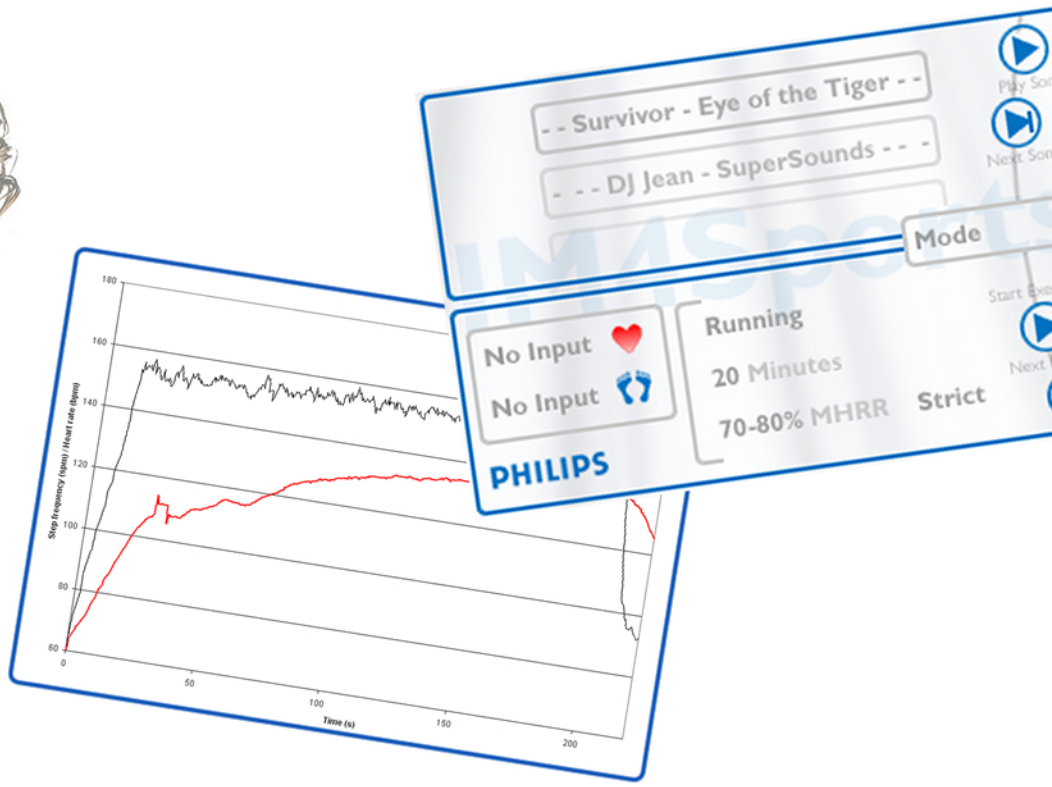
IM4Sports

Interactive Music for Sports

A personalised music system for motivation in sport performance



Master's thesis

Gertjan L. Wijnalda

Philips Research
Vrije Universiteit Amsterdam

PHILIPS

vrije Universiteit amsterdam

Interactive Music for Sports

A personalised music system for motivation in sport performance

Gertjan L. Wijnalda

Master's thesis
August 2005

supervisors
dr. H. Stuckenschmidt (Vrije Universiteit Amsterdam)
dr. S.C. Pauws (Philips Research)
dr.ir. F. Vignoli (Philips Research)

2nd reader
prof.dr. F.A.H. van Harmelen (Vrije Universiteit Amsterdam)

vrije Universiteit *amsterdam*

E-mail g.l.wijnalda@redant.nl
Homepage http://www.redant.nl/g.l.wijnalda/

This thesis was typeset using LaTeX 2ε.

Abstract

Listening to music makes endurance exercising more enjoyable, and motivates athletes to keep up with their exercising goals. The IM4Sports system helps select music that suits a training program, changes playback to reflect or guide current sport performance, and collects data for adapting training programs and music selections.

The system consists of three phases. In the first phase, the user specifies the training program, which typically consists of a number of exercises, each of which is defined by a duration in time or distance, and intensity or target heart rate (zone). Based on the specified training program, the system suggests music for inclusion on the portable device.

In the second phase, during exercising, music is adapted to motivate users, according to the goals specified in the training program. To this end, the system utilises input from a pair of sensors to measure the current physiological state of the user: one pedometer to measure step frequency, and one heart rate meter. All data from these sensors, as well as the chosen music adaptations and music selection, is logged during this phase. Experiments point out that an interval over which a change in music tempo is propagated must be observed, and that the system is able to detect when a stabilisation in physiological state, more specifically heart rate, occurs.

Finally, in the third phase, after exercising, the data that was logged during exercising is used to personalise music selection. The system uses a Bayesian inference method that updates an initial music tempo distribution with the data obtained during exercise. An evaluation on the key properties of flexibility, consolidation capacity and resistance to noise shows that the proposed method performs better than the traditional methods it was based upon.

User tests show a general positive review of system performance, along with highly enthusiastic remarks on the system and its concept.

Keywords

Music, multimedia applications, multimedia signal processing, personalisation, sports

Acknowledgements

While writing this thesis, first and foremost I've had help and wisdom from the most enthusiastic supervisors I could ever have imagined.

There have been a couple of fellow students have been particularly helpful during my research: Mark Vossen, who kindly advised me on his APG algorithm and my writings on it. Tijn Schuurmans, who managed to talk with heart rate belts, pedometers and music players so that actually some noise could be made (and adapted!). Finally, Erwin van Iperen, who gave me the first ideas on using Bayesian statistics for a part of my problems.

The following individuals all gave me an inspirational hand in the process: Jan van Herk at Philips, who made the research prototype from heart rate sensors and pedometers. Paul Meagher at Datavore, who gave some valuable Bayesian hints. Elmo Diederiks at Philips, who designed the user interface. The friendly people at Technogym who provided information on programming their treadmill. Renee Timmers from Radboud University for her suggestions on literature. Jos van den Berg at the VU for his comments on the prototype hardware.

Also, many thanks to my friends, especially Paul Hooijmans, for allowing me to explain what I have been working on for hours in total, and the students at the Media Interaction group at Philips who made my time there so enjoyable.

Last but not least, my mother, my brother Thom and my girlfriend Liesbeth provided their love, support and understanding for which I can never thank them enough.

This thesis is dedicated to my father, without whom I probably never would have written it. You were at the very start of this, and have been my continuous inspiration ever since.

Table of Contents

List of Figures and Tables

Chapter 1

Introduction

1.1 The Problems in Sports Exercising

In present-day life, the trend towards a more service-oriented society and thus towards more daytime office jobs continuously restrains opportunity for physical movement[1]. People are gradually becoming less fit and their weight increases to unhealthy levels. A few years ago this development inspired the spring of heaps of fitness studios, where people participate in individual, often repetitive sports such as (dry) rowing, cycling and running. Currently, over 30 million Americans frequently run for recreation or competition.

In order to be effective, these kinds of sports require perseverance, yet due to the lack of instant rewards often evoke feelings of pain, fatigue and boredom. A possible example effect is that people start exercising three times a week, yet this schedule quickly decreases to once per two weeks or even less. Motivation in these repetitive solo endurance sports is hard to find for large groups of individuals.

1.2 Research Goal

Exercisers have various techniques to cope with the physical demands of exercising, which include switching away their attention to other things. Listening to music can be of help as it attracts attention and is generally perceived as a pleasant activity, while it does not interfere with the physical aspects of the training session. In the next chapter, we will see that music is considered to increase the enjoyment of the exerciser, and hence can be an important motivator for them. Also, we will see that synchronizing the music to the motion of the exercise makes people able to endure the exercise longer.

In this thesis, we will describe a personalised music system called IM4Sports[2]. It is designed for individual sports exercising, in particular running. In Figure 1.1, you can see the research prototype. It consists of a personal computer, a portable music player with flash memory, a pedometer to measure the step frequency of the runner, and a heart rate sensor in the form of a waist belt.

The project's goal is to develop a system that is able to vary the music speed based on the characteristics of the training program and the current performance. To this end, a number of different modes are available which allow the user to freely listen to music, synchronise the music to the user's step frequency or adapt the music to ensure the user stays within a specified heart rate zone.

[1] According to the Central Bureau for Statistics (CBS) in The Netherlands, service-oriented jobs, including government jobs, increased from about 71% in 1987 to almost 80% in 2004.

[2] Interactive Music for Sports

Figure 1.1: Research prototype

Research Goal *The research goal is to develop a system that is able to stimulate, support and entertain an exerciser in endurance sports. In order to meet the goals set for exercising, it should employ music as auditory feedback on user performance. It also should adapt to people's exercise behaviour over time.*

In the remainder of this chapter, we will split up the research goal in a set of development requirements that we will pose on the system, and outline the remainder of this thesis.

1.3 Requirements Analysis

For ease of reference, an overview of all requirements that are described in the following chapter can be found in Appendix B.

1.3.1 General Requirements

The IM4Sports system employs technology to create an application that provides a personalised and adaptive fitness program that is driven by music. The system is meant for durability training programs, hence it should be able to work for all kinds of frequent interval sports such as running, cycling, rowing, stepping, when equipped with the right sensors to measure the pace of the exercise.

Incorporating different sensors in such a system than the ones it was designed for is a straightforward process, since the basic measure in the system, exercise intensity, does

exercise type	duration	intensity	continue
warm up	20 min.	-	to next
stretch	5 min.	-	with present
run	20 min.	50 - 60 %	to next
recover	-	40 %	to next
stretch	5 min.	-	with present
jog	3 km	-	with present
cool down	variable	-	with present

Table 1.2: Example training program

not change. Since we will focus on the system rather than the specific sport for which it is used, we will choose one particular sport. Running is a prime example of a solitary sport for which a music player is often used, there exist several commercially available sensors, lots of literature on running can be found, and most importantly, running is used as cross training for a lot of other different sports. Therefore, we choose running as our target sport and note that the system can be easily adapted to other repetitive endurance sports.

We will define a Functional Requirement **FR 1**, which states that the system's aim is to motivate and entertain runners using music.

We assume an existing fitness program is generated or given by the user as input for the system (Functional Requirement **FR 2**).

In a standard training program, a number of characteristics are always included that jointly define the program (Data Requirement **DR 1**). Other goals for the training program can always be defined in terms of the given variables (for example, burning a certain amount of calories can be estimated by checking heart rate and the exercise duration).

Possible extensions could be *mood* (is this a 'light' or a 'hard' exercise, directly linked to the desired heart rate percentage or kind of exercise, *e.g.* a warm up is a 'light' exercise), or *development over time* (how, for example, step frequency or mood of an exercise change over time).

An example of a training program is given in Table 1.2. The exercise type determines how the exercise will be treated by the system. For example, a warm up exercise will have different implications on the music selection than a running exercise will have. The duration is needed to determine the relative weight of the exercise in the training program. We need that weight to determine which percentage of the music selection should be tailored to that specific exercise. The duration can be specified in time, in distance, left unspecified or be 'variable'. The latter definition means that the exercise lasts until the exerciser wants to finish it. If no duration is specified, this denotes that the exercise lasts until a specific heart rate (zone) is reached, which is specified by the intensity column in the training program. We will go into more detail on the definition of a training program in Section 3.5.

The way in which the system is operated is an important aspect of its functionality. We define the system functionality in three stages, to make user interaction as straightforward as possible. In the first stage, before exercising, all data on the training session (training program, music, etc.) is programmed in the system. Because of its input qualities, which are already familiar to a majority of users, this is done on a personal computer, which we call the *Base Station*. It is easy for the user to change a training program, see what happens to the portable music collection, and determine all kinds of other settings when in the familiar environment of a PC.

The Base Station connects to a special portable music player, called the *Player*, and all

data is transferred to it. The Player does more than what its name suggests, though: apart from music playback it also takes care of music selection, the training program and the logging and handling of physiological sensory data. During exercising, all the user needs to carry is the Player, which is designed to be as small as possible while having all needed functionality.

In the final stage, after exercising, the Player is again connected to the Base Station and the data of the user's performance is transferred to the Base Station. The Base Station then processes the incoming data in order to provide the learning behaviour as described in the research goal.

After briefly recouping the requirements presented in this section, we will go through all three stages and briefly describe their requirements.

General Requirements

FR 1. *The aim of the system is to use music to provide a more stimulating and more entertaining running experience*

FR 2. *The system takes a given training program as input*

DR 1. *The training program should fit a specific profile that allows for standardised processing. This profile includes:*

- Kind of exercise *(Warm Up, Stretch, Walk/Run/Jog, Recover, Cool Down, etc);*
- Duration *(in seconds or in distance);*
- Intensity *(in heart rate percentage)*.

1.3.2 Before Exercising

The most important characteristic of the system is that it should use music to enhance the motivation of the exerciser. This can be done in two ways: selecting the right music for the right exercise, and adapting this music in real time to the user's performance. Possible ways for adaptation include changing the music tempo, dynamics and spatial position. In the first stage, before exercising, only the selection of songs is important.

The determination of the *order* in which songs are played can be determined either *before* or *during* exercising. We will go into depth on this design decision in Chapter 4. Independent on whether the ordering is done before or during exercising, the system needs to take into account all variables that influence song selection, such as the users preference, the physiological state of the user and the desired performance (Functional Requirement **FR 3**).

The IM4Sports system cannot use a hard disk based solution, because a hard disk is a moving part and as such may soon be damaged by the tensions put on it during the training session. This means a flash-based player should be used, as it does not contain moving parts. Because storage space on a portable flash player is limited (4 gigabytes maximum, but more like 512 megabytes - 1 gigabytes on average) and usually (much) smaller than the user's music collection, the system has to give a suggestion for the songs the user can put on her portable device. We define that those songs are selected, that best fit the characteristics of the different exercises in the fitness program (Functional Requirement **FR 4**)

In order to correctly address **FR 4**, additional information on the exercises in the training program is required. These may be calculated on the fly from earlier training sessions or from general user group data. For example, a certain heart rate percentage for a certain kind of exercise means that the exercise has an approximate step frequency at which the user has to run, which varies from user to user and over time as a user gets more trained.

From this requirement follows that the system should have data on all songs in the user's music collection (Data Requirement **DR 2**), such as song title, artist, tempo, genre and duration. Usually all this meta data is readily available when the songs are stored in the popular MP3[3] codec-format, which usually is equipped with so-called ID3 'tags'. A 'tag' is data stored in an MP3 (as well as other formats) which contains meta data on the music as described above.

Meta data can be used to impose constraints on music selection, such as 'don't select more than 10% rock songs' or 'select at least 20% songs by Marillion'. In addition, the song's *meter* can be used to limit song selection to, for example, only two-fold meters, and the song's tempo to select songs that best match the expected average running tempo of an exercise.

Requirements Before Exercising

FR 3. *The system should select music that fits the users (projected) performance*

FR 4. *The system should select songs from the database that best fit the characteristics of the different exercises in the fitness program, and suggest these as the new contents of the Player*

DR 2. *The following meta data on the music should be available to the system:*

- *Song title*
- *Artist*
- *Album*
- *Song duration*
- *Genre*
- *Tempo*
- *Meter*

1.3.3 During Exercising

The adaptation of music should take place during exercising. Small changes in song parameters such as music tempo, dynamics and spatial position may be made to reflect the user's performance (Functional Requirement **FR 5**).

This means, for example, that when the users heart rate is too low, the user has to increase exercise intensity and hence, the music must accelerate. Data on the effects of a change in the tempo of the music on the exercise step frequency are needed before such a change can effectively be implemented in the system (Data Requirement **DR 3**). For example, when the music speeds up with 10 bpm, we need to know whether the change in the user's step frequency will be proportional to this amount.

The system should allow the user to interact with it during exercising. A lot of the settings of the system could be changeable in a menu-oriented interface, in which the user can change parameters of the training program (*e.g.*, the target heart rate zone). This is considered a feature of the final product. In addition, however, the system should allow easy interaction for the basic parameters of the system. We specify song control functionality (play, pause, skip, repeat), exercise control functionality (stop, skip) and mode selection (Functional Requirement **FR 6**). An example is depicted in Figure 1.3.

To achieve a user friendly experience, it is also important that the sensors used to acquire physiological data are easily usable. Hence, they should allow the system to take reliable

[3] MPEG-1 Audio Layer 3, a lossy compression format which allows digital audio data to be compressed to a ratio of about 10:1 depending on codec settings.

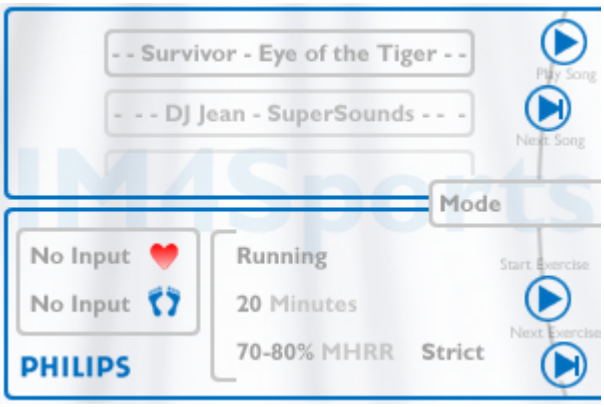


Figure 1.3: Example of user interaction controls

measurements and the user must be able to easily attach and remove them. Their number should not be exaggerated, so possibly the use of sensors should be limited to two, as a high number would restrain the user in exercising instead of motivating her (Functional Requirement **FR 7**). This means only the most important physiological data can be collected and must be used to achieve the desired result, that is, motivation by music.

Requirements During Exercising

FR 5. *The system should adapt the played back music to the user performance*

FR 6. *The system should allow easy interaction during exercising, more specifically:*

- pause song: *because the system may be used in complex environments, such as alongside roads with heavy traffic, a pause button is important - in the example figure 1.3, any part of the screen can be used to pause the music;*
- play/repeat song: *depending on the mode the system is in, the button changes to allow the user to indicate she wants to play a song (in a mode that does not automatically start playing music) or to indicate the song should be repeated (in modes when the song is automatically changed when finished);*
- skip song: *the user wants to skip to another (similar) song;*
- stop exercise: *the user wants to stop exercising;*
- skip exercise: *the user wants to skip the remainder of the current exercise, regardless of whether the exercise is finished or not;*
- mode: *the user can select the playback/exercise mode of the device.*

FR 7. *The sensors used by the system should be easily attachable, removable and usable, and their number should not exceed two*

DR 3. *The effects of a change in a music parameter on the user performance should be available*

1.3.4 After Exercising

The data that is collected during exercising is processed in the final stage, after exercising. To this end, the Player is again connected to the Base Station and all data is downloaded to the Base Station. A (graphical) overview of the users performance may be shown, including music played and tracks or exercises skipped/repeated. Displaying performance may have a motivational effect, as well. However, the most important functionality in this stage is that the system learns from the user's data by taking it into account the next time a training program is determined (Functional Requirement **FR 8**).

Learning is also mentioned in the research goal: the system uses the collected data for its learning functionality. For example, when the user has utilised the system's controls to skip a particular song, the system would assign a lower probability to that song next time the music collection on the Player is determined. When in one particular exercise one particular music tempo has been used, we might select more songs with that tempo the next time the exercise is included in a training program.

To this end, the system should collect data during exercise that allows for an overview of which songs were selected, which buttons were pressed, and which music parameters have been changed in order to adapt to the user's performance, in addition to the user's physiological data read by the system's sensors. This is specified by Data Requirement **DR 4**.

The collected data can then be processed to give recommendations on new music tracks and the removal of previously loaded music, connect to an Internet community to compare performances and get recommendations on tracks that are not currently in the user's music library. This feedback based on the data will not be included in the initial version of the project. However, the data that is collected during exercising, will be processed and used for learning purposes in the system (Functional Requirement **FR 8**).

Requirements After Exercising

FR 8. *The collected data should be used to personalise the system's music selection*

DR 4. *During operation, the system should collect data on:*

- *songs played;*
- *music characteristics adapted by the system;*
- *sensory data;*
- *user interaction.*

1.4 Contributions

In this thesis, an overview of existing literature on techniques related to music and sports is presented (Chapter 2). Creating an intelligent, personalised system for playing music that fits with physical activities crosses borders between sciences. Artificial intelligence (AI) techniques are used for music selection, and techniques from signal processing for music adaptation. We also borrow ideas from (sports)psychology on using music as a motivator, from acoustics on spatial perception, and from movement sciences on music and motion. Conclusions are drawn from this literature overview on the applicability of the presented techniques and knowledge on the current project, and to create a technique to motivate exercisers with music.

According to the devised method in the preceding chapter, we have designed a three-stage system to optimally prepare the music available when exercising, adapting this music

during exercise, and personalising the system after exercise utilising performance data. The design of the system is outlined in Chapter 3.

The author is aware of one system that partly fulfills the research goals in terms of utilising music in exercise, that is, synchronisation of music with step frequency. This system is called StepMan (Fraunhofer, 2004). It currently does not provide personalisation features nor influencing abilities.

Each of the three stages is detailed in an own chapter (Chapters 4 - 6), starting with the selection of music in the preparation stage, followed by the adaptation of music in the exercise stage, and the personalisation of music selection in the feedback stage. Large parts of the described system have been implemented for as far as system evaluation requires. For the remainder of the system, detailed instructions for implementation have been provided.

After the description of the system in the preceding chapters, Chapter 7 contains the evaluation of the various parts of the system, including a small user evaluation of the on-line part of the system. A number of experiments have been devised to find the best values for important parameter settings such as system response time and music adaptation propagation time.

In the final chapter, Chapter 8, we return to the research goal and determine whether the described system fulfills the set goal, and the requirements specified above. We will also present possible enhancements for the proposed design, suggest areas for future research, and indicate opportunities for the commercial outroll of (parts of) the IM4Sports system.

Parts of this thesis, most notably Chapter 3, as well as parts of Chapters 4 (on off-line music selection), 5 (on the behaviour of the system during exercise) and 7 (the on-line evaluation), have been published in *IEEE Pervasive Computing* (see Wijnalda, Pauws, Vignoli & Stuckenschmidt, 2005).

Chapter 2

Motivation and Music

2.1 Motivation in Sports

In the previous chapter, we have seen that a large part of the population take up a repetitive solo sport, such as running, to lose weight and become healthier. However, many of these people suffer from bad motivation and are unable to keep up with their exercising plan. They find it hard to stick to their training schedule because after a training session no direct results are visible.

According to Tenenbaum *et al.* (1999), exercisers employ various strategies to cope with physical demands during exercise. Under low to moderate physical load, that is, *sub-maximal effort*, exercisers can voluntarily divert their attention away from internal sensations (*dissociation*). Sub-maximal effort is defined as a workload below 70% of maximum heart rate. Under higher loads, dissociation becomes increasingly harder and exercisers are forced to switch their attention to feelings of physical discomfort. When an exercise level is reached where feelings of exhaustion prevail, dissociation becomes nearly impossible and exercisers have to directly fight against the pain to continue the exercise.

Occupying the mind with thoughts of things other than physical discomfort enables the exerciser to endure the training longer than would be possible without distraction. In the next section, we will show that music, among other things, can be such a means of distraction.

2.2 Music as a Motivator

The lack of instant rewards is one of the main causes for the motivational problems. By making the exercise more enjoyable, a large increase in motivation can be gained. The use of music appeals to physical exercising: listening to music before and during exercise increases output by dissociation and motivation, and improves affective states. Even at intensities above sub-maximal effort, at both medium and high levels of work intensity, exercisers experience a greater enjoyment when music is played during exercising (Tenenbaum *et al.*, 2004). The hypothesis formed on the cognitive dissociation strategy by Rejeski (1985), is that external auditory stimuli narrow the exerciser's attention and as such pull attention away from feelings of exhaustion and pain.

Music can also be a motivator to sustain effort. The importance of the influence of music on physical exercising is mentioned by Karageorghis & Terry (1997), who found that the playing of music during exercise reduces the rate of perceived exhaustion (RPE). And, when less exhaustion is perceived, it becomes easier for the exerciser to continue the training session. However, as Rejeski (1985) already hypothesised, external auditory stimuli

do not influence the perceived exhaustion at high levels of work intensity, that is, above sub-maximal effort. Boutcher & Trenske (1990) mention that the perception of exertion was significantly lower when music had been played compared to a non-music condition, while heart rate was approximately equal under both conditions. Like Tenenbaum *et al.*, Boutcher & Trenske conclude that the influence of music is dependent on work load. Under low intensity levels, music has a positive influence on perceived exertion. Under moderate to heavy intensity levels, they conclude that although music has no effect on the RPE, it still enhances affective states like under low work load, thus making exercise more agreeable.

Szabo, Small & Leigh (1999) add to this that switching to fast music "during progressive exercise results in the accomplishment of more work", while the exerciser's heart rate does not proportionally increase. The result is attributed to distraction effects, where an additional note is made that the effect of the change in music tempo are dependent on the amount of attention the new music demands (*attention capturing strength*). An additional, but for our research most important, result of Szabo *et al.*'s (1999) research is that music that is synchronised to an exerciser's movements greatly improves motor control, and as such enhances endurance, even more so than non-synchronised music.

Kodzhaspirov, Zaitsev & Kosarev (1986) explicitly mention that the synchronisation of musical tempo to the physical activity is important to obtain a maximum impact of music on the exercise. All of their subjects claimed that the use of music improved their mood, whereas 95.4% said they were looking forward to their training sessions with music.

In a rather philosophical view, Shove & Repp (1995) mention the relationship between biological motion on the one hand and music on the other hand. Different dimensions of music that evoke different feelings include *harmonic progression*, *melodies*, and *rhythm*. Anshel & Marisi (1978) investigate one of these particular aspects of music, namely its rhythmical structure and tempo, and the relationship between this aspect and human locomotion. Their research indicates, for a small group, that music synchronised to physical movement has a particular positive effect on the length that participants where able to endure their tasks. In later research, Mertesdorf (1994) investigates the differences between using music or a computer generated beat, and found no significant differences between the ratings of perceived discomfort for these two forms of musical accompaniment. Also, Mertesdorf's (1994) observation that people like music more than the computer beat had no influence on the feelings of discomfort. When describing the results of his experiments, in which participants were allowed to adapt the music tempo, he notes that while subjects found it pleasant to adjust the tempo of their exercising manually, at the same time they "appreciated the advantages of music with a steady rhythm as a pacemaker".

According to Eitan & Granot (2004), no symmetrical analogy exists between music and motion. For example, an increase in dynamics of the music (*crescendo*) corresponds to a feeling of enhanced speed, while a decrease in dynamics does not correspond to a feeling of slowing down. In their experiments, Eitan & Granot have found certain asymmetrical associations for certain in-song characteristics. The parameters that can be applied to the domain of running are summarised in Table 2.1.

An instrument to assess the effect of different types of music on individual exercisers was developed by Karageorghis, Terry & Lane (1999). Their study aims to predict the psychophysiological responses of music and, hence, to predict how motivating the particular music type will be for a particular group of exercisers.

These results show that music brings enjoyment and motivation for physical exercising, especially below sub-maximal effort. This is the intensity at which most recreational exercisers do their training sessions. An increase in enjoyment and motivation can potentially have an enormous health effect as it increases people's ability to maintain their exercise effort. To investigate the best ways to employ music in sports, we need to understand the basics of human physiology, most importantly heart rate, step frequency and ways to measure their respective values.

Musical parameter	*Direction*	*Motion parameter*	*Direction*
Dynamics	Crescendo	Location	Approaching
		Speed	Faster
		Energy	Higher
	Diminuendo	Location	Moving away
		Energy	Lower
Pitch	Ascent	Location	Moving away
		Speed	Faster
		Energy	Higher
	Descent	Energy	Lower
Tempo	Accelerating	Speed	Faster
	Decelerating	Speed	Slower
Articulation	Tenuto	Location	Approaching
	Staccato	Location	Moving away

Table 2.1: Motion features associated with music parameters
Adapted from Eitan & Granot (2004)

2.3 Human Physiology

In this section, we will describe the basics of human physiology on three important fields: the biomechanics of running, the cardiovascular system, and how the respective associated variables step frequency and heart rate can be measured.

2.3.1 Biomechanics of Running

We define human running behaviour in terms of *steps*. The moment that one of the exerciser's feet touches the ground is considered a step, and a *stride* comprises two steps. A notion attached to running is *stride length*, that is, the distance covered between two steps of one of the feet. The *step frequency* is the number of steps occurring per minute (spm).

We consider the stride length to be more important for covered distance, while the step frequency is more important for measuring exercise intensity. For example, women and men have, on average, the same step frequency while men have a (much) larger stride length, thus resulting in men covering distances faster than women, on average (Novacheck, 1998).

The main difference between different running exercises is the speed of movement, and the swing state. The swing state indicates whether or not both feet loose contact with the ground at the same time. We distinguish four classes of walking: ordinary walking, jogging, running, and sprinting. According to Novacheck (1998), walking speed averages at about 1.2 m/s, jogging at 2.5 m/s, running at 3.2 m/s and sprinting at 3.9 m/s (this range equals 4.3 - 14 km/h), with world-class sprinters reaching speeds up to 9.0 m/s (32.4 km/h).

