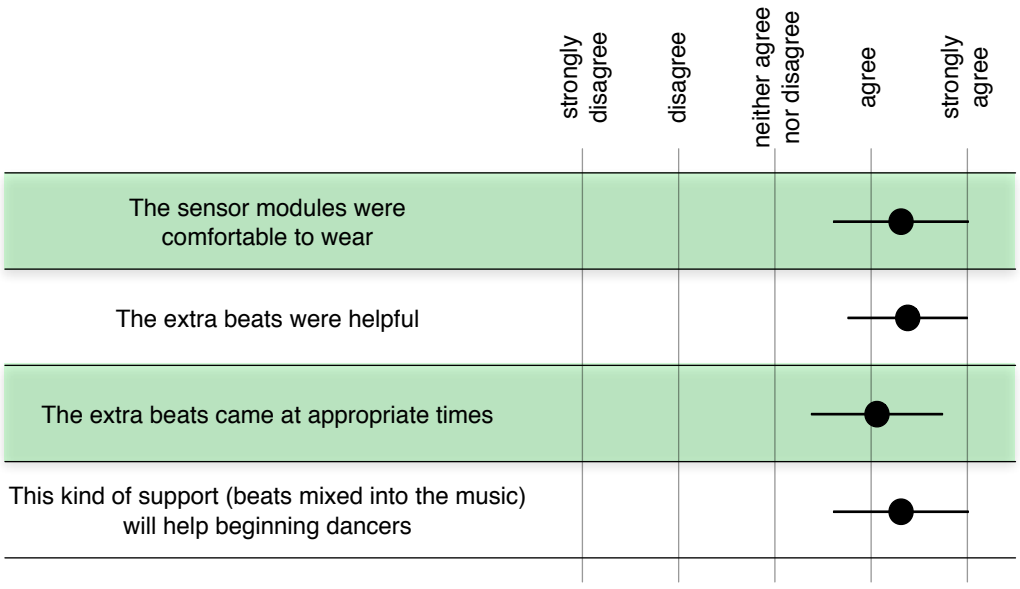

Figure 7.10: Participants' assessment of Saltate!.

std.dev. 0.68). Our participants believed that this kind of feedback will help beginning dancers (avg. 4.31, std.dev. 0.70).

