MOTION PATTERN RECOGNITION FOR INTERACTIVE DANCE

by

Henning Pohl

TECHNISCHE UNIVERSITÄT DARMSTADT

Supervisor: Prof. Dr. Max Mühlhäuser

Reader: Aristotelis Hadjakos

SUBMITTED IN PARTIAL FULFILLMENT OF THE
REQUIREMENTS FOR THE DEGREE OF
MASTER OF SCIENCE

AT

TECHNISCHE UNIVERSITÄT DARMSTADT
DARMSTADT, GERMANY
MARCH, 2010

Contents

List of Figures

Abstract

In this thesis a method to detect patterns in dance movements is described. Such patterns can be used in the context of interactive dance systems to allow dancers to influence computational systems with their body movements. These dance interactions can provide additional information to people shaping an experience and allow them to incorporate their audience into their performance.

For the detection of motion patterns, two different methods were designed to detect motion similarity. Using either dynamic time warping or feature vector comparison, the distance between two given movements can be computed. A custom threshold clustering algorithm is used for subsequent unsupervised classification of movements.

For the evaluation of the presented method, a wearable sensor system was assembled out of available components. Additionally, an evaluation environment was created for the evaluation process itself. To quantify the accuracy of the classification, a custom label space mapping was designed to allow comparison of sequences with disparate label sets.

Based on an evaluation of the system with four participants, this thesis's method is shown to be able to distinguish dissimilar movements. The capability to acceptably classify longer durations of movement activity is shown as well.

German Abstract

Diese Arbeit stellt eine neue Methode vor, die es ermöglicht Muster in Tänzen zu erkennen. Solche Muster sind im Kontext von interaktiven Tanzsystemen von Interesse und erlauben es Tänzern mit ihren Körperbewegungen algorithmische Zeichenprozesse zu beeinflussen. Vor allem in offenen Echtzeit-Gestaltungsprozessen können solche Muster wirken. Sie erlauben es ferner, den Rezipienten einer solchen Arbeit eben jene auch direkt zu beeinflussen.

Es werden zwei verschiedene Verfahren vorgestellt, die in der Lage sind Ähnlichkeiten von Bewegungen zu erkennen: Dynamic-Time-Warping und der Vergleich von Merkmalsvektoren. Mit einer Grenzwert-basierten Clusteranalyse findet dann eine unüberwachte Klassifikation von Bewegungen statt.

Um die beschriebene Methode zu evaluieren, wurde ein tragbares Sensorsystem aus vorhandenen Komponenten zusammengefügt, das die Bewegungen des Trägers erfassen kann. Des Weiteren wurden mehrere Anwendungen für die Aufnahme und Verarbeitung dieser Bewegungen implementiert. Da erkannte und vorgegebene Bewegungen nicht im gleichen Bezeichnerraum koexistieren, wurde eine Abbildung entworfen, um diese zu vereinen und so einen Vergleich zu ermöglichen.

Mit Hilfe von vier Testern wurde das System evaluiert. Es zeigte sich, dass die vorgestellten Algorithmen gut in der Lage sind, Bewegungen zu differenzieren. Auch bei längeren Aufnahmedauern war noch eine akzeptable Klassifikation möglich.

Acknowledgements

To **Prof. Dr. Max Mühlhäuser**, who kindly agreed to supervise this thesis.

To **Aristotelis Hadjakos**, who provided invaluable guidance and help over the course of this thesis. This thesis has been greatly inspired by his work and vision. I sincerely hope that some of this work translates back into something useful for his doctoral thesis. I also need to thank him for his soldering, assembly debugging and general electronics skills, without which I would have been completely lost at times.

To **Clemens Voigt** for proofreading this thesis on such a tight schedule. He also volunteered as study participant, for which I am greatly thankful.

To **Katharina Teske** and **Katrin Aldorf** for participating in the user study as well.

And finally, to my family for ongoing support and encouragement.

Ehrenwörtliche Erklärung

Hiermit erkläre ich, daß ich die vorliegende Arbeit selbständig und ohne Benutzung anderer als der angegebenen Hilfsmittel angefertigt habe. Alle Stellen, die wörtlich oder sinngemäß aus veröffentlichten oder nicht veröffentlichten Schriften entnommen wurden, sind als solche kenntlich gemacht. Die Arbeit hat in gleicher oder ähnlicher Form noch keiner anderen Prüfungsbehörde vorgelegen.

Frankfurt am Main, den 24.3.2010 ________________________________

Henning Pohl

Definition of Musical Terms

To help in understanding the upcoming chapters, a short definition of musical terms used throughout the thesis is given here.

Beat A beat describes the constant dominant pulse of a musical piece. The notion of the beat itself does not include any hierarchy or ordering. A beat is merely the basic background on which more complex notions can be based. Often the tempo of a musical piece is defined as beats-per-minute (bpm).

Measure Beats can be combined into groups, called measures. For example, a $^3/_4$ measure combines three beats into a group where each beat has a note value of one fourth (a quarter note).

Meter The meter defines which beats in a measure are to be stressed. Measure and meter are often used interchangeably. For example, a $^4/_4$ measure is assumed to have the first note as primary accent and the third note as secondary accent. However, this does not always need to be the case. The meter ultimately defines a hierarchy for the beats and is a significant factor in our perception of structure and tempo.

Rhythm A rhythm denotes the general placement of sounds in time. Thus the sequence of notes and pauses and their durations make up the rhythm of a musical piece. Note that this does not refer to any note's pitch as rhythm is purely concerned with the alteration of audible and inaudible time spans (notes and pauses).

Chapter 1

Introduction

In the end, our aim is to examine an ongoing human effort: the desire to integrate the most basic expressions of the soul with the most complex creations of the mind.

— *Mark Coniglio and Dawn Stoppiello*[1]

The title of this thesis already hints at what is to be expected: some patterns are to be discovered. Patterns for something called “interactive dance”. This raises the question of what constitutes interactive dance. A question, that will be answered later on in this chapter. One might also wonder what patterns are and what kind of patterns are to be looked for. In general, a pattern is a recurring event / object / motion / structure or any other aspect perceivable as repetitious. Searching for patterns thus is an action primarily concerned with the *now* and the *past*, relating current observations to ones already passed, possibly extrapolating into the future as well. Here, the focus more specifically is on motion patterns, motion patterns related to dance to be precise. With that direction given, musical aspects will inevitably play a role as well, due to the tight connection of dance and music.

In this chapter the two main aspect of the thesis are described in more detail.

[1]`http://www.troikaranch.org`

1.1 Motion patterns

Motion patterns, in line with the above given definition, are recurring motions. If asked, most people could probably identify motions satisfying that requirement. A group of friends e.g., might have their own secret handshake, an identifying motion of that group. Similarly, a person giving a presentation sometimes might not be able to control his body language, resulting in repeated anxiety gestures like fiddling with clothing. While we are constantly in motion, some parts seem to stand out in a way that reminds us of previous motions. How a repeated motion is recognized, however, is not an easy question.

There is no singular feature that denotes motion similarity. A similarity over a given time span is what makes motions appear similar to us. Not all parts are equally important in determining that similarity though. For example, a motion pattern will have an introduction, a body and a conclusion. The body will be spanning the longest amount of time, with the introduction and conclusion varying in strength and length. This could, for instance, be a circular hand motion. When performed repeatedly the pattern is readily apparent. But when did it start? At some point the structure of the hand motion became visible, but the transition from arbitrary movement to pattern is a smooth one. It is only during the execution of a movement that we can recognize whether it belongs to a group of previous ones.

As can be seen, motion patterns are quite fuzzy. When they start, how long they last and when they end is not perfectly clear and up to some ambiguity. Also, we do not require a perfect repeat of a motion for a motion pattern. In the above given circular motion for example, even larger variations would not keep a human observer from detecting a level of similarity. A system working with motion patterns thus has to be robust to variations in the data and able to make an informed decision on the level of similarity needed for a match. Later on in this thesis, some valid restrictions for dance motions are discussed. Additionally, algorithms that are able to detect patters are presented as well.

1.2 Interactive Dance

The term *interactive dance* also requires an explanation. One could argue that all dancing is interactive. Certainly all non-solo dances contain interpersonal interaction. In fact, dancing is inherently connected to a social context. But dance is not only social, but also musical and interaction could be defined in that context as well. A dancer is certainly acting according to the music and the music might also change in response to dancing (e.g., a DJ will adapt his music to the response he is currently getting from the crowd). In general, interactivity is a feedback system, where actions and reactions are intertwined and dependent on each other.

For the purpose of this thesis, a more specific notion of interactivity is necessary. An *interactive dance* shall be a dance that elicits reactions in a computational system. Basically, dance acts as an input device, similar to a mouse or a keyboard. For dance to be accessible to a computer, an interpretative process is needed, that transforms dance movements into semiotic signs (here based on Charles Sanders Peirce's definition). In later chapters such transformations are described in detail.

With dance movements being accessible to the computer, they can become part in any given sign process inside the machine (machine semiosis). *How* they are interpreted is dependent on the intended application domain. In this section three such domains are outlined.

1.2.1 Interactive Dance in Art

Interactive dance, or movement in general, is primarily a part of installations, performance art and contemporary dance. Often interactive dance is used to give dancers a certain level of control over the music playing or the stage lighting. In other instances, interactive dance is used in generative art where it is used, for example, to control a visualization or the audio output. Using dance and movements as input to an art piece can be seen as making the body an extension of a physical object. Thus, aspects of mechanization in art certainly play a role here, with human motions and expressions being reduced to signals in an algorithmic process. Another recurring artistic theme is the feedback loop, where e.g., actions influence the visuals which in turn influence the actions being performed. The piece of art becomes transitive and in its complete form only exists in presence of the interacting person.

An example for the usage of interactive dance is the *Digital Dance Project*, by the Danish Institute of Electroacoustic Music (DIEM). Here, a system was developed to track body features of dancers. *Movement Study* and *Sisters* are two pieces, that use that capability to influence the music playing [58]. In *Sisters*, for instance, two dancers represent two opposing sides of the artist (henve also incorporating above mentioned notions of the body as a machine) and are able to control sound playback and filters.

One artist, who has been working with interactive dance since early on, is Mark Coniglio[2]. In 1989 he created *MidiDancer*, a system he used in various projects to give dancers interactive control over digital media. In collaboration with other artists through Troika Ranch, he designes multimedia dance theater performances.

Not specifically targeting dance, but movement in general, Todd Winkler builds installations reacting to audience movements. For example, in his *Light Around the Edges* installation, a camera records people moving through a space which adapts to their movements. He notes, that:

> *As sensing technology matures, artists will be compelled to conceive of work where physical interaction, computer interaction, and social interaction are vital to creating new forms of expression and experience.*
>
> — *Winkler [68]*

In addition to installations, he also sees motion input as an exciting prospect for interactive music systems [66]. Thus, "*extending the performer's power of expression*" over existing musical instruments.

[2]`http://www.troikaranch.org`

While the use of motion and dance for artistic purposes is an exciting field on its own, this aspect is not the focus of this thesis. However, some more examples of artistic use of this technology are given in Chapter 2, when detailing sensor choices.

1.2.2 Interactive Dance in Video Games

Using dance as input for video games was pioneered by the *Dance Dance Revolution*[3] series, which started in 1998. In this series, players need to step on sectors of a floor mat according to a given choreography. The sectors basically act as buttons, being triggered by a player's feet. *Dance Dance Revolution* is available in arcade and home versions, and has inspired numerous clones like the *In The Groove*[4] and *Pump it Up*[5] series. Systems, such as those from the *iDANCE* series[6], do scale to up to 32 players and are also used in a fitness context. Some games, like *Dance Factory*[7], use a camera to also analyze hand motions, i.e., asking players to point in a given direction at a prescribed time. Other games, like *Just Dance*[8], eschew the dance mat and use input devices like the *Wii Remote* (the controller for Nintendo's *Wii* console) to detect dance movements. The *ParaParaParadise*[9] game uses motion detector sensors to observe upper body movements, which are the most important component of the *Para Para* dance style.

Dance based video games are a subset of music video games, which are quite popular in general. For further information on the genre and *Dance Dance Revolution* in particular, an article by Smith provides a good starting point [60]. An overview of input devices used in music video games and descriptions of the games themselves can be found in an article by Blaine [12]. Dance in video games is generally almost exclusively used as a means of rhythmic input. Certain steps have to be performed at a given time and points are awarded or deducted based on doing so correctly. In most games dance input is just another form

[3]`http://www.ddronlinecommunity.com`
[4]`http://inthegroovegame.com`
[5]`http://www.piugame.com`
[6]`http://www.positivegaming.com/index.php?id=34`
[7]`http://www.codemasters.com/dancefactory`
[8]`http://justdancegame.us.ubi.com`
[9]Only an emulator is available at `http://www.paraparaparadise.net`

of button presses. As in the case of interactive dance in art, the application of interactive dance was not a target scenario for this thesis. However, this does not preclude aspects of motion pattern recognition being appliable in that context as well.

1.2.3 Interactive Dance in Clubs

Dance clubs are another setting that could profit from interactive dance. One open question in that context is how to enable more audience interaction and interactive dance is one possible way to do so. Before exploring such interactions though, a step back and an examination of the setting are in order. In a dance club, there will be at least one DJ (disc jockey) mixing the music. In addition one or more VJs (visual jockeys) might work on accompanying visuals for the club. Any number of other groups, like Go-Go dancers, aroma jockeys (people mixing fragrances, at an event, to add to the overall experience) or people operating foam machines, might also have a stake in the overall experience. In general, all those stakeholders in the dance club experience already are well trained in interacting with the crowd. A DJ, for example, needs to be able to determine appropriate musical choices and change his performance according to the crowd at hand.

To determine how DJs already interact with their audiences and what they felt could be done to improve said interaction, Gates et al. did a user study, interviewing eleven DJs [25]. DJs were generally confident in their ability to read cues in the audience and react accordingly. They would change their mix and adapt tempo, mood and intensity of the music, to fit an audience's energy level. They might also speak to the audience directly or communicate via body language to further engage the audience. The primary way DJs determine an audience's state was identified as visual. Thus, this mostly includes body language like gestures, winks, smiles, nods, intensity levels and overall audience movement (e.g., to or from the dance floor). DJs also listen to audible cues such as laughter, yelling and cheering. All of those cues require a certain closeness and the DJs in the study pointed out a dislike for settings where they were removed from the audience to some extent. Lack of a good field of view from the DJ booth, for example, already reduces their ability to react to the audience. Regarding systems designed to help them in with their work, DJs were critical of automation, fearing a loss of artistic freedom.

While Gates et al. only interviewed DJs, most of the points brought up will also be relevant to VJs. While concerned with visuals instead of audio, they also work on shaping the club experience and are dependent on cues from the audience to determine appropriate visual choices. However, VJs often enjoy a heightened amount of possibilities when it comes to interacting with the audience. For examples, visuals might incorporate live video feeds and work has been undertaken to enable audience members to stream live video from their mobile handsets to be used as well [20]. A more in-depth introduction to VJing and an overview on the history of VJing can e.g., be found in articles of Annet Dekker [15] [16].

Similar to the work done by Gates et al., nine VJs were interviewed by Engström et al., to determine their thoughts on their practice [20]. One aspect here was how their work relates to the work done by DJs. VJs would often note, how their visual building blocks also feature pulsating elements, that they would then bring in sync with the music playing. Matching the rhythm of those two elements was seen as important for the overall quality of a performance. Like DJs, VJs also did not want any systems automating too much of their workflow, especially in regards to artistic control. VJs generally felt, that they had less interaction with the audience than DJs, noting that often people do not even associate them to the visuals. However, VJs do look for visual cues in the audience to determine the quality of their performance. One VJ for example pointed out that "*[I]f someone is dancing looking at a screen, then we understand that it contributes to the atmosphere*" [20]. VJs also use the DJ as a point of reference, e.g. anticipating music changes when they see a DJ grabbing a new record.

Drawing from their study of DJs, Gates et al. came up with a list of ten human-computer interaction (HCI) recommendations for systems catering to that user group[25]:

Quality of Information DJs are already good at determining the level of excitement in their audience. Any new system would need to provide information that is more detailed and/or more precise.

Audience Information While DJs can read emotional cues in their audience, they are unable to ascertain their musical preferences and background. Software assisting in collecting such background information could be valuable, especially in more diverse settings.

Issues of Voting Giving the audience too much control over what music to play is viewed critically. It disempowers DJs to some extent and hampers with their role of introducing new music. It also could lead to a more chaotic and less well designed experience. Indirect audience interaction is seen by Gates et al. as a more promising choice.

Speed of Musical Changes Instantaneous changes might disrupt the flow of the music and thus negatively alter the overall experience. Careful planning has to be employed for smooth musical arcs over several songs. Short sound events like scratching on the other hand are still a valid option.

Composition The overall composition of a set matters a lot. Software intended to perform music selection on its own should take into account the expert knowledge of a human DJ.

Measurement of Excitement Limiting the measurement of audience satisfaction to people on the dance floor is only half of the big picture. As DJs cater to the whole club, any measure of success should take into account all people and their individual characteristics.

Biofeedback While biofeedback mechanisms are an exciting new way to measure audience properties, HCI designers should take into account if and how a meaningful translation to information for the DJ can be accomplished. As the first point states above, this information needs to be better than cues already available to DJs.

History While DJs would like to know about the tracks played by other DJs before them at the same event, they are often hesitant to share that information themselves. Software compiling such information needs to be aware of possible privacy constraints and should be open to improvisation on the DJs side.

Cognitive Load DJs already have to coordinate a number of tasks when performing. Any new technology should not add to their cognitive load. The goal should be to make menial tasks more efficient, as to allow DJs to focus on the creative aspects of their line of work.

Usage of Information When new information is made available to DJs they should be given full freedom to interpret it in their own way. Automating the creative process or predetermining ways to work with such information would restrict their artistic independence and their possibilities to set themselves apart from their peers.

In general, they argue for systems that provide additional information to, but do not automate decisions for the DJ. This somewhat clashes with the desire to not increase cognitive load, though. However, this can be seen as a delicate balance, where any additional cognitive load has to be connected to at least an equal gain in informational efficiency. Gates et al.'s recommendations also apply to VJs, who face similar challenges. In fact, getting feedback from the audience is even more important for VJs, who might directly incorporate such feedback into their visuals. Especially if generative art is mixed into the video output, complex mappings could be employed to make direct visual use of audience data.

As described above, DJs and VJs rely on subtle cues from the audience in their effort to facilitate an exciting sensory experience for them. Interactive dance can be utilized in this context to provide additional cues about the audience's state. As per the HCI recommendations by Gates et al., this additional feedback should be informative and hence should not be intended for automation purposes. Interactive dance seems most promising in aiding the VJ, as it could easily be adapted for generative art or as a general input in the visual performance process.

In this context, Ulyate and Bianciardi early on explored possible uses of interactive dance in clubs [64]. At the 1998 ACM SIGGRAPH convention, they set up a club where participants could influence real-time graphics via dancing. Among the prototypes explored were:

- A zone with light beams that, when broken by a participant, triggered musical phrases.
- A visualization of a dancer's body heat and motions using the images from an IR camera.
- An array of pads that, when stepped on, trigger musical phrases and accompanying projections.
- A platform, where one could control the music with one's shadow.
- A zone with proximity sensors, reacting to hip movement, where percussion sounds and computer graphics are controllable.

Some of those interactions only work for a single participant, while others work for pairs or larger groups. When testing their interactive dance club, Ulyate and Bianciardi found that those devices that allowed for more freedom of movement, were yielding more satisfying dance interactions than devices such as buttons or pads. In light of the HCI aspects, the role of an "experience jockey" was devised. This person controls the overall experience and changes mappings according to the current situation. With this new role, no additional stress is put on DJs or VJs.

Also interested in dance club interactions, Feldmeier did a user study on their viability and quality [22]. For this purpose, he designed small, disposable sensors to be distributed to dance club patrons at low cost. When one of those sensors detected an acceleration above a given threshold, a signal was send to a base station. From the aggregated signal data the system could determine a set of features. Among them the number of signals received in a two second time frame, the time passed since the last signal arrived or the peak frequency of the data. Ultimately, an estimate for the overall tempo and energy of the crowd was computed. Based on that estimate, the music and lighting were changed. Several user tests on groups of up to 200 participants were done. While some tests were used to determine the technical performance of the system, others had questionnaires to gather participant opinions as well. Evaluating those questionnaires, an overall positive user response to the system was indicated. Participants enjoyed the experience itself and felt that the music and lighting adapted well to their motions.

This thesis is primarily interested in the application of interactive dance in a club environment, too. However, while dance clubs are the usage scenario in mind, this thesis is not a user study in that setting. Instead, the parameters of a dance club setting influenced design decisions made and features evaluated.

1.3 Overview

In this thesis, a method is described to detect motion patterns in dance movements. Motion patterns are one feature potentially useful in a club context to aid DJs and VJs with their work. This thesis details the steps taken to achieve acceptable motion pattern detection for this scenario.

Examination of dance movements requires non-standard means of input. As an inherently multi-dimensional signal, dance movements are far more complex than e.g., mouse movements. Multiple ways to process dance input are presented in this thesis, and evaluated as to their feasibility for dance pattern detection. Specifically, a set of requirements are defined for the usage scenario at hand.

In order to test the proposed system, a prototype was needed. For this purpose, already available components were combined to assemble a wearable system prototype. Some changes were made to pre-existing components to, e.g., provide mounting options better suited to dance movements. The prototype is loosely-coupled to clients by utilizing a publish/subscribe communication model. This functionality builds on top of an existing middleware solution.

For the detection of dance patterns, two different ways to compute motion similarity were build and are presented. As both approaches work on a block level, a segmentation of the sensor data is performed using beat information from the audio track. On top of a block's raw sensor data, additional features are defined and can be used instead or together with the raw data itself. Thus, blocks are first transformed to select desired signal components, before being passed on to dynamic time warping (DTW) and feature vector comparison (FVC) modules that are able to compute pairwise similarity values. On top of this capability, an unsupervised clustering algorithm was designed that classifies motion data blocks according to a threshold criterion, taking into account the block similarities.

As to be able to evaluate the proposed algorithms, an evaluation environment was created. One part of this environment is a recording application. This component, e.g., makes sure to provide comprehensive information to participants and ensures recording integrity. When evaluating the quality of a recording, the class labels of the recorded and given sequences have to be compared. A best-fit label space mapping is presented, which is able to do so. Challenges in interactive dance evaluation are described as well.

The results are presented and it is shown that the proposed methods both can be used to distinguish disparate movements. It is also shown that both methods provide acceptable motion pattern detection even for longer sequences. As the parameter space of both methods is explored, the most important features to use are determined. Performance is also compared by movement and it is shown which movements are easier to classify than others.

Finally, the results of this thesis are put in relation to the target scenario again. The feasibility of dance pattern recognition for interactive dance experiences in clubs is described with respect to previous studies.

Chapter 2

Related Work

While the focus of this work is in dance pattern recognition, a number of other areas overlap with this one. Which sensors to use when it comes to motion, is one relevant aspect for example. More general pattern recognition ideas also influence the work at hand and some are presented here as well.

2.1 Sensors for Interactive Dance

There are a number of different approaches to make dance, or body movement in general, available to the computer. In this section, a number of different approaches are presented. Sensor systems processing dance movements were primarily chosen, but a number of other ones that could easily be adapted to dance motions were included as well. When applicable, the usage scenarios are described as well, providing an insight into a number of interactive dance applications.

2.1.1 Sensing Floors

In sensing floor systems the primary focus is on lower body motion. They can be coupled with other systems for more complete sensor data. Sensing floor systems mostly vary in respect to resolution, modularity, size and response time.

One of the first sensing floor system was Johnstone's *PodoBoard* [34]. The *PodoBoard* is made up of a 36 by 40 grid of aluminum tiles. Metal plates were added to the toes and heels of shoes to be used with it. The four plates are pulsed sequentially resulting in an electrical signal when in contact with the *PodoBoard* and thus allow to determine their position. The shoes are also outfitted with a piezoelectric film that allows to determine the velocity of a shoe when touching the board. The board sends out Musical Instrument Digital Interface (MIDI) messages in response to activity. Foot contact is mapped to *note-on* and *note-off* messages, while moving the foot while in contact with the board is translated into pitchwheel messages. Johnstone built the board for Alain LaMontagne, a clackage[1] performer, who used the board to generate sounds during a performance.

For their "*Magic Carpet*" [46] Paradiso et al. laid out a grid of wires on a carpet to detect motion. The insulation of the wires was made out of a piezoelectric material and thus each cable produced a voltage if pressure was applied to it. The voltage in each wire is checked at 60Hz and if a new peak is detected a *note-on* MIDI event is send out. In this MIDI event the note to be played is determined by the wire number and the velocity of the note by the pressure applied at the wire. If the pressure falls below a given threshold, a *note-off* event is send out. Two doppler radars are used to detect upper body movement activity and direction. Using those inputs a "*relaxing soundscape*" is generated by the system. A user's movement on the carpet is mapped to sound changes on three different levels.

Around the same time, Griffith and Fernström developed a similar system based on optical proximity sensors. Their "*LiteFoot*" [27] system contains 44×44 sensors in an area of 1.76 m^2 and can work in two modes. It either detects reflections or shadows. In the first mode, light is emitted by the sensors and reflected by objects on the floor. In the second one the shadows of objects, illuminated from elsewhere, are detected on the floor. Two software modules were written for the floor. The first one is used to map foot positions on the floor to notes in a MIDI signal. The entire floor can be set to one instrument or divided up into different areas for different instruments. A second module generates visual output dependent on the movement on the floor. All steps are plotted and fade out over time. Griffith and Fernström combined these effects in both an Irish dance and free form modern dance performance.

Building on their previous work, Paradiso and Fernström together with McElligott et al. tried to combine the advantages of both systems. The "*Z-Tiles*" [42] system was a more modular solution and provided better linear response and spatial resolution. Each tile consists of 20 so-called *prexels* – pressure elements.

[1]A seated form of step dance

They are arranged in a pattern that guarantees neighboring tiles interlock. Each tile has four connectors, one for each potential neighbor. The *prexels* consist of a pressure sensitive polymer that changes in conductivity according to the force applied. While initial force response is fast the recovery time is significantly lower. The final network of tiles is self-organizing and can be reconfigured while live. *Z-Tiles* were used in several projects, with one of them being the mapping of location and pressure to MIDI notes. They also experimented with systems sensitive to weight distribution on the floor. For example, this was used for audio signal processing and navigation in a virtual world.

Srinivasan et al. also built a modular system based on pressure sensitive polymers [61]. They significantly increased the resolution of the sensor grid with each 62×53 cm mat holding 2016 pressure sensors. Each mat of sensors is connected to the host system via ethernet, which, using switches, allows for the total number of mats to currently scale up to 128 allowing for sensing floors of up to 26.7 m^2. The entire floor is gapless as the mats are build to allow for a certain amount of overlap. Each mat independently samples at 30Hz and then sends the data to the host machine, which collects the samples from all mats.

2.1.2 Sensors in Shoes

Where the previous section described sensing floors, another option is to move the sensors from the floor to the shoes. Where Johnstone already augmented shoes to work with his sensing floor [34], there are some approaches that solely rely on the shoes and omit the floor. Applications are often somewhat similar to sensing floors, but the spatial restrictions of floors do not apply.

One example of shoes with sensors comes from Paradiso and Hu, who described a concept system where a range of sensors are added to dancing shoes in order to capture dance movements [45]. In the front of each shoe two piezoelectric pads measure the pressure applied by the big and small toes, while another one takes measurements at the heel. A force-sensitive sensor strip is placed in the middle of the shoe, along its axis to measure bending. In the heel of the shoe an accelerometer and a electronic compass are added to measure the orientation of the shoe. For more accurate results they had planed to add a gyroscope as well. An embedded controller and a small transmitter were mounted on the back of the shoe. They sample at 100 Hz and wirelessly transmit the signal to a base station. On top of this per-shoe data, the position of each shoe in the space is determined via either a laser rangefinder or a sonar system. When a sonar is used the shoes will also transmit the measured time delay back to the base station. Finally electric field sensing is used to determine the elevation of the shoes above the stage.

One [47] and two [48] years later Paradiso and Hu together with Hsiao built working prototypes of the above mentioned system. The first one was used in an interactive dance performance, where, for example, samples were triggered and tempo and volume were controlled by dance movements. In the second paper they additionally describe a more complex mapping where music was synthesized based on the movements performed and the dancer's position on the stage.

Kim et al. created the *Shadow Dancer* system that aims to generate a visual dance partner to accompany a human dancer [35]. For this purpose they added pressure sensors to the tip and the heel of step dancing shoes. As they specifically target step dance, limiting themselves to pressure sensor works in their scenario. Four patterns are defined as building blocks for step dance. To find the best matching one, the step intervals are calculated, which provides a sufficiently well-performing classifier. For each pattern a number of video clips were pre-recorded and pre-processed to show only dancing silhouettes (hence the name *Shadow Dancer*). When a pattern is danced, the system projects a matching pattern on a screen to dance along the human dancer.

Where other shoes presented here contain a range of sensors, Fujimoto et al. only added one three-axis accelerometer to the tip of each shoe [24]. For their project they were specifically interested in B-boying (breakdance). When B-boying, the dynamic adaption to the music and the personal expression in the dance movements is of special importance. Taking that into account, Fujimoto et al. designed their system to be customizable for this scenario. Dancers train the system with a set of motion definitions. They also wrote script to define what sound is to be played in response to which motion. The scripting language allows for states (varying mappings) and varying degrees of mapping complexity. This ranges from simple one-to-one mappings of motion to sound, via predicate mappings (movements have to be danced in a certain order) to aggregate mappings (a movement has to be danced x times in a row to trigger a sound). Unfortunately, Fujimoto et al. do not specify how they match motions with each other. They state that they are using a dynamic programming algorithm, but do not state which one. A live performance of the system with two dancers in a dance battle was shown in Kobe on December 2007.

Apart from interactive dance, sensor shoes can also be used in sports. Hockman et al., for example, built a system that changes the tempo of a track according to the pace of a runner [32]. For this, a two-axis accelerometer is fitted into a running shoe and used to detect the step frequency. This detection is performed by smoothing the signal with a median filter, thresholding it and auto-correlating it with itself. This already results in a set of potential beat periods, which is further weighted with a Rayleigh distribution that emphasizes step frequencies more likely to occur in a running human. The detected step frequency is subsequently used for time-scaling the audio track. To enable this, the audio track is beat-tracked in a pre-processing step and this information is used to compute the proper scaling factor.

2.1.3 Camera-Based Systems

Where previously described systems relied on extensive electronics and wiring, camera based systems offer an easy way to capture motions. Naturally, cameras have been used in a number of systems and several platforms exist to build upon. Four different works are presented here that process video data in different ways and are designed for an interactive dance context.