2.3.2 Cardiovascular System

In the IM4Sports system, we need to work with an abstract representation of the effort that the exerciser makes. In the human cardiovascular system, a number of parameters are important:

1. Heart rate;
2. Stroke volume;
3. Cardiac output;

4. Blood flow;
5. Blood pressure;
6. Blood amount.

According to Wilmore & Costill (2004), the *heart rate* is the most informative of all cardiovascular parameters. Therefore, we will focus on heart rate in the remainder of this section. The heart rate is an indication for the endeavour that the heart must make to meet the increased demand that exercising takes on the body, that is, the human heart rate changes during exercise, in order to carry out the exerciser's demands with maximal efficiency.

The *resting heart rate* (RHR) is generally about 60 to 80 beats per minute. However, values as low as 28 (for well-trained athletes) and over 100 bpm (older, unconditioned individuals) have been reported. While exercising, the heart rate increases proportionally with the exercise intensity. Near the highest heart rate value (HR_{max}) the heart rate begins to level off.

Please note that with this information we partly fulfill Data Requirement **DR 3**, which addresses the need of information of the effects of musical changes on the user's performance.

This HR_{max} is the value that you would achieve when you get to the point of exhaustion in an all-out exercise. HR_{max} is a reliable value that only slowly changes over the course of years, from the age of 10 to 15 years.

An approximation of HR_{max} can be made by subtracting your age from 220. This is only an approximation, as the individual number may vary greatly, and is genetically dependent (Edwards, 1996). An example: a 40 year old individual would be estimated as having an HR_{max} of $220 - 40 = 180$ beats per minute. Empirically, we know that 95% of all 40 year old have an HR_{max} within 156 and 204 beats per minute. In 2001, a slightly better equation was posed by Tanaka, Monahan & Seals (Equation 2.1).

$$HR_{max} = 208 - 0.7 \times age \tag{2.1}$$

A truly reliable value for HR_{max} can only be determined from the RHR, and must be obtained when a person is fully rested. Measurements must be taken several times in order to be accurate. A high HR_{max} cannot be used as an indicator of a better individual athletic performance, nor does a low value signal lesser performance capabilities.

Wilmore & Costill (2004) describe the change of heart rate during exercises at submaximal effort: the heart rate increases rapidly, until it reaches a plateau (the *steady-state heart rate*) that matches the amount of effort that the specific rate of work demands. When the required effort changes, it takes 1 to 2 minutes for the heart rate to stabilise on a new plateau.

The (maximal heart rate is important for determining specific training zones, which are expressed in a percentage of HR_{max}, and each feature specific training characteristics. See, for example, the popular book *Heart Zone Training* by Edwards (1996), which is focused completely on the notions of different heart rate zones and their specific uses.

The different *heart rate zones* that are commonly distinguished are usually specified in pieces of 10% of HR_{max} for the values between 50% and 90%. One common distinction is:

- **Rest** the heart rate when in rest;
- **Light** (50-60%) when walking;
- **Easy or Fitness zone** (60-70%) when jogging;
- **Aerobic zone** (70-80%) when running;

- **Anaerobic zone** (80-90%) when exercising hard;
- **VO_2 max** (90%-HR_{max}) when going all out.

Typically, the easy or fitness zone is recommended for burning fat, as 85% of all burned calories in this zone are fat calories. In higher heart rate zones, this percentage of energy used decreases and training becomes less efficient (*i.e.*, burning as much fat with as little endeavour as possible). Similarly, the aerobic zone is used for endurance training and the anaerobic zone for performance training.

2.3.3 Measuring Heart Rate

Instruments for measuring heart rate have been developed for over 10 years by companies like Polar, Timex, Cardiosport, and, more recently, Nike and Philips. The technique is common in fitness studios nowadays, and is used to inform exercisers of the amount of effort they are putting in their training sessions.

People usually have to wear either an earlobe sensor or, preferably, a heart rate sensor belt. The latter is much more accurate as it directly measures the user's heart activity. A heart rate measuring system generally consists of a waist belt transmitter and a separate receiver in the form of a watch-like device. A number of companies produce receivers that can be attached to a personal computer. The receivers come in the form of a small box that can be easily placed near the exerciser in a pocket, or on the arm with an arm strap. Different brands of sensors and receivers can be used interchangeably as long as they do not use vendor-specific codings (this has not been the case with the different variants we used).

When used, the chest belt transmits a pulse for every heart beat, which is picked up by the receiver and transmitted to the IM4Sports software.

Figure 2.2: Waist belt heart rate transmitter

2.3.4 Measuring Step Frequency

At Philips, Goris & van Herk (2003) have developed a tri-axial activity monitor that is able to accurately measure the amount of energy used by an individual. It has been successfully used in a smart shoe, which calculates the step frequency, speed, and measures distance walked while wearing it. For this project, we use a simplified version developed by Van Herk, which only measures step frequency. The class of devices that measure step frequency are called *pedometers*. The pedometer, like the heart rate sensor, was also connected to the computer running the IM4Sports software.

2.4 Music Selection

The proposed system has to fit music to a training program, which implies that a selection from an available music collection must be made that fits the exercise. If we want to synchronise music to the user's movements, we need to select a musical piece that comes close enough with respect to tempo, and then adapt it further. In the next section, we will go into detail on music adaptation, but first we will outline music selection.

Most research on music selection is done in the field of *playlist generation*. At the most trivial level, playlist generation is the process of selecting songs from a large music database that fit certain characteristics, or, *constraints*, which limit the eventual selection. When limited to song constraints, constraints that only deal with the characteristics of single songs (*e.g.* a song should be in the genre of 'rock') and not with the playlist as a whole, arriving at a solution is straightforward, as even with large music databases (in the order of, for example, over 40,000 songs) the process of going through the entire database and selecting by several constraints is solvable in linear time. Several approaches have been suggested, such as constraint satisfaction with backtracking, and integer linear programming.

Constraint satisfaction starts with an empty playlist trying to fill positions in a certain order and as such enlarging the playlist with every step until it is of the desired size. It works with one (partial) solution at a time that it expands, yet keeps a history of already investigated partial solutions.

Integer linear programming, on the other hand, models the problem as a network search problem, selecting the most cost-efficient path between the first and the last position of a problem solution. Because of limitations on the expressiveness, it is not suitable for all attribute relations (*i.e.*, constraints).

When several of these constraints are imposed on the selection process, the risk of conflicting constraints increases and as a result, the complexity of the problem increases, too. In addition, playlist-wide constraints (*e.g.* '50% of the songs should be by Luka Bloom' or 'each song should have a tempo that is higher than that of the previous song') require additional reasoning and processing time that, in the worst case, increases exponentially for every additional place (n^m tries before all n songs are fitted on all m places). When constraints conflict (*e.g.* 'all songs should be by Madonna', combined with 'all songs should be of genre Classical'), no feasible exact solution exists, and it becomes important to approximate a solution as closely as possible. Approximating solutions include local search and evolutionary algorithms.

Local search is an approximation method that starts with an initial random playlist, constantly refining it based on a cost function that can determine how good or bad a solution is. In many of its dominant implementations, it utilises a 'Best Improvement' method. It climbs from solution to solution by determining the value of all neighbours (solutions that can be reached with one change) and picking the best.

Local search is a member of the class of evolutionary programming that works with one solution at a time, whereas *evolutionary algorithms*, another member of that class, take into account a population of solutions and work with all those solutions at the same time.

We will overview each of these methods in the following sections.

2.4.1 Constraint Satisfaction with Backtracking

Constraint satisfaction, a member of the class of constructive search methods, is known to be a non-deterministic polynomial-time hard (*NP-hard*) problem. This means that it is at least NP-complete or harder, which means that currently all known algorithms require time that is superpolynomial on the input size. Therefore, ways to reduce the problem must be found. A comprehensive overview of algorithms for constraint satisfaction has been

written by Kumar (1992) and Tsang (1993).

One way to solve the problem is by employing *backtracking* techniques. The main principle behind this approach is to fit the first feasible song in the first empty playlist position, thus enlarging the playlist in every step. The process is repeated for each concurrent position until either the playlist is complete and fulfills all constraints, or it turns out to be impossible to fill a particular position without breaking any constraints. In that case, the previous position is reviewed and changed, that is, the song is removed from the playlist and the algorithm searches for an alternative. If all assignments for this position fail, the search process focuses on the then-previous position. Then, the next position is again reviewed, and so on, until the playlist is complete or all different combinations have been tried.

Backtracking guarantees that a solution, if one exists, will be found, however, at considerable cost. Simple backtracking does not learn from the exclusion of nodes (*i.e.*, songs). More efficient alternatives have been proposed, all based on more advanced *heuristics*, including first filling positions that have the smallest number of alternative songs and removing positions that limit the possibilities for other positions the most. These efficient techniques include dependency-directed backtracking (Stallman & Sussman, 1977), and reordering the way variables are instantiated (Bitner & Reingold, 1975).

Constraint Propagation

An important notion in constraint satisfaction is *constraint propagation*. The search space is reduced by removing from it the songs that violate constraints. In his review of constraint propagation and search-efficient heuristics, Pauws (2002) mentions the two following well-known notions as the most important that are practical for large song databases and/or large playlists:

- *Node consistency:* for each playlist position, constraints that work on that particular position must not be violated, hence, remove all songs from the search space that do not hold for this position. For example, if the first position of the playlist has to be a 'rock' song, remove all non-'rock' songs from the search space for the first position to make the problem node-consistent for this position.
- *Arc consistency:* if there is a constraint that works on two playlist positions (a *binary* constraint), remove each song from the search space of the first position that, when chosen, does not leave options for the second position. Also, the other way around, remove all songs from the search space of the second position that cannot be reached by choosing any song on position one.

Search-efficient Heuristics

Chronological backtracking is the basic backtracking technique as described above. When a constraint is violated, the previous position is reassigned. The following techniques for more efficient backtracking can be distinguished:

- *Order heuristics* rearrange the order in which the positions of the playlist are considered (using, for instance, the *fail-first principle*, which looks for the playlist position to which the least number of possible songs can be assigned and considers this position first).
- *Look-ahead schemes*, like forward checking, assign songs one by one but look if there are still remaining candidates for the other open positions.
- *Look-back schemes* do not jump back to the previous position when a dead-end has been found, but rather *backjumps* to the position that causes the dead-end. To do so,

it reviews all positions that violate constraints with the current position and takes the most recently instantiated of that to backtrack to. Variations on backjumping include conflict-directed backjumping, *backchecking*, and *backmarking*.

2.4.2 Integer Linear Programming

A suggestion done by Alghoniemy & Tewfik (2000a, 2000b, 2001) is to use an alternative approach, based on integer linear programming. However, they report less-than-optimal performance for playlists in the order of 10-15 tracks, especially given the fact that their catalogue is of limited size (400 tracks, generating a playlist takes about 35 seconds on a SUN Ultra 5).

Their network flow approach uses a network in which each node represents a song. Every arc sports an associated cost, representing the constraints that have to be satisfied. Their algorithm searches for a path through the network with a minimum cost.

The *branch and bound* algorithm (see Papadimitriou & Steiglitz, 1998) that they use is not really efficient (in the worst case, it's exponential), and the way in which they represent song attributes as vectors of binary values poses strict limitations on the kind of attributes and constraints that can be represented in the system.

2.4.3 Local Search

The approach of Aucouturier & Pachet (2002), taking into account the two previous methods, is based on an adaptation of local search techniques called *adaptive search* (see Codognet & Diaz, 2001). An overview of local search is given by Aarts & Lenstra (2003).

An implementation of local search is also used by Vossen (2005), who uses as the basis for his Automatic Playlist Generation algorithm (APG). For his decision to use a variant of local search, he has taken into account the performance of the other classes of algorithms we described earlier. However, equally important was the characteristic of local search, that it is able to quickly find a 'good' playlist and then refines it to a better one – and this process can be stopped at any point in time for an interim result. This makes it a particular good candidate for playlist generation, as the optimisation process ensures that while not all constraints may be satisfied, the algorithm ensures that the system gets as close as possible.

Local search maps the problem in a 'landscape' or *problem space* in which the lowest points represent the best solutions (playlists). The problem space can be seen as the aggregation of all cost functions $f(x)$ determined for all constraints. The process of finding a solution, then, is minimising the cost function by finding the lowest point in the problem space. A local search algorithm takes a starting solution x, and then searches all neighbouring solutions (computed using a neighbouring function $N(x)$, which makes small changes in the playlist x by inserting, deleting, replacing songs or by swapping two songs that are already in the playlist) for a solution y that performs better than x. When the starting point is on a 'hill' (this implicates bad solutions), the local search algorithm ensures the direction towards a 'valley' (which implicates good solutions), usually the nearest one. However, when the bottom of a valley has been found, the algorithm can no longer find lower points in its neighbourhood and thus gets stuck in a *local minimum*. There are several ways to escape from local minima.

Tabu Search

One of these, by Glover (1986, 1989), is by keeping track of places that an algorithms has already visited in a *tabu list* and assigning a temporarily low fitness to these solutions, making sure the algorithm does not visit them again within a specified time interval. This means that eventually the algorithm will be forced to 'climb up' from valleys to find a new

local minimum (which might turn out to be the global minimum x^*, eventually).

Evolutionary Computing

Another way is by introducing a randomisation function, a technique that is frequently used in *evolutionary computing* (for the foundations of evolutionary computing, see Holland, 1975). In an evolutionary algorithm a randomisation function is used to mutate a member of the population, whereas usually recombination is used to create new members.

In the most general form of an evolutionary algorithm, we repeat a number of steps while neither the maximum possible or required fitness nor the maximum number of populations have been reached. First, a new population is generated from the current population. Then, the fitness of this population is calculated, and parts of the current population are replaced with the new population.

The generation of the new population is done by selecting random individuals from the current population who will become parents, and selecting the best individuals; these will be members of the next generation as well. A number of new individuals are migrated to the new population, too, and new children are made by applying crossover to the parent selection, and mutating the results.

The difference between local search and evolutionary algorithms lies in this recombination feature. The probability distribution function governing the generation of new points for the set of possible solutions for blind random search is globally uniform, and for stochastic algorithms such as simulated annealing local search (see below) it is locally uniform. For evolutionary algorithms, interactions between points in the population are established by the recombination of parts of selected parents in the population. These interactions between points is reflected in the resulting probability distribution function, which hence is non-uniform (Eiben & Smith, 2003).

Simulated Annealing

In analogy to the physical annealing of a solid when it is heated to a temperature high enough, *simulated annealing* (Kirkpatrick, Gelatt & Vecchi, 1983) is a variant of the local search algorithmic model which is influenced by a control variable t, which models temperature. The probability that a neighbour y of the current solution x is accepted is defined as 1 when it is better than the current solution, or by $\exp\left(\frac{f(x)-f(y)}{t}\right)$ otherwise. The control variable t is decreased on every algorithm step.

The control variable is set to a high initial value, which makes the algorithm behave like random search. When the control variable decreases over the course of the search, the chance of accepting worse solutions becomes smaller as well.

So, in addition to the specification of possible solutions, a neighbourhood function and a cost function, like any local search algorithm, simulated annealing requires the specification of a so-called *cooling schedule* which determines the behaviour of the temperature t in the algorithm. A number of schedules have been proposed in literature, including some theoretical schedules that guarantee an optimal solution but requiring an infinite computing time. Some alternative schedules have been developed, among which the *geometric cooling schedule*, which is much simpler yet much more cost-efficient.

Voting Principle

An increase in efficiency is presented by Vossen (2005), who names it the *voting principle*. The voting principle helps satisfy constraints that have very few songs in the music collection that change the cost. Because of that, it is hard to find these songs in the normal randomised neighbour selection of the SA algorithm. The voting principle states that for those constraints (in particular those that require the determination of a cardinal value over

the entire playlist, or an integer or a fractional count of attributes in the playlist) it is more efficient to introduce a bias into the neighbouring function. Voting is only used in 1 of every 10 steps in the APG implementation, because most constraints can be satisfied with normal randomised simulated annealing.

The neighbour selection procedure roughly works in three steps. In the first step, one of the moves in $\{replace, insert, delete\}$ is chosen, which respectively replace a song that is already in the playlist, insert a new song on an open position, or delete a song from the playlist. In the second step, all constraints mentioned above get a vote on all possible playlist positions for the selected move. When a position is likely to decrease the penalty for a particular constraint, it will cast a positive vote, and a negative vote otherwise. When a song is considered that does not influence the penalty of one of the constraints that are allowed to vote, the constraint does not vote.

The third step is the selection of a song, in the case of a replace or insert move. For song voting, not all songs in the music collection are regarded, which increases the procedure's efficiency. After a move with a song has been selected, the voting procedure is completed. Then, the simulated annealing algorithm determines whether this move is accepted or rejected, dependent on the resulting change in penalty of the move, and the temperature t.

2.5 Music Adaptation

As we have seen in the first sections of this chapter, music works best as a motivator when it is synchronised to the exerciser's movements. Now that a way to select music is determined, we will outline three ways in which to adapt the music. These are controlling the dynamics, time scaling and spatial positioning.

2.5.1 Dynamics Control

One of the most straightforward ways to influence music is to simply 'turn the volume knob', that is, make the music sound louder or softer. As is shown in Table 2.1, an increase in volume corresponds with the sensation of higher speed. The increased speed perception will have a motivational effect on the exerciser.

Volume control has been a standard feature of nearly every music player implementation on a personal computer.

2.5.2 Time Scaling

Time stretching means playing back the music either slower or faster (*i.e.*, changing the audio stream duration) without affecting its pitch. This latter part is important, since when pitch is shifted, the music starts sounding strange.

The problem with most early algorithms is that they expand or contract the wave forms of the sounds and thereby transpose the pitch of the music as well, in much the same way the record player at the wrong rpm setting would. The analogy is to play a 33 rpm vinyl record at 45 rpm - the tempo of the music will be faster, but the music will sound higher as well, giving a 'Mickey Mouse effect' to the singer's voice. However, a number of algorithms have been proposed that can do time scaling while not altering the pitch.

One of these is the Synchronous Overlap and Add (SOLA) algorithm by Roucos & Wilgus (1985) and Makhoul & El-Jaroudi (1986). Its workings are described in detail by Zölzer (2002). The signal is divided in overlapping blocks. In the overlaps of these blocks, the algorithms searches for similar parts. These similar parts are then chosen as the fading point between the two blocks, and the rest of the blocks is discarded. Figure 2.3 (an adaptation of Figure 7.7 in Zölzer) illustrates this.

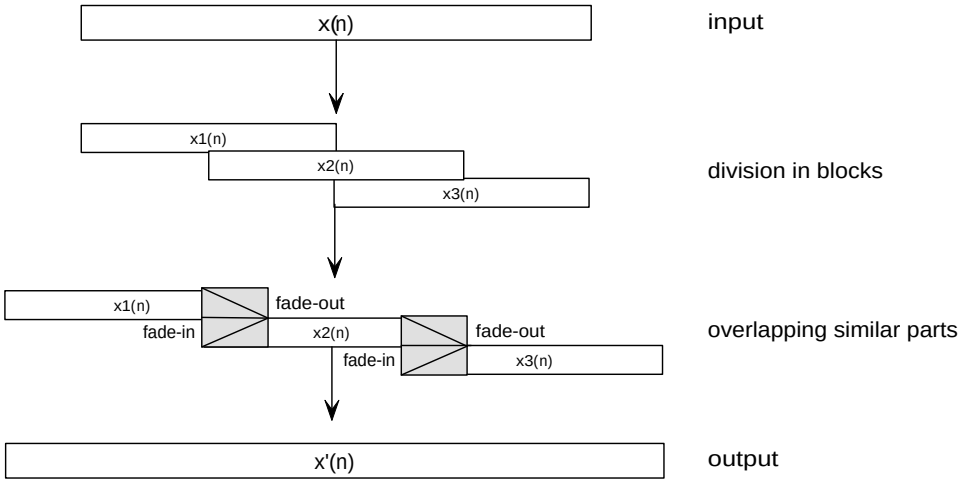


Figure 2.3: Time stretching using the SOLA algorithm

The PSOLA, or Pitch Synchronous Overlap and Add Algorithm (Moulines & Charpentier, 1990), is a variation on the SOLA algorithm focused on voice processing. Its hypothesis is that the sound is characterised by a pitch and hence the knowledge about the pitch can be used to synchronise the time segments to avoid pitch discontinuities. In fact, the algorithm places pitch marks at points in time where the amplitude is maximal or at glottal pulses, and the blocks that are thus created are used to form the new waveform. When stretching, additional pitch marks are added in the same pace as the original ones. For a more detailed description, see again Zölzer (2002).

A popular open source implementation of a variant of SOLA, WSOLA or Waveform Similarity Overlap and Add Algorithm (Verhelst & Roelands, 1993), is made by Parviainen (2002) in his SoundTouch audio processing library. As opposed to SOLA, which uses output similarity as its main synchronising method, WSOLA employs input similarity. The WSOLA algorithm uses a detection mechanism for similar waveforms in order to find better 'cut and paste' points than PSOLA is able to.

All *x*SOLA variants use the same approach, yet differ in the points selected to find overlaps and division points. The removal or duplication of parts of the waveforms comes at a cost, which limits time stretching usability above certain stretching or contraction levels, roughly 10-20% depending on the music type and its application. As an example of the effects of time stretching above this levels, imagine a very short sound, for example the hit on a hi-hat by a drummer. If this sound is stretched over a certain level, the length of the sound becomes longer than naturally possible, which might either sound awkwardly unnatural or plainly impossible, depending on the listener. When the music in which the hi-hat is supposed to sound is contracted too much, it is possible that the entire sound gets lost in the process. This would introduce strange gaps in the drummer's rhythm.

2.5.3 Spatial Positioning

The introduction of the Digital Versatile Disc, or DVD, meant a 5.1 sound configuration is available in many homes, in which five full-range channels (left, center, right, left and right surround, or 'rear') along with one low-frequency effects channel (LFE) could be coded.

This last channel uses only a tenth of the bandwidth of the other channels, hence it is referred to as a '.1' channel. A lot of 3D audio technologies have emerged in recent years that have tried to bring the same surround sound experience from two speakers or headphones. This technique is called acoustic modelling. In Gardner (1999), the underlying principle for most of these techniques is described.

The question is how humans perceive and, more importantly, localise sounds using just two ears. The answer lies in the difference between the sound levels and the moment in time the sound reaches the different ears. This difference, the *head-related transfer function* (HRTF's) can be measured using dummy heads that have microphones in the place of their ears, and computing the difference between the input of the two microphones. The HRTF's for the specified location are then derived using a computer.

A 3D audio system then uses the measured HRTF's to reproduce the sound localisation cues and the listener perceives the sound at the location specified. This process is called binaural synthesis (Gardner, 1999). According to Wightman & Kistler (1989), this process works extremely well when the user's own HRTFs are used to process the localisation cues. However, since measuring the HRTFs is a complicated procedure, most 3D audio systems use a single set to accommodate all users. There is a number of associated problems, as every listener has a different sized and shaped head (see Wenzel, Arruda, Kistler & Wightman, 1993). However, when using headphones the negative effects are reduced, since it can be guaranteed that the sound intended for one ear only reaches this ear and not the other.

A number of techniques is based on this principle, which employ a decoding technique to generate a virtual surround environment, based on a 5.1-encoded source and the technique of noise cancellation (Schobben & Aarts, 2002). The idea behind this technique is that each virtual speaker has its own characteristics and location, and so the sound it produces will reach the ears earlier or later and with its own sound characteristics. The image of these virtual speakers is then summed to two channels, left and right, and so the option to create virtual surround on an ordinary stereo headphone has become a reality.

Linking back to the current project, this means it is possible, although not perfectly, to lead a listener wearing headphones into believing the sound is coming from beyond the headphones. Also, the exact location from which the sound is perceived can be determined. This can be useful in developing motivation techniques.

2.6 Conclusions

We will outline our decisions and conclusions drawn from the conducted literature research on the specific areas of music selection, music adaptation and the indirect influencing of heart rate.

2.6.1 Music Selection

In the development of his Automatic Playlist Generation (APG) algorithm, Vossen (2005) has compared a fairly large number of different ways to implement local search techniques. His APG implementation supports a number of different parameters to configure the algorithm to choose between a number of algorithms and settings. During his tests, he arrived at optimal settings for the problem of playlist generation. These settings favour the use of simulated annealing and the voting principle.

As we will see later on, the IM4Sports project requires only a limited number of constraints for the selection of playlists (or actually play *sets* as we will also see). Therefore, the exact constraint satisfaction approach taken does not have a crucial impact on the project. The APG implementation suffices all requirements the project poses on the constraint satisfaction mechanism, and hence will serve as the method of choice for this project.

In his thesis, Vossen attributes typical performance of his implementation: even for problems with quite some conflicting constraints, an optimal solution approximation is generally found within 2 seconds, for a playlist length of 10 songs and a music collection of 2,500 songs on a PC platform. The problem for song selection in the IM4Sports project is even easier as no ordering constraints are used. For problems with fewer constraints, typical calculation time is less than 300 ms. Thus, the APG implementation does not pose a significant constraint on the system's adaptation speed.

2.6.2 Music Adaptation for Exercise Motivation

In Section 2.2 we have seen that music can be an instant motivator, especially when aligned with exercise tempo. This exact idea forms the foundation of the IM4Sports system. In order to align music, tempo adaptation techniques need to be used. The latest version of SOLA algorithms, WSOLA, provides an efficient technique with better performance and better quality than its predecessors, for which, as we have seen, an open source implementation is already available. Hence, the SoundTouch implementation of WSOLA will be used for song stretching.

2.6.3 Influencing Heart Rate

As the system will be provided with a training program that specifies a particular heart rate zone as a goal for an exercise, we need to measure the current heart rate of the exerciser. Several sensors exist that are able to do so. We also need to influence the users heart rate to reach the level specified in the training program. No safe direct way to manipulate the heart rate exists, which implies we need to influence the heart rate using an indirect method.

One of the important aspects of the system is that it tempts users to modify their exercising effort to fit with a specified training goal (*e.g.*, a certain heart rate level). We have seen we have a way to measure the exercise effort in terms of step frequency and, more importantly, to influence the step frequency using music. We have also learned that a linear relationship between heart rate and exercise effort exists (Wilmore & Costill, 2004). This means we are able to influence the heart rate indirectly, via the step frequency, as long as we remain in sub-maximal effort levels (as we have seen, the heart rate begins to level off near the point of exhaustion). This is illustrated in Figure 2.4.

So, the influencing of heart rate can be done, for example, by increasing music tempo slowly, so the user is encouraged to increase running speed. We assume the discrepancy between the current running speed on the one hand, and the running speed that follows from the training program on the other, can be determined. The resulting percentage can act as an input for the music tempo change, but also for other motivational adaptations.

One could think of an exerciser whose music is at a comfortable level when she is running at the right speed, while the music volume decreases when her speed is too low. Perhaps an even better implementation is to combine this volume adaptation with a surround headphone and play the music at the center position when the running speed is right, but transfer the music to the back or the front when the exerciser runs too fast or too slow, respectively. One could image the music being a virtual training coach that 'runs with the exerciser' like a real coach would, and dictates the running speed - all the user has to do is follow.

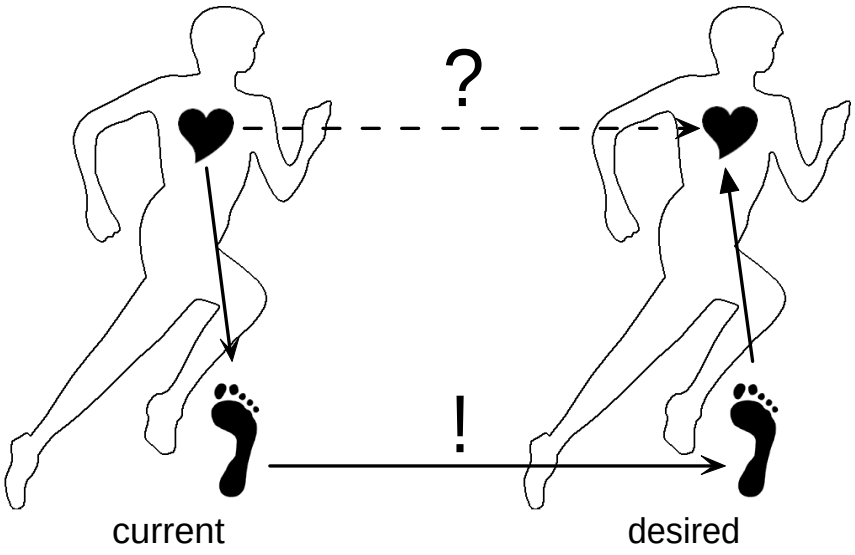


Figure 2.4: Heart rate control via step frequency

Chapter 3

System Design

3.1 Introduction

In this chapter, we will propose our design for the IM4Sports system. As a rationale for the design decisions, we will refer back to the requirements presented in the first chapter.