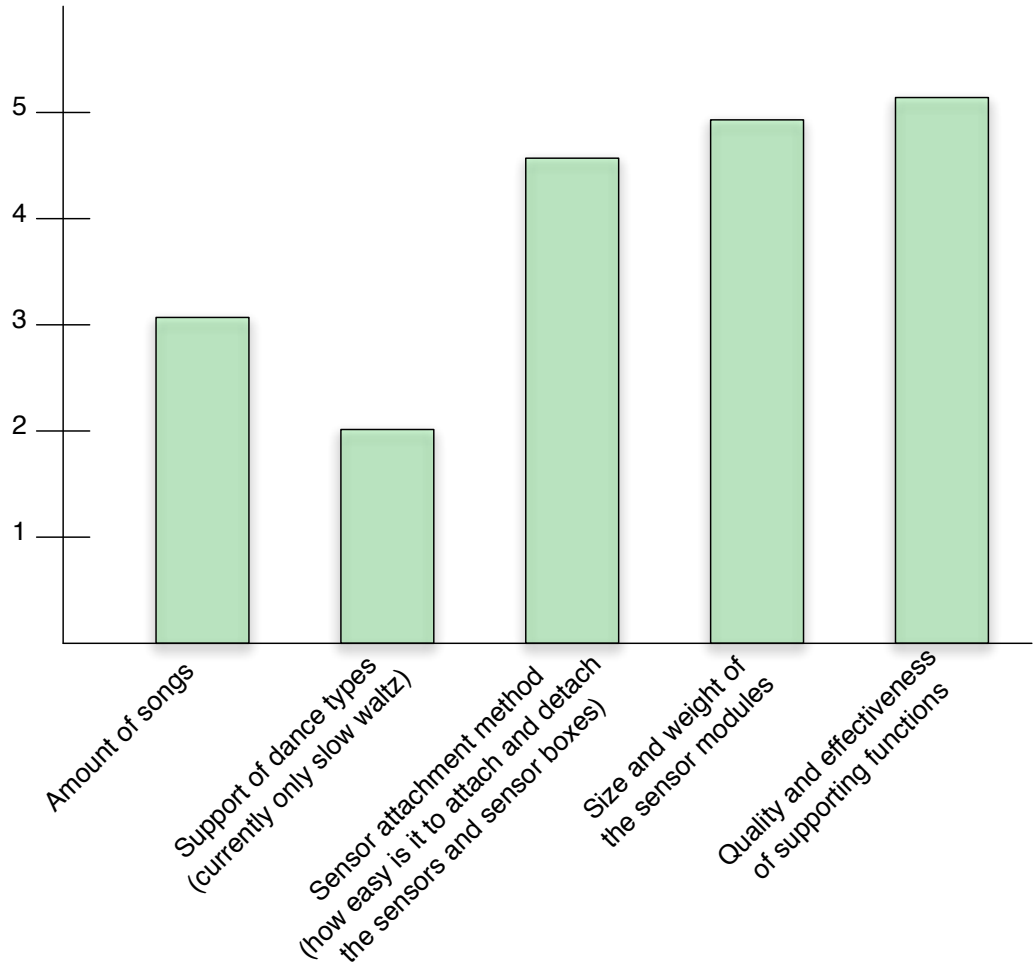

Figure 7.11: Participants' assessment of Saltate!'s relative strengths and weaknesses

In our last question we asked our participants to distribute 20 points among several aspects of Saltate!, assigning more points to specific aspects the better they consider them to be. Two of our participants misunderstood the question so that we could only calculate average values using the opinions of six participants.

7.5 Discussion

The successfully acquired data from four very inexperienced dancers confirmed our assumption that, at the end of the session, the dancing couples would dance more stable (with regards to their step timings' standard deviation), and longer in terms of danced steps per time.

When we consider only the amount of correctly danced steps, we can say that after approximately eight minutes of dancing basic steps or natural turns, it is no problem even for beginning dancers to perform these.

The standard deviation of step timings is suited to determine a dancer's ability.

Our beginning dancers' standard deviation of step timings decreased from about 130ms to 80ms in the basic step phase and from about 125ms to 90ms in the natural turn phase. During Saltate!'s development we measured standard deviations for experienced and highly experienced dancers between about 50ms and 75ms, which leaves a bigger gap between beginning dancers and experienced dancers than the amount of danced steps. The latter value reached values close to 100 percent after already a short training time. Therefore, standard deviations of step timings are better suited to distinguish between beginning and experienced dancers than the amount of danced steps.

When we compare improvements under feedback and under no-feedback condition our sampled values show more improvements under feedback condition when dancing basic steps. A higher improvement was also measured when dancing natural turns, but the differences in standard deviations measured before the training phase were too big to allow any interpretation.

Saltate!'s supporting function does not negatively influence learning.

The fact that the amount of correctly danced steps did not decrease in the first song after a training session with feedback shows that the dancing couples did not unintentionally learn to rely on Saltate!'s feedback to dance. Therefore, we assume that Saltate! successfully avoids the usual negative effect of concurrent feedback, a decreased learning.

Sensor boxes are small and light enough to not disturb their wearer.

The questionnaire's evaluation showed that our efforts to keep the hardware small and of light weight payed off. Our participants agreed that the sensor modules are comfortable to wear. When we take the comments of our subjects into account, they had nothing to complain about the size and weight of the sensor boxes - one subject commented that he "didn't even recognize the sensors". Improvements are possible in the attachment and size of the sensors themselves: Another subject commented that her shoes became pretty tight with shoe insoles and therefore the ball sensors slightly hurt when standing or dancing on the balls. Two subjects proposed to try out attaching sensor boxes at the ankles.

Users considered Saltate!'s support as very helpful.

Our subjects perceived Saltate!'s feedback as very helpful and assumed that this kind of support will help beginning dancers. Three of the more inexperienced dancers commented on the feedback, one of them felt that the extra beats "help a lot when you lost the rhythm", another one said that they could come more often, the third one mentioned that "it was easier to begin the songs with the extra beats", and that it was a "great idea!".

Our subjects' opinion about relative strengths and weaknesses mirrors our efforts: The sensor modules' size and weight, quality and effectiveness of supporting functions, and sensor attachment method were considered to be clearly better than the amount of songs Saltate! supports, or the support of different dances.