Winkler used the Very Nervous System (VNS) of David Rokeby[2] for an interactive dance installation [67]. The VNS requires one or two cameras to monitor a scene. Users can define a grid on top of the video feed, thus defining a set of regions for the camera, or provide arbitrary shapes for those regions. The VNS then

[2]See `http://www.wired.com/wired/archive/3.03/rokeby.html` for an interview with Rokeby on the Very Nervous System (last accessed on March 23rd 2010)

determines the amount of motion in each region. This is done by computing the difference between the current frame and previous ones. All video is converted to 6 bit gray-scale before that difference is computed. There are two modes available when using the VNS: either the difference to recent frames is calculated or the difference to a specifically set frame. In the first mode current motion is detected, while the second mode can detect things not present in the reference frame. If motion is detected an activity value for each region is computed based on the amount of difference detected. Large amount of change in a region yields a high value and vice-versa. Note that not only motion but lighting changes or color changes influence the VNS. For his project, Winkler defined a 4×4 grid and used the first VNS mode to detect motion. Several mappings were explored with the simplest ones being one-to-one mappings of region activity to MIDI notes played. More complex interactions included the mapping of regions to a set of notes, triggered one after the other, changing the tempo based on region activity or using the regions as on-off switches for a virtual sequencer.

Where Winkler used a simple camera, Bevilacqua et al. decided to use a whole 3D motion capture system [10]. Dancers had to wear a number of reflective markers that are tracked by eight video cameras. Using triangulation, the 3D positions of all markers can then be computed for every frame. Ultimately, the animation data is supposed to control the sound. To simplify the needed mapping, Bevilacqua et al. first reduce the size of the raw data by extracting a set of relevant motion features. They do not specify the final mapping but define classes of parameters from which to choose according to each project's needs. One might e.g., be interested in the positions of points in the scene or distances and angles between points on the body. Other relevant features are the velocity and acceleration of body parts. Exploring potential concrete mappings is left by Bevilacqua et al. for future work. Among the potential ones mentioned is a mapping that triggers sounds based on a dancers position on the stage or changes in direction of body parts. They mentioned "*promising results*" for applying dance-controlled filters to pre-existing audio tracks playing. A last area pointed out is sound synthesis based on dance movements.

Similar to the work of Bevliacqua et al. mentioned above, Castellano et al. used cameras to determine a set motion features [13]. Based on the EyesWeb platform, they evaluated two measures: the quantity of motion and the contraction index. The quantity of motion relates to how energetic the movement is, i.e., more vivid movements yield higher values. The contraction index, on the other hand, is computed based on the space occupied by a dancer. Castellano et al. also determine the current *emotion* via a mapping from the two measures to an emotion space. For their visualization the silhouette of the dancer is projected in a color dependent on the recognized emotion. Additionally they use *pDM*[3] to interactively control the expressivity of the music played. Here, the quantity of motion maps to the tempo of the music, the contraction index controls the volume and the detected emotion is used for articulation.

An interesting approach is taken by Guedes, who used video data to detect the frequency of movements [28]. For each frame he computed the difference to its predecessor as the sum of luminance changes. Given sufficient background contrast, the more movement occurs, the higher the frame-by-frame difference will be. In a first step the resulting activity-over-time signal is transformed into the frequency domain. Guedes used a set of 150 band-pass filters (instead of a fast Fourier transform (FFT) because of concerns for speed), to detect frequencies in the range of 0.5 Hz to half of the sampling frequency. In a next step the prominent frequency of the signal is extracted. For this purpose the auto-correlation of a 1 Hz pulse train in the frequency domain with the signals frequency domain representation is used. This is done to emphasize periodic movements and dampen singular frequencies. For one project he then used the dominant frequency to control the playback speed of a piece of music, i.e., faster dancing would also result in higher musical tempo. He also described another usage scenario, where frequency and amplitude data of multiple harmonics can be utilized to synthesize rhythms. From 2003 to 2005 his system was used for two interactive dance performances and one interactive installation.

[3]A set of pure-data scripts found at `http://www.speech.kth.se/music/performance/download/` (last accessed on March 20th 2010)

2.1.4 Wearable Sensors

While sensor shoes already constituted a form of wearable sensor, this category is reserved for systems concerned not just with the feet. Wearable sensors are a large category with a diverse set of sensors being used. With accelerometers, gyroscopes, bend sensors, magnetometers and proximity sensors among the possible choices, very specifically adapted systems can and have been be designed. They all have in common, that they are somehow mounted on the body and e.g., strapped to a participant using velcro. In contrast to floors or cameras, sensor data transmission is an aspect that needs to be covered as well, when sensors are worn. This section describes a number of such systems and their application to interactive dance.

Evaluating Bluetooth as choice for communication with wearable sensor system was the focus of the work of Hromin et al. [33]. For their prototype they built a wearable module that holds a microcontroller, a Bluetooth transceiver and one to five sensors. Sensors tested include accelerometers, flex sensors, temperature sensors, photoresistors and pressure sensors. When running, the wearable module periodically sends IP packets to a Bluetooth access point, which relays them to the host PC via ethernet. To keep the overhead low Hromin et al. decided to only implement the lowest layer of the Bluetooth host stack, which they deemed sufficient for the transmission of raw sensor values. Evaluating their choice of Bluetooth Hormin et al. were generally satisfied with the low power consumption and low costs (compared to 802.11b) of Bluetooth. However, the size limits on Bluetooth piconets (up to seven devices) were deemed to be too restrictive in the context of dance clubs. Hromin et al. also believed that Bluetooth is "*needlessly complex*" which, in combination with their other points, led them to the suggestion that a new wireless protocol for this kind of scenario might be needed.

Three years later Aylward and Paradiso started out to build a similar system [3]. They, however, did implement their own wireless protocol for this purpose. Each sensor has a three-axis accelerometer, a three-axis gyroscope, a proximity sensor (node-to-node) and a radio. The radio is capable of up to 1 Mbps, which is sufficient for 25 sensors being polled at 100 Hz. During their testing they found the range to be at about 50 feet. A custom base station controls the communication on the network (broadcast poll with simple time division multiple access (TDMA) response scheme) and connects to the host PC via USB. With their custom system they were able to achieve uptimes of up to four hours with 289 mAh of power supplied. With the ability to record dance movements of multiple persons simultaneously, they set out to evaluate their system in this kind of scenario. Three ballet dancers received two sensors each (worn at the right wrist and right ankle) and their ballet lesson was recorded. Aylward and Paradiso used to sensor data to determine if all dancers were in sync or whether one of them was '*off*'. While recording, the cross-covariance of each sensor pair is computed (within a one second window). This allows for some lag (the window length) and provides a measure of similarity. In their evaluation, they found that this approach can accurately determine which dancers lag behind. In addition to synchronicity, the level of activity and the separation of gestures are also relevant. As one approach Alyward and Paradiso proposed to use the variance envelope for this purpose. They also note that comparison of overall signal variance to individual axis variance can be useful in determining the dominant sensor.

While Alyward and Paradiso, at the end of their above mentioned paper, discussed possible musical mappings for their sensors, they only explore these a year later [4]. They equipped five dancers with two sensors (an iterated design of their previous sensor system) each. After recording a performance they set out to test the viability of their system in generating music according to that dancing. They came up with a system where instruments could either be controlled by group movements, by solo movements or combinations of those two. For example, the loudness of a violin was controlled by the global activity level of all dancers, while the pitch of the same violin was controlled by how in sync the dancers were as a group. The flute sounds were triggered by that dancer, who deviated from the rest of the group. Latency is mentioned as one problem of this approach, due to the one second window size. The musical mapping was also not yet evaluated in a live performance setting, where the latency would be even higher.

Where Hromin et al. used Bluetooth for communication and noted that 802.11 was comparably less feasible, Fléty set out to evaluate just that [23]. His *WiSe Box* did not have any build-in sensors but was intended as a generic communication module. A Wise Box contains a microcontroller and a WiFi card. One sensor can be connected to the *Wise Box's* analog-to-digital converter (ADC) and is sampled at 16 bit. Sensor values are packaged in Open Sound Control (OSC) messages before transmission and OSC is also used to send commands to the Wise Box. When battery testing the *Wise Box* would run 125 minutes on 900 mAh without a sensor connected. No performance numbers were given for operation with a sensor attached. How many *Wise Boxes* can share a wireless channel is dependent on the sampling rate. Fléty states that at 5 ms periods up to 4 devices can be connected to an access point. He himself has used his system for a performance with five dancers.

How to help in the process of composing was the motivation for Stewart, who also build a wearable sensor system for this purpose [62]. He used accelerometers, flex sensors, a proximity sensor and an orientation sensor, whose readings are all converted to MIDI messages for further processing on a host. Stewart also defined a gesture vocabulary of 18 sparring movements, each with its own custom mapping of sensor readings to sounds. For Stewart this enables what he calls a '*cyclic relationship*'. Here the sound influences the composer's movements which in turn control the sound, thus shaping the creative process. Stewart believes this system enables him to better translate his ideas into an aureal form, noting: "*Technology that can catch, examine and reproduce gesture brings us a few steps closer to tapping into learned and unconscious behavior.*"

Flex sensors were also used by Siegel and Jacobsen [59]. They had eight flex sensors sewn onto elastic supports (normally used for sport injuries) and connected to a wireless transmission unit. There, the sensor signals were converted to MIDI and further relayed. Two sensor sets were built and used in two different interactive dance pieces. As those two were already described in Section 1.2, further details are omitted here.

2.1.5 Additional Sensor Options

An interesting approach was taken by Latulipe and Huskey [37]. Instead of building a custom input device they repurposed a set of Logitech *MX Air* mice to control a visualization. Those mice work like regular ones but do not require a surface to sit upon. A total of 6 mice was used by 3 dancers in several performances. The mouse positions were mapped to control either the locations of 6 objects, more complex parameters for 3 objects or to control only one object in the visualization. Because of the off-the shelf hardware used, however, the dancers were somewhat constricted in their movements. Being required to hold the mice in their hands, they were unable to perform any moves dependent on hand support.

Closely related to dancing, Bayliss et al. used accelerometers embedded into pois, to detect and visualize movement patterns [7]. Pois are spheres attached to the end of leashes that are spun around the body during a performance. They are especially used in club settings and can be glowing or on fire. A two-axis accelerometer is glued to the top of a poi, which is wired to a computer. First only interested in being able to distinguish three basic poi movements, they later chose to translate the sensor data to MIDI. In that translation acceleration measured at the poi is mapped to the pitch of a generated signal, making the movements audible. While the poi movements themselves are not directly a form of dance, they are embedded into a human performance in a musical environment and therefore also can be seen as part of a dance as well. Bayliss et al. in their paper also describe how poi performances play into the club setting and how they interrelate with dance, underlining the notion of a deep connection between the two.

El-Nasr and Vasilakos built a combination of a wearable system and a floor system to enable a lighting influencing experience [18] [19]. They used a *SenseWear*[4] armband, measuring heat flux, skin temperature, near body temperature and galvanic skin response, together with the output of a heart rate monitor. Based on that data, they could estimate a dancer's arousal state. The location of the dancer

[4]See `http://www.sensewear.com` (last accessed on March 20th 2010)

is determined by the pressure sensitive floor and thus the system is able to tell what lights currently affect the dancer. Ultimately, all that data is used to adjust the music (artists had to define a rule system for music-to-arousal mapping), the scene lighting and the lighting of a virtual backdrop to the arousal state. The lighting to arousal mappings are based on light theory and rules they deducted from movies.

Finally, Ng described three different motion-to-music mappings for interactive multimedia performances [44]. In the first one a camera is used to track dancers on stage, wearing color-coded costumes. In the second one facial features are tracked. In the last one drum brushes are outfitted with flex sensors. In all cases, his "*Music via Motion*" framework maps incoming data to MIDI events and wave output and thus is able to e.g., control the pitch of an instrument or the whole background music in a performance.

2.2 Recognition Algorithms

Detecting patterns in motions or working with motions in general, poses a set of challenges. How to detect similarities and what features are relevant for that being the most relevant ones. In this section some approaches to this problems are detailed. As e.g., questions of motion similarity are not only relevant to dance patterns, some works of a related focus were included as well if their approach was deemed transferable to dance motions. In order to not be too general, works on activity recognition were not included. While some aspects are similar, activity recognition is usually dealing with longer time periods, which does not readily transfer to gestures or poses.
In general, recognition algorithms for motions can be roughly split into two groups:

- Algorithms computing features for whole blocks of motion data (segmented according to some criterion) and using those features to determine similarity

- Algorithms computing similarities of temporal sequences directly (such as hidden markov machines (HMM) and dynamic time warping (DTW))

Both of these groups will yield a similarity measure for two given sample sequences, where different distance metrics might be used by either of them.
With a way to determine similarities available, the question of classification is coming up next. Classification could be done via methods such as support vector machines (SVM), neural networks or the *k*-nearest neighbors (*k*-NN) algorithm. In Chapter 4 this is further detailed.

2.2.1 Hidden Markov Models

HMMs are traditionally used in gesture recognition and thus it comes as no surprise that they can also be adapted for dance input and other body movements. In one of the non-dance examples Kunze et al. used accelerometer and gyroscope readings in trying to classify Tai Chi poses [36]. Eight sensors were strapped to the test subjects (one expert and two amateurs) and three different Tai Chi movements were recorded. In line with the notion that in Tai Chi the "*total consumed energy is supposed to be as small as possible*", the squared angular velocity was evaluated as a feature. In their sample the square angular velocity did indeed correlate with Tai Chi experience. However, Kunze et al. note that the sample size is too small to draw conclusions. In another approach, Kunze et al. used a sliding window and for each range calculated the corresponding feature vector. Unfortunately, they do not specify which features were used, except the *75% percentile* (75% of the sensor values are below that feature's value) and the *frequency range power*. Initial classification tests with HMMs performed at up to 85% accuracy. As Kunze et al. state, their results are only preliminary and while they indicate it might be feasible to classify Tai Chai expertise, a validation of that claim is yet missing.

Similarly, Bevilacqua et al. built a custom sensor system with a three-axis accelerometer and a two-axis gyroscope which sends out OSC messages using Zigbee for wireless data transfer [11]. Ultimately interested in gesture input, left-to-right HMMs were used as model and to compute gesture similarity. When learning a left-to-right HMM for a new gesture, the signal is downsampled (typically by a factor of 2) and each value in the signal is mapped to its own state in the HMM. The transition probability from one state to the next is dependent on the downsample rate (1/downsampling factor). The observation function at each

HMM state is a multidimensional gaussian where each dimension corresponds to one sensor axis, with the gaussian mean set to the sensor value. The variance to use is set by the user to control the desired fit. The system was evaluated in an experiment where students needed to match conducting gestures of a teacher. If a student's gestures deviated from the trained gestures the rhythm of the music would deviate accordingly.

Matching whole body motions of varying speed with a set of pre-recorded ones to control avatar motion was the goal of the work of Liang et al. [39].They especially emphasize ways to refine the input signal for motion matching. For example, after automatic segmentation, all motions are normalized in the time dimension by fitting a cubic spline and resampling at the desired resolution. The amplitude of the signal is also normalized to $x' = (x - \bar{x})/Var(x)$. In another pre-processing step, principal component analysis (PCA) is used to extract key features from the signal. The final recognition is performed with HMMs, with one trained per training set. When a matching motion is found, the avatar is animated accordingly. In an intermediate step however, the target animation is first time warped to the speed of the recognized motion. Thus a slow kick and a fast kick are recognized as the same motion (due to the resampling described above) but result in different motions being shown. According to the force of the input motion, generated motions might also be exaggerated.

While the above mentioned approaches were not specifically targeting dance motions, Gutknecht et al. used HMMs for Butoh, a form of experimental dance [29]. Setting out to classify movements, they designed a discreet three-dimensional motion space (‘*intensity*’, ‘*form*’ and ‘*flow*’) yielding a total of 64 motion categories. Dancers are equipped with three-axis accelerometers at the wrist, upper arm and upper leg, whose readings are relayed via Bluetooth. For classification, the values in a two second long sliding window are transformed into a sequence of features. On those features three HMMs (one for each dimension) are used to determine the most likely motion state sequence (remember that motion states are discreet here). The final motion state decision for the block is done using a majority vote. Gutknecht et al. furthermore designed a mapping from the motion space to a custom emotion space and subsequently derived a visualization from the detected emotional state.

2.2.2 Dynamic Time Warping

Another method, somewhat similar to HMMs is DTW. In fact, left-to-right HMMs like the ones used by Bevilacqua et al. and DTW are two solutions to the same problem. The general idea is to compute the similarity of two given sequences that may differ in the temporal domain. While left-to-right HMMs can be adapted to do so by setting the transitional probabilities accordingly, DTW tries to find the overall best fit using a dynamic programming approach.

Tang et al. developed an algorithm to find repetitive patterns in motion capture data of dances [63]. When recording, 35 markers are tracked on a participant and the resulting posture data is normalized (body center as origin and facing frontwards). From the resulting motion sequence a similarity matrix is derived. Postures in two frames are similar if the sum of the point by point euclidean distances is low. Repetitive motions can now be deducted from the similarity matrix, where diagonal patterns of similarity denote sequences, equally changing over time. Tracing patterns in the binarized similarity matrix in some respects is thus equal to similar image processing problems. Tang et al. use DTW to find such traces. Finally, using auto-clustering, patterns are classified as either cyclic or acyclic and an estimate of the cycle period is computed.

DTW was also used by Bettens and Todoroff, who set out to detect gestures in a continuous sensor data stream [9]. For this purpose two sensors (three-axis accelerometer and two-axis gyroscope) are placed on both ankles of a dancing viola player. When performing, the downsampled sensor values are matched against a database of pre-recorded gestures. However, no segmentation of the live signal is tried. The signal is matched against the database at a number of different offsets instead.

2.2.3 Non-Temporal Feature Classification

In addition to the two approaches mentioned above, that work on time series, there are a number of other possibilities for classification. If feature vectors are available for example, similarity of movements can be computed using various distance metrics. Classifiers like a support vector machine (SVM) can also be used to segment the feature space. In this section several different algorithms are presented that illustrate the range of possibilities. With some directly targeting dance, all are at least concerned with body posture and similarities of motions.

One example, where feature vectors and a distance metric are used, is the work by Crampton et al., who set out to use poses as video game input [14]. For sensor input, four three-axis accelerometers are placed on the arms and legs. In the training phase poses are recorded as mean and standard deviation per sensor per axis. When recognizing, the incoming sensor data is compared with all training poses and assigned to the class with the smallest distance. The Mahalanobis distance metric is used, which adapts to existing correlation and is scale-invariant. They tested their algorithm in a user study, where seven participants were instructed to perform eight poses. The study yielded satisfactory results and also indicated that with larger training sets or more training from the players, further improved accuracy can be obtained.

Where other systems use body sensor input, Peng et al. built a system using two orthogonal cameras [49]. Their system is dependent on training data and a set of poses was motion captured from a professional dancer for that purpose. The captured data is used to generate synthesized views of the corresponding poses which is then used to build a tensor. During runtime the video data is similarly transformed into a corresponding tensor. The tensor is decomposed using higher order singular value decomposition, which in turn can be utilized to approximate a pose coefficient vector. This pose coefficient vector will be view-invariant. With the coefficient vectors of the training set several classifiers were trained and subsequently evaluated. In their tests SVM classifiers outranked fixed-threshold and von Mises-Fisher recognizers with recognition rates around 85% and a false detection rate of about 5%.

2.2.4 Algorithms Utilizing Periodic Subspaces

The work of Nevada and Leman also does not fall under any of the previous categories. They developed a method that is able to detect periodic movements in dance [43]. Specifically, they were interested in samba dance and how musical meter is related to the gestures found. For their study two professional samba dancers were recorded on video performing a set of dance sequences. In the recorded video the positions of nine body points were manually annotated resulting in a set of feature vectors with the temporal and spatial resolution of the input video. A periodicity transform [57] is used to find movements correlating with musical meter. For this purpose the meter information for the accompanying music was computed beforehand. When recording, the periodicity transform projects the signal into a periodic subspace, removes that periodicity from the signal and repeats the process for other subspaces. Thus, the proportion of periodicities in the original signal is determined (note how the order of projections is important and must be carefully chosen). Nevada and Leman pick the subspaces to test according to musical meter. This allows them to filter out the strongest periodicities, related to musical meter, in the signal. As this transform is performed independently for both dimensions, found periodicities can be either one-directional or bi-directional. While no classification of gestures was performed, they noted that their approach could provide a useful methodology for dance analysis, with the periodicities potentially being used as classification features.

Chapter 3

Wearable Sensor System

Based on a set of requirements, the sensor options mentioned in Chapter 2 are evaluated. Furthermore the final system design of the prototype is described.

3.1 Requirements

When contemplating the design of the prototype, a number of requirements were taken into account. The sensors should be unobtrusive and should not restrict a dancer's movements. For the same reasons communication to a base station, if needed, should be wireless. The sensors should be able to capture all movements (upper and lower body) and to do so at high sampling frequencies. If battery power is needed, operation should be possible at least for the duration of one capture session. If dance movements to one song are recorded and preparation time is taken into account, this requires at least 15 minutes of battery life. However, much longer system uptimes are desirable. The final system should be able to work in uncontrolled environments. For example, it should work in a dance club setting and thus should not be dependent on constant lighting. Individual dancers should be observable, but scalability to larger groups should remain a possibility. Thus, any dance capturing has to allow for separation of dance movements per dancer. Additionally, it is also deemed beneficial if parts or all of the sensor system can possibly be reused for other projects as well.

3.1.1 Sensor System Choices

As described in the related work chapter, there are a number of sensor designs which focus on dance. However, not all of them are equally fitting to the target scenario. For example, camera-based systems largely depend on lighting conditions. While a constant background lighting can be obtained in a lab environment, dance clubs have wildly varying lighting. This poses problems for algorithms extracting the dancer's silhouette from the video stream. One option here is to use infrared (IR) cameras and illuminate the scene with IR lights. This solves problems with insufficient lighting or wildly varying lighting to some extent. Systems depending on visual input also do not scale well to groups of dancers. With two dancers there is already a good chance for some overlap. Multiple cameras can alleviate the problem somewhat, but only up to a certain degree of occlusion. On the other hand, cameras allow movement detection at a comparably low price and without the need to equip dancers with any sensors. Providing the external conditions such as lighting and background can be controlled, cameras are a viable alternative. This is especially true if only the movements of one dancer are to be recorded. The temporal resolution of the video stream could be a problem for some applications but high speed cameras can be used in those scenarios (albeit at a higher price point).

Sensing floors' main disadvantage is that they are only able to capture foot movements. Upper body movements can not be detected at all and some motions like rock steps are hard to detect for some systems. Differentiating multiple dancers also is hard to do, unless additional sensors are added. Portability of sensing floors is usually poor. They might be modular and thus could be taken apart though. However, they will always be heavier and bulkier than other sensor options. If all recordings are to be taken in a designated space, some of these disadvantages however disappear. It might even be advantageous to have such a system as participants do not need to be wired and can start dancing right away. For this thesis however, a designated room was not available and thus the disadvantages of floor systems outranked other aspects.

Shoes are an interesting option but, like sensing floors, are not able to capture upper body motion on their own. When added to other wearable sensors they offer finer recognition of foot motion. However, if other wearable sensors are already used the benefit of additional foot sensors might be negligible. If e.g., accelerometers already measure leg movements, steps can be inferred. For some dance styles like tap dancing however, a focus on foot movements could be appropriate. Another problem with shoe systems is their limited adaptability to participants. While wearable sensors for example can be adapted to all participants using straps or similar means, shoes cannot adapt in size and only people of a certain shoe size might be eligible participants. In projects primarily interested in foot movement shoe systems can be advantageous, but for this project whole body movement was of interest. Combinations of shoe sensors and wearable sensors could be interesting but come with additional complexity.

Wearable sensors currently are the most versatile option when it comes to dance movements. They can be placed anywhere on the body and are able to capture any motion. In a crowd they enable data collection from individual dancers, independent of occlusion or other outside conditions such as lighting or visibility. With wearable sensors the additional challenge of data transmission back to a base station becomes relevant too. Here, a wireless solution is required. Wires running from the dancers to a base station would restrict their freedom of movement and would prove especially limiting in larger groups of dancers. Analyzing data from wearable sensors, however, is harder than e.g., analyzing data from sensing floors. In sensing floors, foot movement information is readily discernable. In a wearable system on the other hand the movement measured in individual sensors is interdependent i.e., arm movements for example also influence the measurements of sensors placed at the wrists. Wearable sensors also suffer from additional issues, because they are directly attached to a dancer. If e.g., the sensors are worn using straps, they might loosen while dancing thus resulting in an increasing drift in the recorded values. Recording a dance session requires a comparatively lengthy preparation for each participant, as sensors need to be properly attached. When it comes to scalability, wearable systems fare better than other systems as every dancer is independent from the others. Additional stress is only put on the communication channel and the data processing host. A scalable wireless system is therefore required for multiple dancers. Because of the nature of wearable sensors, battery life plays a larger role too. Running an extension cord to each dancer would significantly limit their freedom of movement and practically make some dances impossible (e.g., those featuring severals turns). Therefore, power has to be provided by a wearable module as well. In general wearable system are more complex than other choices but also offer significant flexibility and can be utilized in a more varied range of settings.

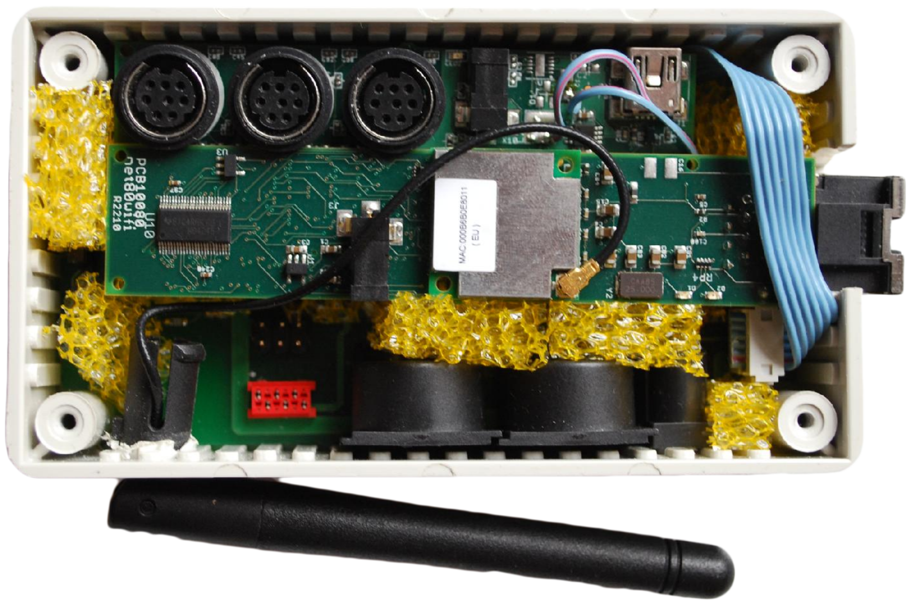


Figure 3.1: Sensor Package Internals

3.2 Hardware

Based on the requirements, a wearable sensor system was chosen for the prototype. The overall system can be split in three parts: the sensors collecting movement data, a communication link to transmit the collected data and a host unit able to control sensor sampling, bundle data from all sensors and interact with external clients. All parts are described in detail in the following sections.

Everything apart from the sensors is bundled in a $11.1 \times 6.2 \times 3.1$ cm box, as shown in Figure 3.1. In this box, an embedded system and a custom interface board together take on the role of host unit. A Wi-Fi module is part of the embedded system with an antenna protruding from the box by about 1 cm. By bundling those parts together in one box, a compact and robust packing is achieved. This is especially important as some interconnections between the parts in the box are soldered manually and are rather fragile.

This box, together with a battery, one sensor and a splitter, is worn around the hips inside a fanny pack (shown in Figure 3.2). This offers a convenient way

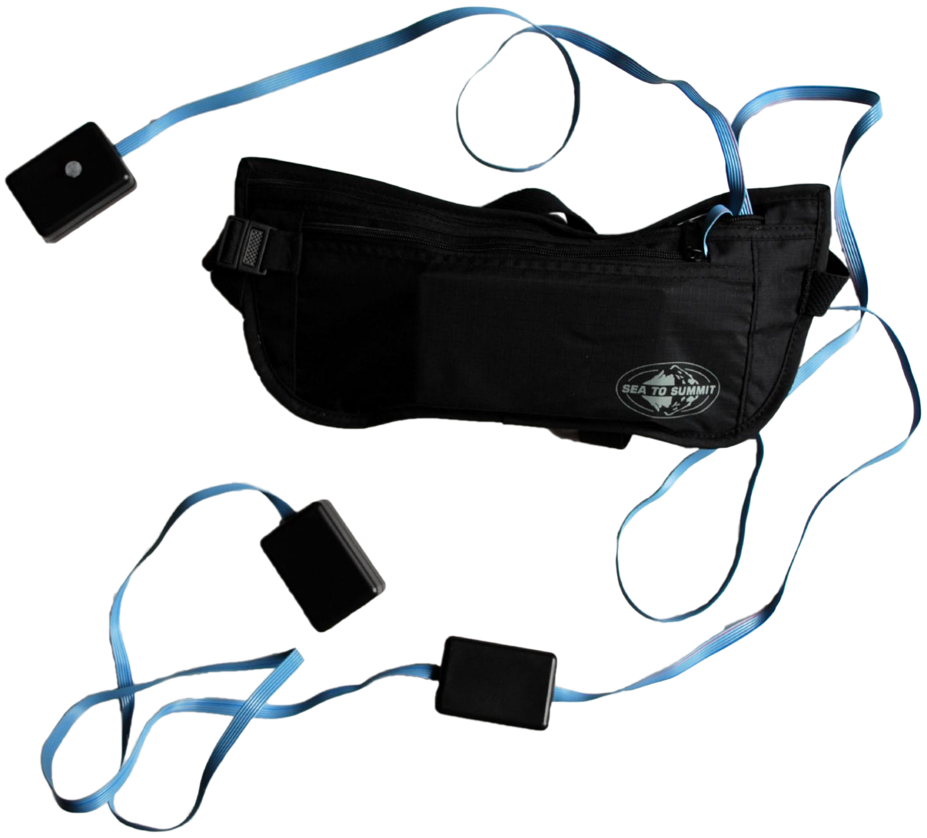


Figure 3.2: Wearable Sensor Package

to wear the bulkier parts of the system without restricting freedom of movement significantly. By having multiple parts of the system inside this bag the number of wire connections exposed is reduced. Also, less mounting straps are needed, thus making the overall system more comfortable to wear.