One of the requirements for the system is that it should suggest music content for during exercising (Functional Requirement **FR 4**). The user's digital music collection will generally be stored on a personal computer in the form of MP3 files. Since exercising will not take place in front of the computer, this means a selection has to be made before exercising.

The requirement that the system should learn from the exercise data (Functional Requirement **FR 8**), means that also after exercise some actions have to be taken.

The requirements imply three different system stages. The following can be distinguished:

1. The off-line stage *before* exercising (**preparation stage**);
2. The on-line dynamical stage *during* exercising (**exercising stage**);
3. The off-line stage *after* exercising (**feedback stage**).

In the preparation stage, the training program and the music selection on the portable device are determined. In the on-line exercising stage, the played back music is adapted to the users performance. In the feedback stage, performance data of the user is collected, presented on screen, and used to personalise the music selection in the next preparation stage.

3.2 Song Order Selection

For the selection of the actual song that should be played during exercising, two different methods can be considered: generating a playlist up front, in the *preparation* stage, or select songs while exercising, in the *exercising* stage.

A major advantage in generating the playlist before exercising lies in the fact that several (good) algorithms for off-line playlist generation already exist (see Chapter 2). Only new constraints would have to be introduced that deal with the specific criteria for song selection in the current project. Another advantage is the availability of greater processing power on the Base Station as opposed to the portable Player.

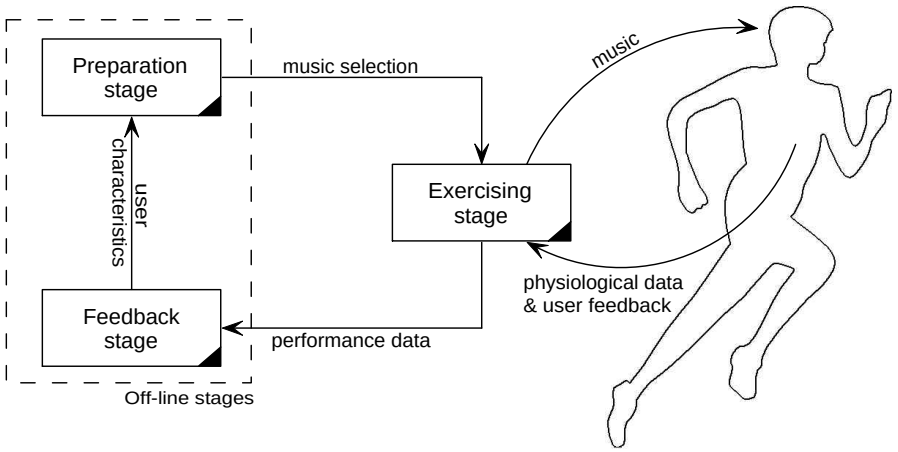


Figure 3.1: The three stages in the system and the data flow between them

However, the major problem with this setup is that not all desired music characteristics can be produced beforehand – the music that would be needed to guide the performance of the user *depends* on the actual performance, which cannot be known beforehand. Three arguments for this statement can be given:

The playlist has an arbitrary length. Take for example a runner who has selected a training program that includes running five kilometers. We don't know how long this will take her, since that depends on her physical condition, experience, current mood, environment, and motivation. Also, suppose she finishes the five kilometres when she's only 500 metres from home and wants to continue running until she gets there. Of course, we would not want the music to stop because the playlist has finished.

The user performance might require a different song tempo during exercising. The proposed method requires synchronisation to the user's step frequency (as opposed to, for example, the other way around). The user's step frequency might demand for a different kind of song than the one that was imagined during playlist generation, since a limitation of song stretching, which we'll come to later, is that not every song can be played at every tempo. A song can only be played a limited percentage slower or faster until the results get too horrid to listed to. Hence, when a much faster/slower tempo is needed than the current song has, a different song has to be selected - and this moment is not known in advance.

The user performance might require a different song characteristic during exercising. When the user is running, her performance might be so disappointing in terms of the predefined training program, that she will need a boost in morale. At this point in time, depending on the users musical preferences, for instance a hard rock song, a song by Marillion, or a song with loud rhythm might get the user back on the desired performance level.

All the problems mentioned above could be circumvented by creating much longer (to counteract the arbitraries in playlist length) and lots of different (to accommodate the need for different songs) playlists. Since the resulting system would allow for a great deal of flexibility, a dedicated algorithm would be needed to switch playlists while exercising. Such an algorithm could, in fact, pick individual songs as well as playlists, thus eliminating the need to generate playlists upfront. Selecting music during exercise would in fact bring even more flexibility to the system. This is why we decide to select songs during exercise instead of up front. However, even though not the entire playlist is generated before exercising,

we still need to select the right songs to take on the portable device, the Player, based on training program expectations.

3.3 System Stages

In the system setup that follows from this notion, the content of the three separate stages can then be defined as follows (Figure 3.1):

1. In the off-line (**preparation stage**) before exercising, the training program and the pre-selection of the music for the portable device are determined (details will be presented in Chapter 4);
2. In the on-line (**exercising stage**), music is selected, played and altered according to the performance of the user (Chapter 5);
3. In the off-line (**feedback stage**) after exercising, performance data of the user is collected, presented on screen, and stored for later reference. The analysed data can then be used as an input for the next off-line preparation stage (Chapter 6).

As described, both off-line stages take place on a personal computer, the Base Station, whereas the on-line stage is embodied by a portable device, the Player. The Player is connected to the Base Station in both off-line stages.

3.4 System Parts

The entire IM4Sports system consists of a number of important main components, which are depicted in Figure 3.2.

The off-line preparation stage is implemented by a single intelligent component, the SongSelect algorithm. As the name suggests, it selects songs from the main music database that will be included on the portable device. It takes into account the training program, and a tempo distribution per exercise. Tempo distributions are made per exercise, and represent the probability that music of a certain tempo will be needed in a particular exercise. At first, these will be based on statistical data from particular gender/age groups, but in time they are refined to reflect a more personalised view on the user's exercise behaviour.

The second important algorithm, MusicMotion, implements the on-line stage. It gets the device content and training program as input, as well as the user's heart rate and step frequency. To determine the current exercise intensity, information about the user's maximum heart rate is retrieved from the user model. The MusicMotion algorithm uses these variables to influence the music the user hears, and tries to motivate and influence the exercising with its musical output. It consists of three components, which will be detailed in Chapter 5.

The third algorithm, TempoInference, learns from previous exercises performed by the current user. Like the MusicMotion algorithm, it utilises information from the used model to reflect the user's characteristics in its workings. Using the exercise data received after exercising from the MusicMotion algorithm, it adapts the appropriate tempo distributions to fit them better to the user's exercise behaviour, by modifying the constraints used for song selection in the next training session.

To be able to fully understand the details of all algorithms, it is important that their input and output from other components of the system are known. The training program, which is used in all the stages, is discussed in the next section. Thereafter, we will briefly describe the user model, before, in the remainder of this chapter, we explain some implementation facets of the processing of physiological input, the propagation of musical output and the communication between system components.

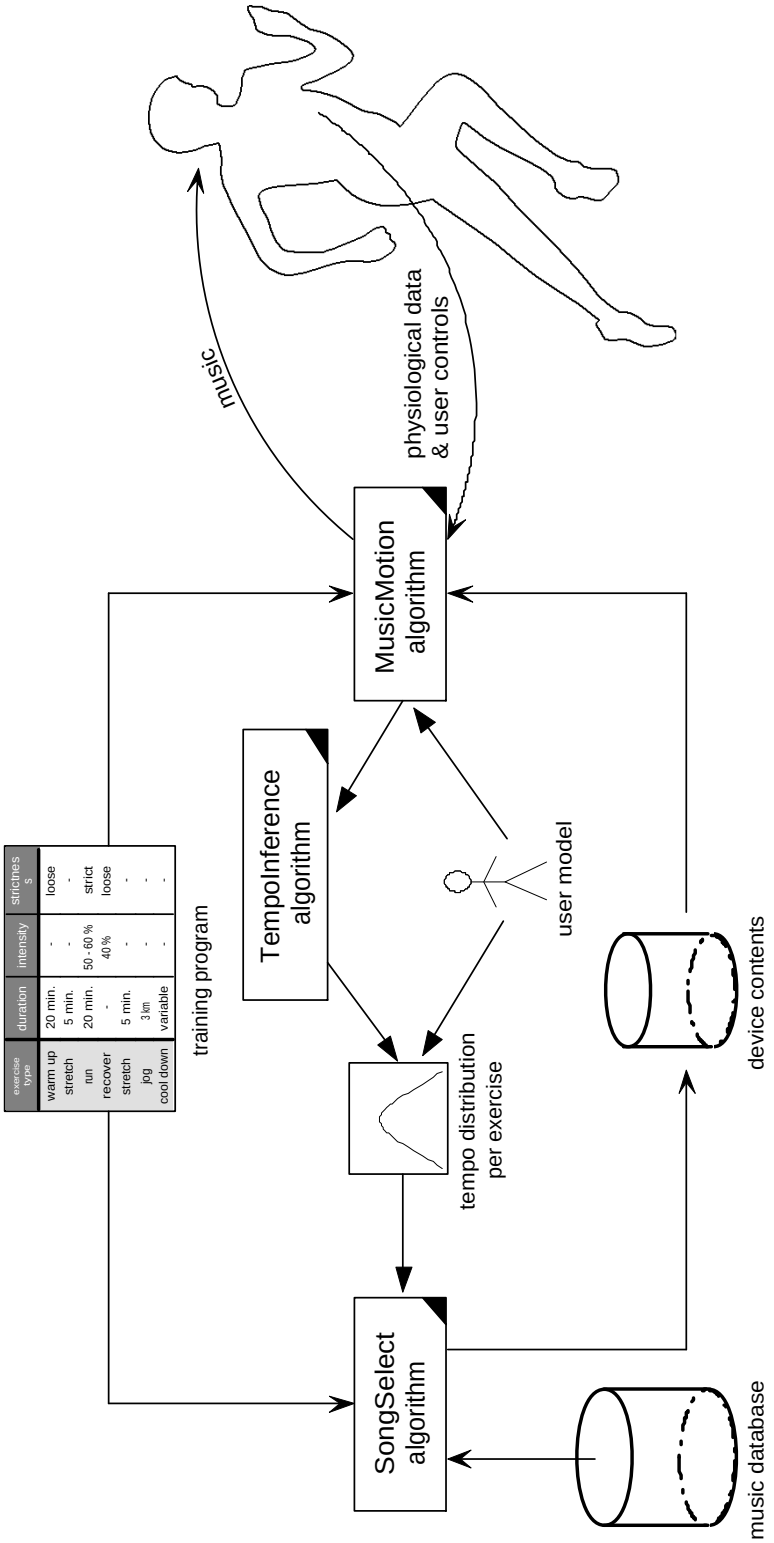


Figure 3.2: The different algorithms in the IM4Sports system

3.5 Training Programs

One of the key parts in the IM4Sports system is the training program, which is an input of the system according to Functional Requirement **FR 2**. It is defined by the user, and defines what behaviour the system should show during the different exerciser that the training program consists of. The following representative examples (Tables 3.3-3.5) were taken from the largest website on running, Runners World (n.d.). All share some common characteristics which we will use to determine the training program format for IM4Sports.

day	*exercise type*	*duration*	*intensity*
Sunday	run	20-30 min	50-60%
Monday	-		
Tuesday	jog	5 min	
	run	5 min	70%
	walk	5 min	
	run	5 min	70%
	walk	5 min	
	run	5 min	70%
	walk	5 min	
	cool down	5-10 min	
Wednesday	-		
Thursday	run	20-30 min	65%
Friday	-		
Saturday	run	40-50 min	60-65%

Table 3.3: Runners World: Polar beginner's program

3.5.1 Common Characteristics

Each training program specifies different exercises with different lengths. The exercise type differentiates between exercises that require a form of running (walking, jogging, running and sprinting), and other types mainly used for recovery or warming up.

The duration of an exercise is generally expressed using either the time of the exercise, or its distance in miles or meters. In some cases, however, the duration in time or distance is unknown and instead the time needed to reach another goal (for example, a specific heart rate zone or a given step frequency) forms the duration of the exercise.

In the IM4Sports system, an additional category is introduced which allows training programs to specify their duration or intensity in step frequency. This category is not generally used in training programs currently, because the step frequency (in steps per minute) is not easily monitored using conventional training devices (usually a heart rate meter watch). It is specified indirectly, though, in the specification of minutes per mile (MPM) when training for 5K, 10K or the (half-)marathon, which roughly indicates the speed of the training and hence, with a fairly constant stride length, a rough estimate of the step frequency.

When we view each day of the week in the tables as a separate training program, we quickly arrive at the definition we use in the IM4Sports system, which includes an exercise type column, a duration column which can be specified in distance, time or state 'heart rate zone', an intensity range in which the heart rate or step frequency zone is specified. In addition, an IM4Sports training program features a column which specifies what happens when the current exercise is finished: it could repeat, or the training program could automatically skip to the next exercise.

day	*exercise type*	*duration*	*intensity*
Monday	-		
Tuesday	run	2 miles	
	run	1:00	70-80 %
	jog		50-60 %
	run	1:00	70-80 %
	jog		50-60 %
	run	1:00	70-80 %
	jog		50-60 %
	run	1:00	70-80 %
	jog		50-60 %
Wednesday	-		
Thursday	run	4 miles	
	walk		50-60 %
	run	100 meters	
	walk		50-60 %
	run	100 meters	
	walk		50-60 %
	run	100 meters	
Friday	-		
Saturday	run	3-4 miles	
Sunday	run	6-7 miles	

Table 3.4: Runners World: Your ultimate half-marathon training program

day	*exercise type*	*duration*	*intensity*
Sunday	run	70 min	65%
Monday	-		
Tuesday	warm up	15 min	65%
	run	5 min	80%
	walk	3 min	
	run	5 min	80%
	walk	3 min	
	run	5 min	80%
	walk	3 min	
	cool down	5-10 min	
Wednesday	jog	30 min	60-65%
Thursday	run	20 min	70%, increase to 80%
Friday	run	45 min.	60-65%
Saturday	-		

Table 3.5: Runners World: Weekend Racer program

```
<trainingprogram>

    <exercise type="IM4S_PACEMATCHING">
        <duration type="time" val="5" />
        <afterExercise type="skipToNext" />
    </exercise>

    <exercise type="IM4S_MOTIVATION">
        <duration type="time" val="10" />
        <intensity type="pace" min="100" max="120" />
        <afterExercise type="skipToNext" />
    </exercise>

    <exercise type="run">
        <duration type="time" val="30" />
        <intensity type="range" min="40" max="80" />
        <afterExercise type="continue" />
    </exercise>

    <exercise type="walk">
        <duration type="distance" val="3000" />
        <intensity type="single" val="40" />
        <afterExercise type="skipToNext" />
    </exercise>

    <exercise type="jog">
        <duration type="heartrate" />
        <intensity type="range" min="50" max="100" />
        <afterExercise type="continue" />
    </exercise>

</trainingprogram>
```

Figure 3.6: A sample training program in IM4Sports XML format

3.5.2 Definition and Specification

A training program in the IM4Sports system is specified in a particular file in XML format. Its XML schema is given in Appendix C. A sample training program is shown in Figure 3.6.

A training program, like the one we see in the example Figure, is specified within the `<trainingprogram>` tag. A training program can contain an unlimited number of exercises (these are specified within `<exercise>` tags). Like in all example training programs included in this chapter, every exercise includes a non-unique `type` identifier which specifies the *exercise type*, such as 'run', 'walk' or 'jog'.

A number of special exercise types have been included to prompt the IM4Sports system to switch to a specific mode (these are 'IM4S_FREEFORM', for free form mode, 'IM4S_CRUISECONTROL', which puts the system in cruise control with fixed speed, 'IM4S_PACEMATCHING', for pace matching mode, and 'IM4S_MOTIVATION', to switch to motivation mode). They carry a `parameter` attribute which has a different meaning dependent on the exercise type. No parameter exists for the free form mode. For

cruise control mode, the parameter specifies a music speed (in beats per minute) which cannot be changed during exercise. The parameters for the other two special exercise types are dependent on the chosen implementation, which is why they will be detailed in Chapter 5.

Every exercise must carry a specification of its duration. The duration can be specified in a number of ways:

1. *Time*, in which a number of seconds can be assigned to an exercise. This would be used for running or jogging exercises with a fixed length in time.

2. *Distance*, which determines the duration in a number of meters that must be covered in this exercise.

3. *Heart rate*, which means that the exercise is considered to be finished when the value specified by the exercise intensity is reached.

A specification of exercise duration in 'heartrate' can be used, for example, for certain cool down exercises which require the heart rate to drop below, for example, 40% of (HR_{max}. The exercise duration is specified with the <`duration`> tag and its required `type` attribute. The `val` attribute specifies the actual length, in seconds when `type` is 'time', in meters when it is 'distance'.

The exercise intensity (specified in the XML file with the <`intensity`> tag) can either be empty, be specified as a fixed value (`val` attribute) or as a range (`min` and `max` attributes). If the intensity is not left empty, it must be specified as a step frequency value (when the intensity type is 'pace') or a percentage of HR_{max} (intensity type 'heartrate'). When exercising, the system will make an effort to motivate the user to stay within the specified heart rate zone or as close as possible to the specified step frequency.

For convenience, it is possible to specify whether, when an exercise's goals are reached, the system needs to automatically advance to the next exercise or continue the present exercise until the user indicates that the system should start the next exercise. For example, an exercise like 'run 5 kilometers' will likely not skip to the next exercise automatically, because it probably will not be finished after exactly that distance, whereas for a repetitive exercise series such as on Tuesday in Table 3.5 an automatic advancing behaviour would be preferred. In the XML training program file, the <`afterExercise`> tag specifies what happens after the exercise, which can be 'skipToNext' exercise or 'continue' this exercise.

Data Requirement **DR 1** is fulfilled by the definition of a training program above, as it specifies the kind of exercise, the duration and the intensity.

3.6 User Model

The user model is the location where some important characteristics of the user are stored. In the current implementation, only age and gender need to be stored, as these are the only variables used in the system. HR_{max} is calculated using these variables with Equation 2.1.

In future extensions, HR_{max} can be inserted when obtained after careful testing. All these values, which potentially influence the behaviour of the system, can be stored in the user model. For example, we could extend the system by storing variables like the user's fatigue curve, music taste, and stride length.

Also, to further personalise the interaction between the user and the system, the user's name and other personal information can be stored. As another example, the current location could have an influence on the behaviour of the system if it would recommend training programs – if the system knows the distance from the current location to the running track where the training session will take place, it could suggest a warming up covering this distance with a few stretching exercises in between.

3.7 Implementation

In the following section, we will describe the implementation of the physiological sensors for heart rate and step frequency, as well as the playback of music.

3.7.1 Physiological Input

Heart Rate

Heart rate is collected using a heart rate belt transmitter and a heart rate receiver, which is attached to a joystick interface. It comes in the form of an easily attached box with a mini jack connector, which we attached to one of the buttons of a joystick interface. The joystick interface, in its turn, was connected to a USB convertor and, utilising a 2 m USB cable, it was connected to the computer on which the system software was running. We chose this setup to minimise the amount of interference the receiver gets from testing setups featuring a treadmill. The magnetic radiation of a treadmill (or similar devices such as hometrainers) can possibly distort the signal picked up by the receiver.

The IM4Sports system utilises a C++ program which continuously reads all input from the USB port. The program, `hrsfm`[1], attaches a time stamp to all incoming heart rates and writes these to its standard output. Thus we assure that all heart rates get a machine time stamp for later comparison. The counterpart of `hrsfm` in the online stage Sensory Framework, PhysioState, processes the messages from `hrsfm` on its standard input. The heart rate messages are of the form `hr` <*time stamp*> <*heart rate*>. The time is the machine time stamp, which is defined as the number of milliseconds from January 1st, 1970, 0:00am GMT.

Some heart rates, however, are not communicated on standard output. The `hrsfm` program contains an internal filter which determines whether a heart rate is possible or not. These heart rates may show up because of interference between the sensor's sender and the receiver, or because the sensor moves around over the exerciser's chest, and occasionally interprets such a move as a heart rate. Heart rates higher than 250 are not possible for a human exerciser at sub-maximal effort. Hence, heart rates above this threshold value are discarded. In addition, `hrsfm` calculates the five-point moving average over the heart rate before it puts in on its output.

The PhysioState component, in its turn, parses the messages from `hrsfm` and sends the actual heart rate to the Scheduler component. These components will be outlined in Chapter 5. In addition, it logs the received values in a comma separated file (csv format) `HeartRate.csv`, in the format <*time stamp*>, <*heart rate*>.

Step Frequency

The step frequency is read through a pedometer as described in Section 2.3.4 communicated through a USB joystick interface, like the heart rate. The `hrsfm` program also reads the step frequency and puts the value on its standard output.

Similar to incoming heart rates, some preprocessing is done by the `hrsfm` program. It calculates a five-point moving average from the incoming step frequencies and puts this value on its output, and also keeps track of the step count (which is increased by one on every incoming step frequency). Also, values that would result in a step frequency above 250 steps per minute (spm) are discarded.

As an additional filter to check for spurious steps, the *kurtosis excess* or 'peakedness' of the sample data is calculated. The kurtosis excess is the fourth moment about the mean, with a correction term to ensure the kurtosis of a normal distribution equals zero, and is

[1] Heart Rate and Step Frequency Monitor.

defined by:

$$\text{Kurt}(x) = \frac{\mu^4}{\sigma^4} - 3 = \frac{\int_{-\infty}^{+\infty}(x-\mu)^4 f(x)dx}{\sigma^4} - 3$$

Kurtosis is a measure of whether the data are peaked or flat relative to a normal distribution. That is, data sets with high kurtosis tend to have a distinct peak near the mean, decline rather rapidly, and have heavy tails. As we expect the step frequency to be fairly equally distributed (that is, not with a large peak in it) over the tested interval, we can use the kurtosis as a measurement to eliminate incorrect input step frequencies.

For example, kurtosis values above 12 (which has proven to be an efficient threshold value in the experiments with the IM4Sports system) indicate that the corresponding value has an exceptional tangent, and thus is quite out of range with respect to the values around it.

All information is written to standard output in the form `sf` <*time stamp*> <*step frequency*> <*step count*> <*kurtosis*>. The output is processed by the PhysioState compontent, in which the kurtosis threshold is enforced.

All received values are written to a comma separated file `StepFrequency.csv`, in the format <*time stamp*>, <*step frequency*>, <*step count*>, <*kurtosis*>.

3.7.2 Music Output

Music output is implemented using the X Multimedia System (XMMS) application by Alm & Alm (2004), which is a multimedia player for UNIX systems. It was chosen as the music player for the IM4Sports system as it is open source and hence easily modifiable if needed, and a decent time stretching plugin is available for it.

The PlayerInterface component in the IM4Sports system receives messages from the Scheduler component regarding the music output of the system, such as 'play song *x*', or 'change the tempo to 148 in a period of 10 seconds'.

Berger (2002) has developed a plugin for XMMS, `sndstretch_xmms`, which allows time stretching of songs by specifying the appropriate stretching level in a percentage of the song's intrinsic tempo. The right value is calculated by the PlayerInterface, when it receives a command from the Scheduler. The stretching performance of the plugin is not very well, as it introduces a large number of audible artifacts into the music output when contracting or stretching to levels above 10%. Hence, Parviainen's (2002) WSOLA implementation was modified to substitute the original plugin's stretching algorithm. Its performance does not include artifacts for stretching levels up to 50% and contraction levels up to 25%.

Unfortunately, XMMS does not have a standard way to communicate to its plugins through commandline parameters. Hence, the PlayerInterface communicates with the `sndstretch_xmms` plugin using an external program, `xmmsspeedcontrol`, which writes to memory it shares with the plugin. It can be called to read the current stretch value or set it. Song changing is done by directly calling XMMS with the appropriate commandline parameter (XMMS can be configured to not spawn multiple instantiations of itself, so starting a new XMMS in effect changes the parameters on the old one and then exits the new instantiation).

In addition to forwarding message to XMMS and its plugin, respectively, the PlayerInterface also logs all commands it issues in a comma separated file `Player.csv` in the format <*time stamp*>, <*music tempo*>. In addition, a file `Songs.csv` is maintained in which on every line a <*time stamp*>, <*song title*> pair indicates song changes made by the PlayerInterface.

Chapter 4

Selecting Music

4.1 Song Selection Process

The selection of a music collection that is transferred to the Player in the first off-line stage is specified by Functional Requirement **FR 4** (the system should suggest a new music collection on the Player that best fits the characteristics of the specified training program). We assume that Data Requirement **DR 2** (meta data on music should be available) is fulfilled by the inclusion of ID3 tags in the user's MP3 collection.

The SongSelect algorithm uses a phased selection process. The on-line song selection is done by the MusicMotion algorithm in the exercise stage, which leaves the contents of the portable device, based on the available music and the training program, to be determined by the SongSelect algorithm. In this selection process the following steps can be discriminated (see Figure 4.1):

1. Use the user preferences to decrease the population of songs to a pool of songs that are liked by the user;
2. Determine what songs from that pool are most probable to fit within the playlist of a given exercise (along with alternatives with a different average speed or mood), and use these as a 'pre-selection pool';
3. Transfer the pre-selection pool to the portable device.

The selection of music according to user preferences is already accounted for in literature, as has been illustrated in Chapter 2. Therefore, in the current project we will focus on the off-line algorithm that selects the music based on a training program. User preferences can be added to the system by either simply removing all non-preferred music from the initial database to obtain a database of 'user-preferred music' (this is depicted in Figure 4.1), or by adding user preference to the off-line selection algorithm.

4.2 Music Selection for Training Programs

In Chapter 3, it was determined that in the preparation stage, songs are selected based on a number of constraints:

1. The training program;
2. The available music (music database);
3. The storage space that is available on the portable device.

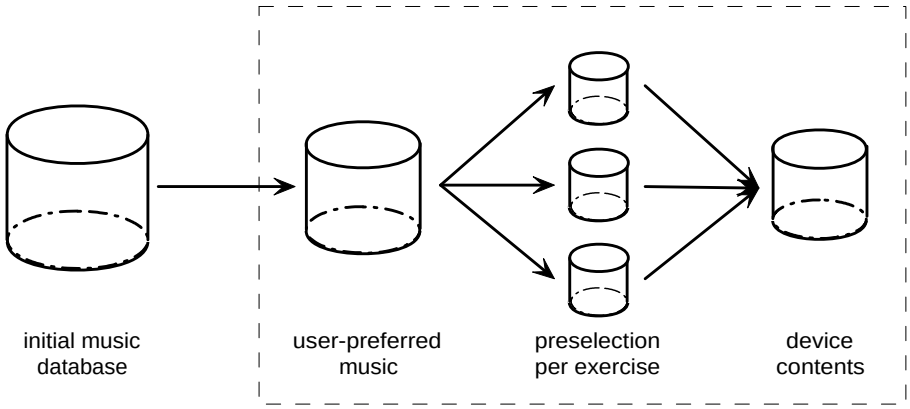


Figure 4.1: The process of selecting the device contents from the music database
The scope of the current project is marked with a dotted line.

The main problem posed in this system stage is how the system should select music based on a training program. We will research the defining characteristics music to fit a particular training program. In the previous chapters, we have already seen that many training programs include jogging, running and walking exercises. The two main differences between these particular exercise types are:

1. The step frequency at which the exercise takes place;
2. The intensity at which is exercised.

In addition, other exercises such as warming up, cooling down, stretching and recovery, can be found in training programs, in which the exercise step frequency does not play a role that is as prominent as in the previous examples. Arguably, although the music for these exercises should fit a certain profile (one might not like to hear fast music when cooling down), the exact tempo is less important, and could be seen as a preference that differs from user to user. Therefore, we will currently focus on exercises in which tempo is a prime concern.

In this chapter, we will outline the design of the preparation stage of the IM4Sports system. First, based on extensive experiments, gender/age group distributions for different types of exercises and heart rates will be determined as a starting point. During the use of the system, these distributions will be personalised towards the user's performance. When a distribution per exercise is obtained, the relative weight of the exercises in the training program is determined and from the weighted exercise distribution the training program distribution is calculated. Finally, the global distribution is mapped to the available device space in order to optimally fill the portable device with music.

4.2.1 Gender/Age Group Distribution

Selecting music based on a training program can be done by first converting the described heart rate (zone) to an average step frequency for the exercise. When a heart rate zone is given, we take the average heart rate for that exercise as the approximation of the entire zone.

Our starting point to make this conversion is an average tempo for every possible user group for all exercises. According to Wilmore & Costill (2004), age and gender are the

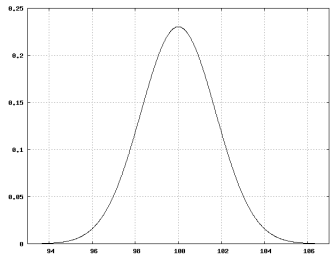


Figure 4.2: An example of a normal distribution with $\mu = 100$ and $\sigma^2 = 3$

two most important user characteristics that determine the level at which the heart has to work at specific exercise intensities. These characteristics are readily available, as opposed to characteristics such as health or fitness. A possible way to obtain this data is to let representative members of the gender/age groups perform all exercises while keeping their heart rates constant at a certain number, and averaging out the results.