Chapter 8

Summary and future work

"Good questions outrank easy answers"

—Paul A. Samuelson

8.1 Summary and contributions

Within this thesis' project we developed hardware and software for Saltate!, a first sensor based system to support dance beginners, and tested it both with experienced (during its development) and inexperienced dancers (during its evaluation).

Saltate!'s sensor boxes are small and light enough to mount them onto a shoe without disturbing the wearer. Two additional sensors could be attached to the sensor boxes with very little effort. Saltate!'s hardware components are easy to handle without expert knowledge in electrical engineering, allowing for easy modifications or application in other domains than dancing.

The system measures dancers' step timings and compares these to beat timings within the music. We gathered enough data to estimate that the typical standard deviation

of these timings of experienced dancers is in the range of about 50ms.

Saltate! eases beat detection for beginning dancers by playing extra sound files to the beats of the music. The volume of these sounds is automatically adjusted by Saltate! according to a dancing couple's performance. Reactions of volunteers showed that this kind of support is taken very positive by beginning dancers.

The software's architecture allows an easy exchange of sound files as well as modifications of timings at which these should be played relative to each bar of a music piece.

8.2 Future work

After conducting the final evaluation, we have several ideas for improving and using Saltate!:

8.2.1 Improved step detection

Our current step detection suffers from two weaknesses:

1. Using only pressure sensors makes it impossible to decide into which direction a foot moves. Though Saltate!'s distinction between forward and backward steps works fine, there is no way at all to identify whether a foot is moving to the left or to the right. The additional use of acceleration sensors should allow Saltate! to decide into which direction a foot is moving. When this information is present, the system could automatically detect if a couple dances one beat behind or ahead of the music.

2. The currently used calibration method and step detection algorithm should be improved. We assume that this is not trivial: Several times, after one song with very good step detection, step detection seriously decreased for the same couple of dancers using

the same calibration values. If accelerometers are included into the system, their data might be very useful for the step timing detection as well - once a foot is set on the ground, is is not accelerated any more. Another solution might be an "automatic recalibration": Saltate! is already able to compensate a couple of not detected steps. If the system assumes that a step has been missed around a specific point of time (at which a step from the dance partner might have been recognized), an analysis of sensor data around this point of time could lead to a better calibration value. Another approach would be the use of calibration free detection algorithms, which could work based on value changes instead of value thresholds.

In order to test more complex new step detection algorithms, a function should be developed that allows the replaying of recorded data to analyze them with new detection algorithms.

8.2.2 Radio network change to API mode

The currently used XBee network in transparent mode leads to huge latencies in case of several broadcasts: If several broadcasts are sent in a short time, the whole network seems to stop for a couple of seconds[1]. By now, this was not a problem as several broadcasts were only sent during synchronizations, which were initiated manually when no song was playing. A more flexible API mode network would allow more frequent messages from the Saltate! software to the sensor modules. This might be useful for an improved step detection. In addition, this should allow an automatic synchronization process during or between songs.

8.2.3 Improved error detection

Currently, each recognized step is mapped to one beat. Before conducting new tests, we propose to mark a step as

[1] Digi's support informed us that this was inevitable

wrong if its time difference to the closest beat becomes too big. Since the couple with the highest dancing ability we tested[2] had a rather huge time differences of up to 200ms, the time difference that Saltate! considers as correct should still be rather huge.

Such an improved error detection would allow a better detection of a "drifting" couple that continues to dance but does not dance at the speed of the music any more.

8.2.4 Quantitative evaluation

Our chosen experimental setup was too easy for couples with some dancing experience. The task to dance only basic steps or only natural turns was not demanding enough for those couples to evaluate Saltate!'s supporting function, as it was - sometimes - not activated at all. We propose to conduct a study similar to the one we made, if possible with an improved version of Saltate!. Variations should be made on the task, which should be made harder.

For a psychological evaluation of the effects of feedback like Saltate!'s on motor skill learning we recommend not to use a widely known dance. To analyze the effects of adaptive feedback it would be better to create new dance steps that are to be performed by only one person. This eliminates all disturbing effects of different relationships of a couple's dancers to each other and minimizes effects of different experience: People who have already taken a dancing course could not make use of their specific knowledge regarding the dance they have to perform any more.