3.2.1 Sensors

The sensors (shown in Figure 3.3) were repurposed from a previous project of Aristotelis Hadjakos [30] with no changes made to them. Each sensor contains an *Analog Devices ADXL330* three-axis accelerometer, an *InvenSense IDG300* two-axis gyroscope and an *Analog Devices ADXRS300* one-axis gyroscope. The second gyroscope is mounted perpendicularly to the first one to obtain a virtual three-axis one. The *ADXL330* has a measurement range of ± 3 g while the *IDG300* has one of ± 500 °/s and the *ADXRS300* one of ± 300 °/s. All sensors are sampled with 10 bit

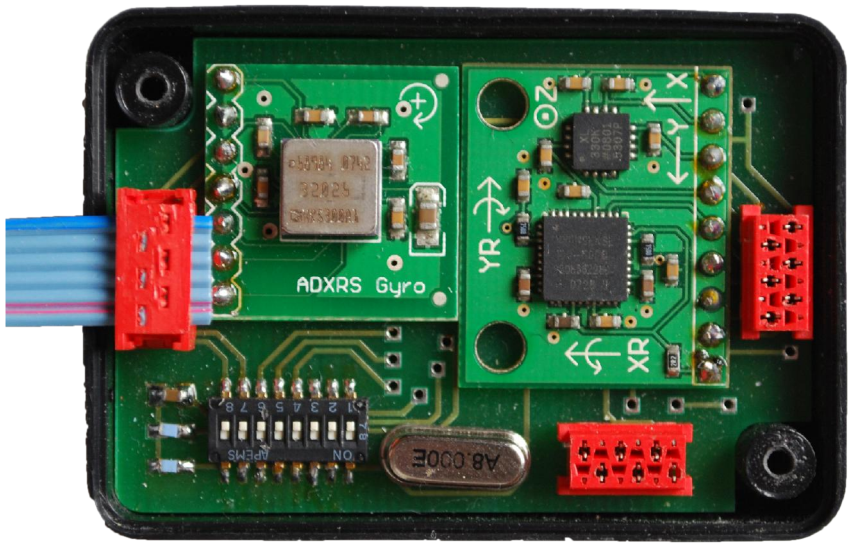


Figure 3.3: Sensor Internals

resolution. The sensors are housed in a $5.2 \times 3.6 \times 1.8$ cm box for added robustness.

While the accelerometers do not need to be calibrated, the gyroscopes are prone to drift. Over time the gyroscopes zero point shifts and a rotational force might be measured while no rotation is taking place. To compensate for such errors, the gyroscopes are continuously calibrated. When sensor data is coming in, it is also buffered per sensor. After a certain number of samples has agglomerated, the deviation for each accelerometer component is computed. If it is higher for any dimension than a given threshold, the sensor is considered in motion and is not calibrated. In the other case, the sensor is assumed to not have moved and any gyroscope readings other than zero are considered erroneous. To compensate for said error the detected gyroscope deviation is deducted from all further gyroscope readings. As this calibration is done on a per sensor basis, a sensor might be calibrated while others are not and vice versa.

To determine if the choice of sensors was appropriate, a dance sequence was recorded and analyzed. The recorded sequence contained side steps and rock steps with arm movement ranging from none to stronger movements with the arms raised above the head. The song used had a tempo of 119 bpm and thus

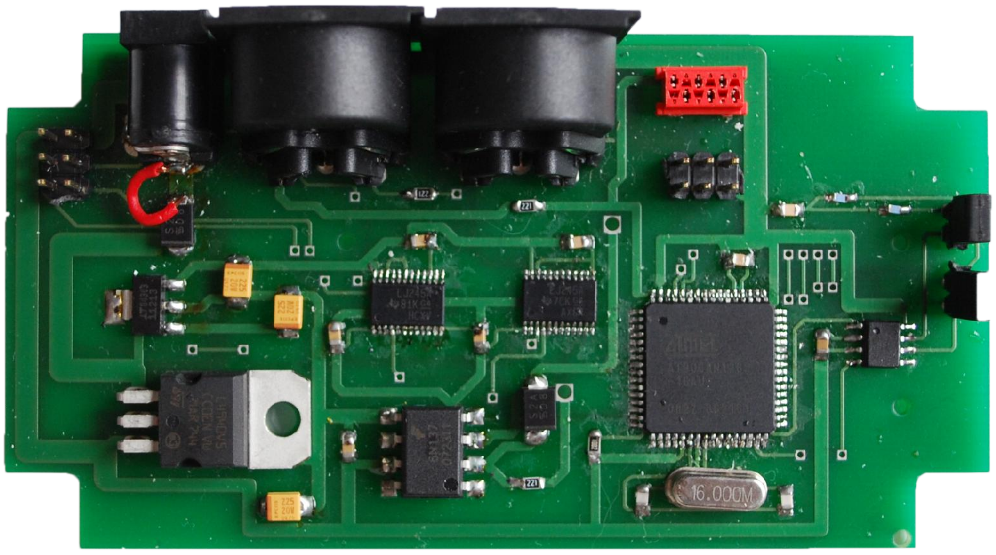


Figure 3.4: Sensor Interface Board

falls into the lower tempo range of house music. Faster music styles like jungle or gabber might elicit more energetic dance movements. Table 3.1 shows the values measured. Note that all measured values lie within the normal range of the utilized sensors. A similar measurement, using the same sensors, has previously been done for movements associated with piano playing by Hadjakos et al. [31].

3.2.2 Sensor Interface Board

To ease working with the sensors, a custom interface board (shown in Figure 3.4) was built by Hadjakos for this prototype. It connects with all sensors via a controller–area network (CAN) bus [1]. The CAN-bus is a broadcast serial bus initially designed for automotive applications, but now also used in other areas. CAN-bus can be used on top of a number of physical layers. For this system, electrical wiring was chosen. While wireless transmission would increase freedom of movement slightly, wiring the sensors also renders batteries in the sensors unnecessary. Furthermore, wiring allows for abundant bandwidth and scales up to a large number of connected sensors. In the current system a 1 Mbit/s bandwidth is provided by the CAN-bus. On the bus, all connected nodes can send messages,

[1]See `http://www.can-cia.org/index.php?id=46` for a more in-depth describtion of CAN (last accessed on March 22nd 2010)

Table 3.1: Sensor ranges when dancing

Sensor	Minimum	0.1 Percentile	Mean	Std	99.9 Percentile	Max
Hip Accelerometer	-3.465 g	-2.649 g	-1.203 g	0.436 g	-0.103 g	0.230 g
Hip Gyroscope	-338.4 °/s	-233.4 °/s	-8.9 °/s	56.2 °/s	157.7 °/s	201.3 °/s
Hand Accelerometer	-3.104 g	-2.744 g	-0.146 g	1.252 g	3.358 g	4.423 g
Hand Gyroscope	-416.5 °/s	-278.5 °/s	64.2 °/s	93.1 °/s	449.7 °/s	525.8 °/s
Arm Accelerometer	-4.120 g	-3.035 g	-1.247 g	0.609 g	1.393 g	2.331 g
Arm Gyroscope	-391.6 °/s	-305.5 °/s	29.4 °/s	76.1 °/s	318.1 °/s	419.0 °/s
Leg Accelerometer	-3.299 g	-2.881 g	-1.299 g	0.461 g	0.232 g	2.595 g
Leg Gyroscope	-484.2 °/s	-161.6 °/s	0.7 °/s	47.3 °/s	179.7 °/s	329.4 °/s

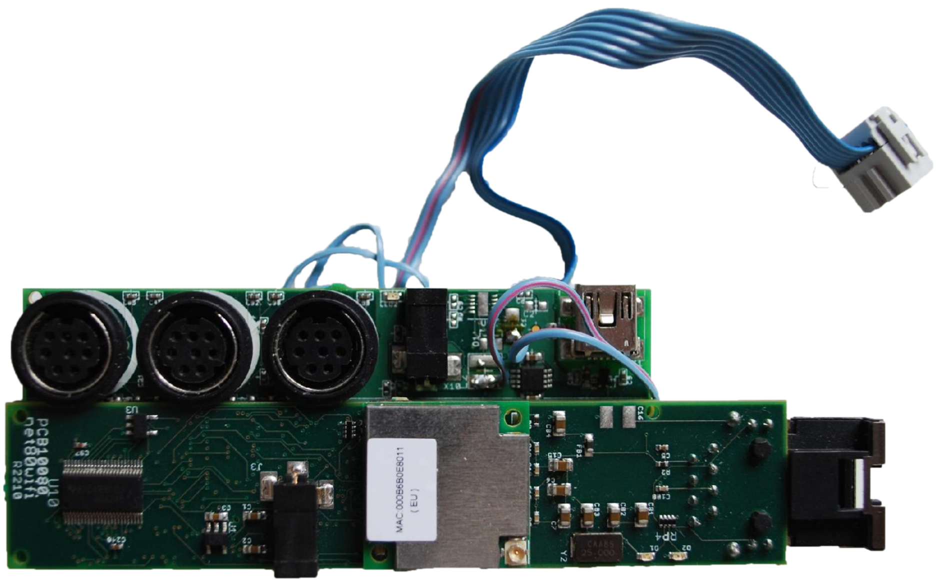


Figure 3.5: Gumstix Internals

but prioritization of more important messages ensures low delays where needed. The CAN-bus also provides error handling and guarantees data consistency on the network. The sensors are arranged in the topology shown in Figure 3.7.

The interface board acts as the host unit and controls the sampling of the sensors. On startup, the board broadcasts a discovery message, triggering a reply from all sensors. In this way, a list of all current sensors on the bus is compiled. Using an internal 16 MHz quartz, an interrupt handler is set to be triggered at 100 Hz. At each invocation, the interface board will broadcast a message to all sensors, notifying them to take a sample. The sensor then reply, sending a message with their current state. This data is collected by the board and further passed on to the embedded system. The board and the embedded system interface using a serial connection. By sending commands over the serial bus, the interface board can e.g., be instructed to start reading sensor data, change the output format or stop data collection. Sensor values are passed back to the embedded system either in textual form or in a binary format for further processing.

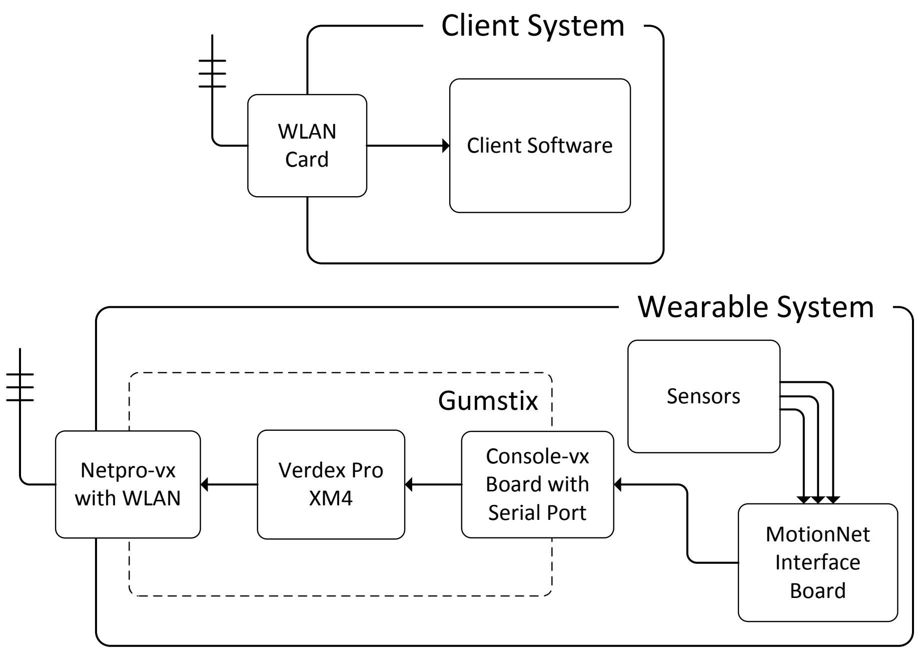


Figure 3.6: System overview

3.2.3 Embedded System

The prototype is built around a Gumstix embedded system. A *Verdex Pro XM4* mainboard is paired with both, a *console-vx* expansion board and a *netpro-vx* expansion board with the optional Wi-Fi module installed. Both expansion boards snap onto the *Verdex Pro XM4* mainboard and the bundle is locked in place using a set of screws. To interface with the Gumstix, a serial cable can be plugged into the *console-vx* or a connection can be established over ethernet or Wi-Fi. The *console-vx* board is also used to connect to the interface board described above, using one of its serial ports.

Communication

For the communication link between the wearable sensor system and external clients it was decided to use Wi-Fi. While Wi-Fi was not the best option for a project like this, some factors still made it an appropriate choice. Other projects have used Wi-Fi for similar purposes before. For example Fléty used Wi-Fi in his *Wise Box* [23] and noted that the performance was “*excellent*”. Bevilacqua et al. designed their system with the *Wise Box* in mind but set out to improve power consumption and size [11]. One of the changes they implemented was switching the Wi-Fi controller for a Zigbee one. Aylward and Paradiso also decided against using Wi-Fi and instead created their own custom wireless protocol [3] [4]. Both of those systems use less power than Wi-Fi. On the other hand, they offer significantly less bandwidth. For those systems this was not an issue as the expected throughput of their sensor data was still less than the bandwidth available. Many systems also use Bluetooth (e.g., [33], [53] or [35]), which requires more power than comparable systems and does not scale well as each Bluetooth master can only communicate with up to seven slaves.

Subsequently the choice for Wi-Fi was made based on the direct availability of an extension board for the Gumstix system. Using another protocol would have required considerable extra work and time. Furthermore, the Gumstix is also used for other projects which require a Wi-Fi module (it is e.g., also used for indoor location awareness). For future systems it would be beneficial, though, to switch from Wi-Fi to another communication solution.

3.2.4 Sensor Placement

In this project four sensors were available. To be able to register whole body movements the sensors were arranged to cover arms and legs. One sensor was placed near the thigh, one at the wrist, one on the upper arm and one at the hips. The arrangement is also shown in Figure 3.7. With only four sensors available, it was not possible to cover both halves of the body. However, in the domain of dance it can be assumed that leg movement of one leg is not independent from the other one. While an equally strong assumption can not be made for arm movement, it was still deemed highly unlikely that all movement is constrained to just one arm. Four sensors also did not allow the placement of sensors on a lower leg or a foot. Instead, preference was given to upper body motions. However, this choice does restrict the system in some aspects. Specifically, the overall system is less sensitive to foot movements. While lifting a foot will also be measurable at the thighs, such measurements are less precise. In future iterations additional sensors at those locations would be preferable.

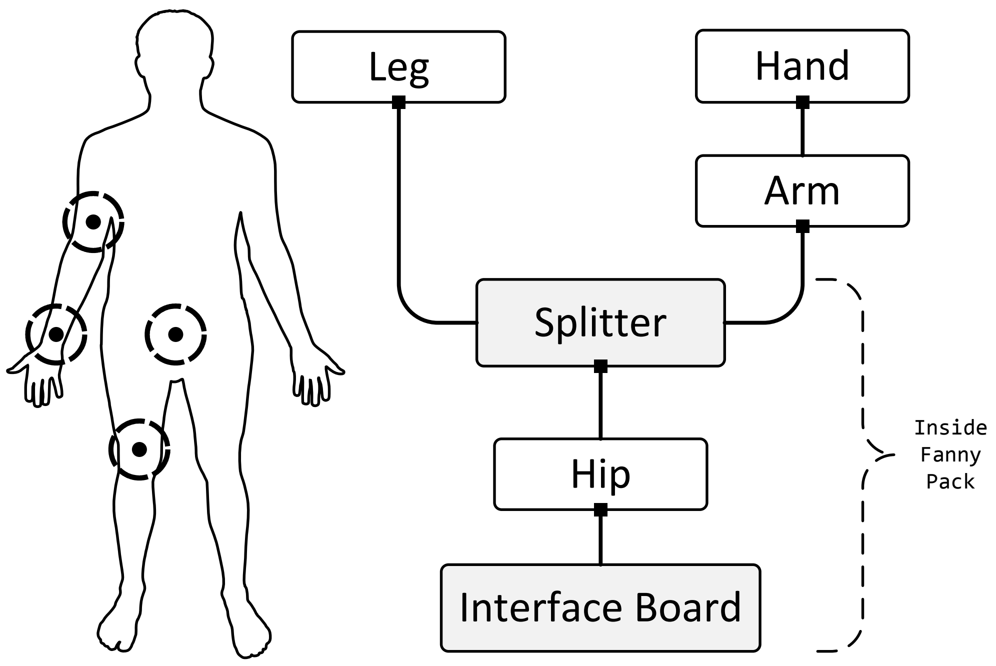


Figure 3.7: Sensor locations and topography

As noted above, the hip sensor is located inside a fanny pack worn around the hips. The other sensors are fixed to the body using a set of adjustable velcro straps. This allows for fast and flexible attachment of sensors. As the straps are adjustable they can be affixed to participants of varying body size.

3.3 Embedded Software

The OpenEmbedded framework is used to compile a GNU/Linux distribution for the Gumstix. OpenEmbedded is a flexible environment, designed to cross-compile code for a range of architectures. For this purpose OpenEmbedded comes with a number of BitBake files. BitBake is somewhat similar to *make* and BitBake files define the tasks to execute and additional metadata. Thus a BitBake file may e.g., describe where to fetch the source for a package, how to compile it and what dependencies that package has. BitBake files can also be used to describe additional user programs to be compiled for the Gumstix.

The program to run on the Gumstix, however, is not compiled using OpenEmbedded itself but is built on top of the MundoCore[2] communication middleware [1]. MundoCore is designed to work on a number of platforms and has bindings for several programming languages. It provides abstractions to ease the development of pervasive and distributed systems. For example, MundoCore has facilities for remote procedure calls, node discovery, publish/subscribe messaging and marshalling. The code to run on the Gumstix is implemented as a MundoCore service and uses a serial class included in MundoCore to communicate with the interface board. For connection with external clients, the embedded software uses MundoCore's publish/subscribe mechanisms. Thus, the embedded system is only loosely-coupled to any client machine.

Receiving commands and sending out collected data is done on a dedicated publish/subscribe channel. By default, outgoing messages are not delivered to the sending system. Hence, the embedded software only receives messages sent by other nodes. In MundoCore a message is always a map of key-value pairs. On top of this, MundoCore also offers facilities to send objects using externalization and serialization. Sensor values are essentially an array of floating point values and, to preserve that structure, are transmitted as a simple binary blob. This allows for low overhead in the message and easy decoding on the client side. Of course, the byte order of the Gumstix and the client system need to match, but in the given scenario that was the case. The number of sensors connected to the embedded system is transmitted as well, but could potentially also be computed from the size of the binary blob.

[2]See `https://leda.tk.informatik.tu-darmstadt.de/cgi-bin/twiki/view/Mundo/BuildCPPoe` for instructions on how to build MundoCore for OpenEmbedded devices (last accessed on March 22nd 2010)

Chapter 4

Dance Pattern Recognition

In this chapter, the steps taken to find patterns in dance sequences are described. Patterns in dance are reoccurring movement sequences at one or more levels. Whether such patterns are apparent or even existent will depend on the dance style at hand. For example, one could easily construct an experimental dance piece without such characteristics. However, most dancing will exhibit some patterns. To look for examples underlining this proposition, one does not need to look further than the next dance studio or dance club. For instance, all forms of ballroom dancing and latin dancing feature the notion of the basic step, sometimes also descriptively called basic pattern.

Dance and music are intrinsically related. In fact, they often even share a name (e.g., waltz, samba, disco or swing). Most dances came into existence with a specific style of music in mind. For example, breakdance evolved from the same cultural background that spawned hip-hop music and rock and roll music inspired the creation of rock and roll dancing. When dancing, certain moves might correlate with the music playing. People headbanging, for example, will do so more vividly in high energy stretches of a rock song, like the chorus. Dance movements are made in close relation to musical meter, also taking into account any accentuations. For instance, the basic steps in a waltz are arranged according to the meter with the first one being the dominant one. As our perception of a song's meter already provides us with a hierarchy of beats, it is only natural that this hierarchy is also reflected in the dance movements.

When analyzing dance we can therefore make the assumption that a close relationship to the music playing does exist. This relationship can be utilized to help in the task of dance pattern classification. As stated above, dancing contains patterns and dance movements are aligned with the meter. Therefore, using meter information for segmentation purposes is a natural choice. While such a segmentation is not necessarily needed for some algorithms (e.g., Tang et al. [63], as described in Chapter 2, transform the problem into a new domain where some computational hurdles no longer hold up), it reduces the complexity of others significantly. Segmenting the input signal into blocks according to meter, and thus restricting the interest in similarity to inter-block similarities also is a useful abstraction in general. While we might search for patterns in sequences smaller than one such block, such similarities would be less meaningful in many instances.

Taking a step back, possible block lengths to consider and examples for corresponding patterns are:

Section Level A song playing is made up of a number of sections (e.g., intro, chorus, bridge, verse and outro), which could be used as a possible segmentation level. In this case, the assumption is that dance patterns might change in relation to section changes. The chorus, for example, often exhibits a different rhythmic and harmonic structure than the verse and thus also encourages different dance movements as well. While this is a valid assumption, it is also rather simplistic. Choruses span several measures, and verses are usually longer than that. Those measures usually are not the same musically. Assuming a constant dance movement, spanning all of them, seems inappropriate. Comparing sections with other sections would also require temporal normalization of some kind, as no constant section length can be assumed.

Measure Level Compared to whole sections, individual measures, spanning a number of beats, seem like a more appropriate choice. However, only for some basic steps one measure is exactly the time needed (e.g., waltz). Others span over two measures (e.g., salsa or cha-cha-cha) or only part of a measure (e.g., discofox, or foxtrot). Thus, patterns at the measure level are probably not a good choice for standard or latin dances in general. However, for casual dancing in a club setting the measure level is more appropriate. Here, basic rock steps and side steps are often the main components being built upon.

Beat Level As stated above, some dance moves last less or longer than one measure. In this cases one might consider looking for patterns on a multi-beat or beat level. Comparing motions on the beat level however, introduces a certain arbitrariness. For example, a step forward is part of many patterns. One could argue, that if such building blocks of larger patterns could be found, those larger patterns could be derived from them. This approach is similar to finding repetitive parts in a string or in DNA sequences. The problem on this level is that a forward step in one movement can be quite different from a forward step in another one. The steps before and after the forward step will influence the movement itself. On the other hand, a complete step sequence (e.g., in a side step) is much more closed against such influences.

Subbeat Level Going one step further, one might try to look for patterns at an even more fine-grained resolution than the beat level. In some respects, one could consider non-segmenting approaches such as the work of Tang et al. [63] to work on this level. However, when segmenting, this is a poor choice, as the problems of beat level segmentation are weighting in even more. Additionally, it is doubtful that patterns at arbitrary subbeat levels exist. Half beat steps are a part of many dance moves but usually come in pairs and could be considered as a union (e.g., two quick steps and one slow step in a chassé).

For this thesis, measure level patterns were deemed the most interesting. As it is evaluated with respect to a dance music setting and not as a means for standard and latin dance, this is a valid choice. For this reason the sensor data is split into blocks of measure length, which subsequently form the basic level for comparison. Thus, when interested in patterns spanning one $^{\mathrm{g}}/_{\mathrm{n}}$ measure and given a song, playing at b bpm and a sensor sampling at f Hz, each such measure would contain:

$$\frac{60b}{\mathrm{g}}f$$

samples. For a 120 bpm song at $^{4}/_{4}$ with motions samples at 100 Hz, this would result in 200 samples in every measure. As can thus be seen, segmenting the data breaks it down into manageable blocks, where a brute force comparison of all possible shifts would be computationally expensive.

In this chapter the segmentation process and the subsequent classification are described in detail.

4.1 Signal Segmentation

As stated above, pattern comparison is to be done on a per-measure basis. Detecting measures, however, is quite challenging, while detection of beats is a more well-researched problem. As this thesis targets a scenario of dance music, all music can be assumed to be in a $^4/_4$ measure anyway, allowing for an easy mapping from a beat level segmentation to a measure level one by means of grouping. Note that the proposed algorithm would also work for other segmentations and the grouping decision is solely based on the type of music and the corresponding motions. For the task of beat detection, one could use the sensor data directly or compute beats from the accompanying audio. While some approaches exist to perform beat detection on motion data (e.g., the work by Enke [21]), for this thesis it was decided to precompute beat delimiters from the audio file used. As beat detection from motion data is less accurate than to do so from audio data, and as the recording environment and setting can be controlled in this context, this is the easier and safer alternative.

Detecting beats in audio files is an area already researched extensively (see e.g., arcitles by Alonso et al. [2] or Goto and Muraoka [26]). For this thesis, *Beatroot*[1] by Simon Dixon [17] was used. *Beatroot* takes audio files as input and generates a list of timestamps denoting the found beat positions. This list is saved in a raw text file with one timestamp per line. When recording a dance session, this file can be loaded in order to use the contained beat information. For details on how the evaluation application handles the beat data, see Chapter 5. Beat information is mapped to the recording timeline and embedded in the recorded data to aid in further processing. Note again that the beat detection does not identify measures but only individual beats. However, as stated above, for dance

[1]Available at `http://www.elec.qmul.ac.uk/people/simond/beatroot` (last accessed on March 22nd 2010)

music and most other forms of popular music, a time signature of $^4/_4$, also known as *common time*, can be assumed and is used here.

4.2 Movement parameters

With the sensor data split into blocks, a measure for block similarity has to be defined. Before doing so, however, it is useful to define some parameters that could be used to discern different movements. Parameters could be defined for the whole block or per sample in a block. To define possible parameters, it is useful to do so with respect to possible dance moves. For this purpose a number of moves were selected from the website of Chihoe Szeto[2], a professional dancer, choreographer and instructor. All dance moves are defined for a $^4/_4$ measure and target club dancing.

"Clown Walk" For this move one starts with putting one heel down in front on the first beat. On the next half the forward foot is turned outside. At the second beat a jump alternates the forward foot which is then turned outside as well at the following half beat. The whole sequence is repeated to complete one measure. Together with this footwork, the arms swing in front of the body, converse to the legs (one alternation per beat).

"Out 'n' Up" Here, one foot is moved to the side on the first beat in a sweeping motion with the body leaning in the opposite direction. It is brought back and afterwards lifted up in the next beat period. In the second half of the measure the movement is repeated with the other foot. On the up motion the arm of the corresponding other side is brought up in front of the chest, staying up during the following sweeping motion.

"Bounce" This is a form of rock step. On each beat the body drops down a bit and straightens up again. When doing so, the drop is slightly faster than the rebound. When going down, one successively shifts the body to either side, thus also bending one knee more than the other. Note how the feet do not move at all. A variation of this move is the "*Ghetto Bounce*", that is simply danced twice as fast.

[2] `http://www.chihoe.com` (last accessed on March 5th 2010)

"Dip in U" This move is also a form of rock step. One starts in a position leaning to one side, drops down a bit, shifts to the other side and goes up again. All of this in a form of a parabola or upper case 'U'. It can be danced with one shift per beat or at a slower speed as well. The hip movement can be accentuated for added effect.

Looking at those moves, some features are readily apparent. For example, in two of them the feet are never lifted up, while in the other two an energetic lift is present. The arm movement is either a more horizontal or vertical one, but might also be left undefined. Accentuation sometimes is a possible differentiator, as some moves have a more constant flow, while others contain distinctive slow-fast patterns. Comparing two moves, the "Ghetto Bounce" and the "Out 'n' Up", the first one features small but fast movements while the other one has more spacious but also slower movements.

In general, energy and spaciousness of a movement seem to differ in those examples. Concerning axes of motion, most moves have a strong sideways component with only one move being performed forward. Similarly, an upwards component is often present, but varies in strength (e.g., jumps vs. bounces). Looking at the moves, some seem 'smoother' than others, as jumping for instance gives a move a more jagged feel. While the moves in the examples are all rather energetic, one can imagine more subtle movements being executed along more subdued portions of a song (a bridge for example sometimes brings upon a sudden drop in musical intensity).

4.3 Block Similarities

Finding reoccurring patterns ultimately is a problem of finding similar blocks. If the difference between two blocks is sufficiently small, chances are the movement in the second one is a reiteration of the first one. There are two problems at hand: how to determine the similarity of two blocks and what threshold to use when grouping them together. While the second problem requires some evaluation and is detailed later on, the first shall be described in this section.

The data inside a block is a multidimensional sequence of samples. For example, consider a system using two sensors, each with one three-axis accelerometer and a three-axis gyroscope. Here, each sample is a 12-dimensional vector of values. The multi-dimensional nature of the data has to be taken into account and should be kept in mind when reading subsequent sections of this thesis. From the raw sensor data in a block, additional meta signals (same temporal domain and a function of raw sensor data) may be derived as well. With more sensors in use, the resulting sample dimensionality might be quite high.

Note that at this stage the data is assumed to be normalized. This is necessary, as the range of gyroscope data for example is much larger than the range of accelerometer data. To ensure equal influence of all sensor components, the range of each component should be roughly equal. However, a perfect normalization is not possible when processing streaming data. Even when maximal and minimal values have been determined for one case, as was shown in Table 3.1, this does not ensure an equal range with a different user or dance. For any given sequence of sensor values there is also no way to ensure that no future value ever exceeds the range of that sequence. Hence, an approximation is to be used for normalization. By recording a test set, an appropriate choice of normalization van be determined. This is no guarantee that normalized values never go beyond the $[-1, 1]$ interval but a reasonably good approximation.

There are a number of ways to compare two blocks of motion data. In Section 2.2 several of them were outlined. For this thesis, two kinds of approaches were evaluated. In the first one, blocks are compared based on features describing them. Hence, a transformation has to be defined, that maps whole blocks to such a feature space. The second approach works directly with the block data. This allows for a more fine-grained similarity computation, albeit at some computational cost. Both approaches are described in detail in the following sections.

4.3.1 Block Definition

For subsequent use, the notion of blocks of motion data is defined below.

Definition. *Let $\mathcal{B}$ be the symbol for a block of raw sensor values. Two blocks a and b can be distinguished as $\mathcal{B}^a$ and $\mathcal{B}^b$.*

Definition. *Let $|\mathcal{B}|$ be the number of dimensions for a block. For three sensors with six degrees of freedom (DOF) this would be 18.*

Definition. *Let $\mathcal{B}_n$ be the n-th sensor dimension with $1 \leq n \leq |\mathcal{B}|$. For example this might be all values for the x-axis of the first accelerometer.*

Definition. *Let $\mathcal{B}_{n_m}$ be the m-th sensor value of the n-th sensor dimension.*

Definition. *Let $\mathbb{B} \equiv \mathbb{R}^{nm}$ be the set of all blocks $\mathcal{B}_{n_m}$.*

Features

As noted above, block data is not limited to the raw sensor values. Instead, higher level features can be computed from the underlying sensor data. Several such possible features are described in this section. Algorithms determining the similarity of two blocks can do so by using any combination of raw sensor data and features.