The end result of this experiment will be a table which includes for every gender/age group and all heart rate zone (say, 50%-90%), and an average step frequency. According to the well-known law of large numbers, when all statistical data is collected and averaged out, a normal distribution will be found.

Say, we have an exercise which goal is to keep the exerciser at 50% of HR_{max}, and a male exerciser of 18 years old. Let us assume that for the 'male/under 20' group, 50% of HR_{max} equals a step frequency of 117 steps per minute with a variance of 3. This means that in order to get a good result for a user from this gender/age group, we need to draw songs from a normal distribution with a mean μ of 117 and a variance σ^2 of 3 $\left(i.e.\ N(117, 3)\right)$.

The idea of the proposed system is to use the gender/age group distribution as a starting point, which will be updated after each training session, when the user has actually performed the specific exercise belonging to that distribution. In this way, the system will become more tuned to the characteristics of its user with every learning step. Updating the distribution is done after exercising, in the feedback stage, and will be described in detail in Chapter 6.

Fatigue Curve

The amount of energy the user can put in the exercise decreases while exercising, when the user gets tired. This phenomenon of decreasing performance has an associated curve called the user's *fatigue curve* (see for example Gabriel, Basford & An, 2000). The fatigue curve shows how a user's performance decreases after exercising for a certain time. For example, a 3 minute run in the beginning of a training session will be ran at a different speed and step frequency than the same 3 minute run at the end of a heavy interval training. In the current implementation, we will not use the fatigue curve, but we mention it because it would be a useful extension when further developing the IM4Sports system.

4.2.2 Global Distribution

When a distribution for each of the exercises in the training program has been found using the method described above, all distributions can be summed to one global distribution. The relative weight of the exercises should be taken into account. This weight w is proportional to the length of the entire training program.

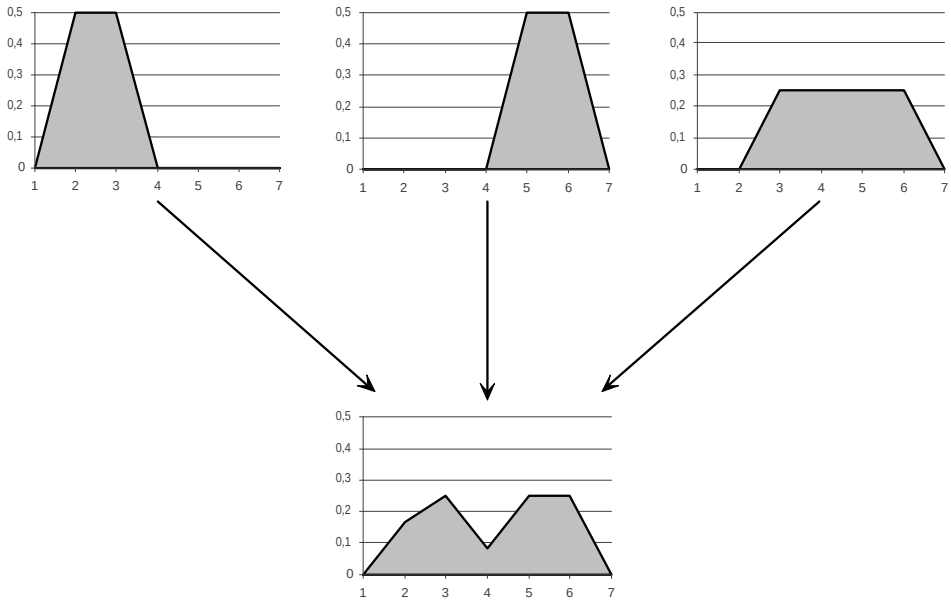


Figure 4.3: Summing three distributions to one global distribution

One of the problems that arise is that some exercise durations may be expressed in kilometres or target heart rate instead of time. For the distance duration exercises, we may multiply the distance with the average running speed, taking into account the user's fatigue curve. For target heart rate, the user's fatigue curve can be taken into account as well, however, this relationship is not that straightforward and has to be investigated further.

Summing distributions is straightforward:

$$G(x) = \frac{\sum_{i=1}^{n} w_i \cdot F_i(x)}{\sum_{i=1}^{n} w_i} \tag{4.1}$$

where n is the number of distributions. All individual distributions $F_i(x)$ are summed to get a global chart, which has to be divided by the summed weights of all distributions in order to be a valid distribution G itself[1].

The process of summing distributions is shown in Figure 4.3 for three non-normal probability distributions with equal weights.

4.3 Constraint Definition

When a global distribution has been established for the entire training program, this distribution can be mapped to total playing time available on the portable device. Available playing time is dependent on the amount of storage space on the device and the file size of

[1] This can be seen by temporarily ignoring the weights, and noting that the surface under the graph of a distribution must always be one: hence, when summing n distributions, the surface under the graph will be $n \cdot 1 = n$. Dividing by the number of distributions results in the surface under the resulting graph to be one again. When we use weighted distributions, dividing by the summation of the weights has the same effect.

attribute	*description*	*type*
song ID	a unique identifier	nominal
title	the title of the song	nominal
artist	the name of the performing artist	nominal
album	the name of the album	nominal
track	the track number on the album	numerical
duration	the song's duration (in seconds)	numerical
file size	the size of the song file (in MB)	numerical
genre	the genre to which the song belongs	categorical
year	the year the song was released	numerical
tempo	the tempo of the song (in beats per minute)	numerical
tempo marking	the description of the song's tempo	ordinal
meter	the meter of the song	ordinal
mood	the overall mood of the song	categorical
key	the tonality of the song	categorical

Table 4.4: Song attributes

individual songs. Finding songs that match the global distribution is a whole other problem for which, as we described in Chapter 2, local search algorithms are a suitable solution. The Automatic Playlist Generation algorithm (APG), when supplied with constraints that specify the demanded requirements, is such a solution.

APG uses formalised constraints that should hold for individual songs or groups of songs (such as all songs that belong to one specific exercise), or constraints that should hold for the entire music selection (*i.e.*, the entire training program). Constraints can reflect every aspect of a song as long as it is defined as a formal attribute of that song, such as the users preferred genres, or the song's tempo. A playlist is said to satisfy a constraint if it meets the restrictions that are defined by that constraint. Otherwise, it is said to violate the constraint.

The APG algorithm is designed to generate playlists, in which the order of songs is important and restrictions on the order can be given. In the IM4Sports system, the order of songs is not important. Therefore, no order-related constraints are instantiated in APG and the generated playlist is considered a play *set* in which songs can only occur once and which satisfies the given constraints, and of which the *order* is ignored.

We define a *song* as a finite vector $s = (a_1, \ldots, a_A)$ of A attributes. Every attribute defines one feature of the musical piece, such as its title, the performing artist, genre, duration, tempo and meter (for an overview, see Table 4.4). The user's *music collection* is then defined as a finite set of all m songs $M = s_1, \ldots, s_m$.

The pre-selection p of music that we want to obtain in this stage of the system is an unordered subset of M ($p \subseteq M$). The size of the pre-selection is not known beforehand, because an exercise may or may not be finished when the exercise requirements are completed. We use the notation $p_i[k]$ to denote the k-th attribute of a song p_i in the play set.

All Different Songs

To ensure that the generated playlist is a set, we must include a global constraint that states that no two songs in the playlist can be equal. Formally, this *all songs should be different* constraint could be defined by (p), where:

- p is a play set,

denoting that it has to hold that $\forall p_i, p_j \in p : i \neq j \rightarrow p_i \neq p_j$, that is, all elements in p must be different.

The Size of the Play Set

The size of the play set is limited by the available space on the portable device that is used for playing. If we assume that the k-th attribute of a song denotes its file size on disk, the *file size* constraint could formally be defined as a triple (p, k, max), where:

- p is a play set,
- k is the attribute number of file size, $1 \leq k \leq A$,
- max is the upper boundary for file size,

denoting that it has to hold that $0 \leq \sum_{p_i[k] \in p} \leq max$, and in which max obviously denotes the available storage space on the Player. We could define the lower boundary, currently 0, instead in terms of max, where we define a percentage in which the play set should fill the device, for example $min = 0.98 \cdot max$, thus denoting that it has to hold that $0.98 \cdot max \leq \sum_{p_i[k] \in p} \leq max$.

Musical Meter

The meter of the selected songs must be of a *duple* or two-fold meter, such as a march, so that either 2 or 4 beats go into each measure. This is needed because due to the two-legged nature of a human runner, duple meter songs are easier to synchronise to. For every step, the music plays for one to eight beats, but the feet always land at one specific beat. In a three-fold or *triple* meter, such as a waltz, the foot impact, which follows a two-fold pattern, shifts in every measure (Novacheck, 1998).

If we assume that the k-th attribute of a song denotes its meter, the *musical meter* constraint could formally be defined by the pair (p, k), where:

- p is a play set,
- k is the attribute number of song meter, $1 \leq k \leq A$,

denoting that it has to hold that $\forall p_i \in p : \text{twofold}(p_i[k])$, where $\text{twofold}(x)$ is a Boolean function that is able to determine whether or not song x has a two-fold meter.

Don't Include Skipped Songs

During exercising, the user can skip certain songs. We would like not to include those skipped songs in the next selection stage, so we ensure the system keeps a blacklist b of skipped songs. This blacklist has the same formal properties as a play set, so that we can formally define a pair (p, b), where

- p is a play set,
- b is a play set with skipped songs,

denoting that it has to hold that $\forall p_i \in p, b_i \in b : p_i \neq b_i$.

Songs Fit Tempo Distribution

The tempo of the songs in the play set should be distributed like the global tempo distribution that we generated in Section 4.2.2. If we assume that song tempo is the k-th attribute of a song, the *tempo distribution* constraint can be formally defined as a triple (p, k, G), where:

- p is a play set,

- k is the attribute number of song tempo, $1 \leq k \leq A$,
- G is the global tempo distribution,

denoting that it has to hold that $p[k] \sim G$, that is, the k-th attribute of all elements in p has to be distributed according to the global tempo distribution G that was found by calculating the weighted sum of all normal exercise distributions.

4.4 Penalty Functions Definition

Satisfying a set of constraints, according to Tsang (1993), is an NP-hard combinatorial problem. Vossen (2005) proved that a rephrase, the playlist generation decision problem APG-D, is NP-hard by defining it in terms of well-known problems that are known to be NP-complete, such as the Hamiltonian path problem, the 3-dimensional matching problem, the subset sum problem, and the independent set problem. This means that it is unlikely that an algorithm exists that can compute a play set that meets a set of given constraints in polynomial time. In Chapter 7, we will evaluate the complexity of the IM4Sports playlist generation problem determined by the constraints defined above, and compare it with APG's.

Please note that if constraints are conflicting, no feasible solution even exists. Local search (see Section 2.4.3) is able to approximate the solution instead of looking for an exact solution. In the APG algorithm, each constraint is translated into a normalised, piece-wisely linear *penalty function*.

Each penalty function is defined to be zero if the constraint is met. If the constraint is not met, the penalty function has a value larger than zero that increases when the constraint is more severely violated. As an example, the penalty function for the *file size* constraint can be formally defined as

$$f(p,k,max) = \begin{cases} 0 & \text{if} \quad 0.98 \cdot max \leq \sigma \leq max, \\ \frac{a-\sigma}{\delta} & \text{if} \quad \sigma < 0.98 \cdot max, \\ \frac{\sigma-b}{\delta} & \text{if} \quad \sigma > max, \end{cases} \tag{4.2}$$

where $\sigma = \sum_{p_i[k] \in p}$, $\delta = \max(0.98 \cdot max - n \cdot \min D_k, n \cdot \max D_k - max)$, n is the current length of the playlist, and D_k denotes the domain containing all possible values for each k-th attribute. The penalty is zero, if the summation of all k-th attribute values (*e.g.*, total file size) is within the range. Otherwise, the penalty is a normalised difference between that summation and the closest lower or upper bound.

A similar penalty function can be defined for the *all songs should be different* constraint, where the penalty increases for each duplicate song. By assigning a low penalty to the *don't include skipped songs* constraint's penalty function, we ensure that skipped songs are not chosen when they are not strictly needed, but allow for the demands of other constraints to 'override' this constraint.

For the *tempo distribution* constraint, the difference from the ideal distribution is penalised. Therefore, we base the penalty function on the distance between two distributions given by the *Kullback-Leibler divergence* or *relative entropy* (Kullback & Leibler, 1951) between those two distributions. The relative entropy KL is defined between two distributions q and r as

$$KL(q,r) = \sum_x q(x) \log \frac{q(x)}{r(x)},$$

which leads to the following definition of the penalty function for the *songs fit tempo distribution* as

$$f(p,k,G) = \frac{\sum_x p_x[k] \log \frac{p_x[k]}{G(x)}}{f_0(p,k,G)}, \tag{4.3}$$

where $f_0(p, k, G)$ represents the upper bound on entropy, thus normalising the function results to a value between 0 and 1. Important properties of the relative entropy is that it is always positive, equals 0 if and only if the two distributions are alike and that it does not satisfy the triangle inequality, thus making it not a "true" distance metric. A scaled version of normalised entropy has been suggested by Fonseca, Almeida, Crovella & Abrahao (2003), which counteracts results being cluttered near 1. A scaled version f^s of our penalty function, analogous to their definition, is given by

$$f^s(p, k, G) = -\log_{10}(1 - f(p, k, G)). \tag{4.4}$$

As an alternative, we could take each of the exercise distributions that G consists of and follow the following steps to determine the *correlation coefficient* between such a distribution and its sample data.

1. We first obtain the *ranks* by sorting the dataset consisting of all $p_i[k] \in p$ in increasing order, and assigning the rank of 1 to the smallest value, 2 to the number appearing second, etc.

2. We take an independent and identically distributed (i.d.d.) sample equal to the size of the dataset from a normally distributed population with the parameters of G (normal order statistic medians).

3. We then sort these in increasing order, too, and obtain the expected values (means) of the resulting order statistics. Each of the obtained values becomes the *rankit* (Bliss, 1967) of the data point with the same rank.

4. We can generate a *normal probability plot* (Chambers, Cleveland, Kleiner & Tukey, 1983) by plotting the rankits on the horizons axis and the data points on the vertical axis.

5. Chambers, Cleveland, Kleiner & Tukey (1983) state that the closer the resulting plot resembles a straight line, the better the data approximates the normal distribution. Hence, we take the correlation coefficient of the resulting plot, which tests for the strength of the linear relationship between the two plotted dimensions. It results in a value between 0 and 1.

When a correlation coefficient for all exercise distributions has been obtained, the global correlation can be obtained similar to obtaining the global distribution. This can be done analogous to Equation 4.1, by summing the correlation coefficients and taking into account their respective weights.

4.5 Determining Player Contents

By associating weights with all penalty functions, the total penalty can be defined as the weighted summation of all penalty functions. The weight $w_i \in [0, \infty\rangle$ of a constraint's penalty function determines the importance of that particular constraint. If it is important that a constraint should be met, as for example in the case of the *all songs are different* or the *duration* constraint, we attach a high weight to it. Weights can be as high as ∞, indicating it should not be violated at all. This is called a *hard* constraint. Other constraints may merely indicate wishes that might be partly violated, such as the *don't include skipped songs* constraint.

In Chapter 2 the workings of local search have been outlined. The most important notion is that in every iteration, small changes on solutions can be made. In particular, the APG algorithm can add songs, delete songs, replace particular songs in the solution with

other songs from the music collection and swap the songs on two positions. The last change is not relevant in the current project, as no order constraints are used by IM4Sports.

The general idea is to find solutions with a penalty as low as possible. If a new solution is better than the original solution, the old solution is discarded and the new solution is taken to the next iteration. If it is worse, it can still be accepted but with a small probability. Several techniques such as simulated annealing with a standard linear cooling schedule are also included in APG to ensure the algorithm does not get trapped in local minima.

According to Pauws (2002), it can be efficient to not convert all constraints into penalty functions but handle them by preprocessing. In this way, the required calculation time for the local search algorithm can be further minimised. However, only unary constraints are a good candidate to use for preprocessing. An example can be leaving out all songs with a meter different from $\frac{4}{4}$, $\frac{2}{2}$ and $\frac{2}{4}$ from the input music collection, as songs with other meters might be difficult to synchronise to for runners. This is also included in the APG algorithm.

Chapter 5

Adapting Music to Motion

5.1 Exercising Stage Design

During exercising, in the exercising stage, the music is adapted to the user's performance (Functional Requirement **FR 5**), and new songs are selected based on the projected user performance (Functional Requirement **FR 3**).

The main outline of all important components in this stage is given in Figure 5.1. The central part in this stage is the *Scheduler*, which implements the MusicMotion algorithm that will be described in this chapter. The Scheduler component is the intelligent part of the system, which processes the input and determines actual song playing tempo, and the changing of songs.

Two other important components are XMMS and the *Sensory Framework*. The Sensory Framework models the actual physiological state of the user. It passes on messages to and from the Scheduler, but also attaches additional information, such as the exact time on which physiological data are received. The X Multimedia System (XMMS) performs the actual playing of music, and the adaptation of music characteristics such as tempo.

In order to calculate the user's (maximal heart rate HR_{max}, which is needed to correctly interpret training programs, the system is provided with a simple user model that contains

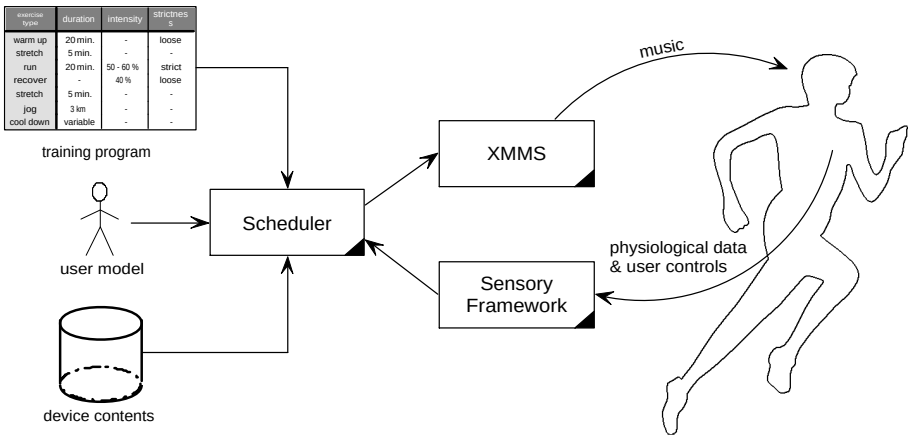


Figure 5.1: Design of the MusicMotion algorithm in the on-line exercising stage

the user's age and gender. When a percentage of HR_{max} is specified by the training program, Equation 2.1 is used and the result is multiplied by the desired percentage. In addition to age and gender definitions, a description of the user's fatigue curve may be included (see also Section 4.2.1). In its simplest implementation, the fatigue curve is specified as a (time, value) fatigue pair that described how many bpm should be subtracted from the value that results from the heart rate percentage calculation after a certain period. The fatigue pair then functions as a correction of the results of the step frequency-from-heart rate calculation.

The task of the Scheduler is three-fold: adapt the music to the user's performance, try to influence the user's performance by adapting the music, and select new songs when needed. The mode of the Scheduler determines its behaviour. The system features four modes, which are partly inspired by the user comments found by Mertesdorf (1994)[1]:

1. *Freeform mode*, in which the Scheduler performs as an ordinary MP3 player and just plays music;
2. *Pace matching mode*, in which the Scheduler follows the step frequency of the exerciser with the music tempo;
3. *Cruise control mode*, in which the Scheduler first tracks the user's step frequency, and locks the music tempo when the 'cruise' key is pressed, and
4. *Motivation mode*, in which the Scheduler matches the music tempo to the user's step frequency, and then tries to influence that performance by slowly changing music play-out tempo.

The freeform mode is a well-known concept and hence will not be discussed any further. We will first outline the pace matching mode, as the other two modes heavily rely on pace matching. Then, we will describe both other modes. The switching of songs does not depend on the mode the Scheduler is in, and will therefore be discussed separately.

In motivation mode (the central mode and the only one that takes into account the training program), all information on music parameters, user variables such as heart rate and step frequency, and which songs are played are logged as specified in Data Requirement **DR 4**. These log files have been specified in Sections 3.7.1 and 3.7.2. We will come back to the use of these log files in the next chapter on the feedback stage, as this is the stage in which the data are required.

5.2 Communication between Components

We already described the communication between the sensors and the PhysioState component, as well as between the Player component and XMMS, in the Sensory Framework in Chapter 3. The communication between all Java components in the system is done using a communication framework called MMa. Java components include the Scheduler, the *PhysioState* component, for reading the sensory input, the Player component which talks to XMMS, and various other components used for wrapping training programs and other helper functions.

MMa (Multimodal architecture) was developed by Vignoli (2003) as a method for supporting more transparent, flexible, efficient and expressive interaction styles. When supporting different interaction styles concurrently, communication between the different components becomes more and more important. MMa allows for different pieces of the same system, called *workers* to run on different computers on separate networks, and still maintain the feel of a single application to the programmer of that application. In its

[1] In Section 2.2 we described that he found that people like to adjust their running tempo manually, but at the same time appreciate it if the music 'dictates' their speed.

id	*attribute*	*definition*	*unit*
physio	?pace	request step frequency	bpm
	!pace	answer step frequency	bpm
	?heartRate	request heart rate	bpm
	!heartRate	answer heart rate	bpm
	?stepCount	request step count	#
	!stepCount	answer step count	#
	?distance	request distance	m
	!distance	answer distance	m
training	?nextExercise	change to next exercise	
	!nextExercise	next exercise	Exercise
	endOfTraining	end of training program has been reached	
userControl	off	turn the system off	
	pause	pause the music	
	play	play/continue the music/exercise	
	skipSong	skip the current song	
	skipExercise	skip the current exercise	
	selectMode	select the mode the Scheduler is in	String
	cruise	change the status of cruise control	
player	tempo	change to a new song tempo	bpm
	tempoReached	a specific tempo is reached	bpm
	?tempo	request current song tempo	bpm
	!tempo	answer current song tempo	bpm
	tempoChangeTime	time in which to change to a new tempo	ms
	volume	change to a new song volume	%
	volumeChangeTime	time in which to change to a new volume	ms
	songProposal	new song proposal	Song
	playNextSong	play the next song (last proposal)	
	songChanged	the current song has changed	Song
	endOfSongNear	the end of the current song is near	
	endOfSong	the end of the current song is reached	

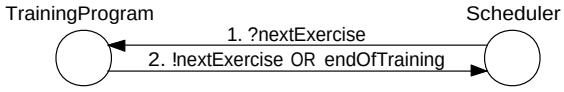
Table 5.2: Available MMa messages in the IM4Sports system

philosophy, it appears to be related to the concept of *agents* (see Wooldridge & Jennings, 1995).

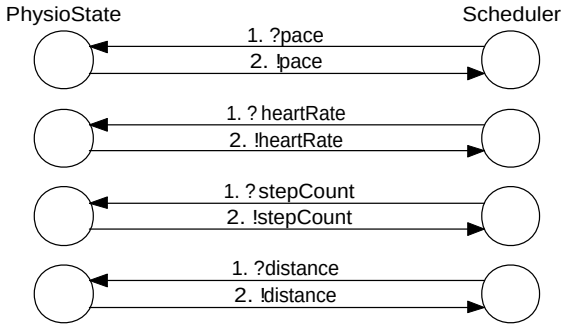
A message sent over the MMa framework can be sent by any worker who is a member of the framework, and is received by all workers who are subscribed to that type of message. A message typically consists of an id, which uniquely identifies the message type, and an attribute- value pair. All message types are defined upfront in a message protocol, so that each worker knows which messages are available in the system.

A list of supported messages can be found in Table 5.2. The list of workers, as far as they have not been mentioned before, include *TrainingProgram*, which wraps the training program, *Player*, which communicates with XMMS, and *UserControl*, which allows for user interaction, which wraps the user model. The *MusicFinder* is not a separate worker, but an intrinsic part of the Scheduler and is able to pass constraints to APG and return the found playlist. The *UserModel* is also part of the Scheduler and gives HR_{max} on demand.

Whenever a new exercise is needed, the Scheduler sends a request to the TrainingProgram and either receives a message with the new exercise, or a message that indicates all exercises have already been sent. A new exercise is needed at the beginning of exercising, when a user wants to advance to the next exercise, or when the current exercise's goals have been reached.



Communication between the Scheduler and the PhysioState component is also straightforward. The Scheduler can make requests for the values stored in the PhysioState, and these are answered by the PhysioState component. These values include step frequency, heart rate, distance ran so-far, and the number of steps recorded by the PhysioState.



In addition to this course of action, the PhysioState can initialise communication, too, by sending the messages tagged with an exclamation mark whenever new sensory data becomes available.

5.3 Tempo Adaptation

In Chapter 2, the different dimensions in which music can be adapted have been described: volume, spatial position, and tempo. Szabo *et al.* (1999) has shown that the effects of aligning (matching) music tempo to the user's step frequency are influential on the exerciser's performance and motivation. Therefore, the focus of this thesis is on music tempo. In its current form, the system will leave volume control to the user. The implementation of volume or spatial position adaptation, if the required hardware is in place, is a straightforward process as the same variables used for adapting tempo can be applied to a spatial position or volume level.

Tempo adaptation has its limits. The WSOLA algorithm that we implemented in the IM4Sports system (see Section 2.5.2) introduces repetitions of waveforms when *stretching* music, that is, slowing the music tempo. When *contracting*, that is, when increasing music tempo, it leaves out parts of waveforms. This process unavoidably introduces artifacts to the resulting waveform which grow more noticeable when the music is stretched or contracted more.

For the system, we introduce the concept of a *song stretch zone*, which is defined by a contraction and stretching level that still sounds acceptable. In general, listeners will not have problems with moderate stretching levels, around 10-15%.

When exceeding moderate stretching, things depend on the music genre and type of instruments in the music. For example, the sound of a hi-hat, with its short 'core' sound, may sound awkward when played solo (separately from other instruments) from about 15% slower. However, when mixed within an entire song, effects may not be noticeable at levels of 35% slower.

The most important characteristic of most music is the vocal. When a familiar song is played stretched or contracted at levels more than moderate, the listener clearly notes that 'something is off'. However, when exercising the attention slips away from the music and is focused more on the rhythm and beat of the song than on its sound quality. This results in stretching levels over moderate, up to 30%, being usable.

Finally, a difference between stretching and constraction has to be made. This last peculiarity, that has to do with the way WSOLA contracts music (Verhelst & Roelands, 1993), is that more than moderate levels are generally unusable when slowing down, because words are stretched so much that short sounds become unidentifiable. When increasing music tempo, this phenomenon does not show, and levels up to 35% are reachable, with 25% being considered a good compromise between sound quality and the awkwardness when hearing a familiar song at a different tempo. At higher levels than a 35% increase in music tempo, large parts of the music get 'lost': words are no longer understandable and syllables are missing.

The song stretching zone, hence, is defined by the interval $\langle 0.85 \cdot \text{tempo}, 1.25 \cdot \text{tempo} \rangle$. For a song with an intrinsic tempo of 100 bpm, this results in the stretching zone equaling $\langle 85\text{bpm}, 125\text{bpm} \rangle$.

The limitations of song stretching result in not every song being feasible for every step frequency. If a song cannot be matched to a particular step frequency, another song must be selected that can be stretched from the current tempo to a tempo that matches the step frequency. This tempo change might even require more than one song change to be carried out completely.

The Scheduler features a configuration file in which settings can be stored and modified. One of these is the song stretching zone, which is defined by the two parameters `maxSongStretchPercentageDown` and `maxSongStretchPercentageUp`. Defaults are 15 and 25, respectively, as described above.

5.4 Pace Matching

5.4.1 Definition

A *pace match* is the situation that occurs when music tempo and step frequency (*i.e.* pace) are considered to be aligned. This is the case when they have equal values, or when music tempo (in a $\frac{4}{4}$ meter) is an even multiple or an integral division of the step frequency. This can be easily understood by taking, for example, a music tempo of 120 beats per minute. This means that, in a $\frac{4}{4}$ meter, 30 measures of each four beats are played every minute.

A step may occur on every beat, as long as the foot impact and the meter follow a regular pattern. So, if a step occurs on the first beat of every measure, the step frequency is 30 and we speak of a pace match. Also, when the step occurs on the first and third beat of the measure (step frequency = 60), it is a pace match. Similarly, the step occuring every beat (step frequency = 120), twice per beat (step frequency = 240) or every two measures (step frequency = 15) are pace match examples.

The system utilises a step frequency monitor, or pedometer, to determine the current step frequency. It is attached to either one of the exerciser's shoes or his belt, where it also functions properly. This sensor, in addition to the heart rate sensor used by the system, brings the total of all sensors to two, which is in concordance with Functional Requirement **FR 7**.