8.2.5 Accuracy determination

We mentioned in 7.1.1—"Independent, dependent, and interfering variables" that we are not sure in how far mean differences of step timings to music beats between different songs are comparable, they vary depending possibly on the

[2] Both danced or have danced in competitions, though not together.

music, sensor attachment, and system measurements. We are sure that the system is very accurate within each song and that the inaccuracies possibly introduced between different songs are smaller than the usual standard deviation. However, if you want to analyze these mean differences, the system's accuracy with regards to those should be determined first.

8.2.6 More dances and different feedbacks

Saltate!'s current version, which provides support only for Slow Waltz, can only be seen as a prototype and proof of concept for a dance training system. Support for more dances is highly desirable, as well as the detection of more dance figures. For dancers who have passed the initial problems of dancing to the beats of the music additional kinds of feedback should be introduced.

If a couple already knows several dance moves, feedback that informs it about the proportion of danced moves could help to actually use all of them when dancing free, i.e., without a predetermined move sequence. Of course, feedback regarding the correct posture is desirable as well, but this will not be possible with sensors only on the feet.

Appendix A

Saltate! questionnaire

The following four pages show the questionnaire we handed out to our participants.

Saltate! Questionnaire

ID: _____

Thank you for taking part in the Saltate! evaluation. Saltate! is a sensor based system to support dance beginners. By taking part in its evaluation you help us to determine in how far the current system might or might not help beginners, and where improvements are necessary. Try to dance as good as you can, and restart dancing should you get out of the rhythm.

On these pages, we have some questions to you. If you have problems to understand them, feel free to ask the instructor for help. If there are several options to choose, and none of them fits perfectly, simply choose the option that fits best.

My participation in this evaluation is voluntary. I accept that the data acquired during my participation will be used for scientific research after anonymization, i.e., my name, or any other data of mine that might be used to identify myself will not be made public.

_____
Signature

Age: _____ Gender (male/female): _____

Occupation: _____

1) I have experience in dancing and dance without problems:

◯	◯	◯	◯	◯
strongly disagree	disagree	neither agree nor disagree	agree	strongly agree

2) I have experience in playing an instrument and can play an instrument without problems

◯	◯	◯	◯	◯
strongly disagree	disagree	neither agree nor disagree	agree	strongly agree

3) Did you know your dance partner before?

◯ No ◯ Yes

Figure A.1: Saltate! questionnaire, page 1

Saltate! Questionnaire

4) At the beginning of the session, I had no problems to dance in step

◯	◯	◯	◯	◯
strongly disagree	disagree	neither agree nor disagree	agree	strongly agree

5) At the end of the session, I had no problems to dance in step

◯	◯	◯	◯	◯
strongly disagree	disagree	neither agree nor disagree	agree	strongly agree

6) I enjoyed taking part in the experiment

◯	◯	◯	◯	◯
strongly disagree	disagree	neither agree nor disagree	agree	strongly agree

7) During the dancing session I felt relaxed

◯	◯	◯	◯	◯
strongly disagree	disagree	neither agree nor disagree	agree	strongly agree

8) The sensor modules were comfortable to wear

◯	◯	◯	◯	◯
strongly disagree	disagree	neither agree nor disagree	agree	strongly agree

comments (if you have any):

Figure A.2: Saltate! questionnaire, page 2

Saltate! Questionnaire

If you recognized the system's supporting function (extra beats mixed into the music) please answer the following two questions

9) The extra beats were helpful

○	○	○	○	○
strongly disagree	disagree	neither agree nor disagree	agree	strongly agree

10) The extra beats came at appropriate times

○	○	○	○	○
strongly disagree	disagree	neither agree nor disagree	agree	strongly agree

comments about the extra beats (if you have any):

11) This kind of support (beats mixed into the music) will help beginning dancers.

○	○	○	○	○
strongly disagree	disagree	neither agree nor disagree	agree	strongly agree

Figure A.3: Saltate! questionnaire, page 3

Saltate! Questionnaire

12) Saltate! is not a completed system. When you have finished the dancing session, you have heard all songs which are currently supported. Other dances than Slow Waltz are not supported yet. We would like to know about the relative strengths and weaknesses of Saltate!, so that we know on which parts we should concentrate further development.