Feature 1. *Instead of raw sensor values, the magnitudes could be used. Given $\mathcal{B}_1$, $\mathcal{B}_2$ and $\mathcal{B}_3$ as the x-, y- and z-axis of the first accelerometer, the magnitude (in euclidean norm) at position m is defined as: $\sqrt{{\mathcal{B}_{1_m}}^2 + {\mathcal{B}_{2_m}}^2 + {\mathcal{B}_{3_m}}^2}$. The sequence of said magnitude can be written as: $||\mathcal{B}_{\{1,2,3\}}||$.*

Feature 2. *Instead of the absolute magnitude of an accelerometer or gyroscope, the relative magnitude can be used. This denotes how the magnitude of one sensor relates to the mean magnitude of all sensor (for both, accelerometer and gyroscope).*

Thoughts on Feature Choice

While an evaluation on which features perform better is left for Chapter 6, a general idea of each features strengths and weaknesses can already be estimated at this point.

Using the magnitude as base for a feature should be a valid choice, if movement is not constrained to one axis. If a dancer were to jump up and down, for example, only one axis would be significant and using the magnitude would be a subpar choice. In general, though, dance movements are not constrained to one DOF and will also not be axis-aligned. As a general measure of activity, under the assumption of free motion, the magnitude therefore seems like one appropriate choice.

The relative magnitude could be useful to determine whether one sensor has recorded significantly more movement than any other one. If one would flail one's arms while standing still this would be the case. However, sensors distributed over a body are not independent from each other. A movement influencing the hips will very likely also influence readings in the arms. While this problem also exists for magnitude measurements, it is less of a problem there, as each sensor is handled independently. With relative magnitudes, this would lead to rather similar values which do not lend themselves well as a differentiating trait.

4.3.2 Whole Block Level Similarities

As stated above the movement inside a block can be described via a set of movement features. Based on those, blocks can be compared. Based on the formal definition of blocks and their features given above, potential block level features can be defined. Note that a block level feature is different from the features described earlier. While those were used to define abstractions from the raw sensor data, they were still in the same temporal domain as the raw sensor data itself. Block level features describe aspects of a whole block. They are a further abstraction on top of the previous one.

Block level features are defined as simple statistical mappings for any dimension of a block. Thus, they can be used for raw sensor data or any derived feature data.

Block Feature 1. *The mean of all sensor values or features of some dimension n:* $\overline{\mathcal{B}_n}$.

Block Feature 2. *The variance of all sensor values or features of some dimension n:* $Var(\mathcal{B}_n)$.

Block Feature 3. *The maximum or minimum of all sensor values or features of some dimension n:* $\max(\mathcal{B}_n)$ *and* $\min(\mathcal{B}_n)$.

Finally, the block's data could also be transformed into the frequency domain for a set of additional features. This is done on a per-dimension basis of the block and hence only requires a one-dimensional fast Fourier transform (FFT). When transforming a signal, it is first multiplied with the *Hann* function, given as

$$Hann(n) = \frac{1}{2}\left(1 - \cos\left(\frac{2\pi n}{N-1}\right)\right).$$

This prevents spectral leakage in the resulting frequency data. The windowed signal is also zero padded to ensure that its length is a power of two.

Block Feature 4. *Given the FFT of a signal $\mathcal{B}_n$, the dominant frequency can be used as a block feature.*

Block Feature 5. *Similarly, the spectral power, as the sum of all FFT coefficients, could also be used.*

Upon closer examination of he given block features, some aspects are noticeable. The dominant frequency block feature, for example, will vary if dance movements change in speed. If the steps in all movements are synchronized to a constant meter, this value will also be constant. Thus, this block feature is only of interest for dance sequences of varying tempi. Similarly, the spectral power is also a poor feature in those circumstances.

When choosing between mean, variance, maximum and minimum for block features, some other aspects should be kept in mind. The maximum and minimum of a sequence are easily influenced by outliers and thus might not provide a reasonably general abstraction. In contrast, both the mean and the variance are much more descriptive. While the mean itself does not provide a sufficient measure of activity, it still is a rough estimate thereof. Similarly, the variance is a good indicator for movement activity (strong movements result in higher values), but does not signify, whether this activity took place at a higher or lower level. Both could be used together, but in general the variance seems like the more promising indicator. Specifically, it allows to distinguish passages with more energetic movements from passages with less vivid ones. While working on activity recognition instead of dance pattern recognition, Ravi et al. [53] also compared different choices for feature vectors. They found that e.g., the standard deviation of their sensor data was the most important value in their feature vectors.

When an appropriate set of block features has been chosen, each block can be mapped to that feature space. For a set of n block features, a feature transform is thus defined as

$$f : \mathcal{B} \rightarrow \mathbb{R}^n.$$

This transform is applied to all blocks with the resulting feature vectors being used to compute a similarity measure between individual blocks. This can be done using any given distance function.

Distance Function

Assuming an appropriate set of features to be used was chosen, blocks need to be compared with respect to them. The basic approach here is to use a distance function, that provides such a similarity measure. Such a function needs to be defined accordingly as

$$d : \mathbb{R}^n \times \mathbb{R}^n \rightarrow \mathbb{R}.$$

Which, for two feature vectors $\mathbf{p}$ and $\mathbf{q}$, is given as

$$d\left(\mathbf{p}, \mathbf{q}\right) = ||\mathbf{p} - \mathbf{q}||.$$

A number of distance functions could be used. Four possible ones are:

Manhattan distance:

$$||\mathbf{p} - \mathbf{q}|| = \sum_{i=1}^{N} |\mathbf{p}_i - \mathbf{q}_i|$$

Euclidean distance:

$$||\mathbf{p} - \mathbf{q}|| = \sqrt{\sum_{i=1}^{N} \left(\mathbf{p}_i - \mathbf{q}_i\right)^2}$$

Cosine distance:

$$||\mathbf{p} - \mathbf{q}|| = 1 - \frac{\mathbf{p} \cdot \mathbf{q}}{||\mathbf{p}|| \, ||\mathbf{q}||}$$

Mahalanobis distance:

Given the covariance matrix S is defined as

$$\|\mathbf{p} - \mathbf{q}\| = \sqrt{(p-q)^T S^{-1} (p-q)}.$$

If the covariance matrix is diagonal this can be simplified to

$$\|\mathbf{p} - \mathbf{q}\| = \sqrt{\sum_{i=1}^{N} \frac{(\mathbf{p}_i - \mathbf{q}_i)^2}{\sigma_i^2}},$$

where σ_n is the standard deviation of the *n*-th dimension in the dataset.

From the given distance metrics, the Mahalanobis distance could not be used as the covariance matrix can not be fully determined from streaming data. It could be an option in more tightly defined settings. In those cases, an appropriate approximation to the covariance matrix could possibly be computed using sample data. With respect to the three remaining options, Salzberg [56] showed, for nearest neighbor searches, that the Manhattan distance is inferior to the euclidean distance, albeit only slightly. Similarly, Qian et al. [51] showed that the cosine distance and the euclidean distance perform almost equally well in nearest neighbor searches. Ultimately, the cosine distance was chosen for this thesis. it is noteworthy, that the cosine *distance* and not the cosine *similarity* was chosen. The cosine similarity values range from 0 to 1 with 0 signifying complete independence of the two input vectors and 1 an exact match. However, as a distance measure is required, this range is inverted.

4.3.3 Block Sequence Similarities

While the previous section presented an approach to compare blocks via a set of block feature descriptors, another possibility is outlined in this section. Instead of comparing abstracted features for whole blocks, the block data itself can be used in a comparison. Thus, such an algorithm should be able to determine the similarity of any two given multi-dimensional time series. However, before dealing with that algorithm one should reexamine the sequences to compare.

As noted in Section 4.3.1, each block can be seen as a collection of data sequences – one for each signal dimension. Additionally some features on such data, like the accelerator magnitude or the relative gyroscope magnitude, were

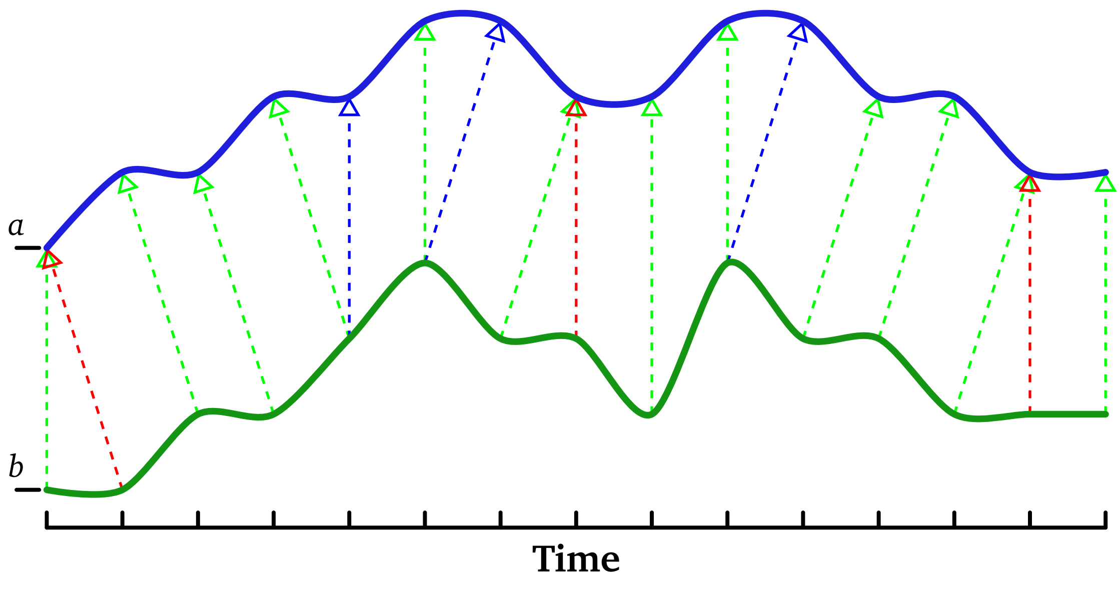


Figure 4.1: Dynamic time warping from signal b to signal a, showing substitution, deletion and insertion

defined as well. Thus, a block of raw data could e.g., be mapped to a block of magnitude data or instead of all sensors in a block, only the first two could be used. In general, a transform is applied that might change the dimensionality of the block data:

$$f : \mathbb{B} \to \mathbb{B}$$

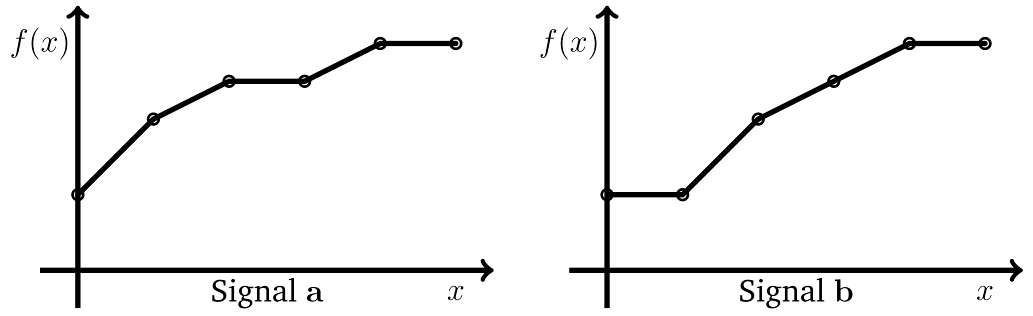
$$f\left(\mathcal{B}^a\right) = \mathcal{B}^b \quad \text{where} \quad |\mathcal{B}^a| \neq |\mathcal{B}^b| \vee |\mathcal{B}^a| = |\mathcal{B}^b|$$

This block transformation is applied to each block, which are then compared pairwise using the dynamic time warping (DTW) algorithm [54], described below. For each two compared blocks, the DTW algorithm yields a distance measure which is subsequently used for classification. As a normalized distance function is used (for details see text on DTW below), the resulting distance values will range from 0 (the same) to 1 (independent).

Dynamic Time Warping

In Dynamic Time Warping, a mapping from an input sequence to a given sequence is found that minimizes the distance between them. The sequences do not need to be of equal length, as DTW corrects for differences in speed, but need to be of the same dimensionality. Running the DTW algorithm on two sequences will yield two measures: a distance between those two sequences, and a so-called *warp path* that describes the best possible alignment of the two sequences. The best possible alignment is that alignment, which minimizes the overall distance between the two sequences.

Note that the DTW definition given in this thesis is based on an article by Salvador and Chan [55]. However, some slight notational changes have been made. When defining DTW, it is helpful to use one-dimensional sequences as examples. Let $\mathbf{a}$ and $\mathbf{b}$ be two such given sequences, whose similarity we would like to determine.



It is readily apparent, that those two sequences are very similar. Sequence b lags behind in the beginning, but catches up later on. Compensating for such small timing differences is one of the beneficial properties of DTW. This is done by warping sequence $\mathbf{b}$ to better fit sequence $\mathbf{a}$. Formally, sequences $\mathbf{a}$ and $\mathbf{b}$ of lengths $|\mathbf{a}|$ and $|\mathbf{b}|$ are defined as:

$$\mathbf{a} = a_1, a_2, \ldots, a_i, \ldots, a_{|\mathbf{a}|}$$

$$\mathbf{b} = b_1, b_2, \ldots, b_j, \ldots, b_{|\mathbf{b}|}$$

Given those two sequences, a warp path W is to be determined, that describes how sequence $\mathbf{b}$ maps to sequence $\mathbf{a}$.

$$W = w_1, w_2, \ldots, w_K \quad \max(|\mathbf{a}|, |\mathbf{b}|) \leq K < |\mathbf{a}| + |\mathbf{b}|$$

$$w_k = (i, j) \quad (1 \leq i \leq |\mathbf{a}|) \wedge (1 \leq j \leq |\mathbf{b}|)$$

The beginning and the end of a warp path are always given, as the beginning and end of each sequence. Warping is only performed inside that window.

$$w_1 = (1, 1)$$

$$w_K = (|\mathbf{a}|, |\mathbf{b}|)$$

Furthermore, a warp path is required to be monotonically increasing and to only be increasing in steps of size one (no part of a sequence may be skipped).

$$w_k = (i, j), w_{k+1} = (i', j') \quad i \leq i' \leq i + 1 \wedge j \leq j' \leq i + 1$$

An overall distance between two sequences can be computed using the warp path. Given a distance function d to use (see Section 4.3.2), this is defined as:

$$\|W\| = \sum_{k=1}^{K} d\left(\mathbf{a}_{w_{ki}}, \mathbf{b}_{w_{kj}}\right)$$

For normalization purposes, this can be adapted to take into account the length of the warp path.

$$\textit{Normalize}\left(\|W\|\right) = \frac{\|W\|}{|W|}$$

To find the warp path that minimizes the overall distance, a dynamic programming approach is taken. In a naive implementation a cost matrix D of size $|\mathbf{a}| \times |\mathbf{b}|$ is constructed and incrementally filled. Each cell in the cost matrix contains the aggregate cost of the best warp path up to this point. The algorithm starts at $D_{1,1}$, the cell denoting the distance between the first element of the two sequences, and terminates at $D_{|\mathbf{a}|,|\mathbf{b}|}$, the end of both sequences. Filling the cells of the cost matrix is done in an incremental fashion on a column by column basis. Each new cell can be seen as an extension of the underlying warp path and three different cases can be distinguished:

Deletion:

$$w_k = (i, j), w_{k+1} = (i, j + 1)$$

Substitution:

$$w_k = (i, j), w_{k+1} = (i + 1, j + 1)$$

Insertion:

$$w_k = (i, j), w_{k+1} = (i + 1, j)$$

The effect of these operations is also shown in Figure 4.1. When computing the value of a new cell, that operation is chosen, which least increases the total

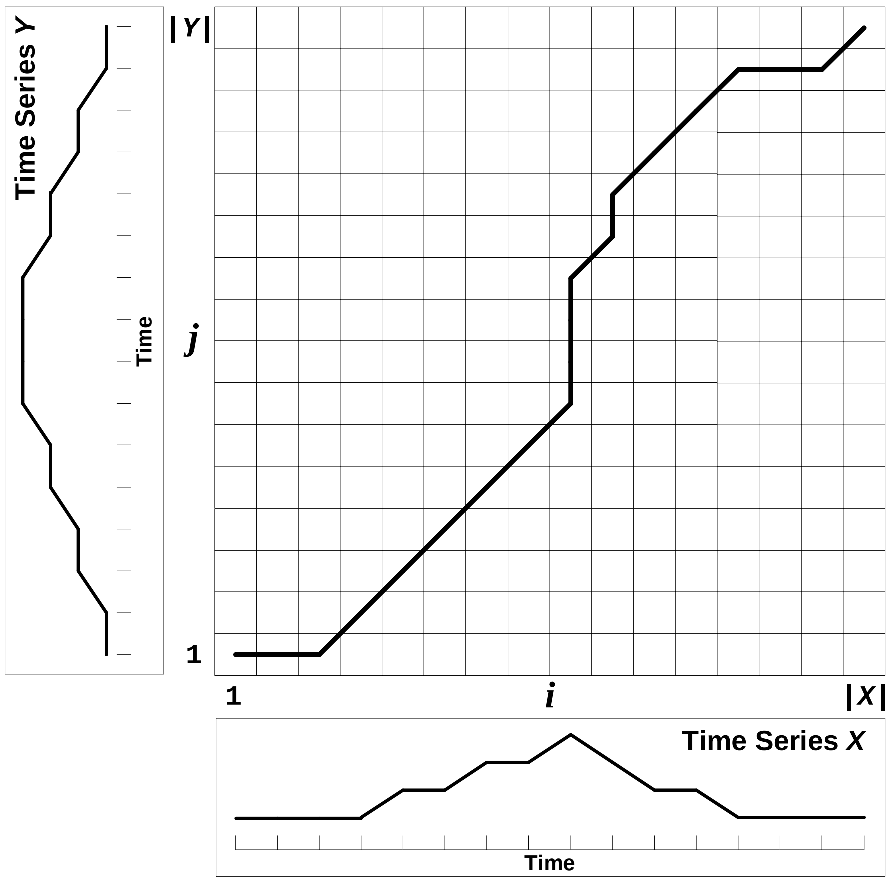


Figure 4.2: Two sequences and their minimum-distance warp path (from Salvador and Chan [55])

warp path cost. For the cell $D_{i,j}$ and distance function d (see above), this is formally given as:

$$D_{i,j} = d\left(\mathbf{a}_i, \mathbf{b}_j\right) + \min\left(D_{i-1,j}, D_{i,j-1}, D_{i-1,j-1}\right)$$

After the cost matrix has been computed, the actual warp path can be determined by a reverse search, starting at $D_{|\mathbf{a}|,|\mathbf{b}|}$. At every cell, the predecessor with the lowest cost is chosen and added to the warp path until the beginning of the cost matrix is reached. An example for two sequences and the resulting warp path is shown in Figure 4.2.

While the above definitions and examples were given for one-dimensional sequences, no changes have to be made to the algorithm to work on multi-dimensional ones. Only the distance function d has to be able to handle multi-dimensional input.

To lower the computational cost of DTW, constraints (see e.g., an article by Lemire [38]) can be used to restrict the number of cells that are calculated. Research by Ratanamahatana and Keogh [52] has provided evidence, that a lack of constraints does not improve on DTW results and accuracy already peaks in severely constraint cases. They furthermore showed, that total computational cost of DTW when using constraints is essentially $\mathrm{O}(n)$, compared to the $\mathrm{O}(n^2)$ complexity of a naive implementation. Note that the space cost for the cost matrix can be reduced to the cost of two columns if the warp path is not needed. In that case only the current and last column are kept in memory, which is sufficient to compute the total warp path cost, but does not allow the reverse search to determine the mapping itself.

FastDTW

For this thesis an improvement on DTW by Salvador and Chan [55] (see Figure 4.3, for an illustration) was used. The *FastDTW* algorithm speeds up DTW computations, by iteratively computing the warp path for a lower resolution, projecting it to a higher resolution and refining it. Lower resolution sequences are computed by averaging samples in a higher resolution, each coarsening reducing the number of data points by a factor of two. In the coarsest resolution the warp

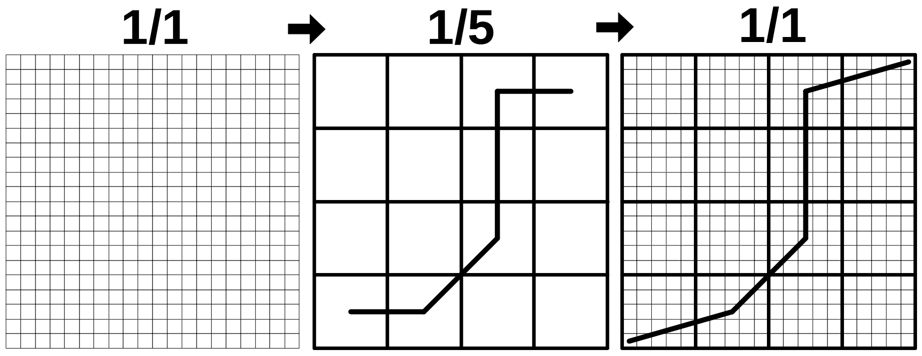


Figure 4.3: Speeding up the warp path computation using multiple resolutions (from Salvador and Chan [55])

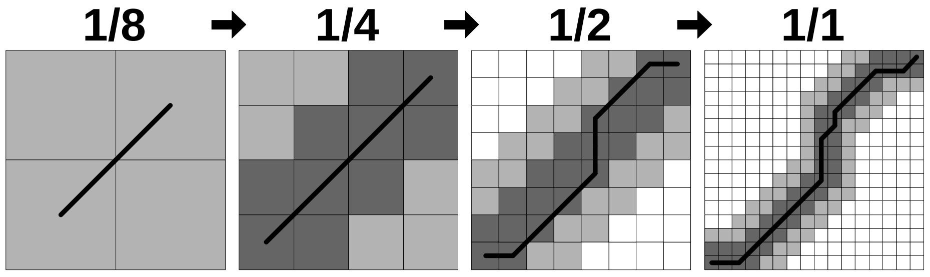


Figure 4.4: Successive refinement of the warp path (from Salvador and Chan [55]). Cells shaded in dark gray are covered by the coarse resolution warp path and are included in the refinement. Light gray colored cells are also included as they lie in a given radius around the warp path.

path is computed equivalent to a regular DTW implementation. When projecting that coarse path to a higher resolution, a more restricted DTW is run. The number of cells required for refinement is equivalent to all cells covered by the path (with a downsampling factor of two in both dimensions this equals four samples per path segment) plus an added boundary of cells in a given radius around the path (see Figure 4.4).

Using that multilevel approach, *FastDTW* works in $O(n)$, similar to constrained versions of DTW. Due to the nature of the algorithm a certain level of error is induced. This is primarily dependent on the radius used. Analysis by Salvador and Chan showed, that for higher radii the error converges to 0. Furthermore, this convergence is significantly faster than in constraint based DTW implementations. An analysis of appropriate *FastDTW* radii for the movement data used here, is given in Chapter 6.

4.4 Classification

In the previous section two algorithms were presented that compute a distance measure for blocks of motion data. Now, further processing steps have to be taken to classify blocks into distinct groups. As no *a priori* knowledge on possible patterns is given, classification is unsupervised and solely depends on statistical information. Furthermore, classification should work in real-time on streams of incoming motion data blocks. Unfortunately, having no set of labels given and the requirement to assign new labels to blocks as they come in, imposes severe restrictions on the classification. For example, a simple *k*-nearest neighbors (*k*-NN) clustering would not work, as the number of desired clusters *k* is not known. Changing assigned labels later on is also undesirable. Thus, algorithms determining clusters reexamining the classes for all available data from the set do not work. An algorithm is needed that preserves previous class assignments and classifies new blocks based on previous classification choices.

For the purposes of this thesis, a threshold clustering approach is proposed, that assigns class labels on-the-fly and unsupervised. Assigned class labels are immutable as demanded above. In this approach, an input stream of motion data blocks shall be modeled as a sequence S, with

$$S = (s_1, s_2, s_3, \ldots) \quad \text{with } s_n \in \mathbb{B}.$$

Individual clusters are modeled as sets of blocks C, while all found clusters go into sequence L. A cluster's index in this sequence also serves as class label.

For the first incoming block $S(1)$, there is no decision to be made. It will be assigned a new class label, thus resulting in

$$L = (\{S(1)\}).$$

For subsequent blocks, the best matching preexisting cluster has to be determined. This assumes a distance function $dist$ is defined, which for the purposes of this thesis will be one of the two above-described ones. In addition to that function, a threshold t has to be provided as well. The distance from a new block $S(i)$ to an already found cluster C in L, is defined as the average distance to C's members.

$$clusterDistance\left(S(i), C\right) = \frac{1}{|C|} \sum_{c \in C} dist\left(S\left(i\right), c\right)$$

The best match of a new block is hence given as the cluster with the smallest distance to, from all available clusters.

$$bestMatch\left(S\left(i\right), L\right) = \min\left\{clusterDistance\left(S\left(i\right), L_n\right) : 1 \leq n \leq |L|\right\}$$

The identifier and the distance to that best match shall be denoted as

$$bestMatch\left(S\left(i\right), L\right)_{ID}$$

$$bestMatch\left(S\left(i\right), L\right)_{dist}.$$

Based on the given threshold t, a new block is considered to belong to the best match's cluster if the distance between them is lower than t. If that is not the case, a new cluster is created.

$$assign\left(S(i), L\right) = \begin{cases} L_{|L|+1} = \{S(i)\} & \text{if } bestMatch\left(S(i), L\right)_{dist} > t \\ L_{bestMatch(S(i),L)_{ID}} \cup S(i) & \text{if } bestMatch\left(S(i), L\right)_{dist} \leq t \end{cases}$$

The threshold has to be chosen carefully for good results. If it is set too low, all blocks are considered unrelated and are assigned their own class. Conversely, a threshold value that is too high, leads to undesired mashing of blocks that should have been distinguished. The threshold value is dependent on the moves used and the features chosen. In Chapter 6, results for different threshold values are shown in detail.

Chapter 5

Evaluation

When evaluating the system, a number of components need to be in place. First of all, a client application has to be available to interface with the wearable sensor system and collect the data. One option then would be to analyze the data while it is recorded. However, it is desirable to decouple recording and analyzing, to allow for more flexibility. Especially exploring the parameter space is not viable in real-time. Thus, analyzing the data requires a second application that is able to run a set of tests on the recorded data. For exchange between those two applications, a data format needs to be specified.

To be able to test a dance movement classifier, the recorded data needs to have class labels. While labeling data manually after recording is possible, it would be a tedious and error-prone process. Instead, it is preferable to do so automatically. For this purpose, a sequence of dance movements is defined before recording and used to provide instructions to participants and in evaluating the classifier. To do so, a file format for dance move sequences has to be defined as well. Also, the recording application has to be able to use that data to generate instructions for participants. These instructions need to be synchronized to the audio and displayed in an easily consumable way.

In general, all time critical parts of the system should be highly synchronized when recording. For example, the timestamps assigned to the incoming sensor data need to be accurate. They also need to be right in respect to the audio playback. As described in Section 4.1, beat locations are precomputed and those timestamps need to be taken into account as well. The recording application has to be able to keep those three time scales in sync and should minimize any delays.

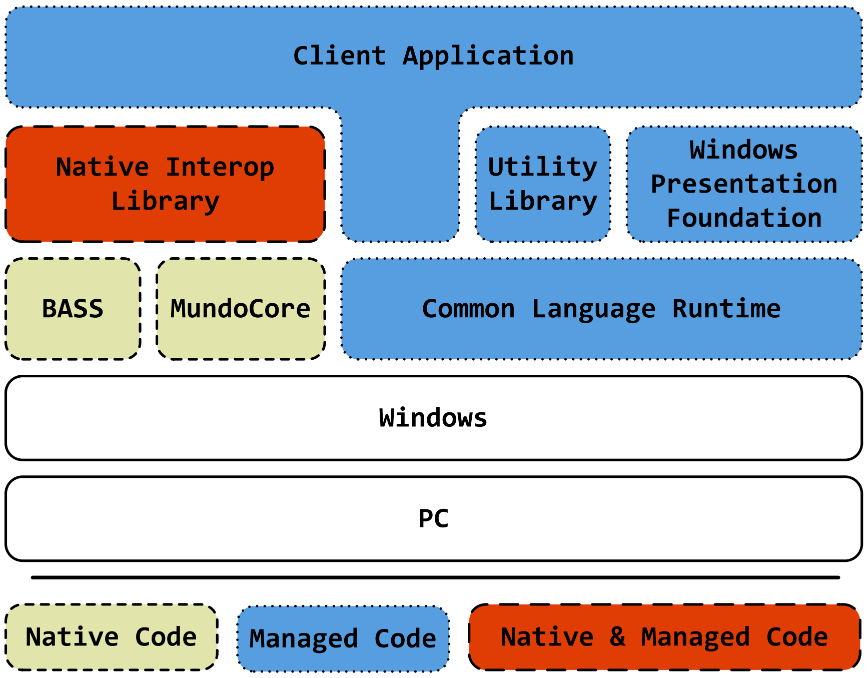


Figure 5.1: Client Software Stack

Finally, the evaluation application should be able to efficiently run a set of tests on the recorded data. Ideally tests can be defined without recompilation and in a convenient way. In this chapter, the implementations for all parts of that system are described.

5.1 Recording Application

Recording a dance session is a critical part of the evaluation. The data collected here will be further analyzed, which requires that possible sources of error in the recording process are minimized. This especially relates to time-critical components and user errors. This section first details how those time-critical components were built and afterwards describes how recording works and what measures were taken to minimize sources for user error.

5.1.1 Time Critical Modules

When recording, it is important to keep track of timing and any extensive delays in critical parts of the application could skew the results. Therefore, all time critical work is done in native code. For this purpose, a `C++` class library was implemented to perform critical operations in native code while being accessible by managed code. It handles audio playback through the BASS[1] audio library (available for Windows and Mac OSX and free for non-commercial use) and also links to MundoCore (see Section 3.3).