The *nearest multiple* is a multiple (an integral division is also considered a multiple) of the song tempo that is closest to the current step frequency and hence is the value that the system should match to. If the song stretching zone does not include an even multiple of the step frequency (*i.e.*, if the music playout tempo can not longer be matched with the current step frequency), the nearest boundary is selected (either the maximum stretch or the maximum contraction), and a different song is selected.

For instance, if the current song has a moderately fast tempo of 125 bpm (with a stretch range of $\langle 106, 156 \rangle$ bpm) and the user walks at a step frequency of 117 spm, the system would slow down the music playout tempo to 117 bpm. If the user starts to run fast at a step frequency of 175 spm, no valid match inside the song stretch range can be found. In this

case, the music playout tempo is sped up to 156 bpm, which is the stretch border closest to the step frequency. When the tempo indicated by that boundary is reached, the song is changed to one that includes the boundary as well as a multiple of the step frequency, to allow for a smooth transition without abrupt tempo changes.

Following the above example: after the playout tempo stretch to 156 spm, a new song is selected to replace the current one, under the condition that the current song has been played long enough (say, 30 seconds, to reduce repetitive song changing over time). The new song is required to have a stretch range that includes the current playout tempo as well as the current step frequency, in order to allow for a transition without abrupt tempo changes. The new song starts to play out at 156 bpm and will be sped up to 175 bpm. The procedure of changing songs will be outlined in Section 5.7.

Certain differences in musical beat go unnoticed by the human ear. Like in visual aspects of human perception, there's a phenomenon called *just notable difference* (JND) between beats. This is the least difference a distinction between two stimuli can be detected. In a two-alternative force choice listening task, a JND for tempo discrimination in the range of 6.2-8.8% has been reported (Drake & Botte, 1993), whereas for synchronised tapping to a steady beat JNDs in the order of 3-4% were found, and these numbers appear to be equal for musicians and non-musicians alike (Povel, 1981).

An exerciser is generally not able to discern (during a short period in time) a difference of about 4-5 beats per minute. Therefore, we could opt to relax our criteria for pace matching to a range of about 4-5 beats as well. The Scheduler carries a dedicated setting for this in its configuration file called `paceMatchRelaxedness`. Its default is 0.

5.4.2 Propagating Changes

When a change in music playout tempo is made instantly, a hick-up in the music stream occurs. This immediately distracts exercisers and thus needs to be prevented. On the other hand, if the change is propagated over a long time, system response time will increase and be too large to accurately match the user's step frequency. Hence, it is important to make sure that the propagation time for tempo changes is neither too small nor too large.

For the propagation time for music tempo in pace matching mode, we will use the notation T_m in this chapter. T_m specifies the time that a maximal change in music playout tempo, as defined by the song's stretch range, takes. The tempo change time function is denoted by t_m. We define the propagation time for a change of 0 beats to be 0 ms ($t_m(0) = 0$). The propagation time for a maximal change is defined by T_m: $t_m(\Delta_{max}) = T_m$, where Δ_{max} denotes the maximal change in music playout tempo (*i.e.*, from $0.85 \cdot$ tempo to $1.25 \cdot$ tempo). Two functions that calculate the actual time to propagate a new playout tempo can be defined.

The first function, in which $t_i(x)$ is defined in terms of $t_{lin}(x)$, assumes a linear relationship between the two specified points. The time required for a particular tempo change from tempo a to tempo b would then be calculated using:

$$\begin{aligned} t_m(\Delta_{a\to b}) &= t_{lin}(\Delta_{a\to b}) \cdot T_m \\ &= \frac{|a-b|}{\Delta_{max}} \cdot T_m, \end{aligned}$$

where Δ_{max} is calculated by $(1.25 - 0.85) \cdot \text{tempo} = 0.4 \cdot \text{tempo}$. The result of choosing Δ_{max} as one of the parameters in the equation is making it dependent on the song's intrinsic tempo. Hence, slower songs will adapt more swiftly than faster songs will do. The reason for is that people are better at discriminating beats when the music is slower, hence, a change in tempo can be propagated faster than when the tempo is already fast. Both Povel (1981) and Drake & Botte (1993) report JNDs in percentages of tempo instead of as fixed values.

The second function, which uses $t_{sqrt}(x)$, assumes a square root relationship between (0,0) and (Δ_{max}, T_m). This results in the following function for the tempo change time:

$$\begin{aligned} t_m(\Delta_{a\to b}) &= t_{sqrt}(\Delta_{a\to b}) \cdot T_m \\ &= \frac{\sqrt{|a-b|}}{\sqrt{\Delta_{max}}} \cdot T_m. \end{aligned}$$

Using $\Delta_{max} = 10$ and $T_m = 500$ ms, a change of 5 bpm would result in a change time of 50 ms using t_{lin}, and about 158 ms using t_{sqrt}. For a change of 10 bpm the values would be 100 ms and about 224 ms, respectively, and for a 45 bpm change the resulting values are 450 ms for the linear function and about 474 ms for the square root. The use of t_{sqrt} favours the use of larger change times even for smaller changes, and hence creates an overall smoother transition.

The parameter in the Scheduler configuration that specifies the system response time in pace matching mode (T_m) is `timeForMaxTempoChangeWhileMatching`. The parameter `changeTimeFunction` specifies the appropriate function. The default is `sqrt` (denoting the use of $t_{sqrt}(x)$), the other possibility is `lin` (denoting $t_{lin}(x)$). When the system has been put in pace matching mode by the special exercise type 'IM4S_PACEMATCHING', the `parameter` attribute in the training program specifies the value for T_m.

Finding an optimal value for T_m is one of the goals for the experiments used for testing the system. These will be described in Chapter 7.

5.5 Cruise Control

The cruise control mode consists of two stages:

1. *Match pace*, the pace of the exerciser is followed;
2. *Lock tempo*, the tempo is frozen on the current tempo.

The user can switch between both stages by pressing the 'cruise' button, like the cruise control in a car works. In addition, manual controls to set the speed may be included that allow the user to speed up or slow down using buttons instead of having to unlock the tempo, run faster, and lock the tempo again. However, the latter way of control is available for users that prefer it.

5.6 Pace Influencing

5.6.1 Method

One of the goals of the IM4Sports system is to motivate an exerciser to stick to his or her training program. This is done by unobtrusively trying to influence the user's running pace. The system uses a four-step method to motivate users to reach and to keep their heart rate within a certain heart rate zone:

1. match the user's step frequency with the music tempo;
2. determine desired heart rate and, from that, desired step frequency and music playout tempo;
3. propagate the change in music playout tempo from the current tempo to the tempo desired by the determined desired music playout tempo;
4. wait for heart rate stabilisation.

These steps are repeated until the exercise goal has been reached. By matching music playout tempo to the user's step frequency in Step 1, the assumption is that the exerciser will keep her movements in time with later changes in music playout tempo in Step 3. The pace matching mode as described in Section 5.4 is used to synchronise music playout tempo to step frequency. As soon as the pace is matched, the system advances to the next step.

The required music playout tempo that the system needs to change to in Step 3 is calculated in Step 2. This is done using the method that we defined in Section 2.6.3 in four steps. First, we determine the current heart rate and the associated step frequency. Then, we look up the current exercise in the training program and determine the desired heart rate for this specific exercise. Afterwards, we calculate the difference percentage between the current and the desired heart rate. And finally, we apply the same percentage to the current step frequency to determine the desired step frequency. Recall that we are only able to do this at sub-maximal effort levels, as only then the heart rate increases or decreases proportionally with the exercise effort. We use the following equation:

$$\text{desired step frequency} = \frac{\text{desired heart rate}}{\text{current heart rate}} \cdot \text{current step frequency} \tag{5.1}$$

An example: suppose the current heart rate is 100 bpm, the current step frequency is 150 spm, the current music playout tempo is 75 bpm (a pace match), and the desired heart rate is 120 bpm. We use Equation 5.1 to calculate the desired step frequency: $\frac{120}{100} \cdot 150 = 180$.

The desired step frequency is transformed into a music playout tempo, which can be a multiple or integral division of the desired step frequency. If the current song cannot be stretched to accommodate the change in music playout tempo, the system changes songs in the same manner as in pace matching mode.

5.6.2 Propagating Changes

In Step 3, a change in music playout tempo is not carried out instantly. An abrupt change in music playout tempo would probably cause the user not to follow the change, either because of ignorance, surprise, or fatigue. So, a change is propagated over a period of time. Like in pace matching mode, two functions that calculate the time in which a change in music playout tempo is propagated can be defined. The general form is $t_i(x)$ which is supplied with two points $t_i(0) = 0$ and $t_i(\Delta_{max}) = T_i$, where T_i denotes the time a change from a song's maximal contraction point to a song's maximal stretch point should take.

Both functions are defined similarly to their matching counterparts. The first, in which $t_i(x)$ uses $t_{lin}(x)$, assumes a linear relationship between the two specified points. A change from tempo a to b is thus defined as:

$$\begin{aligned} t_i(\Delta_{a\to b}) &= t_{lin}(\Delta_{a\to b}) \cdot T_i \\ &= \frac{|a-b|}{\Delta_{max}} \cdot T_i, \end{aligned}$$

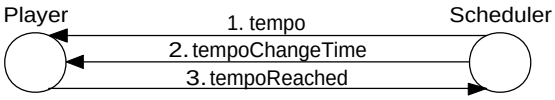
where Δ_{max} is calculated by $(1.25 - 0.85) \cdot \text{tempo} = 0.4 \cdot \text{tempo}$. The other, which makes use of $t_{sqrt}(x)$, assumes a square root relationship between (0,0) and (Δ_{max}, T_m):

$$\begin{aligned} t_i(\Delta_{a\to b}) &= t_{sqrt}(\Delta_{a\to b}) \cdot T_i \\ &= \frac{\sqrt{|a-b|}}{\sqrt{\Delta_{max}}} \cdot T_i. \end{aligned}$$

The `timeForMaxTempoChangeWhileInfluencing` parameter in the Scheduler configuration specifies the propagation time T_i when influencing pace. When the system has been put in motivation mode by the special exercise type 'IM4S_MOTIVATION', the `parameter` attribute in the training program specifies the value for T_i.

The function used to define $t_i(x)$ is specified by the parameter `changeTimeFunction`. The default is `sqrt` (denoting $t_{sqrt}(x)$), the other possibility is `lin` (denoting $t_{lin}(x)$). It is currently not possible to choose different change time functions for matching and influencing as the parameter changes both. This is done to make the resulting system behaviour slightly more transparent when changing settings.

A change is propagated by the Scheduler, by sending a tempo message along with a message specifying the tempo change time to the Player component. When the specified tempo has been reached, the Player notifies the Scheduler of this.



In the experiments conducted to test the proposed system, one goal is to determine an optimal value for T_i. Please see Chapter 7 for the results.

5.6.3 Heart Rate Stabilisation

As we have learned in Section 2.3, Wilmore & Costill (2004) describe that when the required effort of an exercise changes, it may take up to 2 minutes before the heart rate has fully adapted to the change. Thus, it is important that the system waits for the effects of a change to settle before propagating another change. The system described above is tailored to make changes as efficiently as possible, by propagating big changes, if possible, in one restricted period. Therefore, no insuperable problems arise with respect to system response time when a 'cooling period' is used to allow the heart rate to stabilise.

The stabilisation period is dependent on fitness and exercise intensity. Suppose an exerciser starts at a fast walking pace of 6 km/h, and then accelerates to a running speed of 11 km/h. The time it takes the heart rate to stabilise at a value that matches 11 km/h will be longer than it takes to stabilise at a running speed of 9 km/h. Also, the stabilisation heart rate will be lower at 9 km/h than at 11 km/h, given all other exercise circumstances remain the same between both exercises.

In literature, no exact definition of heart rate stabilisation can be found. Therefore, we define heart rate stabilisation to occur when during an interval of 10 seconds the heart rates stay within a 1 bpm deviation of the average of that interval.

We refer to a heart rate by its index number i, starting from index 0. We use the notation hr_i. In addition, we define a function $\text{hr}(t)$ which specifies a heart rate at a specific time t, and a function $t(\text{hr}_i)$ which specifies the time stamp belonging to a heart rate with index i. The notation $\overline{\text{hr}(T)}$ is used to define the average heart rate over the time interval T.

To calculate whether the heart rate stabilises or not, the system keeps a history of all previous heart rates. Over a period of 10 seconds, it checks whether each of the reported heart rates stays within the defined 1 bpm deviation of the average of the zone. If this is true, the heart rate is considered stabilised. Formally, we define the Boolean *stabilisation function* on heart rate index x, as:

$$\text{stabilised}(x) = \forall t \in T : \mid \text{hr}(t) - \overline{\text{hr}(T)} \mid \leq 1 \tag{5.2}$$

where $T = \langle t(\text{hr}_x) - 10, t(\text{hr}_x) \rangle$, in seconds.

This has proven to be an efficient test for stabilisation in the experiments conducted to test the IM4Sports system (see Chapter 7).

Alternatively, the system can continue with the propagation of changes while not waiting for heart rate stabilisation. This behaviour is not recommended for the reasons mentioned earlier. However, it is possible to change Scheduler behaviour with respect to heart rate stabilisation with the boolean parameter `holdChangesUntilHRStabilised`.

5.6.4 Fixed Song

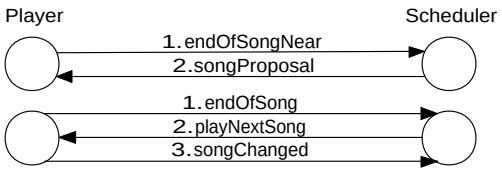
The system allows the selection of one fixed song, which is then played during the entire training session. This can be useful when conducting experiments, and in fact was employed in the evaluation of the system. The associated Scheduler parameters is called `playFixedSong` and sports the *name* of the song that is to be played in exercising mode.

Please note that in fixed-song mode, no restrictions other than technical limitations are posed on song stretching. That means that the fixed song may be stretched as far as 50% and contracted as much as 25%. This allows the song to be used with nearly all step frequencies, as usually a multiple or integral division can be found in this broad range. Of course, for this to hold, the song's intrinsic tempo should not be too low, nor too high.

5.7 Selecting New Songs

A song change may be needed when the user requests it, when the song stretching zone no longer includes the predicted music tempo needed, or when a song is finished.

In the latter case, the Player component notifies the Scheduler in advance, for example 5 seconds before the song finishes, so the Scheduler has time to select a new song, which is sent to the Player component. If the song is really finished, the Player component already has a song proposal, and sends a notification to the Scheduler. The Scheduler instructs the Player component to play the next song and the Player component confirms the song change.



In addition, the Scheduler may take action by sending a song proposal and a `playNextSong` message to prompt the Player component to change the song. The Scheduler always receives confirmation of a song change.

In order to select new songs when needed, the system utilises the APG algorithm to select songs from the available music database.

As in the off-line preparation stage, we need to formulate constraints for APG. However, as opposed to the previous chapter, where play sets where generated to determine the device contents, we will now use APG to select only one song at a time from the database. To ensure that it is not a song that has already been played or twice the same artist in a row, we will define a *not on blacklist* and a *don't repeat artist* constraint.

The most important constraint, however, is the *tempo* constraint that takes into account the predicted step frequency and thus the predicted tempo of the system. Therefore, we will first define how we predict step frequency from the history. When we arrive at a prediction function, we will continue with the formal constraints.

5.7.1 Step Frequency Prediction

In addition to heart rate values, the system also keeps track of the previous step frequency values. It uses this history to make a prediction how the heart rate (and hence the music playout tempo) will progress in the next 30 seconds. This prediction provides valuable information on what song is best to choose next at the moment a song change is required for playback. Making a good prediction effectively minimises the number of song changes needed by eliminating the need to switch between different tempo values. Hence, it has a

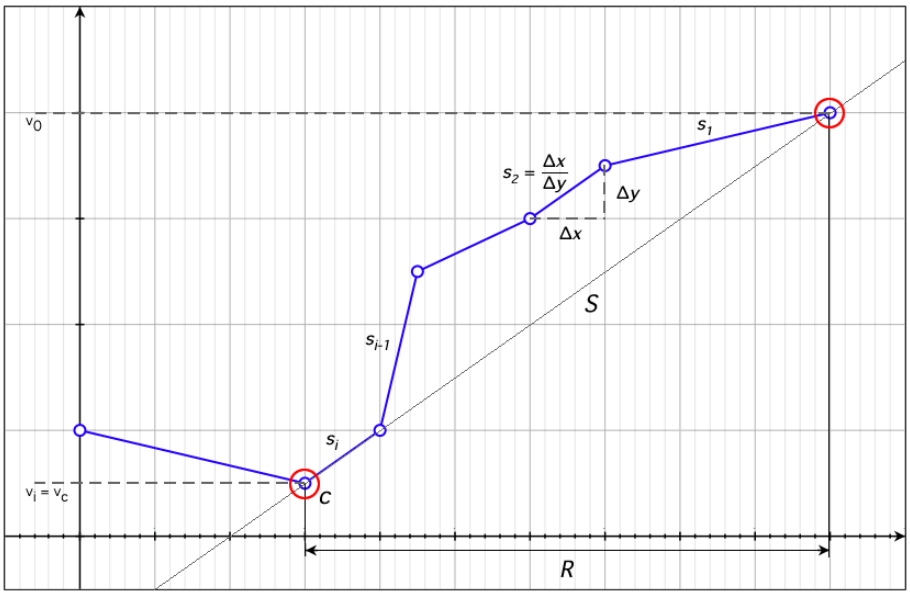


Figure 5.3: Slope prediction of step frequency
Please see text for an explanation of used symbols.

particularly important role in a system that needs to wait for extended periods of time, like the IM4Sports system, for the results of a taken action to become apparent.

The step frequency, like the heart rate, is stored in time-value pairs (t, v) where t is the current machine time stamp, and v is the value of the step frequency in spm.

Traditional prediction methods, such as the well-known *regression analysis* or the *least squares method*, all presume that the data can be approximated by function of a particular form, whose parameters are then changed to optimally reflect the data. The heart rate data is influenced by external circumstances as well as internal, and hence, for the current problem, no such function can be determined beforehand. To make a prediction of the step frequency, we therefore use a method based on *slope prediction* (*cf.* Jiang, Shao, Li & Jia, 2002) instead. This method suggests that we determine the slopes of a particular range in the step frequency history and use these slopes to determine the overall direction in which the step frequency is moving.

Our first step is to determine the direction of the change. We traverse the step frequency values in reverse order, from the last point, until a value that is not equal to the current value has been found. If it is larger, the step frequency is decreasing, if it is smaller, the step frequency is increasing. In the example figure, 5.3, the last point in the history is included on the rightmost side of the graph.

Secondly, we need to determine where the increase or decrease has started. We again traverse the values of the history backward and find the point where the sign of the tangent of the step frequencies changes. We call this the *critical point c*. For example, when the values are decreasing and we come across an increase, this is the critical point. The critical point will function as the 'anchor' from which we make the prediction. The window between the critical point and the current step frequency value is defined as the *critical range*.

Now, we need to determine the slopes of all points in the critical range, and assign a weight to all these slopes. We want to assign a weight to every value ensuring that the weight of a value decreases with the distance from the current value, that is, the earliest point gets the least weight, while the current value has a large weight. This is done by defining the weight of a value as the inverse of its order in the range.

Say that the critical range R includes r (t, v) points. We define[2] $R = [v_i, v_{i-1}, \ldots, v_1, v_0]$ where $v_i = v_c$, that is, the step frequency at the critical point, and v_0 indicates the current step frequency. The slope s_i and the weight w_i as the inverse of its index can be defined for all $i \in [r, 0\rangle$:

$$s_i = \frac{v_{i-1} - v_i}{t_{i-1} - t_i}$$

$$w_i = \frac{1}{i}$$

Finally, the prediction function P is the weighted summation of all slopes, correcting for all weights, multiplied by the number of milliseconds in which the prediction is needed (Equation 5.3).

$$P(t) = v_0 + \frac{\sum_{i=1}^{r} w_i s_i}{\sum_{i=1}^{r} w_i} \cdot t \tag{5.3}$$

5.7.2 Weighting Global Slope into Predictions

One extension of the prediction function is to take into account the global slope from the critical point to the current point in addiction to the prediction made, and use a subjective weight.

The global slope S is defined by

$$S = \frac{v_0 - v_c}{t_0 - t_c},$$

and hence the advanced prediction function $P^+(x)$ is formally defined by taking a weighted average of both the global and the originally predicted slope. The weight $w \in [0, 1]$ expresses the importance of the global slope with respect to the originally calculated value.

$$P^+(t) = (1 - w) \cdot P(t) + w \cdot S \cdot t \tag{5.4}$$

The Scheduler setting in which the importance of the global scope w, is expressed, is `weightForGlobalScopeInPredictions`.

5.7.3 Constraint Definition

Predicted Song Tempo

Now we are able to predict the step frequency, a constraint for the selection of songs can be formulated. If we assume that the k-th attribute of a song denotes its intrinsic tempo, the *tempo* constraint could formally be defined as a triple (p, k, t), where:

- p is a play set of size 1,
- k is the attribute number of song tempo, $1 \leq k \leq A$,
- t is the number of milliseconds in the future for which a prediction needs to be made,

[2] Please note that due to this above definition, an index of $i - 1$ is *later* in time than an index i!

denoting that it has to hold that $p_0[k] = P^+(t)$ and, obviously, p_0 is the first (and only) song in the play set. For example, if we define t to be 30 seconds, the resulting play set should consist of one song of which the tempo is equal to the value of $P^+(30000)$. If such a song does not exist in the music collection on the portable device, the penalty function of this constraint will ensure that it is violated as little as possible by selecting a song which has a tempo close to (a multiple or integral division of) the desired tempo.

Don't Repeat Songs

When exercising, the motivational effects of a song decrease when it is played more often (Karageorghis *et al.*, 1999). Using APG to generate a one-song play set, however, may result in the same song more often. Therefore, the system keeps a blacklist b of already played songs, which has the same formal properties as a play set, so that we can formally define a pair (p, b), where

- p is a play set of size 1,
- b is a play set with already played songs,

denoting that it has to hold that $\forall s \in b : s \neq p_0$. By assigning a low penalty to this constraint's penalty function, we ensure that the song is not chosen when it is not strictly needed. However, when no more songs are available in the portable Player's music collection that fit certain tempo characteristics, the Scheduler will select blacklisted songs anyway.

A Scheduler setting is implemented, `allowSongsPlayedTwice`, which regulates the inclusion of this constraint in the on-line song selection process. It defaults to `false`.

Don't Repeat Artists Successively

In addition, one could want more variety in the music by ensuring artists are not played successively. In a similar fashion, many more constraints can be formulated to ensure, for example, that songs with a specific tempo or mood are not played successively. Only the value of k in the following constraint would have to be changed, to point to the right song attribute. The formal definition of the *don't repeat artists successively* constraint, then, is a triple (p, k, s), where

- p is a play set of size 1,
- k is the attribute number of song artist, $1 \leq k \leq A$,
- s is the current song,

denoting that it has to hold that $p_0[k] \neq s[k]$. The associated (Boolean) Scheduler setting is `allowSameArtistSuccessively` which defaults to `false`.

5.8 User Interaction

Although the MusicMotion process outlined above is largely automatic and system driven, the user is able to interact with the system, too. Functional Requirement **FR 6** states a number of buttons on the Player, whose functionality will be described in this section. Also, the audio signals that the system uses to communicate its current status to the user will be outlined.

5.8.1 Buttons

Play/Repeat Button

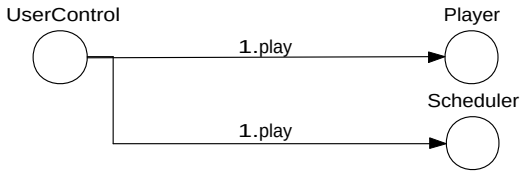
The ‘play’ button simply prompts the system to start the current exercise, or, when music playback is paused, to resume playing the current song. When the music is actually playing, the button changes to a repeat button. This fixes the current song and when it has finished, it will be repeated. Please note that unlike fixed song behaviour which was included as a debug or evaluation Scheduler setting, this does not affect the song’s stretching zone. So, when the limits of the current song stretching zone are reached, the system will change the song no matter the repeat setting.

Alternatively, an audio signal could be played (see Section 5.8.2) that signals the user that the song cannot be stretched further, and the current stretching level is maintained until the user unlocks the song repeat mode. The choice between one of these two possible interaction styles is a matter of preference.

The use of the repeat functionality gives the concerning song a higher priority in the next off-line selection process. This means that it will be more likely that the song is selected for the next training program, provided that its tempo fits one of the exercises in it.

Please note that the extended functionality of the ‘play’ button effectively eliminates the need for a ‘start exercise’ button. An exercise can only be started when it is stopped. This is the case before exercising or when the entire training program has been stopped – but music playback is stopped then, too. Hence, the ‘play’ button functions for starting the exercise as well as the music playback and no separate ‘start exercise’ button is needed.

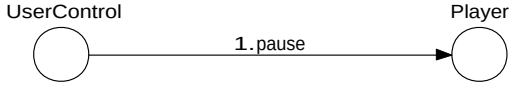
Because of the dual functionality, both a message to the Player component and the Scheduler is sent.



Pause Button

The ‘pause’ button pauses music playback. However, all other system functions are not halted. So, the song may be changed during pause, which would cause the user to hear a different song after unlocking the pause playback mode as before. The new song is selected when the user’s step frequency demands a music playout tempo not included in the current song’s stretching zone. The use for a ‘pause’ button can be seen in complex environments where an exerciser is running alongside other traffic. Potentially dangerous circumstances can be prevented with careful use of the ‘pause’ button.

Communication for the pause button is done directly between the UserControl and Player components.

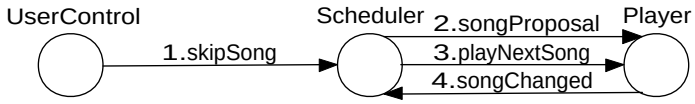


Skip Song Button

When the system has selected a song for playback that the user does not like, the user is able to skip it with the ‘skip song’ button. The use of the skip song functionality gives the concerning song a lower priority in the next off-line selection process. Hence, it will be

less likely that the song is selected for the next training program, regardless of whether its tempo fits one of the exercises in it.

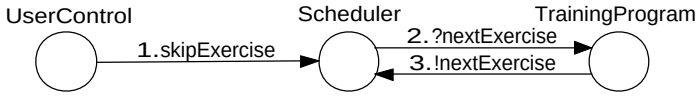
In the case that a user skips a repeated song, the repeat mode will be terminated and neither an extra nor a lesser weight is attached to the song.



Skip Exercise Button

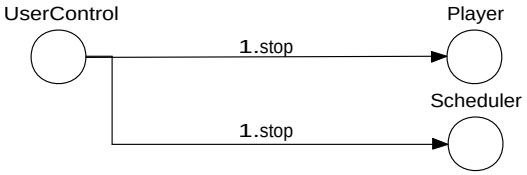
As with the 'skip song' button, an exercise can be skipped as well. As some songs can be specified in the training program to continue even though their goals are reached, this button provides vital functionality for the Player. For example, an exercise like 'run 5 kilometers' is a likely candidate exercise to continue after reaching the goals, as it is not that important whether the user runs for *exactly* 5. For example, the user might run a specific track which length is just 4.9 km. In that case, we would like the user to be able to determine exactly when to continue to the next exercise.

On the other hand, in a training session where the user needs to stretch for 1 minute and then relax for 30 seconds, and repeat this three times, it would not be user friendly if the user needs to press a button after every part. In these cases, it is specified in the training program that the next exercise is automatically started when the goals of the previous exercise are fulfilled.



Stop Button

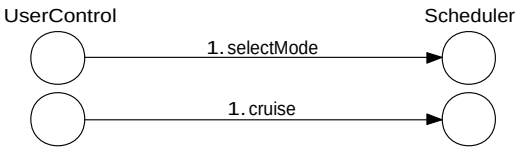
When the user wants to quit the training program and thus exercising, the 'stop' button can be used. This stops music playback as mentioned before. In effect, the 'stop' button is rather a 'pause exercising' button, as the training program can be resumed by pressing the 'play' button.



Mode Selection

The user can use the mode selection switch to select one of the Scheduler modes described in Section 5.1. The training program is considered only in motivation mode, hence the 'skip exercise' button will only work in this mode. When the mode is changed to motivation mode, the training program is reset and the user can start from the first exercise.

In cruise control mode, the user can activate the music tempo lock by pressing the cruise button. All communication goes directly to the Scheduler component.



5.8.2 Audio Signals

The system needs to alert the user of some circumstances. Since the communication from the system to the user is largely based on sound (the user would not look at a display constantly during exercising), only auditory icons are used. Auditory icons are everyday sounds meant to convey useful information to computer users by analogy with the sounds produced by everyday events (Gaver, 1986). Auditory feedback seems to provide immediate and engrossing feedback. Users do not have to think about a *pop!* sound, indicating a button in a visual interface appearing from nowhere, nor about the *whirr. . .* sound a copying machine would make when a copy button is pressed. Research by Gaver & Smith (1995) however states that using sounds in the interface can become annoying after a few exposures. They state that sound should be concise and unobtrusive, and should reflect a number of data dimensions at once to avoid repetition (such as using different sounds for different sizes of objects).