Please distribute 20 “quality points” among the following categories. The more points you give, the better do you consider this category in comparison to the others. You can use free space to experiment with the point distribution. Please write your final distribution into the rectangular boxes.

☐ Amount of songs

☐ Support of dance types (currently only slow waltz)

☐ Sensor attachment method (how easy is it to attach and detach the sensors and sensor boxes)

☐ Size and weight of the sensor modules

☐ Quality and effectiveness of supporting functions

If you have any additional criticism or idea for improvement, or any other comment, please write it down:

Thank you for your participation!

page 4/4

Figure A.4: Saltate! questionnaire, page 4

Appendix B

Pictures and sketches

On the following pages, we present some more photos and sketches of Saltate!'s hardware.

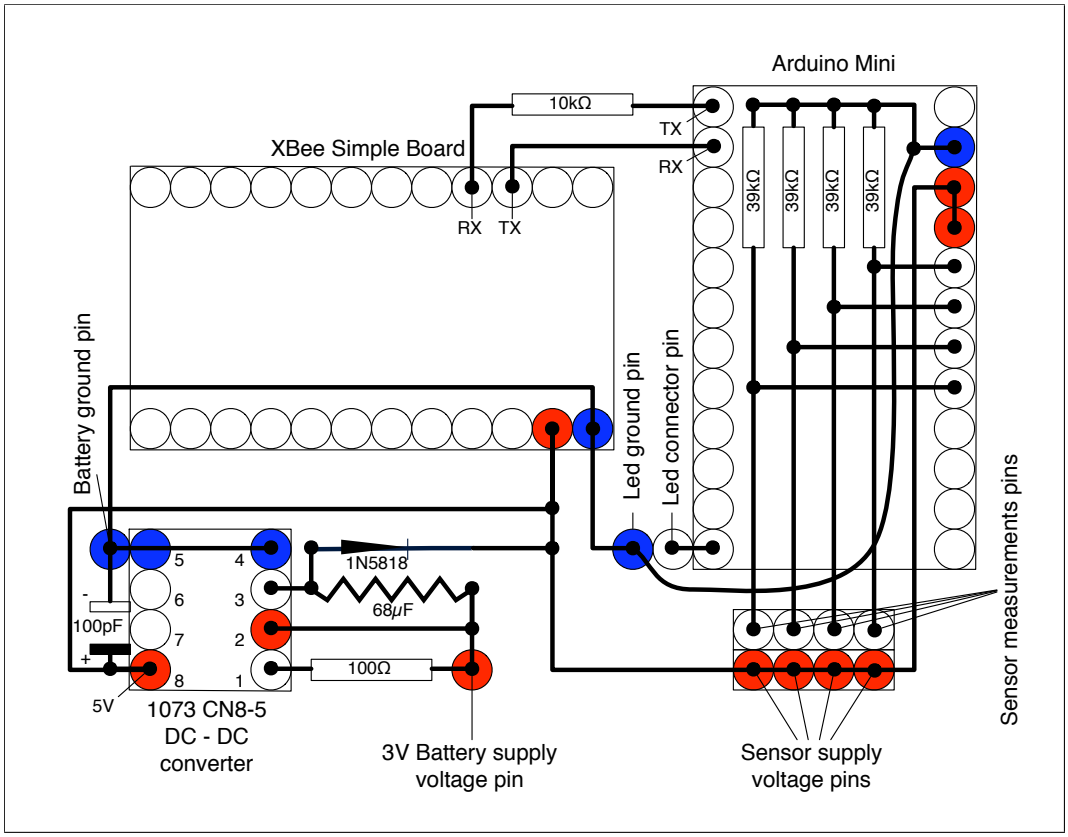

Figure B.1: The sensor boxes' internal wiring. Supply voltage is marked in red, ground voltage in blue.