The beat data, described in Section 4.1, is loaded by this library and used to generate events during audio playback. The BASS library allows for setting callbacks on audio streams, which is used to set callbacks at every beat's position. However, as the audio pipeline introduces some delay, some time passes between a stream being instructed to play and the sound being audible on the speakers. The BASS library allows to query for the average duration of that delay. This value is added to the beat timestamps when setting the callback times. Thus, a callback is triggered not when a sample at a beat's time is decoded but when it is actually output on the speakers. Each callback triggered calls a delegate function, previously passed to the audio player instance, and passes on a timestamp for the current beat. This timestamp is not based on the beat definition but is using a high accuracy timer singleton inside the library. In that way beat times and sensor times are using the same timescale.

To interface with the wearable system, MundoCore is used on the client side as well. For this purpose, a MundoCore service is implemented that subscribes and publishes on the same channel as the embedded system. Thus, these two components of the system do not need to be directly coupled but only need to know a common channel name. The client can send messages via that channel to start or stop recording on the embedded system. When recording is started, sensor data is also coming in on that same channel. As described in Section 3.3, sensor data is transmitted as a binary blob. On the client, that data is unpacked and transfered from its native array format into a managed collection. Also, a timestamp is assigned the moment that data is coming in, using the same timer singleton as the beat events. In order to not block the MundoCore service, sensor data is put into a thread-safe queue before further processing in the recording application. As GUI updates triggered by new sensor data can take some time, decoupling the data collection from the GUI is preferable. Otherwise, the data

[1]Available at `http://www.un4seen.com` (last accessed on March 22nd 2010)

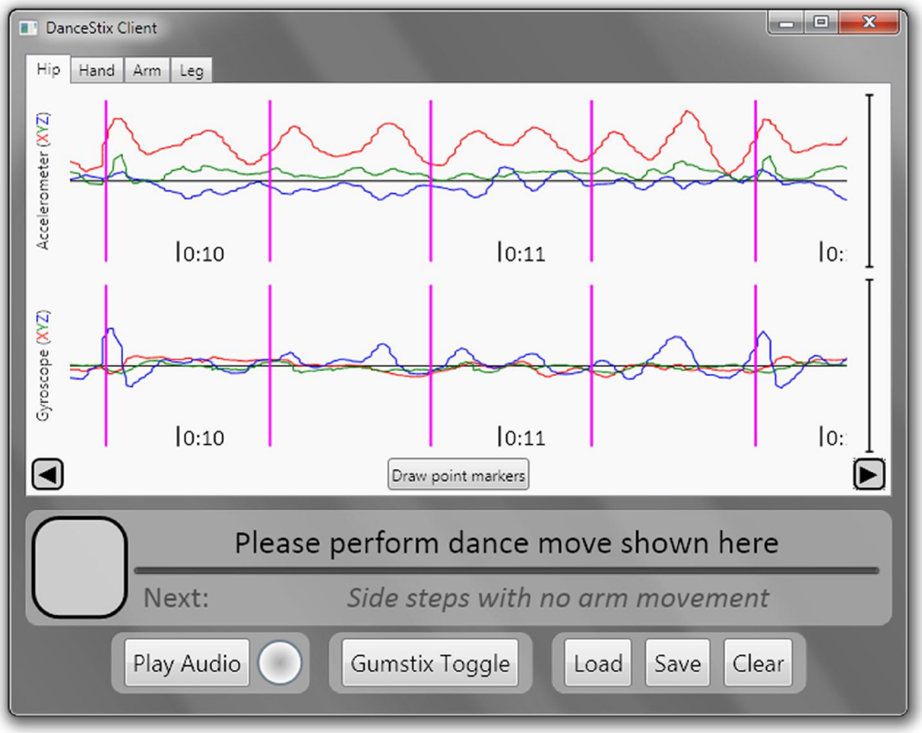


Figure 5.2: Client Software showing a recorded session

receiving thread might stall and incoming sensor data would not be handled immediately.

5.1.2 User Interaction

When recording, participants have to be instructed on what moves to perform at every given time. This information needs to be highly visible and informative. In combination with instructions on the current moves to dance, it is desirable to provide instructions on the next moves as well. This eases transitions to new segments and prepares participants for upcoming changes in style. One possibility would be to simply identify the upcoming moves. However, without a countdown timer, showing *when* that change will occur, this information is not sufficiently useful. When providing timing information, it is advisable to inform participants of the current progress also; Hence, making it clear how much time is remaining.

Furthermore, some feedback on the status of the recording is desirable as well. This primarily includes feedback whether the data collection is working properly.

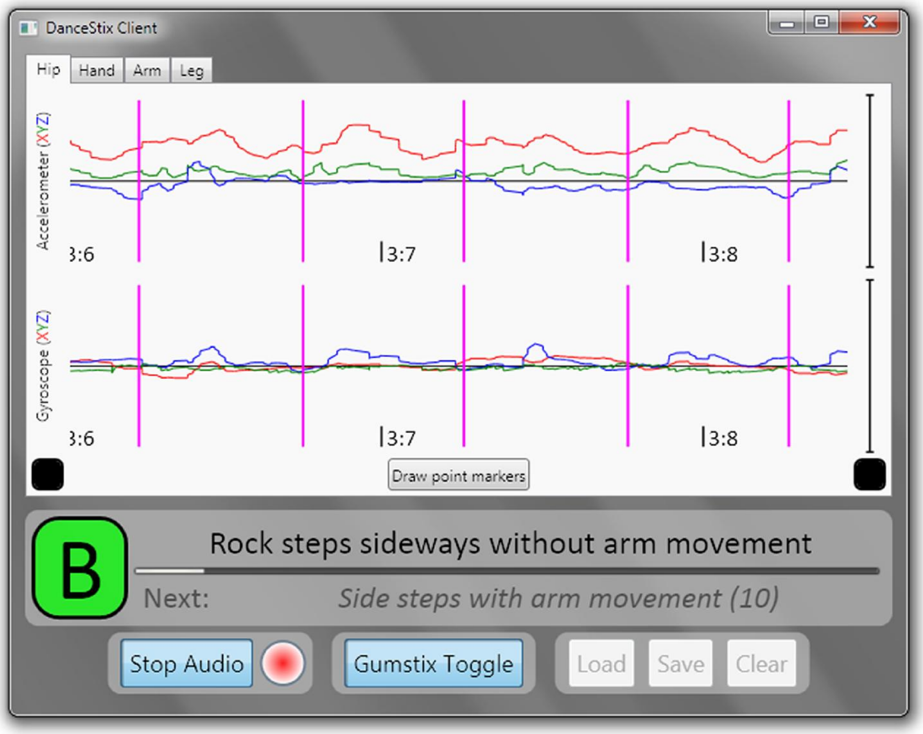


Figure 5.3: Client Software during a recording session

Additional data, like beat information or segmentation could be provided as well.

The actual application used in recording a dance session is shown in Figure 5.2. Additionally, Figure 5.3 shows the application during a recording session. It contains three different areas. In the upper area of the window is a graph showing the recorded sensor values. A tab control is used with one tab for each available sensor. A separate graph for accelerometer and gyroscope data is shown inside the tab control. When recording, this graph is updating live to show the sensor values currently coming in. In addition to the sensor values, beats are denoted in the graph with pink vertical bars and time labels are added at second intervals as well. If the application is not currently recording, the graph can be scrolled using buttons below it. There is also an option to draw point markers to better identify the samples in the graph. This is useful in detecting whether the time interval between samples is indeed constant. If markers are unevenly spaced in the graph, this could, for example, indicate connectivity issues. In general, this area provides feedback as to whether sensor data is recorded correctly.

In the area in the middle, instructions are provided to participants when dancing. On the left, a class identifier is shown to the participant. This is done to provide a more easily recognizable sign for the current pattern. In addition to the label, each class is also assigned a color which is shown in this area as well. Taking colors across the entire hue range makes sure class color identifiers are distinguishable. In addition to this, the instructions for the current steps and for the upcoming ones are shown. Next to the upcoming steps label, the participant is also informed about how many more beats the current step pattern will remain active. In this way, participants have an expectation of when to switch dance movements. Furthermore, the bar in the middle fills up as the recording progresses, giving participants a rough idea of how much longer a recording is going to last.

The bottom of the window contains several buttons to control the application. This includes saving recorded sensor data or loading it again. By pressing the button in the middle, the embedded software on the Gumstix (see Section 3.2.3) is instructed to either start transmitting sensor values or to stop doing so. Finally, the button on the left starts playing the audio file or stops playback. While recording, a glowing orb next to that button is pulsating in sync with the beats. To ensure a smooth start of a recording, a lead-in is added to the audio. When the play button is pressed, the music does not start right away, but a click track will count down one full measure. The intervals in the click track are interpolated from the beat information and hence based on the song playing.

5.1.3 Dance Description Format

As described earlier, the dance movements to be danced have to be predefined to enable an evaluation of the proposed algorithm. In the previous section, it was also outlined how participants are informed about what to do when recording. In this section, the `.steps` file format, which can be used to define those instructions, will be described. In this format, a beat level movement description is used. However, movements can be defined that span more than one beat. Thus, the format can be adapted to most dance scenarios. Note that this is not a full dance movement description, like the *Laban* [65] or *Benesh* [8] notation formats. The purpose of the `.steps` format is to describe sequences of movements but not the movements themselves. Individual movements are only assigned instructional labels. Those can either be read and acted out directly by participants or explained and shown to participants beforehand.

Fields in a `.steps` file are identified by the first keyword on the line. In general, a `.steps` file contains a header and a body. However, they are not specifically denoted and in fact no strict order of definition has to be adhered to. All class identifiers are positive integers excluding zero. Macros can be defined and use string identifiers instead of numerals. Any line not starting with any of the keywords is considered a comment. Currently available keywords are:

```
CLASS, SUBCLASSES, STEP, STEPS, DEF, ENDDEF, PAT
```

The CLASS keyword is used to define new dance classes. If needed, those can be further broken down as will be described. For each class, a numerical identifier and a textual description need to be provided. Such a definition might look like this:

```
CLASS 1 Side steps with no arm movement
CLASS 2 Rock steps sideways without arm movement
CLASS 3 Jumping around
```

For some movements, it might be interesting to define submovements. For example, in cha-cha-cha the base step is defined over two measures and is subdivided into two steps (starting on the second beat) followed by a chasse over two beats and another two steps and a second chasse to return to the starting position. Overall, the movement spans eight beats. As can be seen, the base step class could be split into four distinct subclasses. While one could argue that the cha-cha-cha base step only contains two different subpatterns, those are executed in different directions and thus result in differing steps. The `.steps` format allows definition of such subclasses via the SUBCLASSES keyword. If used, one first provides the identifier of the class to further specify, followed by a list of new movement identifiers. For example, if class 1 can be further subdivided in four distinct subclasses one would write:

```
SUBCLASSES 1 101 102 103 104
```

The STEP and STEPS keywords define actual steps to be danced. For those two commands the order in which they are executed is important. While they do not all need to be specified in one block, they are parsed top to bottom in the `.steps` file. The STEP keyword only defines which class to use in one beat period. If a movement stretches over multiple beat periods, the STEPS keyword can be used with an additional length parameter added after the class parameter. The step description is independent of time signature and solely defined on the beat level. Thus, a movement spanning an entire $^4/_4$ measure, for example, has to be defined with a step length of 4. An example of a movement over two measures is given below.

```
STEP 1
STEPS 2 3
STEP 3
STEPS 1 3
```

Some movement sequences might be reoccurring throughout a song. The DEF, ENDDEF and PAT keywords are used to define such macros and add them to the step sequence. The DEF and ENDDEF keywords act as block delimiters. When starting a block, a string identifier has to be added after the DEF keyword. This identifier is later used by the PAT keyword. Inside the block, steps can be defined for later use. When a PAT keyword is read later, the defined macro is added at that position. Inside a block, STEP and STEPS commands can be used to define the actual moves for it. It is also possible to use already defined macros in a macro definition via the PAT keyword. In the example below, an eight measure long macro for the chorus is defined and inserted.

```
DEF CHORUSPATTERN
STEPS 2 32
ENDDEF

PAT CHORUSPATTERN
```

An utility library provides functions to load `.steps` files. When loading the library provides options to already transform the step data before passing it on to the caller. For example instead of a list of class identifiers a list of subclass identifiers can be obtained. It is also possible to get a list of class identifiers per measure and not per beat.

5.1.4 Recorded Data Exchange Format

```
<!ELEMENT tgf (comment?, sync+)>
<!ELEMENT comment (#PCDATA)>
<!ELEMENT sync (names, data)>
<!ELEMENT names (name+)>
<!ELEMENT name (#PCDATA)>
<!ELEMENT data (#PCDATA)>
```

Type Definition 1: DTD for recorded data exchange format

As the recorded data has to be processed by another application, it has to be saved to disk in some format. In general, it is considered useful to be able to revisit previous recordings. For this purpose, the `.tgf` format is used. This extensible markup language (XML) based format has been defined by Hadjakos for a previous project utilizing the sensors and is inherited unchanged. A document type definition (DTD) for such a `.tgf` file is shown in Type Definition 1.

In each `.tgf` file, a number of `sync` blocks can be included. Each `sync` block denotes a collection of values which share a common clock. For example, a `.tgf` could include one block for sensor data and another one for metronome data. The actual data goes into `data` blocks inside the `sync` blocks. It is not described in XML form, but as a block of text with one sample set per line (for a three-axis accelerometer this would come out to four values, including a timestamp). The individual values are separated by whitespace and saved as textual floating point representations. Names for each data column are given in a `sync`'s `names` block. By convention the timestamp should be the last column of each line of data.

A `.tgf` file can be used to save recorded data from multiple participants and is also used to save beat data as well. While beat timestamps are already defined in another format (see Section 4.1), those timestamps are relative to the beginning of the song they were derived from. While a `.tgf` allows multiple `sync` blocks, each with a clock of its own, they are synchronized. A timestamp in one block also is a valid timestamp for any other clock. This allows for different sample timestamps for different blocks, but ensures that the sample timestamps share a common time base. The beat data is mapped to the recording timeline, as mentioned earlier, and included in the `.tgf` as well.

5.2 Evaluation Application

When evaluating, the recorded data is put through a number of tests to determine the quality of the classifier. When running a test, the question is how to determine the quality of the classifier. When the possible classes are known before classification, this is easy. In that case, each class classified as a different one is an error. However, here the classifier has no class definitions available and assigns class labels on its own. As described earlier, the classes for the intended sequence are available as well in the `.steps` format. However, those class identifiers are a completely different set than the assigned ones. The assumption that identifiers can be matched using simple equality therefore does not hold.

In this section, one approach is defined that maps identifiers from one space to another in a best-fit manner. This enables the determination of a classifier's accuracy. Furthermore, as it is desirable to define tests outside of the evaluation application's source code, a test format was defined. This XML format, which is loaded by this application at runtime, is described in this section as well.

5.2.1 Test Definition

The evaluation software will load an XML file defining all tests when running. This makes adapting tests easier and allows for running new tests without recompiling the evaluation application. For each test, it has to be defined which files to use, what features to test and with which parameters to do so. A DTD for the XML format used for this purpose, is shown in Type Definition 2.

As can be seen, three levels of grouping are available. The whole test set contains a number of test groups. Those groups themselves are nothing but a convenient way to easily exclude a number of tests at once. Each test inside such a group then is defined for one input file and writes all output to one directory. Inside a test, a feature case is used to describe various possible transforms on the

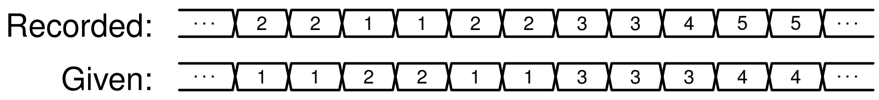
```
<!ELEMENT TestSet (TestGroup*)>
 <!ELEMENT TestGroup (Test*)>
   <!ATTLIST TestGroup   Name    CDATA    #REQUIRED>
   <!ATTLIST TestGroup   Active    (True|False)    #REQUIRED>
  <!ELEMENT Test (File, Directory,InstructionFile,Cases)>
     <!ATTLIST Test   Name   CDATA   #REQUIRED>
     <!ATTLIST Test   BeatGrouping   NMTOKEN   #REQUIRED>
     <!ATTLIST Test   Active   (True|False)   #REQUIRED>
   <!ELEMENT File (#PCDATA)>
   <!ELEMENT Directory (#PCDATA)>
   <!ELEMENT InstructionFile (#PCDATA)>
   <!ELEMENT Cases (FeatureCase+)>
    <!ELEMENT FeatureCase (Features, Sensors, Method, Thresholds)>
      <!ATTLIST FeatureCase   NormalizeCostMatrix   (True|False)   #REQUIRED>
      <!ATTLIST FeatureCase   Suffix   CDATA   #REQUIRED>
     <!ELEMENT Features (#PCDATA)>
     <!ELEMENT Sensors (#PCDATA)>
     <!ELEMENT  Method EMPTY>
      <!ATTLIST Method   xsi:type   (DTWMethod|FeatureMethod)   #REQUIRED>
      <!ATTLIST Method   Radius   NMTOKEN   #IMPLIED>
      <!ATTLIST Method   FeatureFunc   (Variance|Mean|All)   #IMPLIED>
     <!ELEMENT Thresholds EMPTY>
      <!ATTLIST Thresholds   Min   NMTOKEN   #REQUIRED>
      <!ATTLIST Thresholds   Max   NMTOKEN   #REQUIRED>
      <!ATTLIST Thresholds   Increment   NMTOKEN   #REQUIRED>
```

Type Definition 2: DTD for test definition input data. Different feature cases e.g., might use different radius settings for the dynamic time warping (DTW) algorithm.

5.2.2 Identifier Mapping

As noted above, the class labels assigned automatically and the class labels expected, are in two different label spaces. A mapping has to be determined to quantify the quality of a classifier. This section describes the algorithm used for that purpose.

Consider the following two sequences:



If identifiers had to match exactly, the above sequence would contain nine errors. Clearly, this would be wrong as e.g., class 4 in the given sequence and class 5 in the recorded sequence are matching up. The classifier just assigned identifiers differently. A human observer would see that identifiers can be mapped to achieve a better fit. For the sequence above, such a mapping (from recorded to given) would be:

$$\{1 \rightarrow 2, \quad 2 \rightarrow 1, \quad 3 \rightarrow 3, \quad 4 \rightarrow 3, \quad 5 \rightarrow 4\}$$

However, this mapping introduces another problem. Now the number of errors in the recognized sequence would be zero. Assigning two different classes, where only one class should have been recognized, is clearly an error. Thus, the mapping has to be an injective function and no two classes from the recorded sequence may map to the same class in the given sequence. We require:

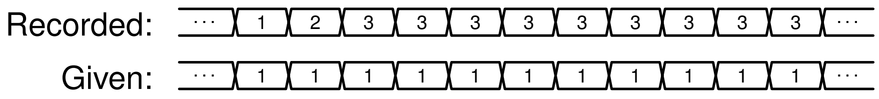
$$f(x_1) = y = f(x_2) \Rightarrow x_y = x_2$$

With that requirement in place, the mapping could be changed in two ways:

$$\{1 \rightarrow 2, \quad 2 \rightarrow 1, \quad 3 \rightarrow \emptyset, \quad 4 \rightarrow 3, \quad 5 \rightarrow 4\}$$

$$\{1 \rightarrow 2, \quad 2 \rightarrow 1, \quad 3 \rightarrow 3, \quad 4 \rightarrow \emptyset, \quad 5 \rightarrow 4\}$$

While the first mapping would result in a sequence with two errors, the second one would only result in a set with one error. Which mapping to chose here is not a clear choice though. Either mapping is valid. However for this thesis, when evaluating, the best mapping is chosen. In the given example, the class 4 in the recorded sequence would thus be considered a classification error while class 3 would not. Another aspect of that choice can be observed in this pair of sequences:



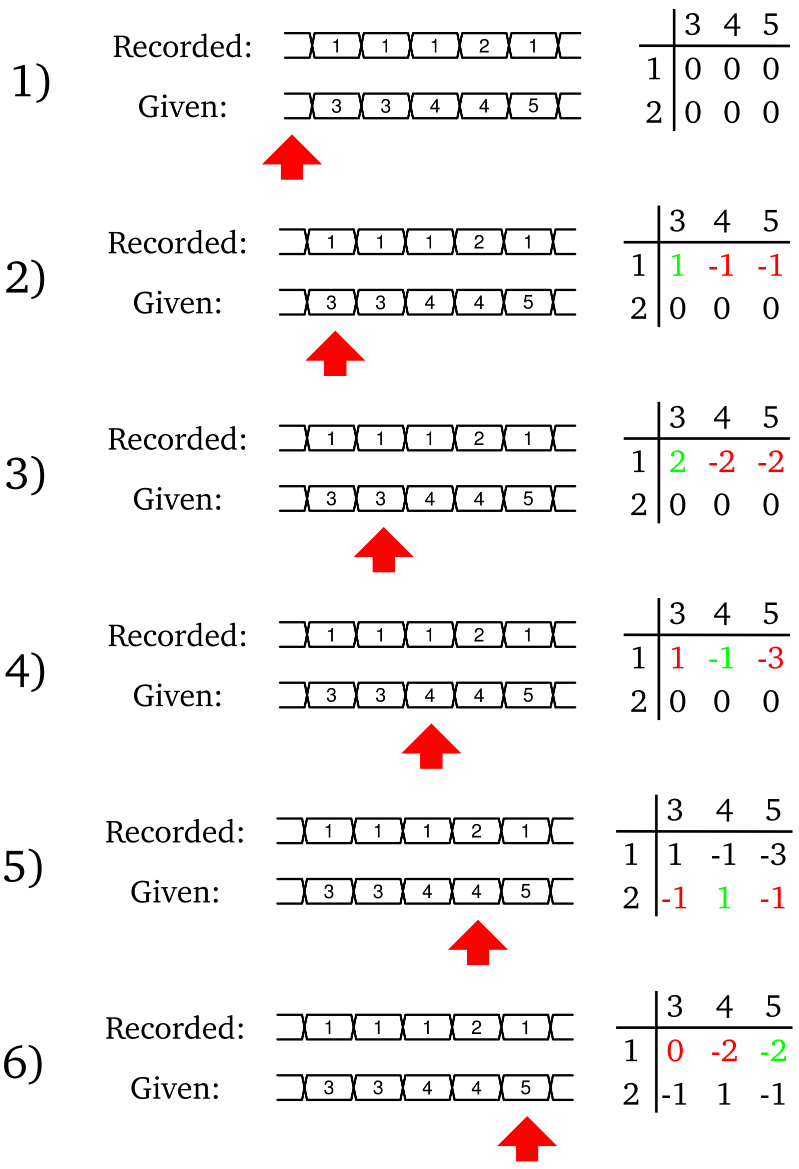
If the mappings were assigned in a first-come, first-served fashion the result would be:

$$\{1 \to 1, \quad 2 \to \emptyset, \quad 3 \to \emptyset\}$$

This would result in a bad fit of the two sequences. Mapping class three to class one, on the other hand, would be a much better choice. In general, small classification errors should not influence the overall classification score excessively. Therefore, a mapping should minimize the overall error.

When searching for the best fitting mapping, a naive approach would be to explore all possible mappings. However, for n different classes in the recognized sequence and m different classes in the given sequence, there are n^m possible combinations, if multiple assignments are allowed. For larger numbers of classes, this grows too fast for efficient computation. Given the required injective function constraint, the best mapping only has to be chosen from a smaller set and can be computed more efficiently.

A matrix buffer is used to aggregate a quality measure for each mapping. This buffer of size $n \times m$ is initialized to zero. Each recognized class identifier is assigned a row of the matrix and each given class is represented by a column in the matrix. Now both sequences are traversed. At each position in the sequence, a recognized class r and a given class g are read. Intuitively mapping $r \to g$ would yield a locally better result than any other mapping. Thus, the value in the corresponding cell of the buffer is increased by one while all other values in the row of r are decreased by one. Hence, higher values in a cell signify a better mapping while lower values denote the opposite. An example with two recognized classes and a given sequence with three classes is shown below:



1)

	3	4	5
1	0	0	0
2	0	0	0

2)

	3	4	5
1	1	-1	-1
2	0	0	0

3)

	3	4	5
1	2	-2	-2
2	0	0	0

4)

	3	4	5
1	1	-1	-3
2	0	0	0

5)

	3	4	5
1	1	-1	-3
2	-1	1	-1

6)

	3	4	5
1	0	-2	-2
2	-1	1	-1

After traversing the input sequences, the cost buffer can be used to determine the best mapping. To do so, in each iteration the highest scoring mapping is extracted and the corresponding row and column are deleted from the buffer. In case of ambiguity, a mapping is randomly chosen. This is done until all available options are used up. Below, this process is shown for the buffer computed in the previous example:

1) $\begin{array}{c|ccc} & 3 & 4 & 5 \\ \hline 1 & 0 & -2 & -2 \\ 2 & -1 & 1 & -1 \end{array}$

Choose best mapping: from class 2 to class 4

$\{1 \rightarrow \emptyset, \quad 2 \rightarrow \emptyset\}$

2) $\begin{array}{c|ccc} & 3 & \cancel{4} & 5 \\ \hline 1 & 0 & \cancel{-2} & -2 \\ \cancel{2} & \cancel{-1} & \cancel{1} & \cancel{-1} \end{array}$

Add to mappings and cross from matrix

$\{1 \rightarrow \emptyset, \quad 2 \rightarrow 4\}$

3) $\begin{array}{c|ccc} & 3 & \cancel{4} & 5 \\ \hline 1 & 0 & \cancel{-2} & -2 \\ \cancel{2} & \cancel{-1} & \cancel{1} & \cancel{-1} \end{array}$

Choose best mapping: from class 1 to class 3

$\{1 \rightarrow \emptyset, \quad 2 \rightarrow 4\}$

4) $\begin{array}{c|ccc} & \cancel{3} & \cancel{4} & 5 \\ \hline \cancel{1} & \cancel{0} & \cancel{-2} & \cancel{-2} \\ \cancel{2} & \cancel{-1} & \cancel{1} & \cancel{-1} \end{array}$

Add to mappings, cross from matrix and terminate

$\{1 \rightarrow 3, \quad 2 \rightarrow 4\}$

This process is guaranteed to terminate as the matrix is reduced in each iteration. It will also choose the best possible mapping. In each step, the best mapping is extracted, thus no other possibly better choice is ignored.

When reducing the cost matrix, one can also take the class frequencies into account. One error in a class that occurs twenty times should be less significant than one error in a class that only occurs five times. This normalization is done after a cost matrix has been built. In an optional additional step, each element in the cost matrix is divided by the number of times its given class is occurring. The following cost matrix reduction is equivalent to the above shown one, but is working with floating point numbers instead of integers. While the cost function used in the first case is one that tries to minimize the total number of errors, the second one tries to do so with the relative amount of errors.

5.2.3 Output

For each test run, a number of data files are generated. They are further processed using a set of Python scripts to generate graphs. By using an additional set of script for that purpose, results from multiple test can easily be compared and combined. The intermediate files used are:

Threshold Error Information For each tested threshold value (see Section 4.4), the number of found errors is outputted.

Best Error Plot Given the result with the best performing threshold, a list of block errors is generated. For each block in the sequence, a zero is written to the output when the classification worked correctly and a one if the classification was erroneous.

Recognized Classes In line with the previous output, the assigned class label sequence of the best result data is saved.

These intermediate files can be used to e.g., create graphs showing:

- The performance over a threshold interval
- Comparisons of different feature sets
- The classification errors over time

5.3 Evaluation Setting

For the evaluation, a song to use and an according choreography were to be defined. The decision ultimately was made to use a remix of Lady Gaga's song "*Just Dance*" for this purpose. It is 4 minutes and 54 seconds long, at a speed of 119bpm. With a prominent bass drum track it allows for good beat detection and contains no ambiguous parts, that could possibly throw off participants. At 119bpm, it is also in the lower speed range for dance music, making it easier for participants to follow it. A long lead-in, containing only a drum pattern, eases starting off and gives participants some time to get used to the rhythm.
A set of six dance movements was defined to be danced on top of this song. With each one to be danced in multiples of one $^4/_4$ measure, they are given as:

A. Side steps with no arm movement

B. Rock steps sideways without arm movement

C. Rock steps sideways with arm movement

D. Side steps with arm movement

E. Side steps with arms up in the air

F. Standing still with head bopping

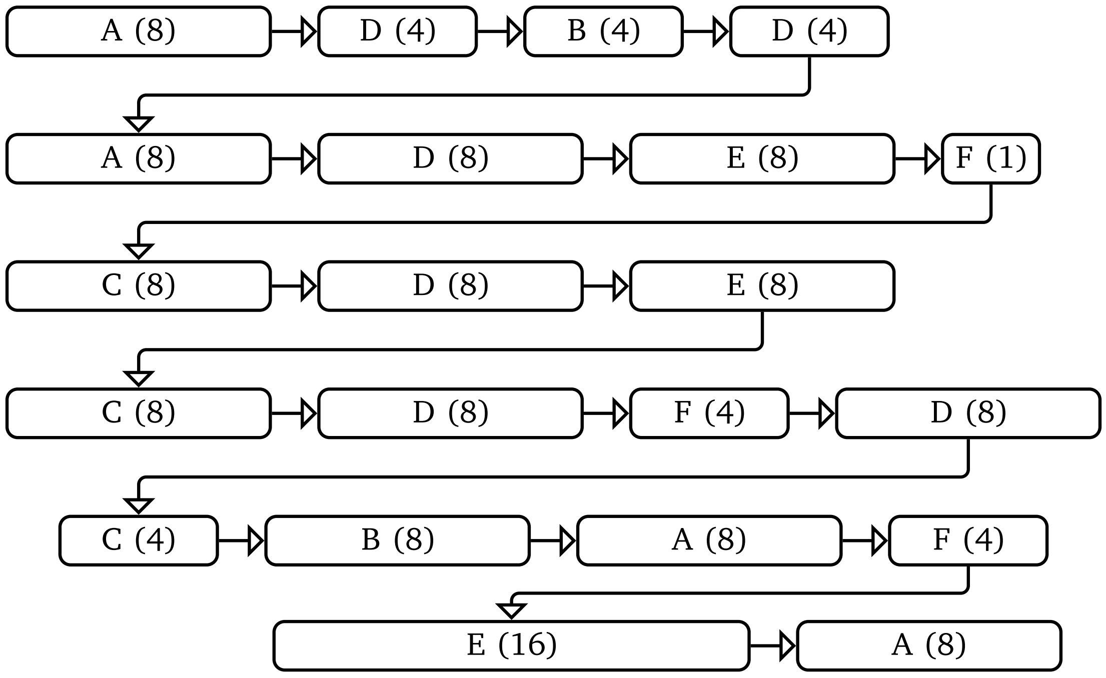


Figure 5.4: Dance movement sequence used in the evaluation

As can be seen, there are two distinct foot movements combined with three possible arm movements. In addition, there is a resting pose, used in three short intermissions present in the song. The movements were chosen as to allow participants to take part in the evaluation without lengthy training. While more complex movements were used for theoretical purposes (see Section 4.2), those were deemed too complex for non-professional dancers in an ad hoc evaluation session.