The most important signal is the signal that tells the user that an exercise's goals have been reached, or that the training program is finished. In order of magnitude, this could be a 'smaller' sound for an exercise that's finished, and a 'larger' sound for a whole training program. Upon the signal that an exercise's goals are reached, the user can choose the moment that she wants to advance to the next exercise.

The start of a new exercise is signalled by a different sound. This is played at the end of an exercise that automatically advances, in conjunction with the 'goals finished' sound. It is also played when the user presses the 'next exercise' button.

When the user opts for the second song repeat style (see above), that is, to prioritise song repeat above accurate stretching, an audio signal must be played to signal the system needs to change the song but isn't able to because of the song lock. This signal sounds roughly like a *buzz!* sound, signalling some part of the system is in error.

Finally, a small *click* sound is played when the user presses a button that does not have an associated sound such as the 'skip exercise' button. In this manner, the user knows that the system has registered the button press. This prevents the user from having to look at the display during exercise.

Implementation of audio signals must be limited to absolutely necessary sounds like the ones described above, since sounds in addition to the music get obtrusive and annoying quickly.

Chapter 6

Inference-Based Personalisation

6.1 A Personalised Tempo Distribution

The most important consideration in making a selection of music based on a training program for the IM4Sports system, is that the tempo must be right. As described in Chapter 4, the proposed system will use a tempo distribution per exercise. This is a normal distribution and as such is defined by its mean (μ) and its variance (σ^2).

We use the gender/age group normal tempo distribution as the starting point for obtaining a personalised tempo distribution. The TempoInference algorithm in the feedback stage of the system will update the gender/age group distribution for each individual user, when data from the performed exercises becomes available after each training session (*cf.* Functional Requirement **FR 8**). This requires the logging of all data, which is done during the exercise stage in accordance with Data Requirement **DR 4**.

6.1.1 Prior Distribution

In each learning step, the *prior distribution* is the distribution obtained as the result of the previous learning step. In the first learning step, the gender/age group distribution is the prior distribution.

As an example, we consider a hypothetical exerciser who is male and between 20 and 25 years old, and a hypothetical exercise 'jogging'. We assume that for the target gender/age group of all males between 20 and 25 years old the music tempo used in a 'jogging' exercise is normally distributed with parameters $\mu = 100$ and $\sigma^2 = 10$.

Now suppose that the value of μ is not a discrete value. Instead, we consider μ to be a normally distributed, as well. In the distribution of μ, the mean m' defines the most probable value for μ, and the variance σ'^2 expresses the inverse of the *certainty* we have in that particular value. We formally define a prior normal distribution for μ with parameters (m', σ'^2):

$$\mu \sim N(m', \sigma'^2).$$

The confidence or certainty c we have in the mean value of a particular normal distribution is indicated by Equation 6.1:

$$c = \frac{1}{\sigma^2} \quad \text{for} \quad \sigma \neq 0. \tag{6.1}$$

6.1.2 Sample Data

Each time the user performs the specific exercise, data is collected in a file `Player.csv` (see Section 3.7.2). For each exercise, this data contains each music tempo and the length

it was used.

For example, say that the user has run for 180 seconds. Of these 180 seconds, the user has run 90 on 100 bpm, 75 on 101.5 bpm and the remaining 15 on 103 bpm. The mean m of the sample set can now be determined[1]:

$$\begin{aligned} m &= \frac{1}{n}\sum_{i=1}^{n}(x_i) \\ &= \frac{(90 \cdot 100) + (75 \cdot 101.5) + (15 \cdot 103)}{180} \\ &= 100.875. \end{aligned}$$

We will use similar sample data for each individual exercise to find a better value for m' and thus for μ, and this value will have to be updated and become more precise with every new exercise performed by the user.

6.1.3 Learning Objective

In the next two sections, we will see that traditional inference methods assume that our confidence in a particular value for m' increases with every learning step. The traditional point of view is that 'data is certainty', and hence every bit of data increases confidence in the value that happens to be the mean of the distribution. This is partly due to the assumption that the data is linear, and independently identically distributed (i.d.d.). Instead, the IM4Sports system works with non-stationary data – it is never 'finished learning', as the fitness for a user of the system will continuously change, and hence the needed music tempo.

As opposed to a i.d.d. view, we will introduce a method which takes the learning abilities of the traditional methods, and upgrades them so that the posterior variance σ''^2 and thus posterior confidence c'' is dependent on the differences in prior distribution and sample data. In this way, if the sample data is a distribution that is significantly different from the prior distribution, but has a large certainty (when the data has a small variance), the resulting confidence in the data will be smaller (and the resulting variance larger) because the system is 'confused' between which of the inputs (prior or sample data) it should believe. Similarly, when the sample data confirms the prior data by being alike, the resulting confidence will be greater than the confidence in the prior data.

We define three criteria on which the methods will eventually be evaluated. Because exercisers use the system while working on their fitness, user characteristics are bound to change, albeit slowly. Hence, we evaluate *flexibility*, that is, a good personalisation method should still be able to learn even after a large number of learning steps, without being locked in a certain position.

The second criterium is that a good personalisation method should be able to *consolidate* a number of (almost) equal learning steps by expressing an increased confidence in the posterior distribution (*i.e.*, the variance should be decreased).

The final evaluation criterium is how the methods cope with *noise*. Noise, that is, sample data with low certainty, should not be taken into account as much as sample data with high certainty. The posterior distributions should reflect the certainty of the sample data, while at the same time avoiding locks due to a lack of flexibility.

The evaluation results on these criteria are presented in Chapter 7. In this chapter, we will investigate the traditional methods for inference-based learning, and outline our proposed method.

[1] The variance of the sample set is not used in the traditional inference method, which is why we will return to the sample variance later.

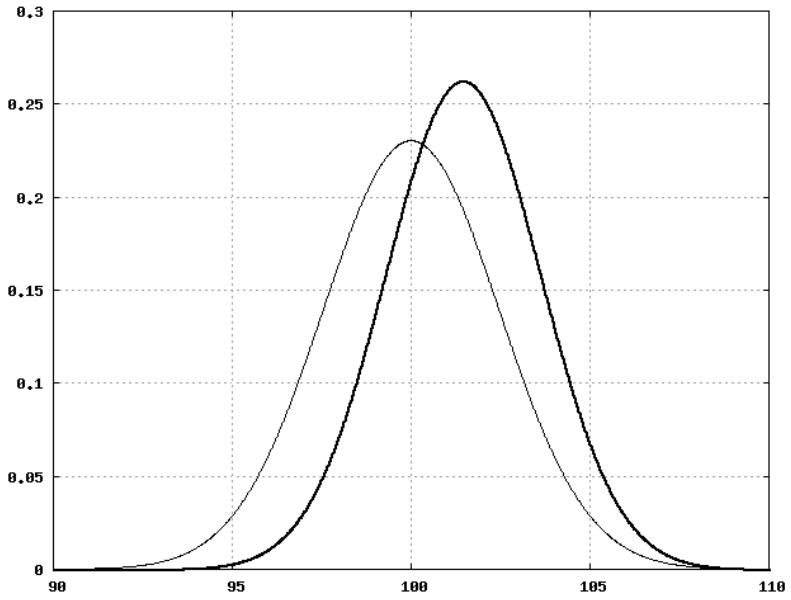


Figure 6.1: Learning step from a normal distribution with parameters ($m' = 100, \sigma'^2 = 3$) to an updated normal distribution (bold) with parameters ($m'' = 100.29, \sigma''^2 = 2.31$)

6.2 Traditional Inference Learning

If the prior distribution of μ is a normal distribution with mean m' and σ'^2, the posterior distribution of μ will also have to be a normal distribution with mean m'' and σ''^2 (Section 8.8, Winkler & Hays, 1975). The posterior distribution is defined using Bayes' Theorem (Bayes, 1763):

$$m'' = \frac{(1/\sigma'^2)m' + (n/\sigma^2)m}{(1/\sigma'^2) + (n/\sigma^2)} \tag{6.2}$$

$$\sigma''^2 = \frac{1}{(1/\sigma'^2) + (n/\sigma^2)} \tag{6.3}$$

Bayes and his theorem are discussed in detail by Stigler (1982).

The posterior mean is a weighted average of the means of the prior and sample distributions. The value of n reflects the number of exercises that are contained in the sample data and as we are always considering data from one experiment, it currently is equal to 1.

So, for our example, this means the posterior distribution will have the parameters

$$m'' = \frac{(1/3)100 + (1/10)100.875}{(1/3) + (1/10)} \approx 100.29$$

$$\sigma''^2 = \frac{1}{(1/3) + (1/10)} \approx 2.31.$$

Note that since the variance is now smaller than it was (2.31 versus 3), in the next learning step the influence of the sample data will most likely be smaller, as well. The difference between the prior and posterior distributions of μ is depicted in Figure 6.1, and an example of a traditional inference progression is given in Table 6.2.

The posterior distribution of μ, m'', is not the same as the distribution of the music tempo, even though values of the mean of both distributions are equal. μ is the mean of the

	mean	*variance*	*certainty*	
prior distribution	100	3	0.33	
sample data 1	100.875			
posterior distribution	100.20	2.31	0.43	(↑)
prior distribution	100.20	2.31	0.43	
sample data 2	102			
posterior distribution	100.33	1.88	0.53	(↑)
prior distribution	100.33	1.88	0.53	
sample data 3	87			
posterior distribution	98.22	1.58	0.63	(↑)
prior distribution	98.22	1.58	0.63	
sample data 4	92			
posterior distribution	97.38	1.36	0.74	(↑)

Table 6.2: Example progression using traditional inference learning

music tempo distribution, whereas m' is the prior mean of the distribution of μ, and m'' its posterior mean. Similarly, σ^2 is the variance of the music tempo distribution, σ'^2 and σ''^2 are the prior and posterior variance of the distribution of μ, respectively.

As the resulting distribution is used for personal tempo selection whereas the input really is the distribution of music tempo as it was used for a large group of people, the variance σ^2 of the gender/age group distribution can be neglected or it can be taken as a guideline for determining the starting variance.

The value of σ^2 can also be considered a *design choice*. We could define σ^2 to be larger or smaller depending on the desired behaviour of the system. Smaller values imply a smaller variance in the used tempo, whereas larger values imply a greater tempo range for the songs that are selected for this particular exercise.

In any case, σ^2 is a constant, keeping the same value after each learning step. Since the variance of the distribution remains unchanged, the resulting distribution of music tempo in this particular exercise will be much like the distribution in Figure 4.2, but with a mean μ of 100.29.

In the above Equation 6.3, every added bit of sample data makes the variance of the posterior distribution smaller. The decrease of the posterior variance is currently fixed, as it is dependent only on σ'^2 (variance of the prior) and σ^2 (tempo variance). Remember that σ^2 is currently a constant (it's 10 in every new step). This makes σ'^2 the only changing variable in Equation 6.3. Its value is replaced by that of σ''^2 in each new learning step. The posterior variance will thus be approaching zero. This behaviour can also be observed in Table 6.2.

6.3 Sample Variance-Based Inference Learning

As we outlined before, we would like the posterior variance to be dependent on the variance of the sample data. In this way, the certainty of the end result would be affected by the certainty of the sample data. To this end, the *sample variance*, denoted by S^2, can be used as an estimator for σ^2.

However, according to Winkler & Hays (1975), Section 6.2, S^2 is not an unbiased estimator and needs to be corrected. This leads us to define the *modified sampling variance* $\hat{S}^2$ as an unbiased estimator for σ^2 ($\mathbf{E}\{\hat{S}^2\} = \sigma^2$), in which n refers to the number of seconds

in one learning step:

$$\hat{S}^2 = \frac{1}{n-1}\sum_{i=1}^{n}(x_i - m')^2 \tag{6.4}$$

where the sample population is expressed by x_i with $i = 1, 2, \ldots, n$. We can proof $\hat{S}^2$ is an unbiased estimator.

Proof. An estimator is defined as unbiased when it holds that $\mathbf{E}\{\hat{X}\} = x^2$. In our case, this means we need to solve $\mathbf{E}\{\hat{S}^2\} = \sigma^2$.

$$\begin{aligned}
\mathbf{E}\{\hat{S}^2\} &= \mathbf{E}\left\{\frac{1}{n-1}\sum_{i=1}^{n}(x_i - m')^2\right\} = \frac{1}{n-1}\sum_{i=1}^{n}\mathbf{E}\left\{(x_i - m')^2\right\} \\
&= \frac{1}{n-1}\sum_{i=1}^{n}\mathbf{E}\left\{((x_i - \mu) - (m' - \mu))^2\right\} \\
&= \frac{1}{n-1}\sum_{i=1}^{n}\mathbf{E}\left\{(x_i - \mu)^2\right\} - 2\mathbf{E}\left\{(x_i - \mu)(m' - \mu)\right\} + \mathbf{E}\left\{(m' - \mu)^2\right\} \\
&= \frac{1}{n-1}\sum_{i=1}^{n}\sigma^2 - 2\left(\frac{1}{n}\sum_{j=1}^{n}\mathbf{E}\left\{(x_i - \mu)(x_j - \mu)\right\}\right) + \frac{1}{n^2}\sum_{j=1}^{n}\sum_{k=1}^{n}\mathbf{E}\left\{(x_j - \mu)(x_k - \mu)\right\} \\
&= \frac{1}{n-1}\sum_{i=1}^{n}\sigma^2 - \frac{2\sigma^2}{n} + \frac{\sigma^2}{n} \\
&= \frac{1}{n-1}\sum_{i=1}^{n}\frac{(n-1)\sigma^2}{n} = \frac{(n-1)\sigma^2}{n-1} = \sigma^2
\end{aligned}$$

□

On a side note, this means the sampling distribution will be a Student's or *t*-distribution (Gosset, 1908) instead of a normal distribution. However, for large values of n like the ones we use (an n of 1 means 1 second, and a typical exercise lasts a couple of minutes) the resulting (shifted) *t*-distribution is a good approximation of a standard normal distribution.

Now $\hat{S}^2$ can be calculated for our example values and substitute it for σ^2 in Equation 6.2 and 6.3.

$$\hat{S}^2 = \frac{(90 \cdot (-0.875)^2) + (75 \cdot 0.625^2) + (15 \cdot 2.125^2)}{(180 - 1)} \approx 6.22$$

$$\begin{aligned}
\sigma''^2 &= \frac{1}{(1/\sigma'^2) + (n/\hat{S}^2)} \\
&= \frac{1}{(1/3) + (1/6.22)} \approx 2.02
\end{aligned}$$

This results in the progression of Table 6.4, in which we can clearly see the relatively big influence of the third learning step as opposed to the relatively small influence of the fourth learning step, which we assigned a sample variance of 1 and 20, respectively.

6.4 Mean Distance-Based Personalisation

Now we have seen the two most frequently used methods for i.d.d. inference learning, the traditional and the sample-variance based method, we review our original objective again, as posed in Section 6.1.3. We have defined the confidence c we have in the data as the inverse of the variance σ^2 (Equation 6.1).

[2] See also the lemma on *Variance* on Wikipedia.

	mean	*variance*	*certainty*	
prior distribution	100	3	0.33	
sample data 1	100.875	6.22		
posterior distribution	100.29	2.02	0.50	(↑)
prior distribution	100.29	2.02	0.50	
sample data 2	102	6.22		
posterior distribution	100.43	1.53	0.65	(↑)
prior distribution	100.43	1.53	0.65	
sample data 3	87	1		
posterior distribution	92.31	0.60	1.67	(↑)
prior distribution	92.31	0.60	1.67	
sample data 4	92	20		
posterior distribution	92.31	0.59	1.69	(↑)

Table 6.3: Example progression using sample variance-based inference learning

Intuitively, the behaviour of both methods is not what we would like, given our definitions, although it might be expected from a statistician's point of view. Suppose the user ran the exercise a couple of times, with a mean exercise tempo of 102. Then, suddenly, the next mean exercise tempo is 87, like in Table 6.4. We see that m'' shifts a bit towards 87, which is like we expect, and σ''^2 becomes smaller. The resulting variance (and thus the resulting confidence, which is a function of σ^2 only!) is independent on the value of m'.

In other words, with every added bit of information the certainty about the value of μ increases, *no matter what that bit of information is!* We have explained earlier that this is caused by the assumption of independently identically distributed data in the traditional model. The behaviour that we would like, is for the third learning step to increase the posterior variance, since it decreases certainty. That is, it makes us less confident about the validity of the posterior mean.

Informally, we would say that we were pretty certain of a mean around 102. Then, a new mean of 87 was found after parsing exercise data, which is in conflict with our previous confidence in the mean being about 102. And to make things worse, the confidence in this new mean is high:

$$c_{sampledata} = \frac{1}{\hat{S}^2} = \frac{1}{1} = 1.$$

This should actually change the posterior mean m'' more than proportionally to the sample mean m', as happens using sample variance-based inference learning, yet it should also decrease c as we clearly have conflicting beliefs. The shifting of the posterior mean to somewhere in the middle does not make us feel confident that this is the right value. We will upgrade the sample-variance based method to a new inference learning method called *Mean Distance-Based Personalisation*, which copes with the intuitive problems by weighting the change in posterior variance by the distance from the prior mean. Please note that although this method redefines the posterior variance, no changes are inflicted upon the posterior mean, which remains the same as in the sample variance-based inference learning method.

We first introduce the concept of the *variance change* v. Due to the effect of the sample data, a change in variance between the prior and the posterior distribution occurs, which is

$$v = \sigma'^2 - \sigma''^2. \tag{6.5}$$

The variance change is a positive real number, since the posterior variance is currently always smaller than the prior variance. In the case of our example, in the first learning step v is $3 - 2.02 = 0.98$. The variance change has a larger value when the sampling data has a

smaller variance, and as such could be used as an indicator of the importance of a particular learning step (*i.e.*, the greater v, the greater the importance of the learning step).

Because of the learning objective we formulated and the additional reasons mentioned earlier, we need to make the influence of the variance change dependent on the distance between the mean of the sample data m and the prior mean m'. The more distant from the prior mean the sample data is, the more the posterior variance should be increased. Again intuitively, this matches to what we already described: the more different the data is from what we already believed, the less confident we will be in the resulting posterior mean.

To this end, two opposing force functions can be defined, which both have an influence of the posterior variance: a positive variance change v which is strong when the sampling mean m is near the prior mean m', and a *negated variance change* $-v$ which gets stronger when m is further from m'.

Modified Posterior Variance with 1 Standard Deviation Limit

If we take a simple approach, we could define the *modified posterior variance* $\hat{\sigma}''^2$ as being dependent on whether or not m is within one standard deviation from m' :[3]

$$\hat{\sigma}''^2 = \begin{cases} \sigma'^2 - v & \text{if} \quad |m - m'| \leq \sigma', \\ \sigma'^2 + v & \text{if} \quad |m - m'| > \sigma'. \end{cases}$$

We immediately see the problems associated with this approach: an m just within one standard deviation of m' would decrease the posterior variance, whereas one just outside would increase the posterior variance. Instead, we need to find a more fluent approach where no hard limits are imposed on the sample variance.

Continuous Modified Posterior Variance

In the continuous method we propose, both the variance change and the negated variance change are influencing $\hat{\sigma}''^2$. Both have a weight dependent on the value of the prior distribution at the place of m' (the weight of the negated variance change $(1 - w)$ is actually the complement of the weight of the variance change w):

$$\hat{\sigma}''^2 = \sigma'^2 - w \cdot v + (1 - w)v. \tag{6.6}$$

In our example, the prior distribution has parameters $m' = 100$ and $\sigma'^2 = 3$, while $m'' = 100.875$ (remember m'' remains unchanged with respect to the sample variance-based method). Then, the value $f(m'')$ of the prior distribution at m'' is defined as

$$\begin{aligned} f(x) &= \frac{1}{\sigma\sqrt{2\pi}} e^{-(x-\mu)^2/2\sigma^2} \\ f(100.875) &= \frac{1}{\sqrt{3}\sqrt{2\pi}} e^{-(100.875-100)^2/(2\cdot 3)} \\ &\approx 0.203 \end{aligned}$$

Defined as a percentage of m', the relative weights for the variance change and the negated variance change can be calculated[4]:

$$w = \frac{f(m') - |f(m'') - f(m')|}{f(m')} \tag{6.7}$$

[3] Note that we really should have written $\sigma'^2 - -v$ in the second part of the equation.

[4] The upper part of the equation is there to ensure an equal processing whether m'' is bigger or smaller than m'. We take the distance, which is $|f(m'') - f(m')|$ and subtract it from $f(m')$ to ensure we get the lowest of the two values.

Using Equation 6.7, we get $w \approx 0.88$. Entering w in Equation 6.6 then finally gives us

$$\begin{aligned} \hat{\sigma}''^2 &= 3 - 0.88 \cdot 0.98 + 0.12 \cdot 0.98 \\ &\approx 3 - 0.8624 + 0.1176 \approx 2.26. \end{aligned}$$

As opposed to the value 2.02 for the original posterior distribution, we see a small effect by the introduction of the weights on the variance change. Now, let us see how this works for the entire table from our example (Table 6.4).

	mean	*variance*	*certainty*	
prior distribution	100	3	0.33	
sample data 1	100.875	6.22		
posterior distribution	100.29	2.26	0.44	(↑)
prior distribution	100.29	2.26	0.44	
sample data 2	102	6.22		
posterior distribution	100.44	1.75	0.57	(↑)
prior distribution	100.44	1.75	0.57	
sample data 3	87	1		
posterior distribution	91.89	2.86	**0.35**	**(↓)**
prior distribution	91.89	2.86	0.35	
sample data 4	92	20		
posterior distribution	91.91	2.50	0.4	(↑)

Table 6.4: Example progression using mean distance-based inference learning

As we would expect, the introduction of the sample data in the third learning step greatly increases the posterior variance. As we said before, this means a decrease in certainty about the mean. as an additional affirmation of our expectations, we see the fourth learning step again increases confidence and thus decreases variance.

Chapter 7

Evaluation

In this chapter, we will evaluate each of the three stages in the system. For the first stage, evaluation will consist of introducing the already performed evaluation of the used APG algorithm, whereas for the exercising stage it will include user tests of the system. For the final stage, we will evaluate how the proposed inference method holds up against other methods.

7.1 Preparation Stage

Because music selection using local search is not a new topic and the used implementation not unique, we will not perform an extensive evaluation of the presented constraints. Instead, we would like to refer to Vossen (2005) for an evaluation of the APG algorithm with representative constraints.

We suffice with a proof of the complexity of the IM4Sports selection problem related to the decision problem of automatic playlist generation (APG-D). In his thesis, Vossen reduces the subset sum problem to APG-D. The NP-completeness of the subset sum problem is proven by Karp (1972). Its definition: given a set of integers and an integer s, does any subset sum to exactly s?

The *file size* constraint defined in Section 4.3 relates to this problem: given a play set of songs p with file size $p_i[k]$, is there a subset of file sizes that sums to exactly max? It can be formulated as a special case of the SumGlobal constraint which is defined in Vossen's (2005) thesis. In Section 3.4, he proves that an instance of the automatic playlist generation decision problem that includes this constraint can be reduced from the subset sum problem, and thus the decision problem is NP-hard.

Even though system response time in the off-line stages is not critical, the APG algorithm should not slow down the system in a level that might annoy the user. In practice, we not require a *file size* constraint to sum to an exact max file size, which makes the problem in effect less hard. The average running time of the APG algorithm with the constraints of the off-line stage never exceeded 1 second, which hence never lays a significant burden on system response time.

7.2 Exercising Stage

Four experiments to evaluate the working of the on-line part of the system have been conducted. The first two, as briefly mentioned in Chapter 5 already, include finding optimal values for T_m and T_i. The third is a small user evaluation of the on-line stage, and the fourth concerns determining heart rate stabilisation.

The selection of new songs has not been formally tested. However, as the constraints are much simpler than those in the off-line selection, and the resulting play set consists of only one song, average running time is below 300 milliseconds which in effect never has a significant effect on system response time.

For the first two experiments, we assume a change in music tempo cannot be propagated instantaneously, and instead, no matter whether we need to match the user's step frequency or want to influence it, we need to take into account a propagation time. We assume that without it, changes in music would be too abrupt and become annoying. However, a value that is too large would make the system late in responsiveness.

For both matching and influencing equations have been presented which calculate the propagation time, $t_m(x)$ and $t_i(x)$, respectively. These equations take as input x the number of beats per minute that will be changed. The outcome is defined by a linear or a square root function multiplied by a value T_m or T_i. Which function, t_{lin} or t_{sqrt}, is used, can be specified by a Scheduler setting in the MusicMotion algorithm, yet for the experiments we use the linear function t_{lin}. Both T_m and T_i are defined in terms of Δ_{max}, the maximum change within the current song's stretching zone (*i.e.*, from the song's maximum contraction point to the song's maximum stretching point). That is, the result of $t_m(\Delta_{max})$ is T_m and $t_i(\Delta_{max})$ is T_i. So, these values represent system response time and system propagation time, respectively, when a maximum change in music tempo for the current song needs to be propagated.

7.2.1 Response Time for Pace Matching

Participants and Equipment

In the first experiment, seven healthy recreational runners between age 22 and 26 years (two females, five males) took part. The experiment set-up included a programmable treadmill (TechnoGym RunRace HC1200), the IM4Sports prototype (in pace matching mode) equipped with in-ear headphones for music playback, and a pedometer for step frequency input. During the experiment, by utilising the system's fixed song setting, we used one particular song, *Train in Vain* by The Clash (1979), which has an intrinsic tempo of 121 bpm. This song was chosen because of its repetitive character, where no big changes in musical accompaniment and dynamics are made during the progression of the song, and because of the relative unfamiliarity of our group of participants with this particular song. During the experiment, because of the fixed song setting, no limits to the song's stretching zone were imposed, allowing the music tempo to be changed freely within an approximate $[-50\%, +50\%]$ zone.

Experiment Layout

The participants were instructed to arrive in sports clothing and with running shoes, and were given the opportunity to do a warming up of about 3 minutes. The instructions to the participants after warming up made clear that they were asked to run for 470 meters, and that the speed of the treadmill would change during the experiments. At specific times, after each speed change, they were asked to give a rating to what extend they liked the last 10 seconds of music as background to the speed change in their running on a scale from 1 (very bad) to 5 (very good). If desired, the participants could use their fingers to indicate their rating. It was made clear that this rating should not be about the music itself but rather the sound quality of what they hear, and the change of music tempo with respect to running tempo. After the last condition, participants cooled down for about half a minute at a speed of $3,5$ km/h.

The treadmill was programmed with a warming up of 50 meters at 8 km/h. During this warming up, the music was started. Immediately after the warming up, without stopping

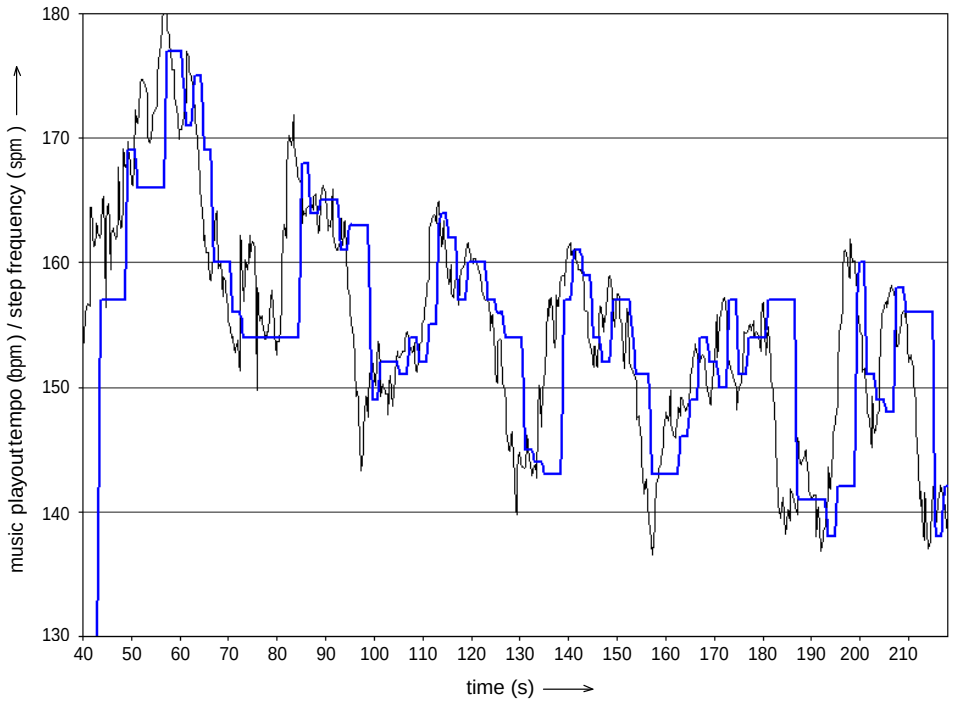


Figure 7.1: Sample pace matching graph
The thin line is the step frequency (input), the thick line the music playout tempo (output). Used values in this graph for T_m are 500, 5000, 0, 10000, 1000 and 2500, respectively.

either the music or the treadmill, the experiment begun. The treadmill was programmed to increase its speed to 12 km/h, remain at that speed while the participants run 50 meters, and decrease back to 8 km/h and stay at that speed while the participants run for 20 meters. The speed changes themselves were not programmable on this particular treadmill type, and each took about 10 seconds. There were six speed changes, and during each the parameter T_m was changed.