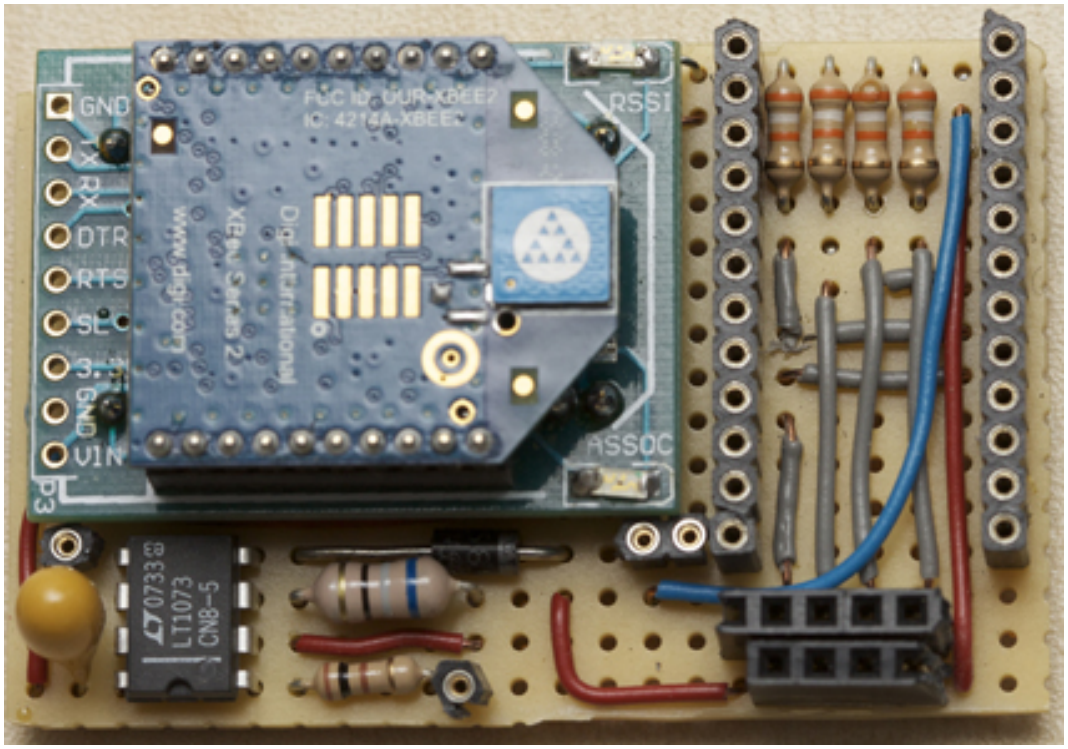

Figure B.2: The sensor boxes' electronics. We removed the Arduino Mini in the top right corner to show the resistors and cables soldered beneath it. The simple board in the top left corner beneath the XBee module is soldered onto the circuit board.

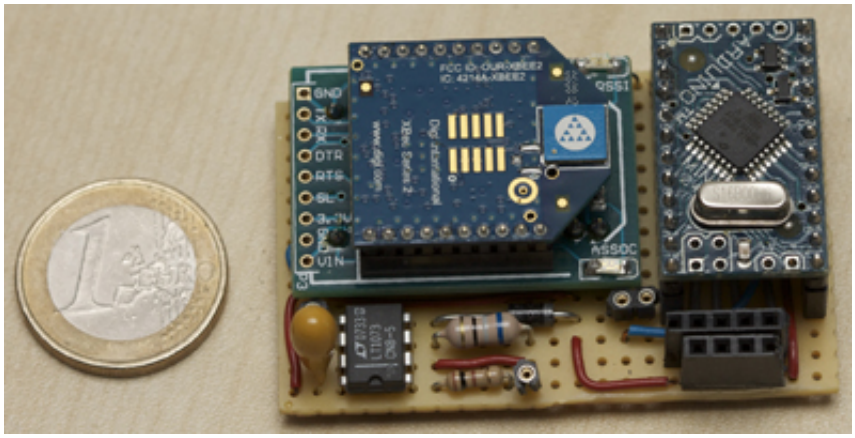

Figure B.3: The sensor boxes' electronics. The XBee module is mounted on a Simple Board in the top left corner, the Arduino Mini is positioned in the top right corner. Sensor connections are in the bottom right corner, the bottom left corner transforms 3V input voltage from the batteries to 5V voltage for the Arduino Mini and the Simple Board.