The whole *4:54 m* long song at the given speed results in 145 blocks. Each block is equivalent to one measure in the song. The prescribed sequence of dance moves is shown in Figure 5.4.

5.4 Evaluation Procedure

Four participants helped in evaluating the algorithms used in this thesis. Before starting a recording session, all participants were instructed on the testing procedure. Specifically, it was pointed out what parts of the recording application's interface to focus on and what to expect. Also, the dance moves to be performed were explained to them. While recording, additional help was provided. This includes oral notification of upcoming transitions and movement instructions.

5.5 Evaluation Issues

When recording a dance session, no participant was able to dance the prescribed moves without errors. Especially at transitions from one movement class to another, mistakes were made. While instructions were provided, sometimes some transitions followed each other too fast for some participants. Also, longer stretches of one movement class led to some participants getting too comfortable with one move and having problems switching to a new one. Most of those problems can be attributed to the experience level of the participants. The choreography was not specifically trained beforehand and none of the participants were used to this kind of task. Dancing according to a predefined motion set also was not something they were used to.

While the sensors themselves worked well, the straps holding them in place turned out to be a suboptimal choice. While they allowed the system to be worn by participants of varying stature, they could not ensure a tight fit over the duration of an entire recording session. While securely fastened in the beginning, the forces exhibited by the dance movements made them more loose over time. Due to this all participants had to manually refasten the sensor straps while dancing. This inevitably led to some noise in the recording. Furthermore, with the sensors slightly shifting, an additional error is introduced.

Chapter 6

Results

Based on the recorded motion data from several participants, the performance of the dynamic time warping (DTW) and feature vector comparison (FVC) algorithms is evaluated. Several aspects are of interest at this point:

- How well can distinct movements be distinguished?
- What parts of the motion data have the highest influence on the results?
- Are parameter choices participant-dependent?
- How strong is the influence of the sequence length?

In this chapter, those questions are answered. As general convention, all error measures in this section are given as percentages. The error rate will be shown on the y-axis, with the range always set to $[0, 100]$. Often the x-axis is used to show the threshold being used. With the similarities between blocks always in an $[0, 1]$ interval (as explained in Chapter 4), this also translates to the x-axis. However, the two methods used have different ranges of interest in this respect. Therefore, the range used here often varies. In several examples, results are shown for the whole recording and the first half of the recording. This was done, because recording issues were mostly concentrated in the later half and a focus on the first one provides a clearer view on the results.

It is worth mentioning that the error rate is sometimes deceiving. Consider an algorithm that assigns a new label to each block. As the identifier mapping (see Section 5.2.2) tries to find the best fit, some blocks are still seen as correct. Specifically, an amount of blocks equal to the number of distinct classes in the given sequence is considered correct. Applying that logic to a sequence of 20 blocks with four distinct given classes, 20% would be seen as correctly classified even though no blocks were grouped. For longer sequences, this problem becomes less of an issue, as long as the number of distinct given classes does not

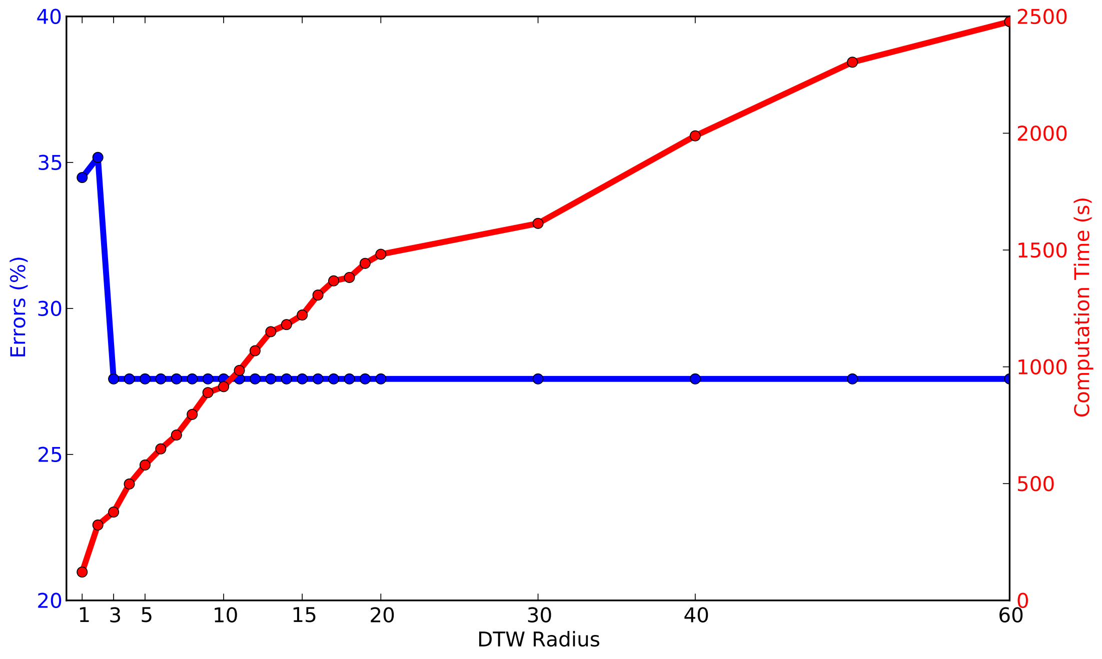


Figure 6.1: Comparing cost and error rate of different *FastDTW* radius choices.

increase correspondingly.

The other extreme is a classification that assigns all blocks to the same class. The identifier mapping will connect that class with the given class of the highest frequency. To put this in perspective, consider a 20 block sequence with one 10 block spanning class and two classes spanning 5 blocks. A recognized sequence of only one class would map to the 10 block long given sequence. Thus, 50% of the data is considered as correctly classified. This problem becomes less of an issue when the number of distinct given classes goes up or given classes are more uniformly distributed.

6.1 Suitable Dynamic Time Warping Parameters

In a first step, appropriate radii for use with DTW were to be determined. As mentioned in Section 4.3.3, this value influences the computational cost and accuracy of the DTW algorithm. A multitude of DTW radii were tested on a full dance recording (145 blocks), with results shown in Figure 6.1. As can be seen any radius higher than 3 does not result in less errors. The computational

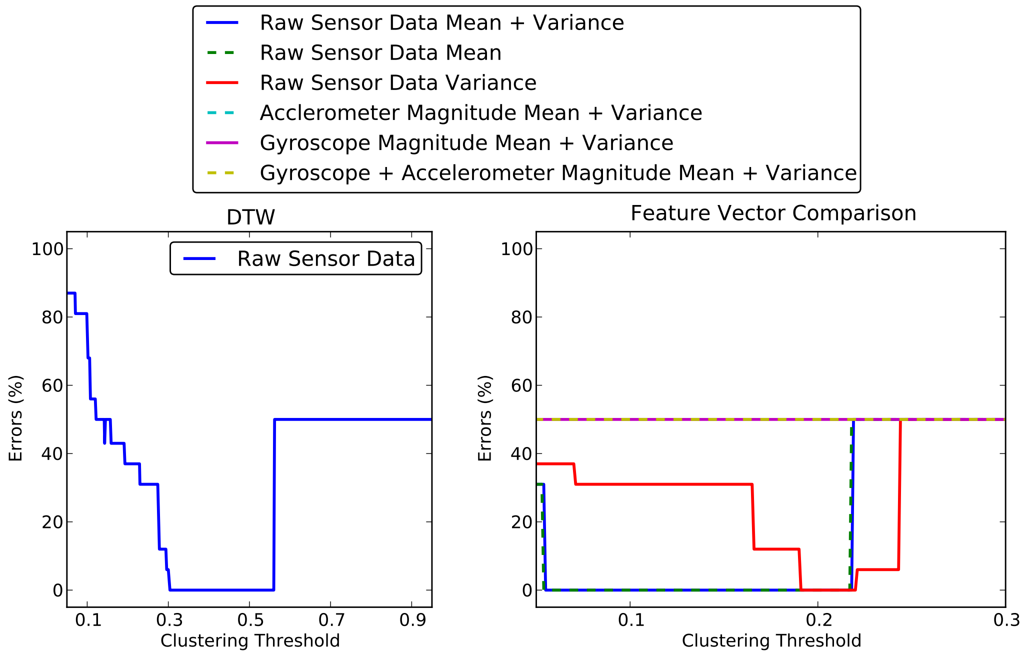


Figure 6.2: Comparison of error rate when detecting differences between side steps with and without arm movement.

cost increases significantly, though. Based on this result, a DTW radius of 4 was chosen for all subsequent DTW calculations.

6.2 Two Motion Comparison

To determine the suitability of both approaches, DTW and simple block similarity measures, they were tested on a sequence of two different motion patterns. Sixteen blocks from a recorded session were used. With the song used, sixteen blocks (measures) span about 32 seconds in time. From those sixteen blocks, eight blocks were side steps without arm movement and eight blocks were sidesteps with arm movement. As can be seen in Figure 6.2, the DTW approach was able to correctly differentiate between those two sequences with zero errors. Note that there is a range of threshold values being appropriate. Classifying blocks based solely on whole block similarity worked equally well for some feature choices. In this case, the mean of the raw sensor data seemed to be an appropriate choice. Using the variance as well made no difference, while the variance alone performed somewhat worse. Other features did not fare as well at all.

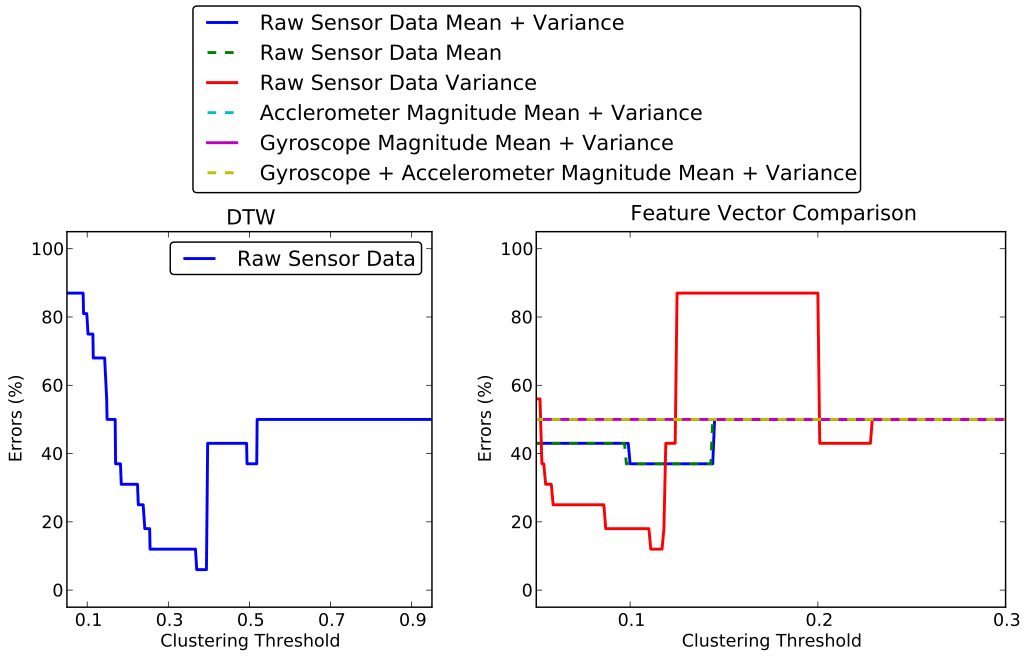


Figure 6.3: Comparison of error rate when detecting differences between side and rock steps with arm motion.

Another comparison of two different moves is shown in Figure 6.3. Here, eight blocks of side steps with arm movement were compared to eight blocks of rock steps with arm movement. Compared to the previous two moves, these two seem to be harder to differentiate. While the variance now comes out as the best performing feature when using FVC, its accuracy varies wildly. The DTW algorithm also has a smaller window of working thresholds. All this reflects, that side steps and rock steps are quite similar to this system. To some extent this can be attributed to a lack of feet sensors, that would be particularly important in distinguishing those two motions.

A last comparison was done for side steps with the arms either moving in front of the body or being up in the air. Figure 6.4 shows, that those two movements could be classified comparatively well. The performance of the FVC seems quite in line with the behavior exhibited in the first comparison. This indicates, that a combination of the mean and variance of the raw sensor data is the best performing overall set of block feature to use. In all following comparisons this combination is thus chosen.

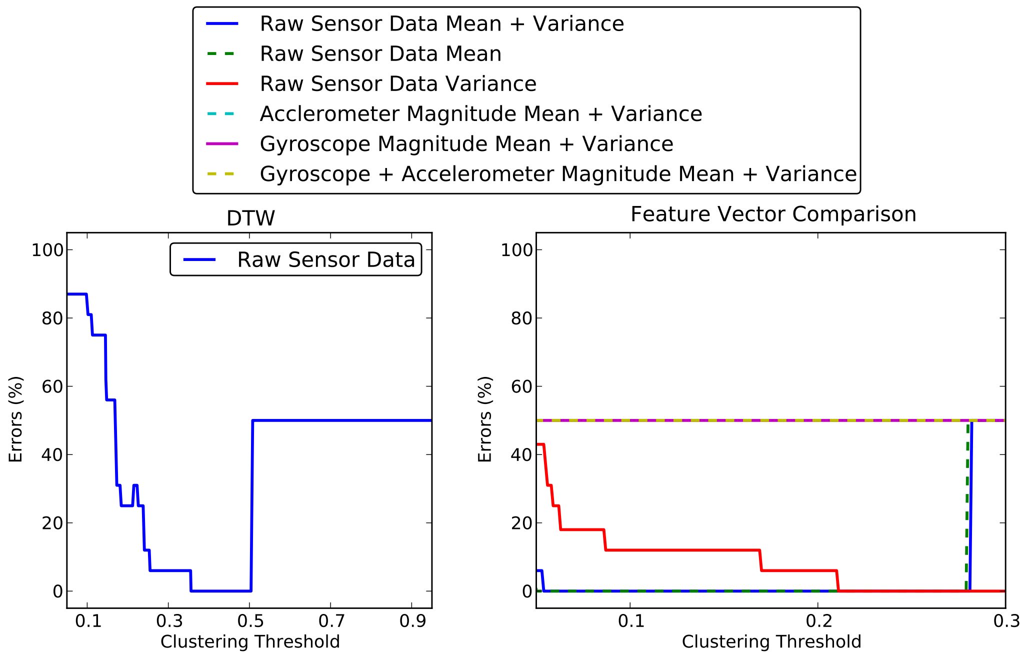


Figure 6.4: Comparison of error rate when detecting differences between side steps with arm motion and with arms up in the air.

6.3 Influence of Feature Choices

As described in Section 4.3.3, the DTW algorithm is not limited to working on the raw sensor data, but can also use any of the various features described in Section 4.3.1. For evaluation purposes, several such features were tested on a 33 block long subsequence of a dance recording. This sequence included five different given classes to be distinguished and lasted almost 67 seconds.

Figure 6.5 shows the performance of several features. While it is readily apparent that some features are worse than others, which is best demands further investigation. For example, several features perform well for a short window of lower threshold values. Lower thresholds affect the clustering to be less inclusive (demands higher similarity for clustering). At the same time, those features yield significantly worse error rates for just slightly higher threshold values. This indicates that those features generally yielded blocks with high similarity. While a demanding threshold still led to a good classification, higher threshold tended to lump them all together.

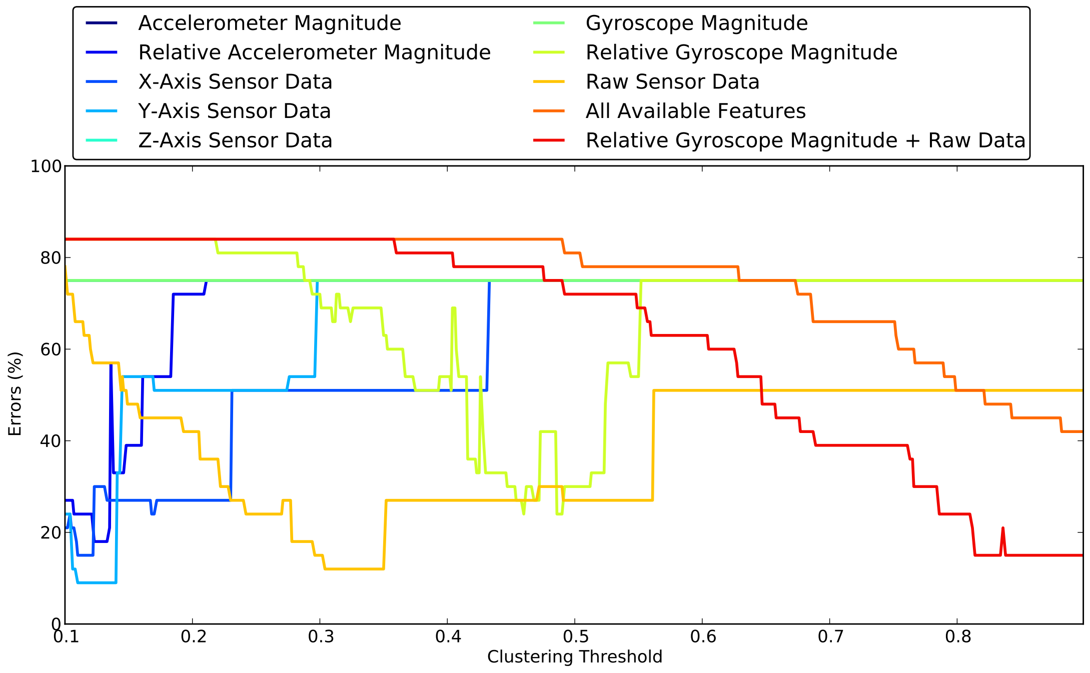


Figure 6.5: Comparison of different feature choices for use with the DTW algorithm.

On the other hand, the raw sensor values led to rather promising results. Comparably low error rates are achieved over a large threshold window, showing that a good differentiation of classes was possible. While some other features outperformed the raw sensor data at some thresholds, those can be seen as outliers in the overall big picture. The relative gyroscope magnitude also performed quite well, but only over a much smaller threshold window.

Combining two well performing features does not necessarily yield a higher performing one. For example, the raw sensor data and the relative gyroscope magnitude fare quite well on their own. Combining them however, does result in a feature with significantly decreased performance. Looking at the graph provides some information on why that is the case. Said feature and the feature combining all available ones, only start performing better at high threshold values. This indicates, that all resulting blocks were detected as being very dissimilar. With high distance values between blocks, only an equally high threshold could result in any clustering at all.

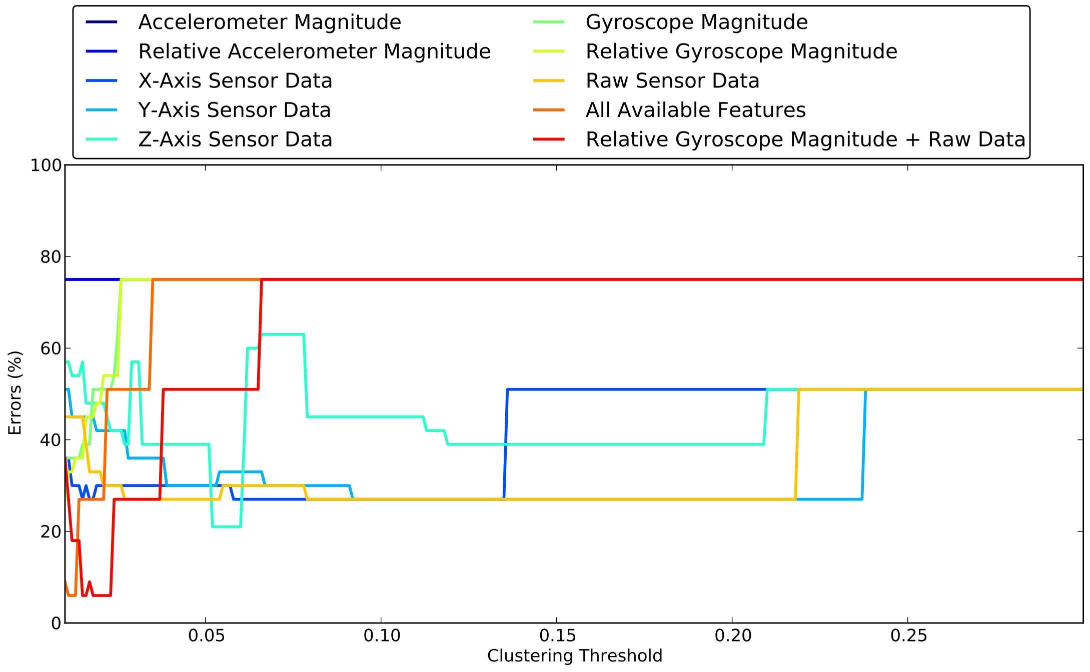


Figure 6.6: Comparison of different feature choices for use with the FVC algorithm.

Using the same block sequence and feature set with the FVC approach, led to less conclusive results. Note that the features in this section do not correspond to the block level features of this approach. As mentioned above, the FVC here uses the mean and variance of whatever underlying block data as block level features. As can be seen in Figure 6.6, the raw sensor data and y-axis sensor data seem to perform roughly equally. Some other features perform well at low threshold values but exhibit rapidly declining performance for higher ones.

The results shown above indicate that the raw sensor data itself is the most promising feature. While other features might excel the raw sensor data at certain thresholds, this is not the case in a more general view. Thus, other possible features sets are eschewed for the rest of the tests and the raw sensors values are used in all comparisons coming up.

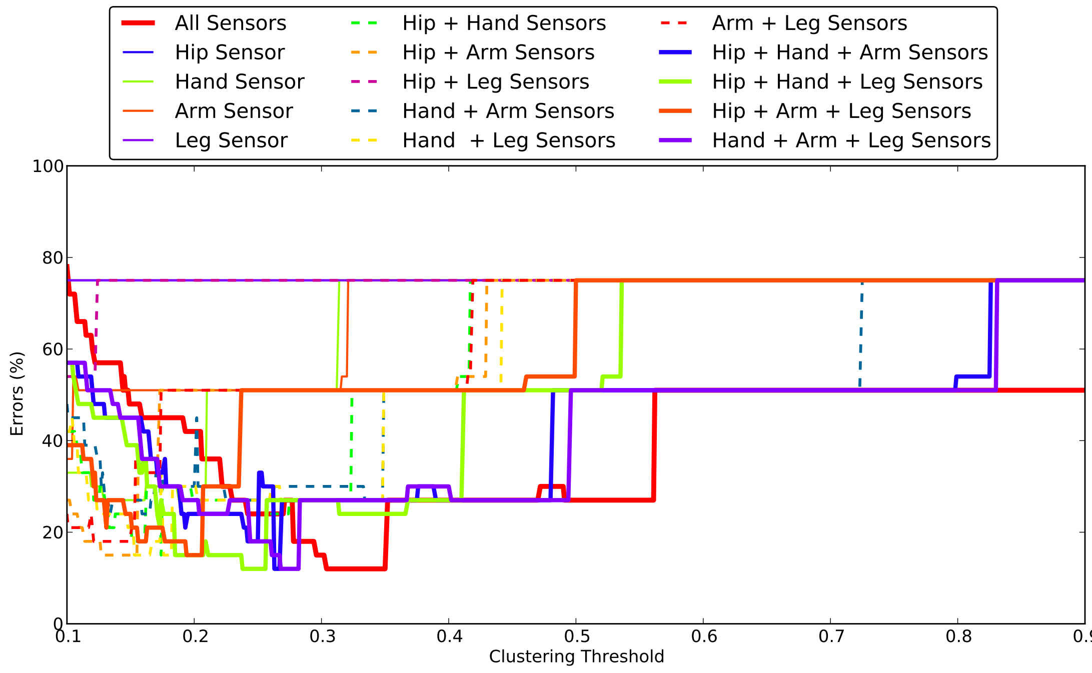


Figure 6.7: Comparison of different combinations of sensor data for use with the DTW algorithm.

6.4 Influence of Sensor Choices

With the best features to use being determined, another question is which sensors have the most influence on the overall result. For this purpose, all possible combinations of sensors have been tested for their viability. While the resulting graph is somewhat overwhelming at first sight, a number of aspects can be observed. For instance, a certain banding is visible. Four distinct horizontal lines are visible that most results converge to. This is directly related to the number of classes in the given sequence. For this test blocks 20 to 53 (see also Figure 5.4) of a dance recording were used, the same sequence as in the feature comparison above. In this sequence, there are four given classes of length 8 and one that is only 1 block long. The threshold determines the amount of clustering with each jump down in the resulting graph signifying a new correctly detected class. An 8 block error in a sequence of 33 blocks corresponds to about a 25 point increase of the error rate. The jumps in the graph roughly correspond to that number. As one class is only 1 block long in the given sequence, a wrong classification here is not immediately apparent.

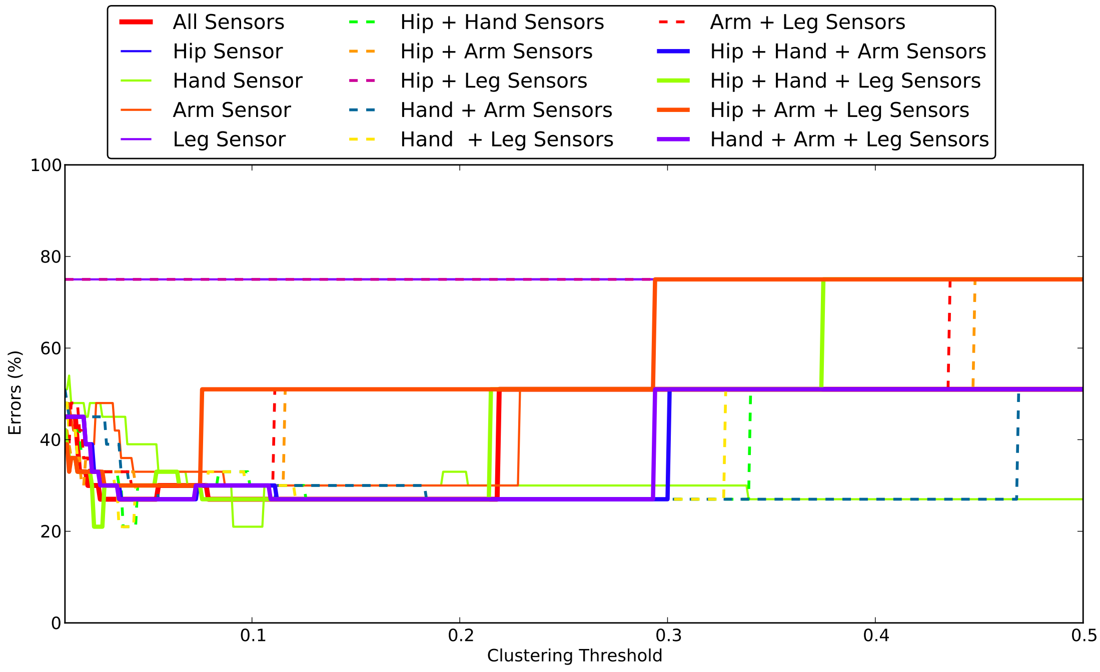


Figure 6.8: Comparison of different combinations of sensor data for use with the FVC algorithm.

It can also be seen, that the combination of all sensors performed best. Leaving out the hand sensor had the most impact on the resulting classification, closely followed by the arm sensor. The hip sensor had the least influence on the result, with the leg sensor only slightly more important. The further away from the body center a sensor is, the higher its impact on the classification seems to be. This should come as no surprise, as the movements in the limbs are usually more pronounced than at the body center.

As earlier, the same tests were also run using the FVC algorithm. As can be seen in Figure 6.8 this led to similar results.

6.5 Influence of Sequence Length

The length of a sequence has some influence on the classification. One aspect here is that over time, slight changes to a dance move are more likely to occur. While it is comparatively easy to perform the same move for 20 seconds, it is harder to repeat the same move after 2 minutes have passed. Thus, one would expect, that a classification of shorter time spans contains less errors. To test

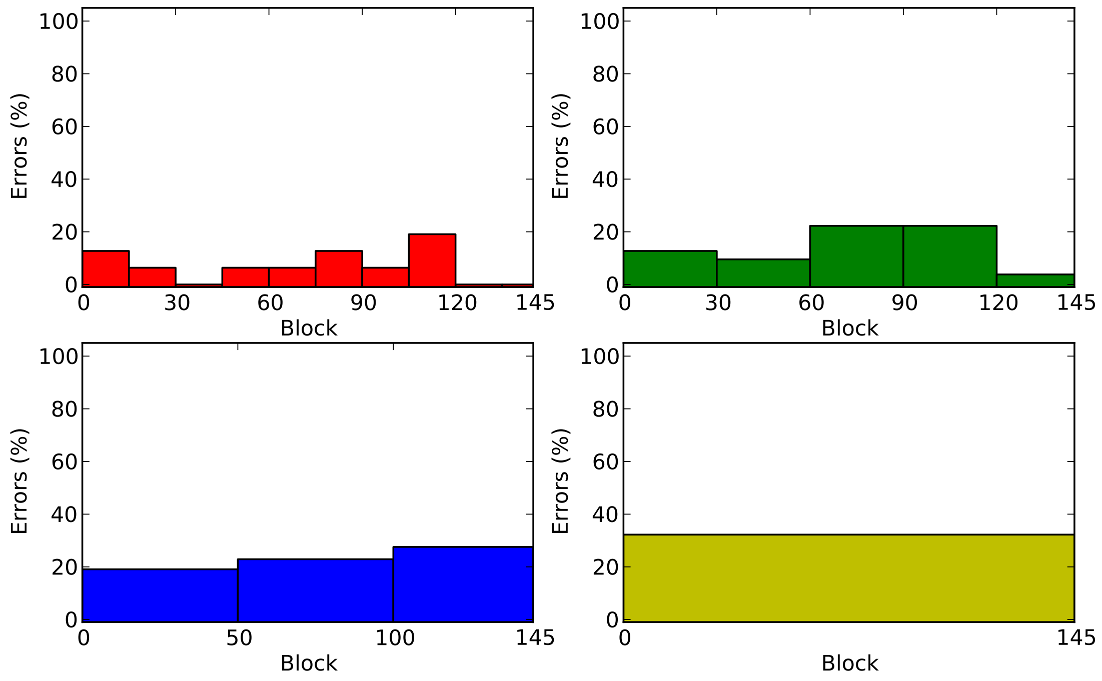


Figure 6.9: DTW-based classification results for several subsequences of sensor data. The width of each bar denotes the data range being used.

this, one recording was analyzed in various windows. Looking at the results in Figure 6.9, a certain increase in overall error is apparent. While analyzing sequences of 15 block length, the maximum error was below 20%, the overall error in a classification of the whole sequence was at slightly over 30%. The data also shows peaks in error rate in later parts of the recorded sequence. Some of that can be attributed to the sensor straps loosening and required fastening motions.