After the experiment, participants were asked which of the speed changes they liked best, and why.

Conditions

The testing range of T_m was determined between 0 and 10000 ms. When T_m is set to 0, a discrepancy between pace frequency and music tempo is resolved immediately by setting the music tempo to the pace frequency. The expectation is that the participants will consider the resulting changes too abrupt. On the other hand, large values for T_m result in long differences between pace frequency and music tempo, which means that the system does not match pace accurately anymore. Therefore, we expect to find an optimal value for T_m between 0 and 10000 ms.

The set of all six conditions was $\{0, 500, 1000, 2500, 5000, 10000\}$. The order in which the values were used was varied between participants to compensate for potential sequential effects following from using a fixed order for all participants.

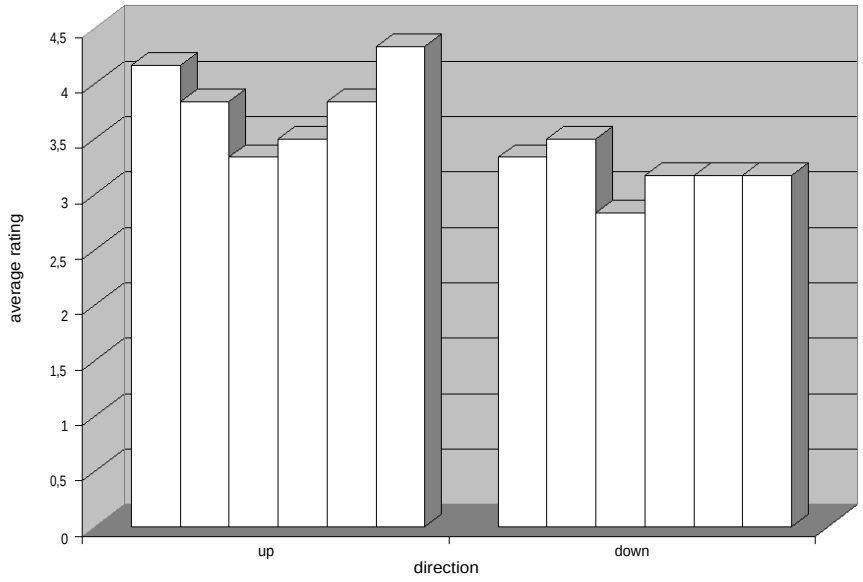


Figure 7.2: Average rating of T_m values
For each direction, the values increase from 0 to 10000 from left to right.

Results

A sample graph of one of the experiments, in which the music follows the participant's step frequency, is shown in Figure 7.1.

As Figure 7.2 shows, both short and long response times tended to be rated higher than mid response times when speeding up running. We could not arrive at statistically sound findings when slowing down running. It appeared that participants largely based their judgement on the extent to which music playback was kept in time with their steps. This synchonisation was hard to achieve since the use of a treadmill produced undesirable effects: abrupt speed changes made runners vary their step frequency and stride length in erratic ways.

Due to this preference, we devised an alternative measurement for the performance of the system. We have suggested that while pace matching should not be done instantaneously, the response time should be as small as possible. Hence, as an alternative measurement, the mean squared error of the music playout tempo graph with respect to the step frequency graph may be taken. This favoured the used of short response times in the range of 0 to 500 ms. As shown in Figure 7.1, short response times track changes in step frequency more accurately than larger response times do.

The participants rated the behaviour of the system in pace matching mode as very good, with an average rating over 3.5. They also noted that behaviour when speeding up was considerably better than when slowing down. This is consistent with their ratings; speeding up was rated with a 3.8 on average, while slowing down received an average rating of 3.2.

Participants were not able to select one of the conditions as a best experience. They indicated that this was due to their occupance with the running and listening to the music, which absorbed their concentration completely (this is also suggested by the research of Tenenbaum *et al.*, 2004).

7.2.2 Propagation Time for Pace Influencing

Immediately after, or in three cases before, the first experiment, the same group of participants took part in a second experiment. The goal of the second experiment was to determine an optimal value for T_i, the system propagation time while influencing pace.

Participants and Equipment

Like in the first experiment, seven healthy recreational runners between age 22 and 26 years (two females, five males) took part. The experiment set-up included the IM4Sports prototype in motivation mode, equipped with in-ear headphones for music playback, and a pedometer for step frequency input. During the second experiment the same song, *Train in Vain* by The Clash (1979), was used as in the first experiment.

Experiment Layout

The participants were instructed to step-in-place in time with the music, while the music tempo varied in playout tempo. There were four of such tempo variances, and during each the value for T_i was changed.

Because the motivation mode of the Scheduler utilises pace matching, a setting for T_m has to be determined as well. For the purpose of this experiment, T_m was set to 0 to minimise the influence of pace matching.

Conditions

For pace influencing, like pace matching, a value too small makes changes too abruptly and thus is hard to follow by the user. However, a value too large means the user can follow the changes in music tempo quite well, but the response time of the system overall becomes too slow. This a value must be found which is as small as possible, while not causing changes too abrupt. Therefore, the testing range of T_i was determined between 0 and 20000 ms.

The set of all four conditions was $\{0, 5000, 10000, 20000\}$. The order in which the values were used was varied between participants to compensate for potential sequential effects following from using a fixed order for all participants. The music playout tempo was changed from 110 bpm to 120 bpm, and back, for each condition.

A ‘step frequency influencing mode’ was constructed by setting the Scheduler motivation mode with a step frequency goal specified instead of a heart rate value or zone. First, a pace match between the user’s step frequency was established. Then, the music playout tempo was changed with a propagation time influenced by T_i. If the user’s step frequency had followed the music playout tempo perfectly, no additional tempo change was necessary until the user’s step frequency changed with respect to the music playout tempo. However, when the user had not reached the new music playout tempo, the process repeated and a new match had to be made before the system tried to stimulate the user to the desired music playout tempo again.

Results

It appeared, while conducting the experiment, that when T_i was set to 0, participants were completely surprised with music playout changes, which occurred instantaneously. This resulted in participants loosing track of the music tempo completely, which confirmed our earlier assumption that system propagation time can not be defined as a very small value. Hence, this condition was discarded.

The mean squared error (MSE) was used to assess the match in the remaining three conditions ($T_i \in \{5000, 10000, 20000\}$) between the user’s step frequency with respect to

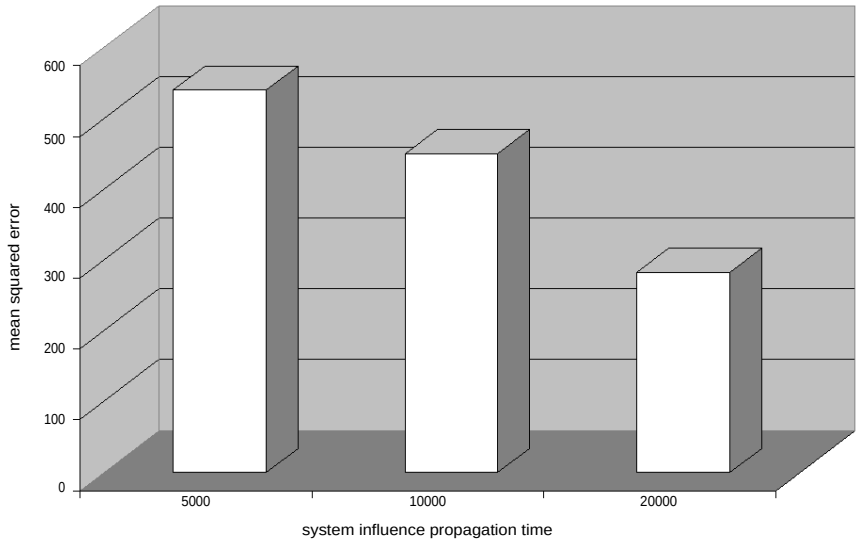


Figure 7.3: Mean squared error for different values of T_i

the music playout tempo (see Figure 7.3). It appeared that participants still had difficulties in synchronising with the music if the music tempo changes were done with propagation times of 5 and 10 seconds; they showed a 'jerky' stepping synchronisation behaviour as response to the music tempo change. As indicated by the mean squared error, 540, 450.4 and 283.7, respectively, best synchronisation performances were achieved with a propagation time of 20 seconds.

The graph suggests that larger propagation times might work even better. However, when asked, all participants rated the behaviour of the system in influencing with the maximum grade of 5. Four participants commented that they "didn't even have to think of matching the music tempo" when T_i = 20000, because "it feels so natural to follow the music tempo when changes are made gradually", as one participant commented.

7.2.3 User Evaluation

After the experiments, participants were asked for their opinions on this mode, and for their subjective comparison between the modes of pace matching and pace influencing. These opinions included questions on their exercising and music listening behaviour; the frequency of exercising, the use of music during exercise, the device used for music playback.

Participants agreed on their rating of the performance and motivational quality of the system, which they expressed as very good. All but one recreational runners would certainly consider using a motivation system such as IM4Sports would it be available and affordable. These participants preferred the motivation mode over the well-known freeform mode. The one participant who would not consider using the IM4Sports system claimed that he is able to control his heart rate better than a system could. However, even the participants that usually did not listen to music while running were highly enthusiastic about the system.

Participants were divided between liking the pace matching or the motivation mode best. Each had a different motivation for their taste. The arguments included two from participants that did not regularly use training programs and run merely for recreation

without a more specific goal than ‘staying fit’. These participants prefer the pace matching mode, which they agreed ‘motivates more than regular music does’. The other participants were positive about pace matching mode, but preferred the extra motivational impulse motivation mode could give. Two participants envisioned problems with determining what a good training program is for a specific goal, and suggested that the final system includes an advisory ‘wizard’ for recommending training programs.

7.2.4 Heart Rate Stabilisation

As we have seen in Section 5.6.3 (Equation 5.2) the heart rate needs time to adapt to a change in exercise intensity (up to 2 minutes). The system needs to wait for the heart rate to stabilise before propagation additional changes, if needed, in music tempo. Therefore, we have defined that heart rate stabilisation occurs when heart rates stay within a 1 bpm deviation of the average of a 10 second interval for the duration of that interval.

Participants and Equipment

For the third experiment, three healthy recreational runners between age 23 and 38 years (all male) took part. The experiment set-up included a programmable treadmill (TechnoGym RunRace HC1200) and the IM4Sports prototype, equipped with a heart rate monitor, and a pedometer for step frequency input. During this experiment, no music playback was used.

Experiment Layout

The participants were instructed to run with the speed of the treadmill, while the treadmill speed varied. There were three runs, and in all one such tempo variation occurred.

All runs started with a warming up of 30 seconds at 6 km/h. After the warming up, the treadmill speed increased to one of three values. This speed was retained for 3 minutes. Thereafter, the speed was decreased to 6 km/h for cooling down.

Conditions

The testing range of running speeds was determined at 9, 10 and 11 km/h.

Results

In Figure 7.4, a sample heart rate graph of a test run can be found. With the experiment, nine of such graphs were collected, and possible candidates for heart rate plateaus were identified in the printed versions of the graphs using visual approximation. This was done to test the definition given in Section 5.6.3.

Let us first realise that heart rate stabilisation is an elastic notion. Although the detection of heart rate stabilisation is an important part of the MusicMotion algorithm in motivation mode, it does not have to be exact. That is, if the heart rate is detected to be stabilised as much as 10 seconds too early or too late, that does not have a large influence on the performance of the algorithm, as we have assumed earlier that the predictions on heart rate with respect to the exercise intensity are quite good. This also appears from the previous experiment, if the correct propagation time is used.

The stabilisation detect algorithm that implements Equation 5.2 was able to find candidate plateaus that fit our definition. However, it also appeared that the found plateaus largely match the visual approximations determined by a human. The mean square error of the detection algorithm with respect to the manually identified plateaus averaged between 150 and 200, which is much lower than what we identified as a good MSE value in the previous experiment.

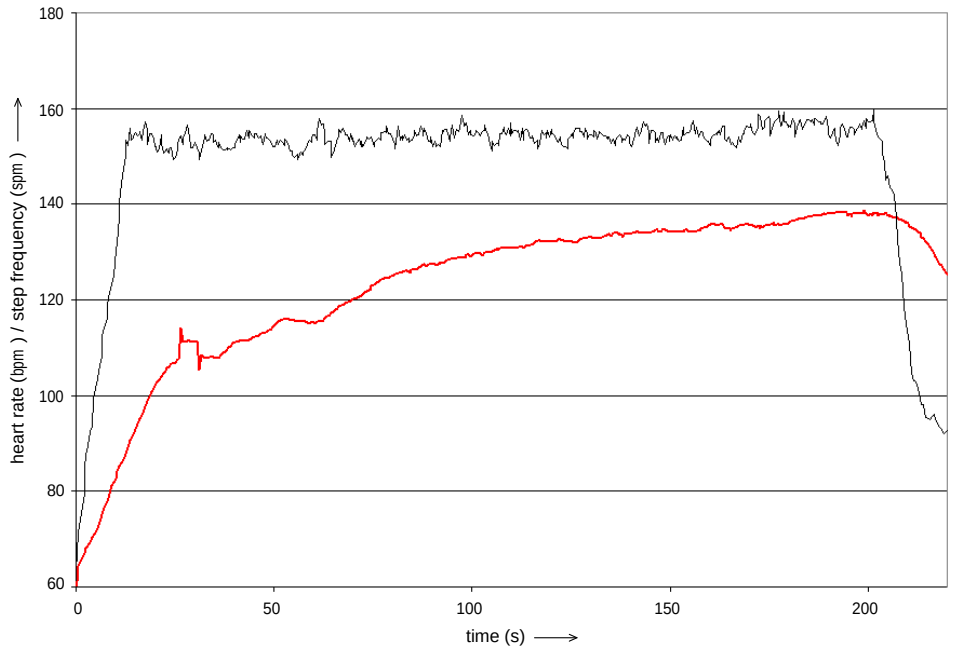


Figure 7.4: Development of heart rate over time, while a steady step frequency is maintained
The pace is indicated with a thin line (mostly above), the heart rate with a thick line (mostly below).

7.3 Feedback Stage

For testing the proposed inference method, we will formally define three aspects that we will judge the presented methods on. These are flexibility, consolidation capacity, and the method's resistance to noise. We perform the evaluation, while noting that it is perhaps not entirely fair with respect to the traditional methods, because they were not meant to be used on non-i.d.d. data. The main purpose of the evaluation is to show the additional possibilities the new method brings to the proven concept of Bayesian inference.

7.3.1 Flexibility

To test whether or not the proposed method is flexible when using large sample sets, a test set of twenty one sample distributions has been generated. The testing set T consisted of

$$T = \Big\{N(120, 10), N(121, 9.75), \ldots, N(140, 5)\Big\}.$$

These sample distributions reflect a mean music tempo that increases with 1 on every learning step, but also an increasing certainty (*i.e.*, decreasing variance of 0.25 every learning step) over the course of the personalisation.

From each distribution, we draw a hundred samples. In addition, we draw twenty extra samples from the first distribution. As sample data for the learning steps, a twenty point moving average over the samples has been taken, hence the twenty extra samples from the first distribution. In this way, 2100 sample distributions have been computed from the twenty one initial distributions. These are used as sampling data for the three different inference methods.

We define the *flexibility* of an inference method to be that it is as close as possible to

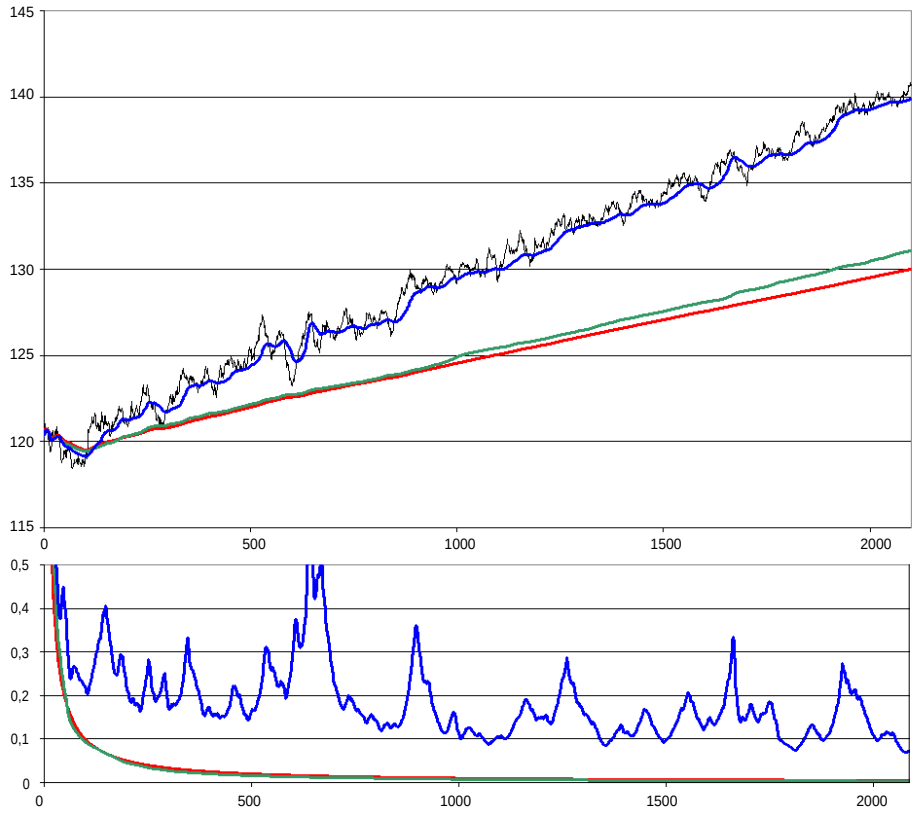


Figure 7.5: Flexibility evaluation of different inference methods. *(a)* mean *(b)* variance
(a) On the right hand side of the graph from top to bottom: sample mean (thin), mean distance-based method (thick), sample variance-based method, traditional method.
(b) Both traditional methods approach 0, mean-distance based method above.

the final distribution (*i.e.* $N(140, 5)$). We expect the Mean Distance-Based Personalisation method to be the most flexible because the variance is based on the sample data instead of decreasing with the amount of learning steps.

In the evaluation of the methods, we will refer to the inference methods using T for the traditional method, S for the sample variance-based method, and M for our proposed Mean Distance-Based Personalisation method. A graph of the resulting behaviour is shown in Figure 7.5

After 250 learning steps, the mean of the sample data is gradually increasing to 122.5. We see that both traditional inference methods T and S reach values at just under 121, whereas M is near 122. More importantly, for both traditional inference methods, the posterior variance has decreased to values as small as 0.03. The method M reflects the uncertainty in the ever-changing sample data better, with a variance of about 0.2.

After 1000 learning steps, the sample mean approaches 129.5 and the posterior mean of both traditional methods leap behind at 124.5, whereas method M is about 128.9. The posterior variance is about 0.01 versus 0.1, still. The differences gradually become clearer.

And, after completing all 2100 learning steps, we see that the sample mean has almost reached the specified 140 and the variance is close to 5. Both traditional methods, at

that moment, have been stuck at about 129 for a long time already, whereas the proposed method M approaches 140, too, rapidly. The posterior variances reflect the flexibility of the traditional methods vs. the proposed method, which are about 0.004 and 0.1 respectively.

The difference in posterior mean of about 11 beats per minute can become a major problem when selecting music based on this value, as during exercising, when the music is synchronised to the user's movements, music generally cannot be stretched more than 15%.

7.3.2 Consolidation Capacity

The *consolidation capacity* of the methods is tested by supplying all methods with a hundred learning steps with each exactly the same mean and variance, say, $m' = 100$ and $\sigma' = 10$. We first calibrate the methods with ten learning steps of $m' = 150$ and $\sigma' = 10$. The method with the best consolidation capacity is the method that is both closest to the mean of the learning steps, and has the smallest final posterior variance.

After all hundred learning steps, the traditional methods T and S perform exactly equal, with the posterior mean being about 104.5 and the posterior variance 0.09. The mean-distance based method M performs a bit worse with respect to variance, 0.11, yet the final posterior mean is 100. Due to the method being dependent on method S's posterior variance (remember $\hat{\sigma}''^2$ and v are defined in terms of σ'') it can never change the posterior variance faster than that method.

7.3.3 Resistance to Noise

The final evaluation criterion is testing how all methods perform with *noisy sample data*. We define a testing set consisting of ten learning steps with constant same sample data to calibrate the methods ($m' = 150$ and $\sigma' = 10$ like above), and introduce highly uncertain sample data with $m' = 100$ and $\sigma' = 100$.[1] We then re-calibrate the methods with an additional ten (150,10) learning steps, and again introduce the same uncertain sample data. We repeat this process ten times in total. The method that is least affected by the uncertain sample data and the most by the certain sample data is considered the the method that is the most resistant to uncertain data.

Like we expect, both S and M, which both calculate posterior variance by taking into account the sample variance are not affected much by the uncertain data. When the first uncertain sample is processed, posterior mean drops by over 4 points using method T, where both other methods are only affected 0.5 points. The posterior variance of method M is a little larger than that of the method S.

After all 110 learning steps have been completed, we see that once the posterior mean of method T dropped to about 145, it never went over 146 during testing. This once again implies the lack of flexibility in this method. Methods S and M, on the contrary, performed roughly equal with respect to posterior mean (it averages at about 149.5). The only difference is the posterior variance which shows method S is more certain on the sample set than method M (final posterior variance is 0.09 and 0.21, respectively).

7.3.4 Results

Judging from the tests outlined above we see that for the IM4Sports system, the proposed Mean Distance-Based Personalisation method is both more flexible, better at consolidation of sample data and less affected by uncertain sample data than both traditional inference methods.

The only weak point addressed in the performed tests is that the method is dependent on the sample variance-based method, and thus can never progress faster than that method. If

[1] The certainty c of this data set would be $\frac{1}{100}$, which is highly uncertain, indeed.

we want the method to consolidate its state faster, we can modify Equation 6.7 by attaching a larger weight to the positive variance change. This gives the method a preference to consolidation, whereas attaching a smaller weight to the variance change would indicate a preference towards the rejection of prior information.

We can conclude that Mean Distance-Based Personalisation fulfills our learning objective and that it can be used for personalisation in the IM4Sports system as a better method that the traditional methods. Further research should be done to investigate its use for other systems, especially since its main characteristics (fast, efficient, and easily implementable[2]) might show that it can be a light-weight alternative to wide-spread personalisation techniques such as machine learning and neural networks.

[2]In fact, most testing was done using an ordinary Excel spreadsheet with a small number of programmed macro's.

Chapter 8

Conclusion

8.1 Accomplishments

Our research goal was to develop a system that is able to stimulate, support and entertain an exerciser in an endurance sport, such as running. We have employed music to satisfy this goal by adapting it to user performance, according to an indirect method utilising step frequency distilled from literature. Requirements that specify the research goal have been formulated and in the course of this thesis, we have shown that each of them has been met by the proposed solutions.

System Design

We have defined the formal properties of a training program, and presented a complete system design, called IM4Sports, to meet the research goal. The system design consists of three stages, for each of which we have described the functionality in each of these stages.

In the first, the *preparation stage*, the system should suggest music that optimally fits a proposed training program. We have devised a method for doing so, utilising and extending pre-existing technology. Our method allows for the creation of a probability distribution of music tempo for every exercise that is in the training program, and for summing these distributions in one global distribution for the entire training program. This technology was a constraint satisfaction system based on local search, and we have supplied additional formal constraints and penalty functions that suffice the problems posed by our research goal.

For the second stage, the *exercising stage*, we have selected a way for adapting music based on tempo. We have proposed four exercise modes, each with a specific characteristic that would cope with the various user preferences that have been described in literature. We have devised methods for matching step frequency with music tempo, as well as influencing step frequency by adapting music tempo, and a method for the prediction of step frequency over time. Also, we have defined song selection in this on-line phase, which is again based on constraints that have been formally defined and which include our prediction method. Finally, we have suggested user interaction controls that would make all of the system's possibilities available to the user.

In the final stage, where *feedback* is given to the user and the system, we want the system to learn from the user performance during exercise to personalise the music selection. To this end, we have shown two inference methods that are able to update our music tempo distributions with exercise data, and noted their shortcomings with respect to non-stationary data. To counteract the problems associated with these methods, we have devised a new method, which is an upgrade of the traditional methods, that weights the certainty of the

sample data into the certainty of the posterior distribution.

System Evaluation

We have evaluated the functionality of the system in all three stages. For the first stage, we have indicated the difference in constraints with the evaluation of the used technology already performed. We formally showed that the new constraints and used constraint set imply no difference in complexity with respect to the constraints used in performance indications given in literature. Therefore, we can conclude that typical performance time for the selection of music in the IM4Sports system is less than 5 seconds. Due to the off-line nature of this stage, no additional requirements on performance are posed, in which case 5 seconds is within performance requirements.

We have implemented the entire on-line part of the system and performed three tests with it, as well as performed a user evaluation with the small group consisting of the other experiments' participants. These experiments show that the system is able to match the user's step frequency, that it can influence the user's step frequency, as well, and that it can determine whether or not the user's heart rate is stabilised or not. We have determined optimal parameter settings for both the matching and influencing functions.

The user evaluation shows that participants generally like the system, and would consider using the system for regular exercising. The participant's preference was divided between pace matching and influencing modes, which are both intrinsic parts of the system. Most important, participants indicated that they are enthusiastic about the motivational qualities of the system.

The last evaluation, of the personalisation features of the system, comprised a comparison of the traditional inference methods with the proposed method. On three different aspects, namely flexibility, consolidation capacity, and the method's resistance to noise, we have shown the proposed method works better than traditional methods for the IM4Sports system.

8.2 Quality Assessment

Determining the motivational qualities of this particular system requires assessing each of its stages with large user groups. A possible method could consist of employing the rate of perceived exertion (RPE) for both a group running with the IM4Sports system, a group running with ordinary music, and a 'control' group on different occasions. We have seen that both off-line stages are in fact preparatory with respect to the (next) on-line stage, which makes testing the system largely a matter of testing the on-line stage.

While we have not addressed the motivational quality of the current IM4Sports implementation by direct experiments, we can compare the system to results achieved in literature. Both Tenenbaum *et al.* (2004) and Boutcher & Trenske (1990) have written about the effect of music on exercisers, and both concluded that music brings down RPE for sub-maximal effort, and in addition enhances affective states and makes exercising more agreeable. Thus, exercising with music in itself can be considered motivational, at least more than exercising without music. In addition, Szabo *et al.* (1999), Kodzhaspirov *et al.* (1986) and Anshel & Marisi (1978) address the motivational qualities of music that is synchronised to the user's movements. This behaviour, displayed by the system in both pace matching and motivation modes, is said to greatly enhance motor control and lower RPE for the exercise. We can conclude that the abilities of the system to match music tempo with step frequency are likely to give it greater motivational qualities than exercising with music alone.

Remains the additional behaviour displayed in motivation mode, where the system adapts music playout tempo according to training program goals. The author is not aware

of any projects currently displaying similar behaviour, including Fraunhofer's (2004) StepMan, and has not been able to find literature on the motivational qualities of music adapted in this way. However, in the evaluation of the on-line stage we have seen that participants are tempted to follow music without focusing their attention on the music tempo. They rated the system's behaviour, without exception, as excellent, and commented that it required virtually no endeavour to follow the music tempo.

8.3 Recommendations

We will now suggest a number of improvements based on the limitations in the current implementation, and extensions that can be made to the system in its current state. In addition, we will indicate where further research can be helpful.

8.3.1 Improvements

One of the most important aspects of any control system, which IM4Sports in its on-line stage is, is its accuracy. While in the current implementation no big concerns rise with respect to the system accuracy, we can still indicate a few places where accuracy could be increased.

One of these places in the system lies in the exclusion of stride length in the current implementation. As we have seen in Section 2.3.1, stride length is an important aspect of human running behaviour. Along with step frequency, the dominant unit of measure, it determines the actual intensity of exercise. We could make the system more accurate by measuring stride length and incorporating this in the heart rate influence loop suggested in Section 2.6.3 (this comprises influencing the heart rate indirectly, by influencing step frequency as the representative of exercise intensity). Including would require searching literature or conducting experiments to understand the influence of stride length and step frequency on performed workload (that is, exercise intensity). Measurements can be done straightforward in the current system, as the pace sensor used is able to measure covered distance, and thus stride length, in parallel to step frequency.