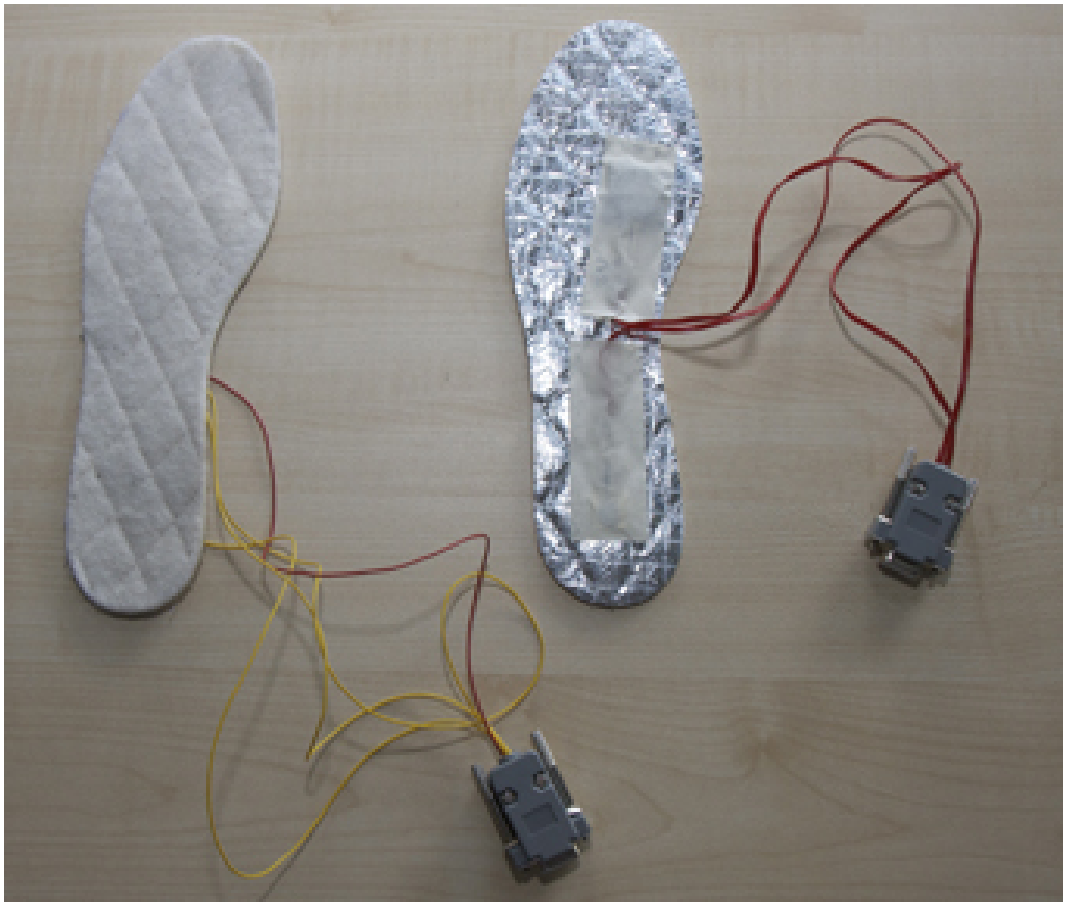

Figure B.4: Two shoe insoles with attached sensors: Top view on the left side, bottom view on the right sight.

Figure B.5: The bottom of one of Saltate!'s sensor boxes. The ruler is flexible and can be pushed under a shoe lace.

Figure B.6: A sensor box with attached sensor cables.

Figure B.7: An opened sensor box.

Bibliography

Sofia Dahl and Roberto Bresin. Is the player more influenced by the auditory than the tactile feedback from the instrument. In *Proceedings of the Cost-G6 Conference Digital Audio Effects*, 2001.

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

Urs Enke. Dansense: Rhythmic analysis of dance movements using acceleration-onset times. Master's thesis, RWTH Aachen University, Aachen, Germany, September 2006.

Fahrmeir, Künstler, Pigeot, and Tutz. *Statistik*. Springer, 2007.

Björn Hartmann, Leith Abdulla, Manas Mittal, and Scott R. Klemmer. Authoring sensor-based interactions by demonstration with direct manipulation and pattern recognition. In *CHI '07: Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 145–154, New York, NY, USA, 2007. ACM. ISBN 978-1-59593-593-9. doi: http://doi.acm.org/10.1145/1240624.1240646.