Similar results are also obtained using the FVC algorithm. Figure 6.10 shows, that only slight variations are present in comparison to the DTW algorithm.

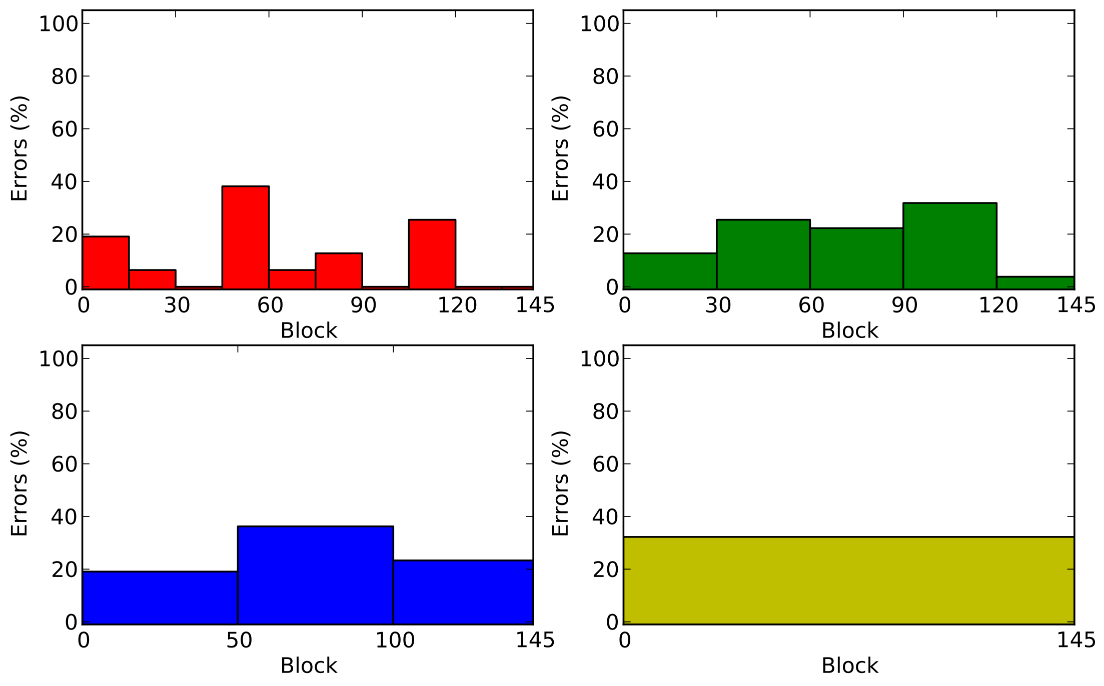


Figure 6.10: FVC-based classification results for several subsequences of sensor data. The width of each bar denotes the data range being used.

6.6 Comparison of Several Recordings

Figures 6.11 and 6.12 show how well the DTW and FVC approaches were able to classify recordings from four participants. For both approaches, all sensors were used with no additional features on top of the raw sensor data. As can be seen, some participant's motions were more easily discernable than other's. Also, each participants seems to have a separate best performing threshold. However, all curves exhibit somewhat similar behavior over the threshold range.

As mentioned earlier, the quality of the recordings degraded over time. Especially loosened sensors could have skewed the results. To sidestep some of these issues, a comparison was also done based only on the first half of the recorded data. As can be seen in Figures 6.13 and 6.14, this led to better classification results.

While the previously mentioned graphs showed the change of the error rate for a range of thresholds, a closer look at the resulting classification is still missing. From all the thresholds tested above, the best performing one is chosen and the resulting classification is further detailed. Figures 6.15 and 6.16 show at which point of the recorded data the classification went wrong.

As earlier in this section, the test was repeated using only the first half of each recording. Figures 6.17 and 6.18 show the resulting classification errors.

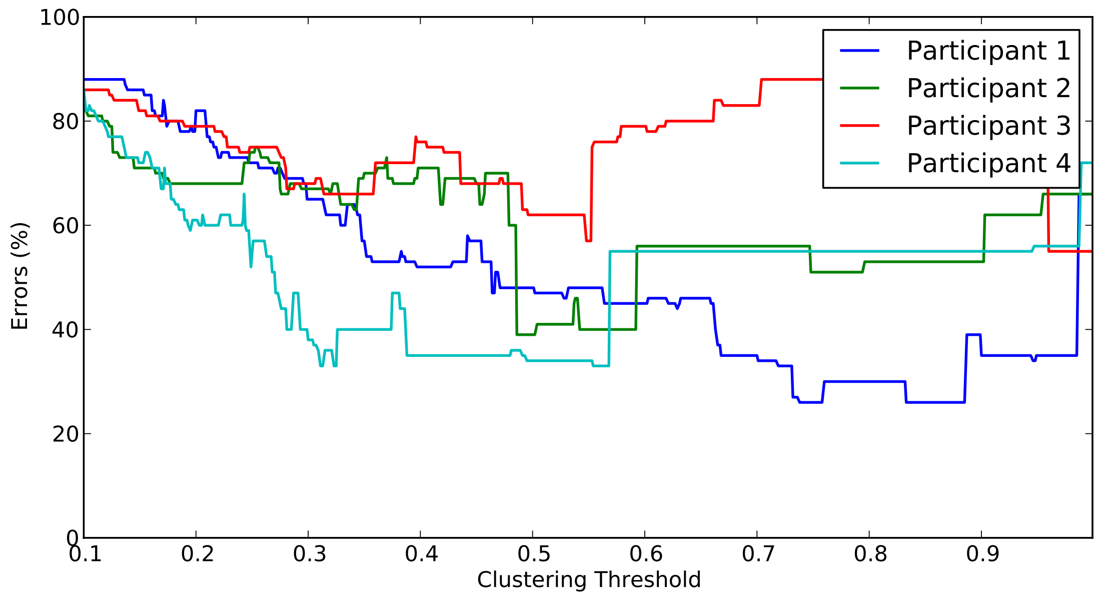


Figure 6.11: Comparing DTW-based classification results from multiple participants. Error rate computed using whole recording.

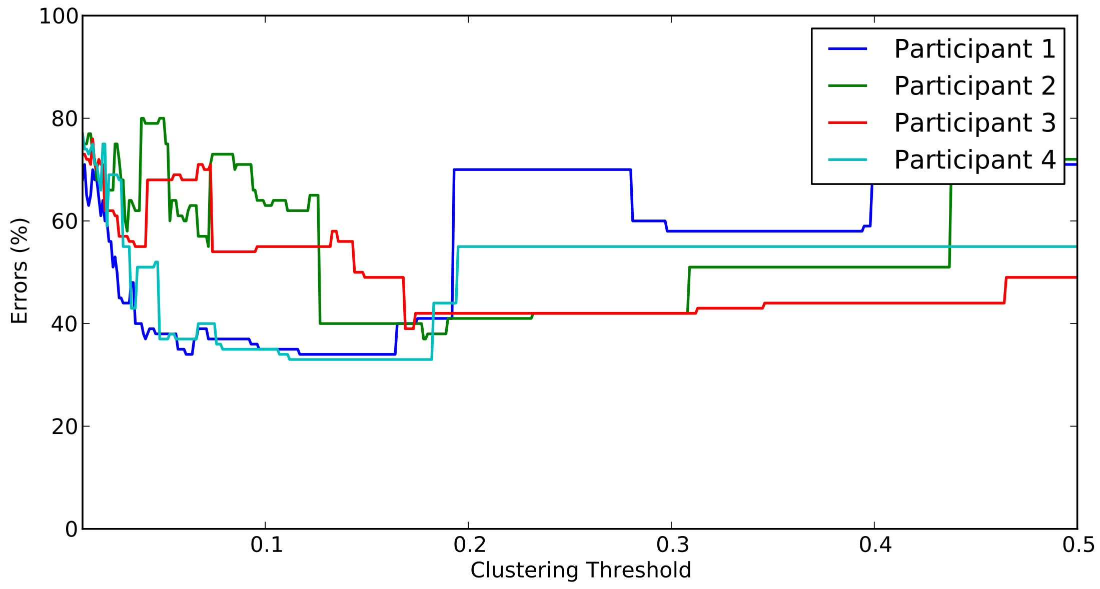


Figure 6.12: Comparing FVC-based classification results from multiple participants. Error rate computed using whole recording.

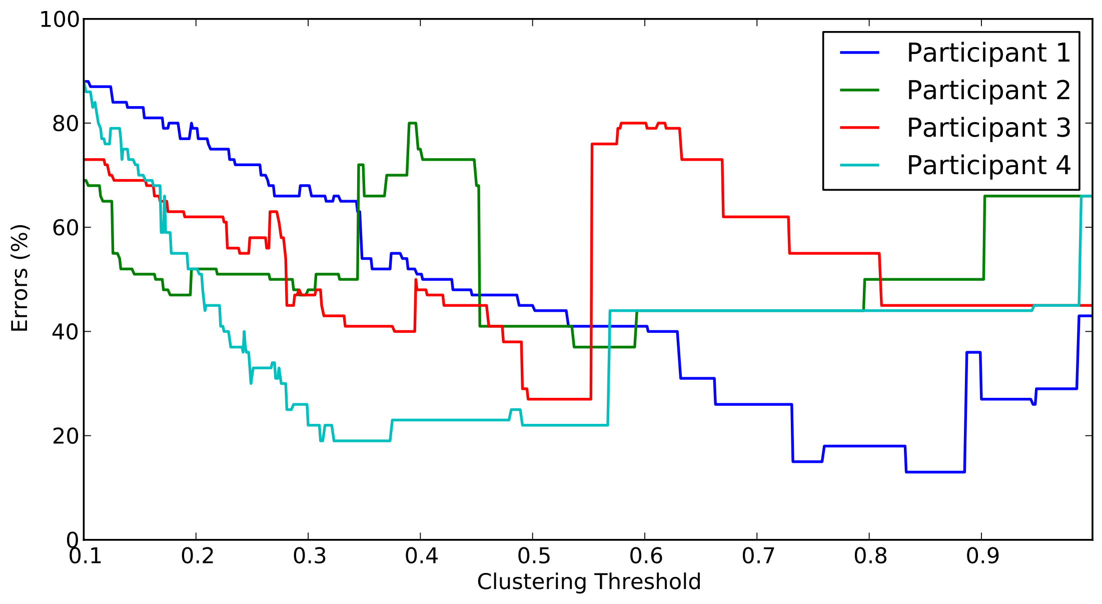


Figure 6.13: Comparing DTW-based classification results from multiple participants. Error rate computed using first half of recording.

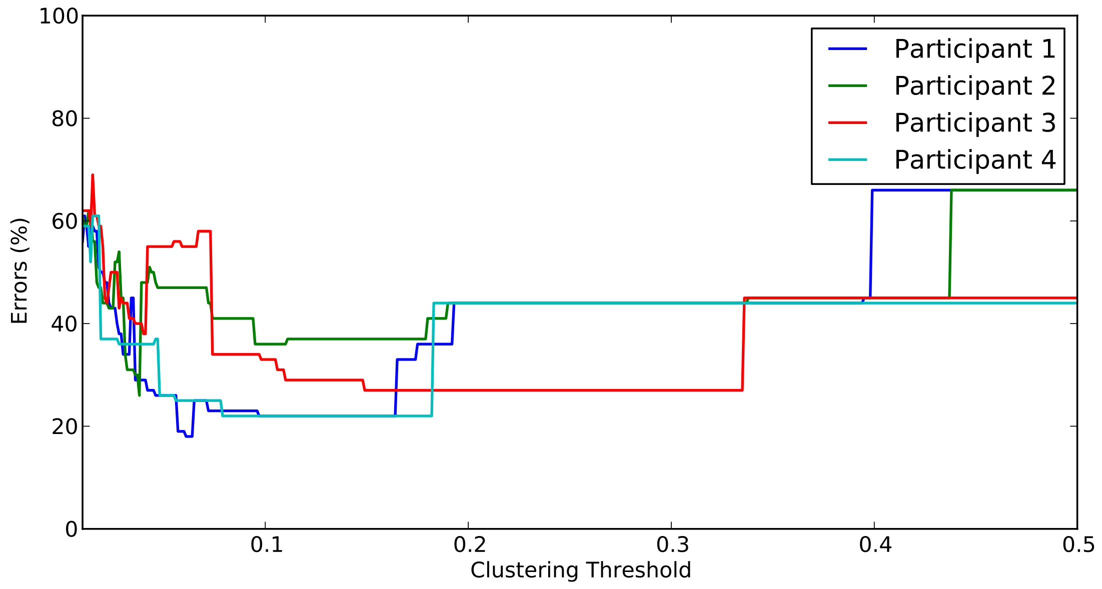


Figure 6.14: Comparing FVC-based classification results from multiple participants. Error rate computed using first half of recording.

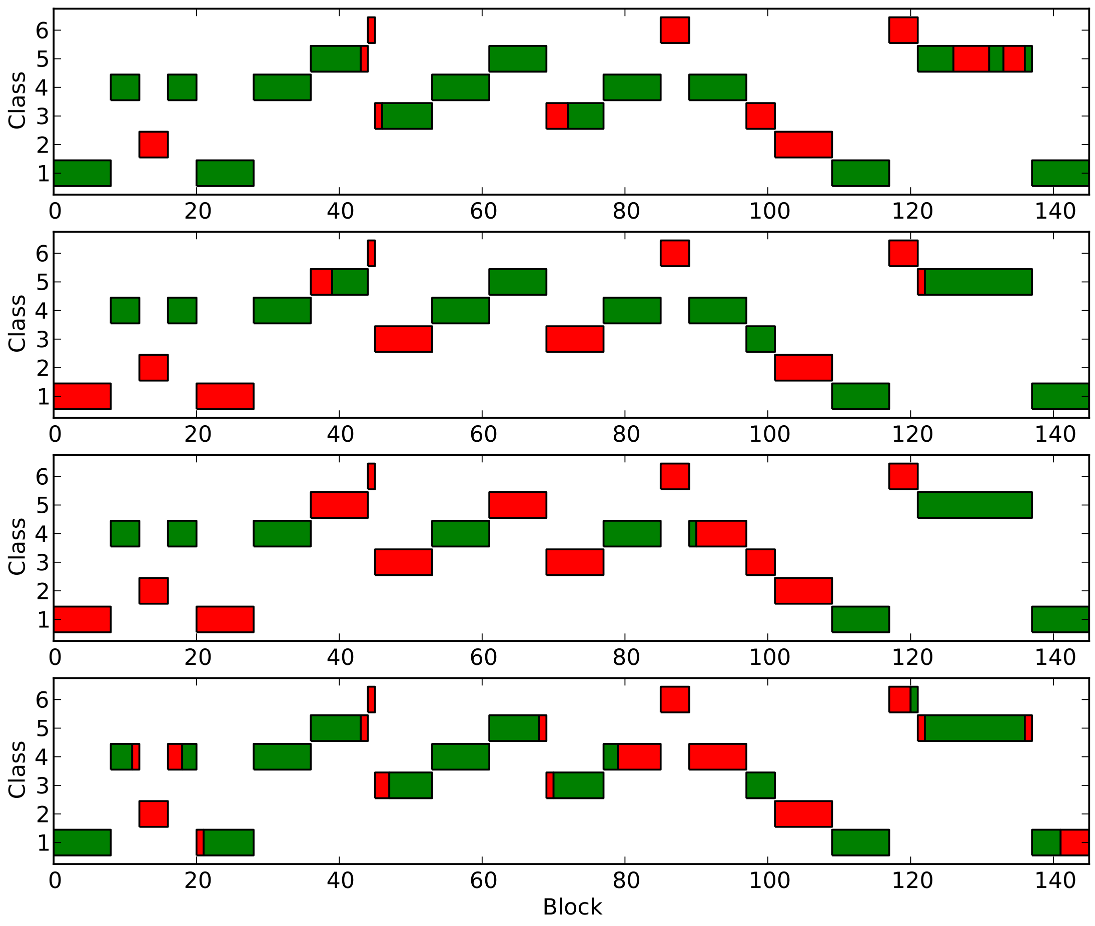


Figure 6.15: Classification errors when using DTW-based classification. Green blocks show correctly classified blocks, while red blocks denote erroneous classification.

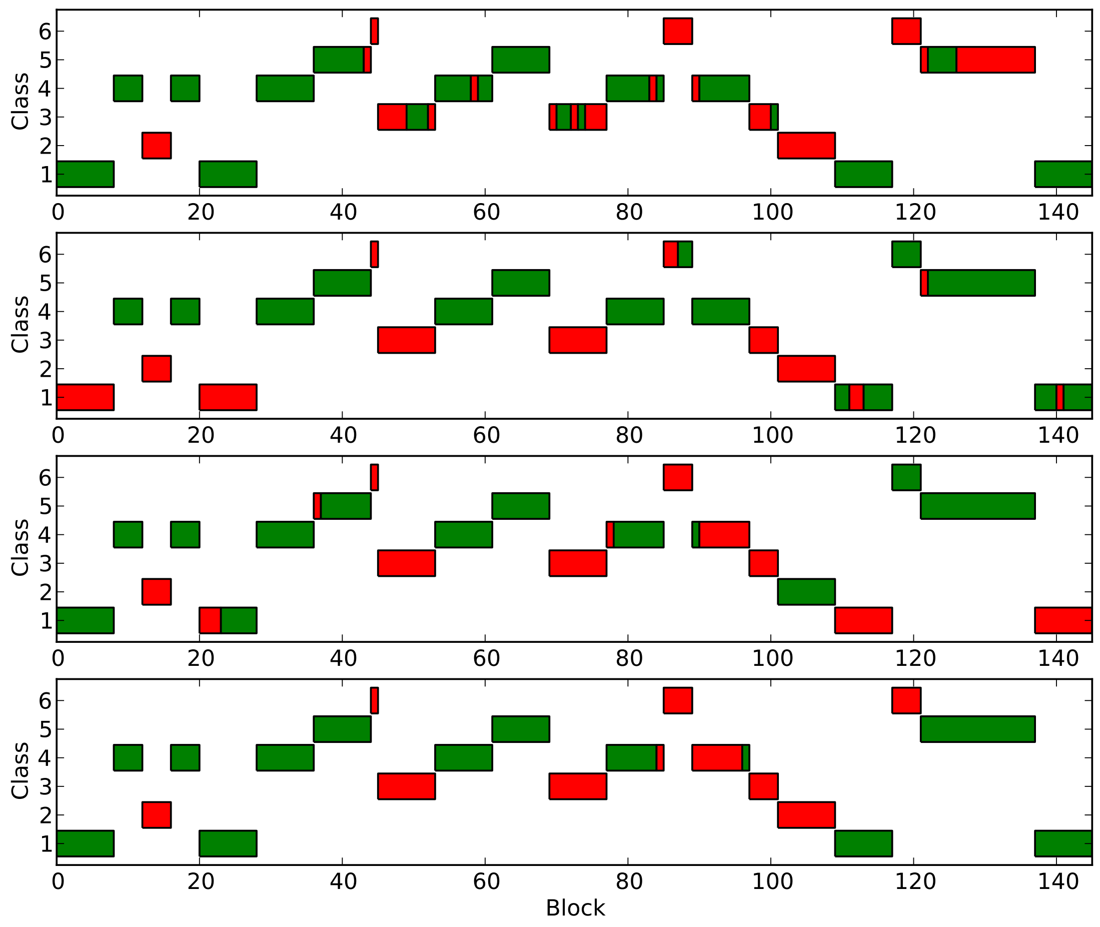


Figure 6.16: Classification errors when using FVC-based classification. Green blocks show correctly classified blocks, while red blocks denote erroneous classification.

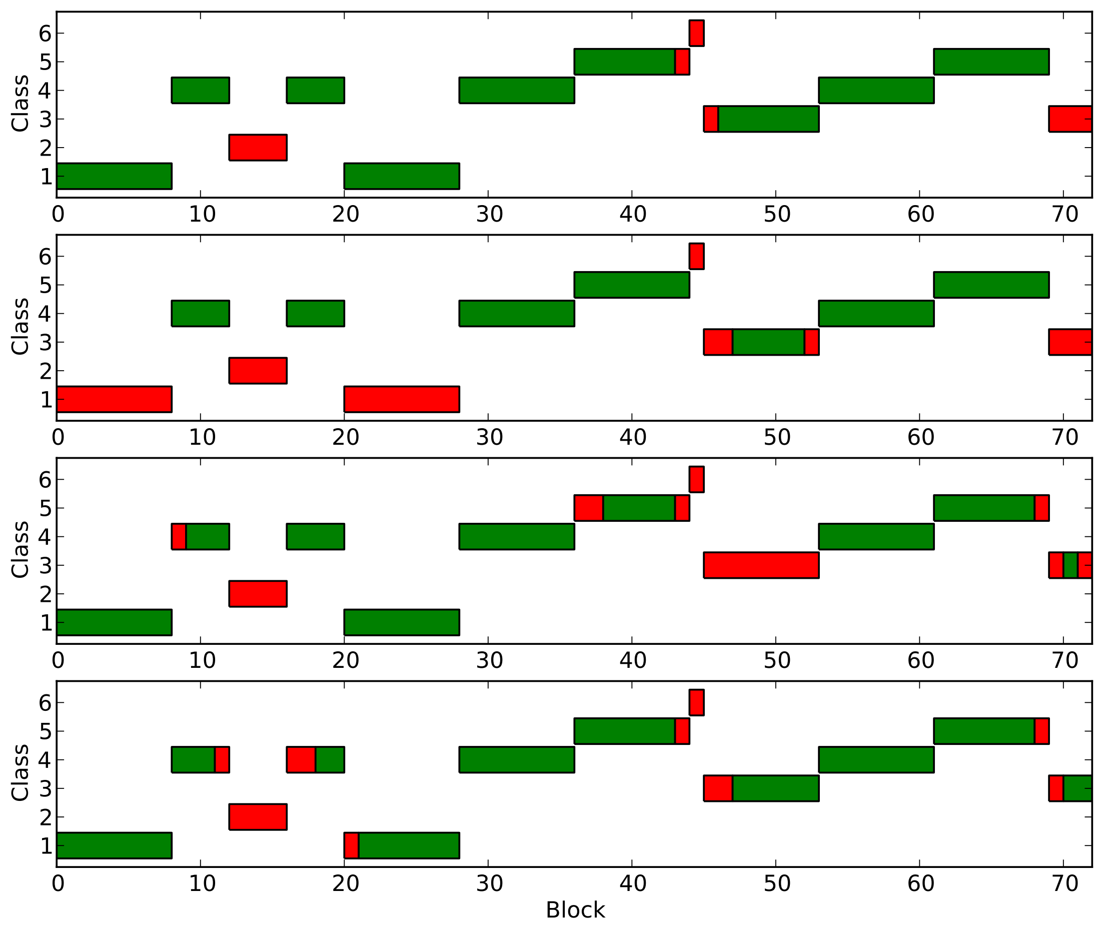


Figure 6.17: Classification errors when using DTW-based classification on the first half of each recording. Green blocks show correctly classified blocks, while red blocks denote erroneous classification.

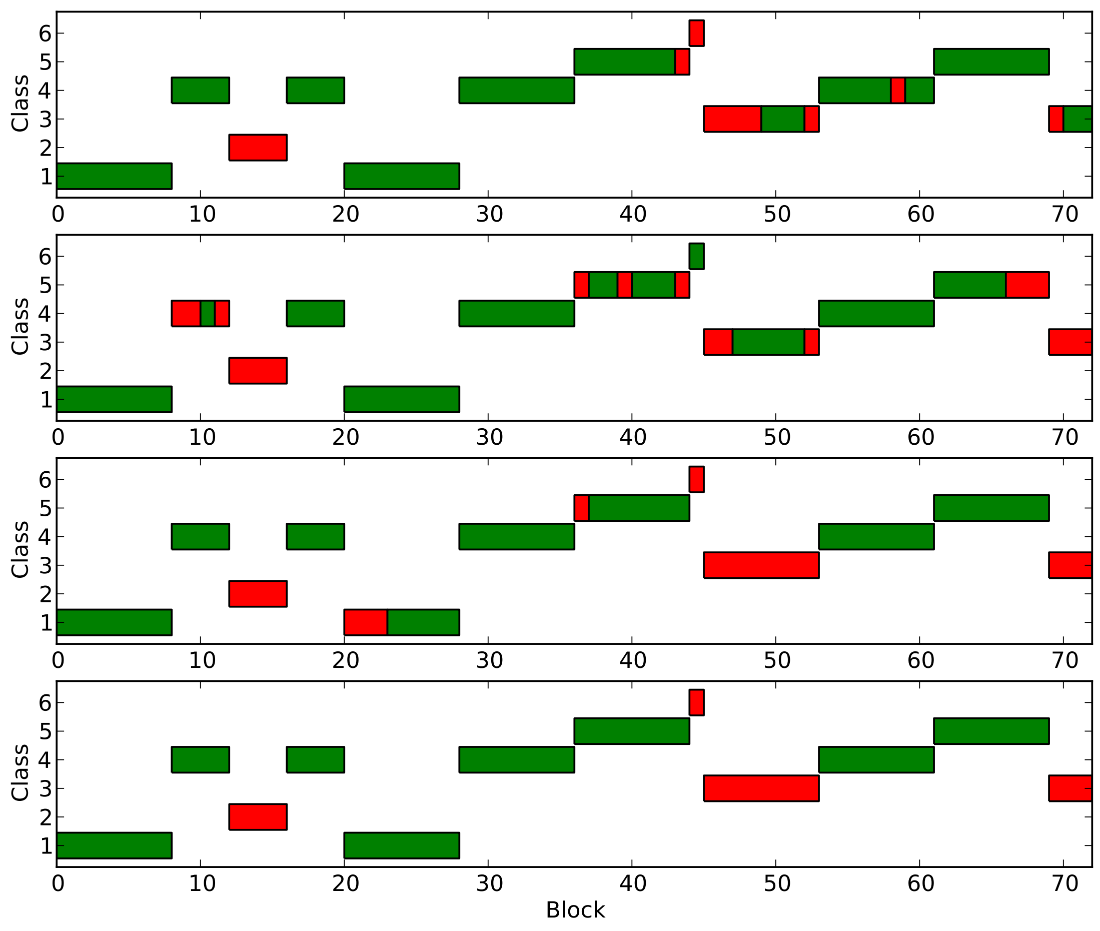


Figure 6.18: Classification errors when using FVC-based classification on the first half of each recording. Green blocks show correctly classified blocks, while red blocks denote erroneous classification.

6.7 Comparison of Error Rate by Dance Move

While an overall error measure provides a general performance estimate, determining the error per dance move helps with a more in-depth understanding. Figures 6.19 and 6.20 show such data for one participant. Note that the full recording was used for this and thus, higher error rates than in partial comparisons are to be expected. In the DTW approach classes 2 and 6 seem to be an issue. Those two coincidentally are also the two classes with the least amount of motion being required for them. On the other hand, more vivid movements were detected comparatively well.

In the FVC approach on the other hand, a much harsher difference in performance can be observed. While three classes were detected really well, the other three were always misclassified. The most likely reason for this behavior is, that the recognized classes could have fallen in two categories: some that are pretty similar to each other and some that are not. Because one overall threshold is chosen, a decision is made to either being able to differentiate the first category or the second one. Here a threshold being able to distinguish side step motions was not appropriate to distinguish e.g., side steps from rock steps.

While the first two figures only showed the error rates from one participant, Figures 6.21 and 6.22 do so for all participants. The variance in error rate is deceiving to some extent, as the different recordings have different base error rates. Thus, the relative error rates are the most interesting aspect here. Note that the relative error between the given classes is roughly similar to the one from a single participant (shown above). This further stresses, that the algorithms' performance is dependent on the motions used.