Another place where system accuracy can be increased is in the propagation of music tempo changes. Currently, these are calculated using a linear or a square root function which determines the function values. The end result is the time over which the music tempo is changed. However, an implementation limitation in the communication between the IM4Sports system and the XMMS music player allows music tempo to only change linearly. An improvement could be to adapt music playout tempo according to a square root function. This would make adaptations start faster at first and progress slower further on, when the exerciser is aware that music tempo is changed. Further research should point out whether this behaviour is significantly better than linear propagation.

An additional improvement, which also would improve system accuracy, is the inclusion of the fatigue curve. This improvement has been suggested already in Section 4.2.1, where a possible spot for its incorporation in the system has been suggested. In addition to the determination of music tempo based on training programs, the fatigue curve could assist in predicting heart rates during exercise.

8.3.2 Extensions

User Preference in Music Selection

Currently, the system does not take into account user preferences when selecting music content for the portable player, nor when playing back music during exercise. Because a broad range of algorithms for the selection of user-preferred music already exists, we

decided to focus on other, more distinctive aspects of the system for the present research. However, it is likely that motivational increases can be gained by playing back much-liked songs when a boost in exercise morale is required. To allow this playback behaviour, a mix of user favourites and not-so-well-known songs should be available on the portable player. It would also require additional constraints and knowledge for the music selection.

Training Program Recommendation

In the current implementation, training programs are considered user input. A user needs to know exactly what goals she wants to reach and, more importantly, how to achieve those goals by putting together a well-balanced training program. Obviously, not all exercisers are able to determine what a good training program is, let alone create one. As a possible extension, the system could suggest a training program based on the desired goals, which could be very broad, such as 'loose weight' or 'get more fit'. In this way, the system could act as a virtual coach with respect to exercise preparation.

Virtual Fitness Coach

The feedback capabilities of the IM4Sports system, which utilise logging all data that can be collected using the attached sensors and system decisions, can be used to further improve this 'virtual coach' behaviour. For example, the system could praise the user for sticking to the training program, or help the user to determine what went wrong by asking specific questions. Such a virtual coach could use exercise data to influence decisions on a new training program recommendation. For example, if a user is known to have problems with sticking to running exercises that exceed 3 km, the system could learn to split these exercises in shorter runs divided by stretching exercises.

Internet Community

One last suggestion on possible extensions is linking the system to an virtual community on the Internet. This potentially provides the user with many opportunities: compare training programs and goals, compare and discuss performance, discuss training-related topics in a forum, or chat with professionals that can help the user overcome particular training problems. Such a community could enhance user bonding and commitment to the system and hence the user's training program. From a system perspective, an online database with user data could allow for the automatic determination of user similarity. From that, the automatic comparison of similar music collections to suggest new songs and artists to the user based on similar user's preferences, the comparison of training programs and the comparison of personalisation states could enhance the system by making it more interactive, more enjoying and more surprising to its users.

8.3.3 Further Research

The most important aspect of the system that requires further investigation is its motivational quality. As we described above, we have firm grounds to believe that the system will be able to motivate people while exercising, but a practical test will be required to address problems with the current implementation. We have performed a small evaluation of the on-line stage of the system which shows no major problems, but in a comprehensive user evaluation of the entire system in all its three stages might indicate additional focus points for improvements.

A test of the entire system will require the design of a user interface and interaction process. Suggestions for both in the most important stages in the system have been made in this thesis. For example, we have indicated the functionality of the buttons in the on-line

stage, and described the user interaction process of the preparation stage. Further research, however, will have to point out how people prefer to interact with an interactive music system during exercise.

To successfully evaluate the off-line preparation stage, a large number of gender/age group music tempo distributions for a number of exercise types need to be obtained. This requires extensive research with hundreds of participants, but can greatly enhance user satisfaction by limiting the calibration of the system in its simplest way to the input of gender and age.

For the personalisation capabilities of the system we have suggested an inference method called Mean Distance-Based Personalisation. We have indicated its advantages over traditional inference methods for our application. These need to be evaluated properly, with real running data, after gender/age group music tempo distributions are available. It would be interesting to learn whether the method is able to provide advantages in other applications, as well, by applying and comparing it in other situations. Further research could also compare this method with other learning algorithms such as machine learning, neural networks, and evolutionary algorithms.

8.4 Commercial Opportunities

As the research for the IM4Sports project was conducted at Philips Research, Eindhoven, we will place the project in a commercial context and suggest opportunities for its further development towards consumer applications.

In the introduction to this thesis, we have mentioned the trend of people exercising more. We have also suggested that people need help with finding motivation to continue exercising. In this chapter, we have concluded that, based on literature, the IM4Sports system presented in this thesis can be a valuable help for giving motivation to exercisers, and hence, a solution to the problem presented in the introduction. For the system, if it were implemented according to the presented specifications and developed into an end-user system, this means a large potential user base, which could make a commercial roll-out an interesting opportunity.

Philips, as a high-tech company, has recently formed an alliance with Nike to bring their respective digital and athletic expertise towards sport. They have described their goal as to "develop innovative technology product solutions which create a richer, more motivating environment for physical activity". Currently, the line-up includes digital audio players which all have excellent anti-shock properties and can be attached to the body during exercise. Only one, the MP3Run (Nike-Philips, 2004), offers capabilities that go beyond playing music like all other players do. It features a speed and distance sensor which can register time and tempo during training sessions, and can communicate this info using speech synthesis when prompted by a button press. Surprisingly, music playback is considered a separate function and is not adapted to the training data at all. Hence, music playback does not differentiate these players from competitors such as Apple's iPod shuffle and other flash-based audio players.

If not implemented in full, parts of the IM4Sports system could enhance players like the MP3Run and thus provide an extra unique selling point for these audio players. For example, the inclusion of pace match and cruise control modes would help exercisers and provide an interesting feature not currently matched by any competitor. In a next stage, when the market is ready, the pace influence mode can then be introduced to get further ahead of the competition, feature-wise. The reason to not include this feature from the start is because it requires the development of a new and easily attached sensor for heart rate measurement, while the step frequency sensor is available in an end-user version right now.

Another application for the IM4Sports system is another division of Philips, which recently has gotten a large amount of media attention: Customer Health & Well-being (CHW).

Philips' mission statement has seen a shift in focus towards personal health applications and Philips consequently has become a large player in health. One of the applications for an IM4Sports-based system could be rehabilitation.

Rehabilitation is a form of medical treatment, often physical therapy, aimed at the restoration of lost capabilities. These are often caused by disease or a serious accident. For example, after a brain stroke, certain physical functions may be lost or may be incomplete. A patient would need to learn to walk or bike again, or might need to build up muscular tissue after being forced to stay in bed for extended periods.

The treatment consists of daily exercise routines, which largely consist of repetition. There is a large common ground between these exercises and repetitive solo sports such as running and cycling, as these exercises often combine parts of these solo sports, quite often are done alone (after initial clinical treatment) and are repetitive in nature. Hence, rehabilitation exercises could potentially be made much easier with proper music background, in a fashion similar to those of repetitive solo sports.

Appendix A

Terminology

Beat pattern See: ↪**rhythmical structure**

Bpm See: ↪**beats per minute**

Beats per minute The tempo of the song is often notated in the amount of beats per minute. The beat in this definition is the same that is defined in the ↪**meter**. Often there are four beats per measure ($\frac{4}{4}$).

Crescendo A gradual increase in music volume.

Decrescendo See: ↪**diminuendo**

Diminuendo A gradual decrease in music volume.

Genre Musical genres are categories which contain music which share a certain style or which have certain elements in common (*Musical genre*, from Wikipedia, n.d.).

Harmony Harmony is the art of using pitch simultaneity (or chords, actual or implied) in music. It is sometimes referred to as the "vertical" aspect of music, with ↪**melody** being the "horizontal" aspect - a harmony are several melodic lines or motifs being played at one particular moment in time.

Heart rate The times the heart beats per minute.

Heart rate capacity The theoretical maximum on the times the heart beats per minute.

Heart rate zone The percentage of the maximum heart rate (See: ↪**heart rate capacity**) at which you exercise in order to meet certain goals. For example, you exercise at 50-60% to burn fat.

Key The key of a musical piece is the *tonal center* of that piece, and is defined by the ↪**scale** the music uses. From music theory we know that certain keys are *compatible* with each other.

Measure A measure is defined in music as *distance in time*. The measure divides a rhythmic pattern in equal parts of 2, 3, 4 or more beats, and thus establishes a ↪**beat pattern**. The ↪**meter** defines the length and advancement of the measures.

Melody A melody is a series of linear events or a succession of tones (pitches) in music. In this succession, phrases and motifs are created. These are usually repeated within songs in various forms.

Meter How many beats go in each measure of a song. Most often a song is 'in the meter of' $\frac{4}{4}$ beats per measure, in which each measure has four beats and each beat has the length of a quarter note. Other common meters include $\frac{2}{2}$, $\frac{4}{8}$ and $\frac{3}{4}$ (the meter of a waltz). The meter defines the length of the ↪**beat pattern** of the music. A *duple* meter means that the music features 2 or 4 beats per measure, *triple* meter that it has three beats per measure.

Pace The step frequency in times per time unit ($^{n}/_{t}$), that is,how many times a minute the runner's feet hit the ground.

Rhythmical structure The rhythmical structure of a song is the pattern in which accents in the music are structured. When a song has a ↪**meter** of $\frac{4}{4}$, and the accents are 'on the beat', there's often a strong accent on the first and a less strong accent on the third beat (to make things more complex, this is because $\frac{4}{4}$ is a so-called 'composed' meter, which means it is really in a $\frac{2}{2} + \frac{2}{2}$ meter). The accent structure is sometimes denoted as '• · • ·' or, more frequently, 'X · x ·' (the large X denotes the strongest accent). When the accent is on the 'off-beat' (second and fourth beat, of which the second beat is most important), the beat pattern is '· X · x'.

Scale A scale contains all the notes of a certain ↪**key**. The *tonica* or base of the scale is the first and defining note of the scale. An example is the c major scale, which contains the notes c d e f g a b. The e minor scale contains e f♯ g a b c d.

Speed The distance per time unit ($^{d}/_{t}$).

Staccato The term *staccato* denotes that a particular note to which it is assigned should be played shorter. Nevertheless, the tempo of the music remains the same, so the remainder of the note is silent. Hence, the notes are played loose from each other, implied by the meaning of the Italian word from which it is derived, *staccare*, meaning 'to pull loose'.

Tempo The tempo is the speed in which the music moves. In other words, tempo is the speed with which the ↪**measure**s follow each other. In classical music, tempo was notated with Italian words and categorised in five main groups: the very slow tempi (*e.g. largo*, *adagio*), the slow tempi (*e.g. andante*), the moderately fast tempi (*e.g. allegretto*, *moderato*), the fast tempi (*e.g. allegro*) and the very fast tempi (*e.g. allegro vivace*, *presto*, *veloce*).

All these tempo names only give an approximation of the real tempo. The invention of the metronome (a device that can be set to tick a certain times per minute) gave musicians a possibility to find the exact tempo. Tempo is measured in ↪**beats per minute**).

Tenuto The term *tenuto* denotes that a particular note should be played a little stronger than the surrounding notes. It is often combined with a ↪**staccato** sign if the articulation should be joined by playing the note shorter.

Appendix B

Requirements Overview

Functional Requirements

FR 1. *The aim of the system is to use music to provide a more stimulating and more entertaining running experience*

FR 2. *The system takes a given training program as input*

FR 3. *The system should select music that fits the users (projected) performance*

FR 4. *The system should select songs from the database that best fit the characteristics of the different exercises in the fitness program, and suggest these as the new contents of the Player*

FR 5. *The system should adapt the played back music to the user performance*

FR 6. *The system should allow easy interaction during exercising, more specifically:*

- pause song: *because the system may be used in complex environments, such as alongside roads with heavy traffic, a pause button is important - in the example figure any part of the screen can be used to pause the music;*
- play/repeat song: *depending on the mode the system is in, the button changes to allow the user to indicate she wants to play a song (in a mode that does not automatically start playing music) or to indicate the song should be repeated (in modes when the song is automatically changed when finished);*
- skip song: *the user wants to skip to another (similar) song;*
- stop exercise: *the user wants to stop exercising;*
- skip exercise: *the user wants to skip the remainder of the current exercise, regardless of whether the exercise is finished or not;*
- mode: *the user can select the playback/exercise mode of the device.*

FR 7. *The sensors used by the system should be easily attachable, removable and usable, and their number should not exceed two*

FR 8. *The collected data should be used to personalise the system's music selection*

Data Requirements

DR 1. *The training program should fit a specific profile that allows for standardised processing. This profile includes:*

- Kind of exercise *(Warm Up, Stretch, Walk/Run/Jog, Recover, Cool Down, etc);*
- Duration *(in seconds or in distance);*
- Intensity *(in heart rate percentage).*

DR 2. *The following meta data on the music should be available to the system:*

- *Song title*
- *Artist*
- *Album*
- *Song duration*
- *Genre*
- *Tempo*
- *Meter*

DR 3. *The effects of a change in a music parameter on the user performance should be available*

DR 4. *During operation, the system should collect data on:*

- *songs played;*
- *music characteristics adapted by the system;*
- *sensory data;*
- *user interaction.*

Appendix C

Training Program XML Schema

```
<xs:element name="trainingprogram">
 <xs:complexType>
  <xs:sequence>
   <xs:element maxOccurs="unbounded" name="exercise">
    <xs:complexType>
     <xs:attribute name="type" type="xs:string" use="required" />
     <xs:sequence>
      <xs:element name="duration">
       <xs:complexType>
        <xs:attribute name="type" type="xs:string" use="required" />
        <xs:attribute name="val" type="xs:unsignedShort" use="optional" />
       </xs:complexType>
      </xs:element>
      <xs:element minOccurs="0" name="parameter">
       <xs:complexType>
        <xs:attribute name="val" type="xs:unsignedShort" use="required" />
       </xs:complexType>
      </xs:element>
      <xs:element minOccurs="0" name="intensity">
       <xs:complexType>
        <xs:attribute name="type" type="xs:string" use="required" />
        <xs:attribute name="min" type="xs:unsignedByte" use="optional" />
        <xs:attribute name="max" type="xs:unsignedByte" use="optional" />
        <xs:attribute name="val" type="xs:unsignedByte" use="optional" />
       </xs:complexType>
      </xs:element>
      <xs:element name="afterExercise">
       <xs:complexType>
        <xs:attribute name="val" type="xs:string" use="optional" />
       </xs:complexType>
      </xs:element>
     </xs:sequence>
    </xs:complexType>
   </xs:element>
  </xs:sequence>
 </xs:complexType>
</xs:element>
```

References

Aarts, E. & J. Lenstra (eds.) (2003). *Local search in combinatorial optimization*. Princeton, NJ: Princeton University Press, 2nd ed.

Alghoniemy, M. & A. H. Tewfik (2000a). Personalized music distribution. In: *Proceedings of the 2000 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP00)*. Istanbul, Turkey: IEEE.

Alghoniemy, M. & A. H. Tewfik (2000b). User-defined music sequence retrieval. In: *Proceedings of the eighth ACM International Multimedia Conference, Part II (IMC00)*. Los Angeles, CA: ACM.

Alghoniemy, M. & A. H. Tewfik (2001). A network flow model for playlist generation. In: *Proceedings of the 2001 IEEE International Conference on Multimedia and Expo (ICME01)*. Tokyo, Japan: IEEE.

Alm, M. & P. Alm (2004). *XMMS: X Multimedia System v1.2.10*. On the Internet: http://www.xmms.org (accessed 6 June 2005).

Anshel, M. H. & D. Q. Marisi (1978). Effect of music and rhythm on physical performance. *Research Quarterly*, vol. 49, pp. 109–113.

Aucouturier, J.-J. & F. Pachet (2002). Scaling up music playlist generation. In: *Proceedings of the 2002 IEEE International Conference on Multimedia and Expo (ICME02)*. Lausanne, CH: IEEE.

Bayes, T. (1763). Essay towards solving a problem in the docrine of chances. *Philosophical Transactions of the Royal Society of London*, vol. 53, pp. 370–418.

Berger, F. (2002). *sndstretch_xmms plugin v0.7*. On the Internet: http://www.geocities.com/harpin_floh/sndstretch_page.html (accessed 7 June 2005).

Bitner, J. & E. M. Reingold (1975). Backtrack programming techniques. *Communications of the ACM*, vol. 26, pp. 832–843.

Bliss, C. I. (1967). Statistics in biology. *Statistical Methods for Research in the Natural Sciences*, vol. 1, pp. 108–111.

Bonants, R. J. M. (2003). *Influence of immersion and biofeedback on intrinsic motivation in the sport setting*. Technical Note 2003/00876, Philips Research, Eindhoven, NL.

Boutcher, S. H. & M. Trenske (1990). The effects of sensory deprivation and music on perceived exertion and affect during exercise. *Journal of Sport and Exercise Psychology*, vol. 12, pp. 167–176.

CBS (n.d.). *Centraal Bureau voor de Statistiek (StatLine application)*. On the Internet: http://statline.cbs.nl/ (accessed 18 Jun 2005).

Chambers, J. M., W. S. Cleveland, B. Kleiner & P. A. Tukey (1983). *Graphical Methods for Data Analysis*. Duxbury Press.

Clash, The (1979). Train in Vain. *London Calling*. Composed by Jones/Strummer, 3:09. Released on Epic, #36328.

Clayton, M., R. Sager & U. Will (2003). In time with the music: The concept of entrainment and its significance for ethnomusicology. In: *ESEM03 CounterPoint*, vol. 1.

Codognet, P. & D. Diaz (2001). Yet another local search method for constraint solving. In: *Proceedings of the 2001 International Symposium on Stochastic Algorithms (ISSA01)*, pp. 73–90. London, UK: Springer Verlag.

Dibben, N. (2002). *The role of bodily sensations in emotional experience with music*. Sheffield, UK: University of Sheffield.

Drake, C. & M. Botte (1993). Tempo sensitivity in auditory sequences: Evidence for a multiple-look model. *Perception & Psychophysics*, vol. 54, pp. 277–286.

Dunn, K. (2004). Music and the reduction of post-operative pain. *Nursing Standard art&science*, vol. 18(36), pp. 33–39.

Edwards, S. (1996). *Heart Zone Training*. Cincinnati, OH: Adams Media Corporation.

Eiben, A. E. & J. E. Smith (2003). *Introduction to Evolutionary Computing*. Berlin, D: Springer Science + Business Media.

Eitan, Z. & R. Y. Granot (2004). Musical parameters and images of motion. In: R. Parncutt, A. Kessler & F. Zimmer (eds.) *Proceedings of the 2004 Conference on Interdisciplinary Musicology CIM04*. Graz, AT: ESCOM.

Fonseca, R., V. Almeida, M. Crovella & B. Abrahao (2003). On the intrinsic locality properties of web reference streams. In: *Proceedings of the 22nd Conference on Computer Communications (INFOCOM03)*, 1, pp. 448–458. Edmonton, Canada: IEEE Communications Society.

Fraunhofer (2004). *StepMan*. On the Internet: http://www.igd-r.fraunhofer.de/IGD/Abteilungen/AR3/Projekte_AR3/stepman_AR3/index_html_en (accessed 18 Jun 2005).

French, J. C. & D. B. Hauver (2001). Flycasting: On the fly broadcasting. In: *DELOS Workshop: Personalisation and Recommender Systems in Digital Libraries*, pp. 89–93.

Friberg, A. & J. Sundberg (1999). Does music performance allude to locomotion? A model of final ritardandi derived from measurements of stopping runners. *Journal of the Acoustical Society of America*, vol. 105(3), pp. 1469–1484.

Gabriel, D. A., J. R. Basford & K.-N. An (2000). Neural adaptations to fatigue: implications for muscle strength and training. *Medicine and Science in Sports and Exercise*, vol. 33(8), pp. 1354–1360.

Gardner, W. G. (1999). *3D audio and acoustic environment modeling*. Arlington, MA: Wave Arts.

Gaver, W. W. (1986). Auditory icons: using sound in computer interfaces. *Human-Computer Interaction*, vol. 2(2), pp. 167–177.

Gaver, W. W. & R. B. Smith (1995). Auditory icons in large-scale collaborative environments. In: R. Baecker, J. Grudin, W. A. S. Buxton & S. Greenberg (eds.) *Readings in Human-Computer Interaction: Toward the year 2000*, pp. 564–569. San Francisco, CA: Morgan Kaufmann, 2nd ed.

Glover, F. (1986). Future paths for integer programming and links to artificial intelligence. *Computers and Operations Research*, vol. 13, pp. 533–549.

Glover, F. (1989). Tabu search - part I. *ORSA Journal on Computing*, vol. 1, pp. 190–206.

Goris, A. H. C. & J. J. van Herk (2003). *Tri-axial activity monitor for the measurement of human energy expenditure*. Technical Note 2003/00772, Philips Research, Eindhoven, NL.

Gosset, W. S. (1908). The probable error of a mean (under pseudonym "Student"). *Biometrika*, vol. 6(1), pp. 1–25.

Holland, J. D. (1975). *Adaptation in natural and artificial systems*. Ann Arbor, MI: University of Michigan Press.

Jiang, C., M. Shao, Y. Li & P. Jia (2002). Accelerating clustering methods through fractal based analysis. In: *Fractal workshop at the 8th ACM International Conference on Knowledge Discovery & Data Mining (KDD02)*. Edmonton, Canada.

Karageorghis, C. I. & P. C. Terry (1997). The phychophysical effects of music in sport and exercise: A review. *Journal of Sport Behaviour*, vol. 20, pp. 54–68.

Karageorghis, C. I., P. C. Terry & A. M. Lane (1999). Development and initial validation of an instrument to assess the motivational qualities of music in exercise and sport: The Brunell Music Rating Inventory. *Journal of Sports Sciences*, vol. 17, pp. 713–724.

Karp, R. M. (1972). *Complexity of Computer Computations*, chap. Reducibility among combinatorial problems, pp. 85–103. New York, NY: Plenum Press.

Kirkpatrick, S., C. D. Gelatt & M. Vecchi (1983). Optimization by simulated annealing. *Science*, vol. 220(4598), pp. 671–680.

Kodzhaspirov, Y., Y. M. Zaitsev & S. M. Kosarev (1986). The application of functional music in the training sessions of weightlifters. *Soviet Sports Review*, pp. 39–42.

Kullback, S. & R. A. Leibler (1951). On information and sufficiency. *Annals of Mathematical Statistics*, vol. 22(1), pp. 79–86.

Kumar, V. (1992). Algorithms for constraint satisfaction problems: a survey. *AI Magazine*, vol. 13, pp. 32–44.

Logan, B. (2002). Content-based playlist generation: explanatory experiments. In: *Proceedings of the 2002 International Conference on Music Information Retrieval (ISMIR02)*.

Makhoul, J. & A. El-Jaroudi (1986). Time-scale modification in medium to low rate speech coding. In: *Proceedings of the 1986 International Conference on Acoustics, Speech and Signal Processing (ICASSP86)*, pp. 1705–1708. IEEE.

Mertesdorf, F. L. (1994). Cycle exercising in time with music. *Perceptual and Motor Skills*, vol. 78, pp. 1123–1141.

Moulines, E. & F. Charpentier (1990). Pitch synchronous waveform processing techniques for text-to-speech synthesis using diphones. *Speech Communication*, vol. 9(5/6), pp. 453–467.

Nike-Philips (2004). *MP3Run*. On the Internet: http://www.nike-philips.com/product_details.jsp?lang=en&product=MP3RUNpsa260 (accessed 17 Jul 2005).

Novacheck, T. F. (1998). The biomechanics of running. *Gait and Posture*, vol. 7, pp. 77–95.

Pachet, F., P. Roy & D. Cazaly (1999). A combinatorial approach to content-based music selection. In: *Proceedings of the 1999 IEEE Interaction Conference on Multimedia Computing and Systems (ICMCS99)*, vol. 1, pp. 457–462. Firenze, IT: IEEE.

Papadimitriou, C. H. & K. Steiglitz (1998). *Combinatorial optimization: algorithms and complexity*. Mineola, NY: Dover Publications.

Parviainen, O. (2002). *SoundTouch sound processing library*. On the Internet: http://www.iki.fi/oparviai/soundtouch/ (accessed 27 Apr 2005).

Pauws, S. C. (2002). *Playlist generation by constraint satisfaction*. Technical Note 2002/00282, Philips Research, Eindhoven, NL.

Povel, D.-J. (1981). The internal representation of simple temporal patters. *Journal of Experimental Psychology: Human Perception and Performance*, vol. 7, pp. 591–596.

Rejeski, W. J. (1985). Perceived exertion: an active or passive process? *Journal of Sport Psychology*, vol. 7, pp. 371–378.

Roucos, S. & A. M. Wilgus (1985). High quality time-scale modification for speech. In: *Proceedings of the 1985 International Conference on Acoustics, Speech and Signal Processing (ICASSP85)*, pp. 493–496. IEEE.

Runners World (n.d.). *Runners World Magazine*. On the Internet: http://www.runnersworld.com (accessed 26 May 2005).

Schobben, D. & R. M. Aarts (2002). Three-dimensional headphone sound reproduction based on active noise cancellation. In: *Convention paper 5713, AES 133th Convention*. Los Angeles, CA.

Shove, P. & B. H. Repp (1995). Musical motion and performance: theoretical and empirical perspectives. In: J. Rink (ed.) *The practice of performance: studies in musical interpretation*, pp. 55–83. Cambridge, UK: Cambridge University Press.

Stallman, R. & G. J. Sussman (1977). Forward reasoning and dependency directed backtracking. *Artificial Intelligence*, vol. 9, pp. 135–196.

Stigler, S. M. (1982). Thomas Bayes' Bayesian inference. *Journal of the Royal Statistical Society, Series A*, vol. 145, pp. 250–258.

Szabo, A., A. Small & M. Leigh (1999). The effects of slow and fast-paced classical music on progressive cycling to voluntary physical exhaustion. *Journal of Sports Medicine and Physical Fitness*, vol. 103, pp. 588–601.

Tanaka, H., K. D. Monahan & D. R. Seals (2001). Age-predicted maximal heart rate revisited. *Journal of the American College of Cardiology*, vol. 37, pp. 153–156.

Tenenbaum, G., G. Fogarty, E. Stewart, N. Calcagnini, B. Kirker, G. Thorn & S. Christensen (1999). Perceived discomfort in running: Scale development and theoretical considerations. *Journal of Sports Sciences*, vol. 17, pp. 183–196.

Tenenbaum, G., R. Lidor, N. Lavyan, K. Morrow, S. Tonnel, A. Gershgoren, J. Meis & M. Johnson (2004). The effect of music type on running perseverance and coping with effort sensations. *Psychology of Sport and Exercise*, vol. 5, pp. 89–109.

Thompson, W. F., E. G. Schellenberg & G. Husain (2001). Arousal, mood and the Mozart effect. *Psychological Science*, vol. 12(3), pp. 248–251.

Tsang, E. P. K. (1993). *Foundations of Constraint Satisfaction*. London, UK: Academic Press.

Verhelst, W. & M. Roelands (1993). An overlap-add technique based on waveform similarity (WSOLA) for high quality time-scale modification of speech. In: *Proceedings of the 1993 International Conference on Acoustics, Speech and Signal Processing (ICASSP93)*, pp. 554–558. IEEE.

Vignoli, F. (2003). *MMa: architectural framework and algorithms to support multiple input modalities*. Technical Note 2003/00663, Philips Research, Eindhoven, NL.

Vossen, M. P. H. (2005). *Local search for automatic playlist generation*. Master's thesis, Eindhoven University of Technology.

Wenzel, E. M., M. Arruda, D. J. Kistler & F. L. Wightman (1993). Localization using nonindividualized head-related transfer functions. *Journal of the Acoustical Society of America*, vol. 94(1), pp. 111–123.

Wightman, F. L. & D. J. Kistler (1989). Headphone simulation of free-field listening (I: Stimulus synthesis, II: Psychophysical validation). *Journal of the Acoustical Society of America*, vol. 85(2), pp. 858–878.

Wijnalda, G. L., S. C. Pauws, F. Vignoli & H. Stuckenschmidt (2005). A personalized music system for motivation in sport performance. *IEEE Pervasive Computing*, vol. 4(3), pp. 26–32.

Wikipedia (n.d.). *Wikipedia: The Free Encyclopedia*. On the Internet: http://www.wikipedia.org/ (accessed 5 Oct 2004).

Wilmore, J. H. & D. L. Costill (2004). *Physiology of Sport and Exercise*. Champaign, IL: Human Kinetics Publishers, 3rd ed.

Winkler, R. L. & W. L. Hays (1975). *Statistics: Probability, Inference and Decision*. New York, NY: Hol, Rinehart and Winston, 2nd ed.

Wooldridge, M. J. & N. R. Jennings (eds.) (1995). *Intelligent Agents*, vol. 890 of *Lecture Notes in Artificial Intelligence*. Berlin, D: Springer Verlag.

Zölzer, U. (ed.) (2002). *DAFX: Digital audio effects*. Chichester, UK: John Wiley & Sons.

Index