James F. Knight, Huw W. Bristow, Stamatina Anastopoulou, Chris Baber, Anthony Schwirtz, and Theodoros N. Arvanitis. Uses of accelerometer data collected from a wearable system. *Personal Ubiquitous Comput.*, 11(2):117–132, 2007. ISSN 1617-4909. doi: http://dx.doi.org/10.1007/s00779-006-0070-y.

Kazuhiro Kosuge, Tomohiro Hayashi, Yasuhisa Hirata, and Ryosuke Tobiyama. Dance partner robot -ms dancer-. In

Proceedings of the 2003 IEEE/RSJ International Conference on Intelligent Robots and Systems, 2003.

Doo Young Kwon and Markus Gross. Combining body sensors and visual sensors for motion training. In *ACE '05: Proceedings of the 2005 ACM SIGCHI International Conference on Advances in computer entertainment technology*, pages 94–101, New York, NY, USA, 2005. ACM. ISBN 1-59593-110-4. doi: http://doi.acm.org/10.1145/1178477.1178490.

Daniel Levitin, Karon MacLean, Max Mathews, and Lonny Chu. The perception of cross-modal simultaneity. *Proceedings of International Journal of Computing Anticipatory Systems*, 1999.

D.W.V. Linden, J.H. Cauraugh, and T.A. Greene. The effect of frequency of kinetic feedback on learning an isometric force production task in nondisabled subjects. In *Physical Therapy*, volume 73, 1993.

Teemu Mäki-Patola and Perttu Hämäläinen. Latency tolerance for gesture controlled continuous sound instrument without tactile feedback. In *Proc. International Computer Music Conference*, 2004.

N.H. McNevin, G. Wulf, and C. Carlson. Effects of attentional focus, self-control, and dyad training on motor learning: Implications for physical rehabilitation. In *Physical Therapy*, 2000.

Alex Moore. *The Ballroom Technique*. Imperial Society of Teachers of Dancing, 1994 edition, 1982.

Akio Nakamura, Sou Tabata, Tomoya Ueda, Shinichiro Kiyofuji, and Yoshinori Kuno. Multimodal presentation method for a dance training system. In *CHI '05: CHI '05 extended abstracts on Human factors in computing systems*, pages 1685–1688, New York, NY, USA, 2005. ACM. ISBN 1-59593-002-7. doi: http://doi.acm.org/10.1145/1056808.1056997.

J. A. Paradiso, K. Hsiao, A. Y. Benbasat, and Z. Teegarden. Design and implementation of expressive footwear. *IBM Syst. J.*, 39(3-4):511–529, 2000. ISSN 0018-8670.

R.A. Schmidt and G. Wulf. Continuous concurrent feedback degrades skill learning : Implications for training and simulation. In *Human factors*, 1997.

Daniel Spelmezan and Jan Borchers. Real-time snowboard training system. In *CHI '08: CHI '08 extended abstracts on Human factors in computing systems*, pages 3327–3332, New York, NY, USA, 2008. ACM. ISBN 978-1-60558-012-X. doi: http://doi.acm.org/10.1145/1358628.1358852.

Masami Takahata, Kensuke Shiraki, Yutaka Sakane, and Yoichi Takebayashi. Sound feedback for powerful karate training. In *NIME '04: Proceedings of the 2004 conference on New interfaces for musical expression*, pages 13–18, Singapore, Singapore, 2004. National University of Singapore.

C.J. Winstein, P.S. Pohl, C.Cardinale, A.Green, L. Scholtz, and C.S. Waters. Learning a partial-weight-bearing skill: Effectiveness of two forms of learning a partial-weight-bearing skill: Effectiveness of two forms of feedback. In *Physical Therapy*, volume 76, 1996.

Matthew Wright. Problems and prospects for intimate and satisfying sensor-based control of computer sound. In *Symposium on Sensing and Input for Media-Centric Systems*, 2002.

Gabriele Wulf. *Attention and Motor Skill Learning*. Human Kinetics, 2007.

Haiyan Zhang and Björn Hartmann. Building upon everyday play. In *CHI '07: CHI '07 extended abstracts on Human factors in computing systems*, pages 2019–2024, New York, NY, USA, 2007. ACM. ISBN 978-1-59593-642-4. doi: http://doi.acm.org/10.1145/1240866.1240942.

Index

Typeset December 27, 2008