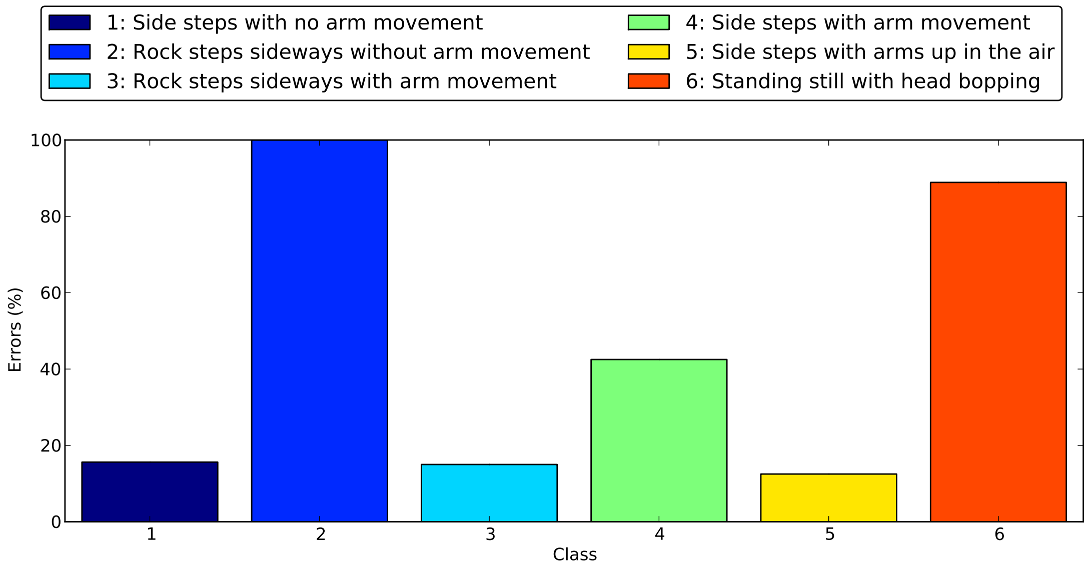


Figure 6.19: Comparing error rates by given class for one participant using DTW-based classification

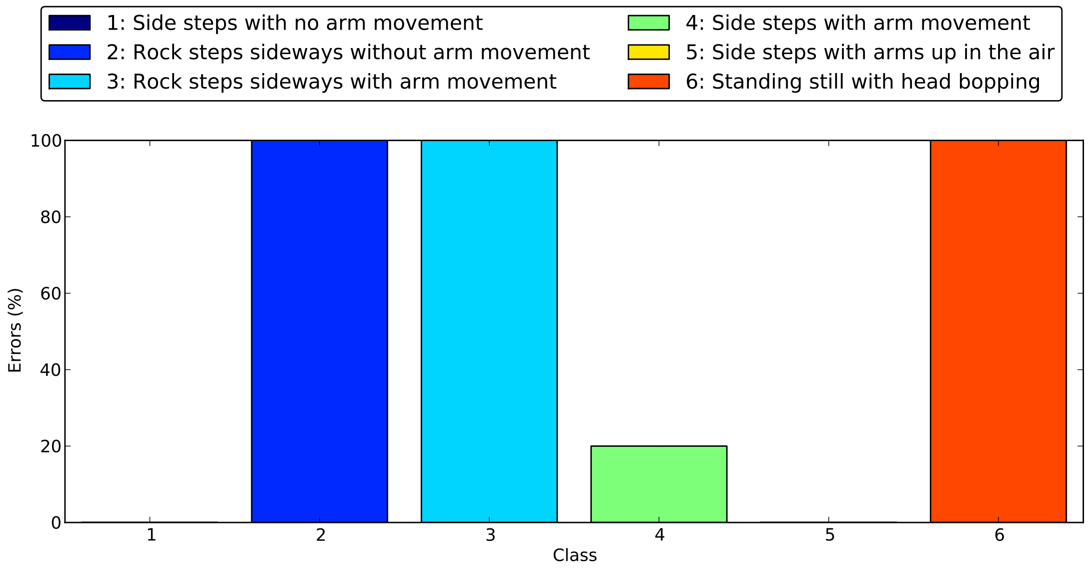


Figure 6.20: Comparing error rates by given class for one participant using FVC-based classification

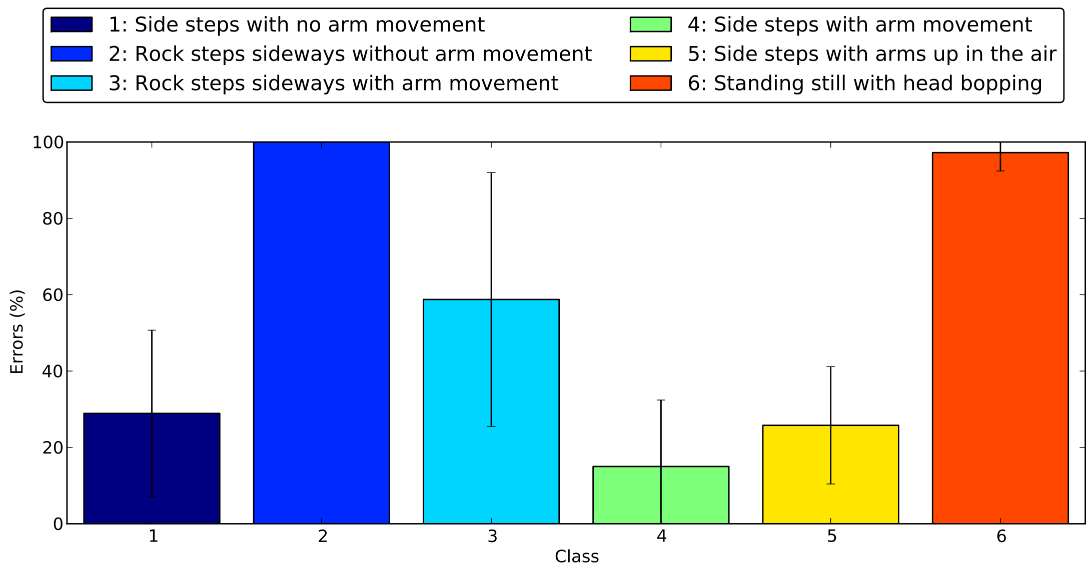


Figure 6.21: Comparing multi-participant error rates by given class using DTW-based classification

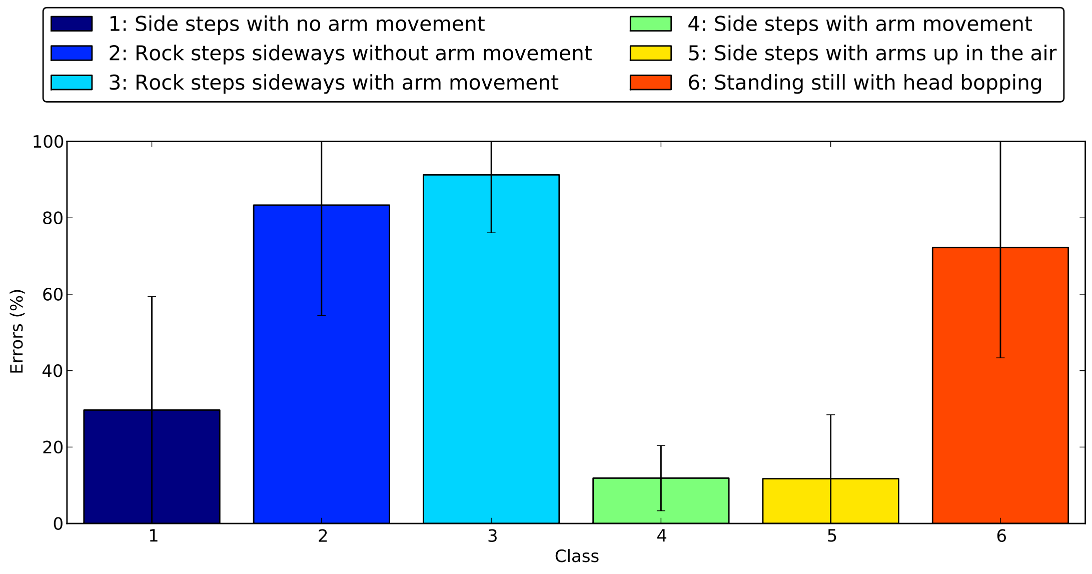
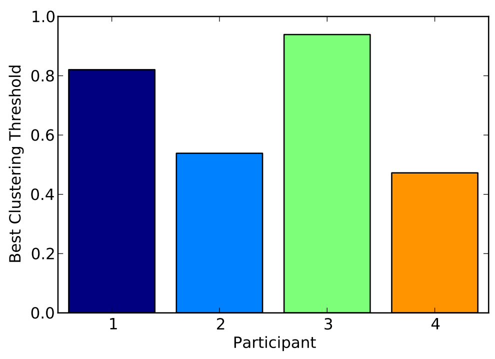
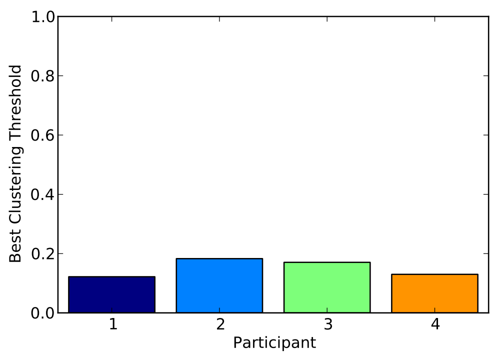


Figure 6.22: Comparing multi-participant error rates by given class using FVC-based classification

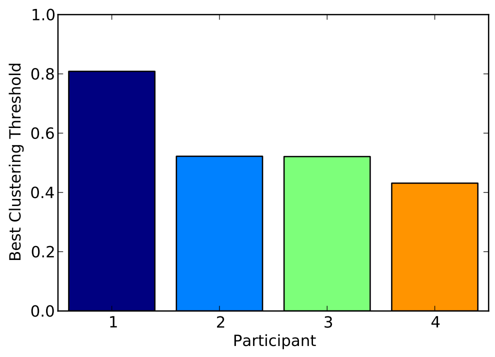


(a) Dynamic Time Warping

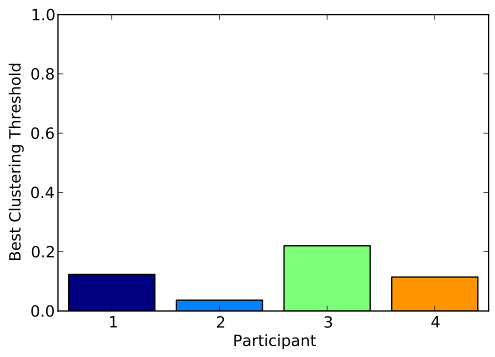


(b) Feature Vector Comparison

Figure 6.23: Means of lowest error yielding thresholds per participant



(a) Dynamic Time Warping



(b) Feature Vector Comparison

Figure 6.24: Means of lowest error yielding thresholds per participant for first half of each recording

6.8 Threshold Choice

In most previous comparisons the error rate was given as a function of the threshold being used. It could also be seen in Section 6.6 how the chosen threshold varies with each participant and performs best at different ranges. In Figure 6.23, a comparison of the best performing thresholds is shown. While that graph shows the performance over the whole recorded sequence, Figure 6.24 does so only for the first half.

For DTW the threshold is roughly in an $[0.5, 0.8]$ interval. In the FVC approach on the other hand, a $[0.1, 0.2]$ interval seems more appropriate.

Chapter 7

Conclusion & Future Work

In this thesis, a system has been described that is able to detect recurring patterns in dance movements. In this chapter, the contents of the thesis are again reevaluated, based on the scenario outlined in the introduction. Furthermore, an outlook is given on potential extensions of the work at hand.

In the first chapter of this thesis an overview of interactive dance and motion patterns was provided. It was described, how interactive dance is used in the fields of art, video games and clubs. As clubs are the main scenario for this thesis, this setting was described extensively. Especially the audience–performer relationship and how performers could utilize technology for their practice were examined more closely.

In the second chapter several existing hardware solutions for handling dance input were presented. Additional interactive dance projects were presented as part of this overview as well. This chapter also detailed a number of existing approaches for dance movement processing.

The third chapter started with a definition of hardware requirements applying to devices in the thesis's scenario. The sensor options described in the second chapter are evaluated according to those criteria. Based on those requirements a prototype system was assembled, repurposing existing hardware. It was described in the following chapters too.

In chapter four the algorithms used for dance pattern recognition were defined. However, before doing so dance movements were examined more closely. This was done to, for instance, determine appropriate segmentation choices for the input. From a set of examples, distinguishing features of dance movements were identified. Based on those preliminaries a formal definition for blocks of motions was given. Two algorithms were described that are able to determine the distance between two such blocks. First, the feature vector comparison (FVC) approach does so by comparing blocks based on whole block descriptors. Secondly, a dynamic time warping (DTW) approach determines the distance by examining the sequence of values inside those blocks. With two means available to provide distance measures, an unsupervised clustering algorithm was described that builds on top of that.

In the fifth chapter, the steps taken to evaluate the proposed methods were described. This included the description of two different applications and the evaluation itself. For the recording application it was described how user interactions were designed to help participants when recording and how the application itself ensures proper data handling in the process. Together with a description of the steps needed for evaluation, all custom file formats used in the process are detailed as well. After recording a dance session, a second application is used to compute performance measures for the algorithm. This application was described as well. This included a definition of a label space mapping algorithm that allows to compare sequences with different class label spaces, determining the best fit between two such sequences. Finally, the evaluation procedure and related issues were described too.

Finally, in chapter six, the results from the evaluation were presented. For both methods, appropriate sensor features to use and parameters to set were evaluated. It was also shown how sequence length and given dance movements influence the resulting classification.

7.1 Conclusion

In Chapter 6 several properties of the work done in this thesis were shown. First of all, both evaluated methods were able to correctly distinguish two given motions in one sequence. It was also demonstrated that, among the features available, using the raw sensor data yielded the best results so far. Comparing multiple recordings, common properties of the threshold curves were identified and a rough range of applicable threshold values was defined.

Working with data from real dance recordings, error rates of about 30% have been achieved. Omitting more erronous data at the end of the recordings, this error rate goes down to about 20%. The results indicated that some movements were harder to distinguish than others. Especially movements eliciting low sensor responses were problematic. However, more pronounced movements were recognized much better.

The work on pattern detection was done in the context of interactive dance in clubs. In Chapter 1, this scenario was described and the question remains how the results of this work relate to it. This relates back to the study of Ulyate and Bianciardi [64], who found that such means of interactions that allow for more freedom of movement also accounted for the more satisfying user interactions. The system described in this thesis very much aligns with that notion, providing a powerful abstraction of complex human motion.

When discussing the results, the work of Feldmeier should be taken into account again. He showed how simple mappings (he used a measure of overall crowd energy to control music and lighting) can be used to facilitate interactive night club experiences and described how patrons enjoyed shaping the experience themselves [22]. Similar uses would be appropriate for dance patterns as well. For example, a VJ could use the current pattern identifier in his mix to control switching between scenes or to trigger clips to be shown.

Another aspect to keep in mind is that the presented system is not necessarily targeted at crowds. While it scales to multiple users, that scenario would bring up additional questions of how to interpret the sensor data. On the other hand, the system is well suited for individual users embedded within a crowd. As mentioned in Section 3.1.1, wearable systems are a good fit in this scenario. This, for example, enables use cases where a designated member of the VJ or DJ team is embedded within the crowd. Dancing together with other patrons, she could partake in the crowd experience while simultaneously shaping the outer influences of that experience by e.g., mapping her movements to control the lighting.

The presented method also relates back to the human-computer interaction (HCI) recommendations for DJ applications, by Gates et al. [25] (see Section 1.2.3). More specifically, the presented system relates to:

Quality of Information DJs are perfectly able to judge a crowd's excitement, they only need a glimpse to do so. However, this ability does not easily translate to dance patterns. Here a longer observation might be needed to be able to discern different patterns. Thus, additional information could be provided by the presented system to close that gap.

Measurement of Excitement Dance patterns are one additional way to gather information on dancers. Especially the FVC method is tailored towards abstractions of motion, such as energy level. Also, recognizing a pattern that previously preceded particularly energetic movements could indicate a similar change coming up soon.

Biofeedback The presented method fits this recommendation rather well. The additional cues provided add additional value to a DJs practice, where such information is yet unavailable. Further means have to be provided to yield a meaningful mapping though.

Cognitive Load Dance patterns are a comparatively high level description of motions. A multi-dimensional data stream of high temporal frequency is reduced to a symbolic identifier stream. Thus, any additional cognitive load on the DJ's side is rather small.

Dance patterns could potentially fit into a VJs or DJs workflow quite well. However, appropriate mappings to further evaluate this were not within the scope of this thesis. Nonetheless, the general feasibility of the proposed method has been shown. Recognizing patterns in dance motion could thus be an exciting extension of the dance club experience.

7.2 Future Work

There are a number of areas in this thesis that are open to extension. This ranges from hardware improvements to algorithmic ones. Several possible improvements are detailed in this section.

7.2.1 Hardware

While the current wearable system worked well, there is some room for improvement. An important aspect here are the mounting straps used. As the sensors loosened over time when recording, a better solution is needed. One option would be to use elastic bandages. While those would provide the needed fixation, they are quite cumbersome to put on and remove though. Another possibility would be to sew sensors directly into textiles. If tight fitting, sensor shifting would not be an issue. However, this limits one sensor set to be only used by participants of similar stature.

The sensors themselves could be upgraded to also include magnetometers. This would enable an additional set of features worth exploring. For example, absolute motion data based on accelerometers and gyroscopes is quite error-prone, although, there are approaches, such as Kalman filtering, to compute an estimate of the absolute sensor orientation (see e.g., papers by Bachmann et al. [5, 6]). Magnetometers, on the other hand, enable easier pose estimation, an interesting feature for dance applications. More advanced sensors, such as InvenSense's *MPU3000*[1] could also be considered. This chip contains motion processing capabilities right on the chip and provides sensor orientation data to hosts. This enables offloading of some motion processing tasks to the sensors themselves and reduces the load on the communication channel and the host application. Potentially, this would also enable to run the entire algorithm on the wearable system itself.

The last hardware related aspect is the communication channel used. While Wi-Fi was used for this thesis, this is far from the optimal solution. Especially with respects to battery life, other wireless communication options would be

[1]See `http://invensense.com/mems/gyro/mpu3000.html`

preferable. Thus, while other factors prescribed the use of Wi-Fi, future systems should explore other options. As described in Section 3.2.3, custom designed wireless systems have shown good results in the past and would be one interesting option.

7.2.2 Algorithm

As other people have had some success with using principal component analysis (PCA) and independent component analysis (ICA) as pre-processing steps in activity recognition (see e.g., [40]), it could be worthwhile to explore the applicability for dance data as well. Overfitting could possibly be an issue with that approach, though. The extent of that problem and whether it can be avoided would need to be evaluated. Also using PCA, Perlibakas has tested additional distance metrics for face-recognition [50]. Whether some of them could also be used to improve dance pattern recognition is another possible extension.

Currently, computing the distance of a new block to an existing cluster is quite inefficient. While this is no big problem when using FVC, comparing blocks using DTW is quite expensive. If a way was found to combine previously grouped blocks, only one similarity computation would be needed for each cluster. Potentially, two blocks could be combined by time warping to a common length and blending them together.

With some movements, having only one threshold for clustering was insufficient. Consider a case, where two groups of movements exist. In the first one are movements that are quite distinct. Each of them is different from the others. In the second group are movements that are rather similar. Considering only movements from the first group, a rather high threshold value would be appropriate. Similar movements would be clustered together with only a slight risk for mislabeled movements. In the second group however, such a threshold would lead to different results. As those movements are only differing to a smaller extend, a high threshold value just groups them all together. Here, a lower threshold is needed to be able to tell them apart. Potentially, an adaptive thresholding scheme could be employed to solve this problem. This could, for example, require finding features strongly correlating with threshold values and training a threshold estimator based on that data.

7.2.3 Evaluation

An evaluation with professional dancers and a trained choreography would allow a more precise measure of performance. Currently, no completely error-free recording is available for testing. While several repeated recordings reduced the number of errors, they still could not provide the best possible evaluation data.

Another option would be to record all dances on video as well. Afterwards, a list of observed motions could be compiled manually and compared with the recognized sequence of classes. This would require a substantial amount of work and time though.

Finally, it would be a worthwhile endeavor to test the system in a more concrete setting. This would require building a system to make recognized dance patterns available to DJs or VJs. Both, Musical Instrument Digital Interface (MIDI) and Open Sound Control (OSC), could be used for this purpose. For OSC one could build on top of the Gesture Description Interchange Format (GDIF)[2] described by Marshall et al. [41].

[2]See `http://www.gdif.org` (last accessed on March 22nd 2010)

Bibliography

[1] Erwin Aitenbichler, Jussi Kangasharju, and Max Mühlhäuser. MundoCore: A light-weight infrastructure for pervasive computing. *Pervasive and Mobile Computing*, 3(4):332–361, 2007. ISSN 15741192. doi: 10.1016/j.pmcj.2007.04.002. URL `http://linkinghub.elsevier.com/retrieve/pii/S1574119207000296`.

[2] Miguel Alonso, Bertrand David, and Gaël Richard. Tempo and beat estimation of musical signals. In *Proceedings of the 5th International Symposium on Music Information Retrieval (ISMIR'04)*, volume pp, pages 158–63, Barcelona, Spain, 2004. URL `http://ismir2004.ismir.net/proceedings/p032-page-158-paper191.pdf`.

[3] Ryan Aylward and Joseph A. Paradiso. Sensemble: A Wireless, Compact, Multi-User Sensor System for Interactive Dance. In *Proceedings of 2006 Conference on New Interfaces for Musical Expression*, pages 134–139, Paris, France, 2006.

[4] Ryan Aylward and Joseph A. Paradiso. A compact, high-speed, wearable sensor network for biomotion capture and interactive media. In *Proceedings of the 6th international conference on Information processing in sensor networks - IPSN '07*, page 380, New York, New York, USA, 2007. ACM Press. ISBN 978159593638X. doi: 10.1145/1236360.1236408. URL `http://portal.acm.org/citation.cfm?doid=1236360.1236408`.

[5] Eric R. Bachmann, Robert B. McGhee, Xiaoping Yun, and Michael J. Zyda. Inertial and magnetic posture tracking for inserting humans into networked virtual environments. *Virtual Reality Software and Technology*, 2001.

[6] Eric R. Bachmann, Xiaoping Yun, and Robert B. McGhee. Sourceless tracking of human posture using small inertial/magnetic sensors. In *Proceedings 2003 IEEE International Symposium on Computational Intelligence in Robotics and Automation*, pages 822–829, Kobe, Japan, 2003. IEEE. ISBN 0-7803-7866-0. doi: 10.1109/CIRA.2003.1222286. URL `http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=1222286`.

[7] Alice Bayliss, Jennifer G. Sheridan, and Nicolas Villar. New shapes on the dance floor: influencing ambient sound and vision with computationally augmented poi. *International Journal of Performance Arts and Digital Media*, 1(1):67–82, April 2005. ISSN 1479-4713. doi: 10.1386/padm.1.1.67/1. URL `http://www.atypon-link.com/INT/doi/abs/10.1386/padm.1.1.67/1`.

[8] Rudolf Benesh and Joan Benesh. *An Introduction to Benesh Dance Notation*. A&C Black, London, 1956.

[9] Frédéric Bettens and Todor Todoroff. Real-time dtw-based gesture recognition external object for max/msp and puredata. In *Proceedings of the 6th Sound and Music Computing Conference*, pages 23–25, Porto, Portugal, 2009.

[10] Frédéric Bevilacqua, Lisa Naugle, and Isabel Valverde. Virtual dance and music environment using motion capture. In *IEEE Multimedia Technology and Applications Conference Proceedings*, pages 1–4, Irvine, CA, 2001. URL `http://music.arts.uci.edu/dobrian/motioncapture/Bevilacqua-mtac-proceeding.pdf`.

[11] Frédéric Bevilacqua, Fabrice Guédy, Norbert Schnell, Emmanuel Fléty, and Nicolas Leroy. Wireless sensor interface and gesture-follower for music pedagogy. *New Interfaces For Musical Expression*, 2007. URL `http://portal.acm.org/citation.cfm?id=1279740.1279762`.

[12] Tina Blaine. The Convergence of Alternate Controllers and Musical Interfaces in Interactive Entertainment. In *Proceedings of the 2005 conference on New interfaces for musical expression*, pages 27–33, Singapore, Singapore, 2005. National University of Singapore.

[13] Ginevra Castellano, Roberto Bresin, Antonio Camurri, and Gualtiero Volpe. Expressive Control of Music and Visual Media by Full-Body Movement. In *Proceedings of 2007 Conference on New Interfaces for Musical Expression*, pages 390–391, New York, NY, 2007.

[14] Nick Crampton, Kaitlyn Fox, Hannah Johnston, and Anthony Whitehead. Dance, Dance Evolution: Accelerometer Sensor Networks as Input to Video Games. In *2007 IEEE International Workshop on Haptic, Audio and Visual Environments and Games*, pages 107–112. IEEE, Oktober 2007. ISBN 978-1-4244-1570-0. doi: 10.1109/HAVE.2007.4371597. URL `http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=4371597`.

[15] Annet Dekker. Synaesthetic Performance In The Club Scene. In Grethe Mitchell, editor, *3rd international conference on Computational Semiotics for Games and New Media*, pages 24–30, Teeside, UK, 2003. URL `http://www.cosignconference.org/downloads/papers/dekker_cosign_2003.pdf`.

[16] Annet Dekker. Dancing in the Light of an Information Overload. In *Inter-Society for the Electronic Arts Conference 2004*, pages 18–21, 2004. URL `http://www.montevideo.nl/en/nieuws/detail.php?id=67`.

[17] Simon Dixon. Evaluation of the Audio Beat Tracking System BeatRoot. *Journal of New Music Research*, 36(1):39–50, März 2007. ISSN 0929-8215. doi: 10.1080/09298210701653310. URL `http://www.informaworld.com/openurl?genre=article&doi=10.1080/09298210701653310`.

[18] Magy Seif El-Nasr and Thanos Vasilakos. DigitalBeing: an Ambient Intelligent Dance Space. *Informatics and Computer Science Intelligent Systems Applications*, 16802, 2006. URL http://magyweb.net/conference/FuzzIeee2006.pdf.

[19] Magy Seif El-Nasr and Thanos Vasilakos. DigitalBeing — Using the Environment as an Expressive Medium for Dance. *Information Sciences*, 178(3):663–678, Februar 2008. ISSN 00200255. doi: 10.1016/j.ins.2007.08.025. URL http://linkinghub.elsevier.com/retrieve/pii/S0020025507004069.

[20] A. Engström, M. Esbjörnsson, and O. Juhlin. Mobile collaborative live video mixing. In *Proceedings of the 10th international conference on Human computer interaction with mobile devices and services - MobileHCI '08*, page 157, New York, New York, USA, 2008. ACM Press. ISBN 9781595939524. doi: 10.1145/1409240.1409258. URL http://portal.acm.org/citation.cfm?doid=1409240.1409258.

[21] Urs Enke. *DanSense: Rhythmic Analysis of Dance Movements Using Acceleration-Onset Times*. Master thesis, RWTH Aachen University, 2006. URL http://hci.rwth-aachen.de/materials/publications/enke2006.pdf.

[22] Mark Christopher Feldmeier. *Large group musical interaction using disposable wireless motion sensors*. Master thesis, Massachusetts Institute of Technology, 2002. URL http://www.media.mit.edu/resenv/pubs/theses/Feldmeier-SM.pdf.

[23] Emmanuel Fléty. The Wise Box: a multi-performer wireless sensor interface using WiFi and OSC. In *Proceedings of the 2005 conference on New interfaces for musical expression*, pages 266–267. National University of Singapore, 2005. URL http://portal.acm.org/citation.cfm?id=1085939.1086024.

[24] Minoru Fujimoto, Naotaka Fujita, Yoshinari Takegawa, Tsutomu Terada, and Masahiko Tsukamoto. Musical B-boying: A Wearable Musical Instrument by Dancing. *Lecture Notes In Computer Science; Vol. 5309*, 2008. doi: 10.1007/978-3-540-89222-9_17. URL http://portal.acm.org/citation.cfm?id=1483460.1483478.

[25] Carrie Gates, Sriram Subramanian, and Carl Gutwin. DJs' perspectives on interaction and awareness in nightclubs. In *Proceedings of the 6th ACM conference on Designing Interactive systems - DIS '06*, page 70, New York, New York, USA, 2006. ACM Press. ISBN 1595933670. doi: 10.1145/1142405.1142418. URL http://portal.acm.org/citation.cfm?doid=1142405.1142418.

[26] Masataka Goto and Yoichi Muraoka. *Music Understanding At The Beat Level Real-time Beat Tracking For Audio Signals*, pages 157–176. Lawrence Erlbaum Associates, 1998.

[27] Niall Griffith and Mikael Fernström. LiteFoot: A floor space for recording dance and controlling media. In *Proceedings of the 1998 International Computer Music Conference*, pages 475–481, 1998.

[28] Carlos Guedes. Extracting Musically-Relevant Rhythmic Information from Dance Movement by Applying Pitch Tracking Techniques to a Video Signal. In *Proceedings of the 2006 Sound and Music Computing Conference*, pages 25–33, Marseille, France, 2006.

[29] Jürg Gutknecht, Irena Kulka, Paul Lukowicz, and Tom Strieker. *Advances in Expressive Animation in the Interactive Performance of a Butoh Dance*, pages 418–433. Springer, 2008. doi: 10.1007/978-3-540-79486-8_33. URL `http://www.springerlink.com/index/hq3q49n1484q6n75.pdf`.

[30] Aristotelis Hadjakos, Erwin Aitenbichler, and Max Mühlhäuser. SYSSOMO: A Pedagogical Tool for Analyzing Movement Variants Between Different Pianists. In *5th International Conference on Enactive Interfaces*, Pisa, Italy, 2008. Edizione ETS.

[31] Aristotelis Hadjakos, Erwin Aitenbichler, and Max Mühlhäuser. Potential Use of Inertial Measurement Sensors for Piano Teaching Systems : Motion Analysis of Piano Playing Patterns. In Kia C. Ng, editor, *Proceedings of the 4th i-Maestro Workshop on Technology - Enhanced Music Education*, pages 61–68, 2008.

[32] Jason A Hockman, Marcelo M Wanderley, and Ichiro Fujinaga. Real-time Phase Vocoder Manipulation by Runner's Pace. In *The 9th International Conference on New Interfaces for Musical Expression*, pages 90–93, Pittsburgh, PA, 2009. URL `http://archive.notam02.no/arkiv/proceedings/NIME2009/nime2009/pdf/author/nm090090.pdf`.

[33] Dennis Hromin, Michael Chladil, Natalie Vanatta, David Naumann, Susanne Wetzel, Farooq Anjum, and Ravi Jain. CodeBLUE: a bluetooth interactive dance club system. In *GLOBECOM '03. IEEE Global Telecommunications Conference*, volume 5, pages 2814–2818. IEEE, 2003. ISBN 0-7803-7974-8. doi: 10.1109/GLOCOM.2003.1258748. URL `http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=1258748`.

[34] Eric Johnstone. A MIDI Foot Controller - The PodoBoard. In *International Computer Music Conference*, pages 123 – 126, San Francisco, CA, 1991.

[35] Yoonji Kim, Donggi Jung, Sehwi Park, Jumin Chi, Taewoo Kim, and Seny Lee. The Shadow Dancer: A New Dance Interface with Interactive Shoes. *International Conference on Cyberworlds*, page 3, 2008. URL `http://portal.acm.org/citation.cfm?id=1488624`.

[36] Kai Kunze, Michael Barry, Ernst A. Heinz, Paul Lukowicz, Dennis Majoe, and Jürg Gutknecht. Towards Recognizing Tai Chi — An Initial Experiment Using Wearable Sensors. In *IFAWC 2006 - Third International Forum on Applied*

Wearable Computing, Bremen, Germany, 2006. doi: 10.1.1.109.1322. URL TowardsRecognizingTaiChi---AnInitialExperimentUsingWearableSensors.

[37] Celine Latulipe and Sybil Huskey. Dance. Draw: exquisite interaction. In *Proceedings of the 22nd British CHI Group Annual Conference on HCI 2008: People and Computers XXII: Culture, Creativity, Interaction-Volume 2*, pages 47–51. British Computer Society Swinton, UK, UK, 2008. URL http://www.bcs.org/upload/pdf/ewic_hc08_v2_paper12.pdf.

[38] Daniel Lemire. Faster retrieval with a two-pass dynamic-time-warping lower bound. *Pattern Recognition*, 42(9):2169–2180, 2009. URL http://linkinghub.elsevier.com/retrieve/pii/S0031320308004925.

[39] Xiubo Liang, Qilei Li, Xiang Zhang, Shun Zhang, and Weidong Geng. Performance-driven motion choreographing with accelerometers. *Computer Animation and Virtual Worlds, 20*, 2(3):89–99, 2009. doi: 10.1002/cav. URL http://www.ingentaconnect.com/content/jws/cav/2009/00000020/F0020002/art00003.

[40] Jani Mäntyjärvi, Johan Himberg, and Tapio Seppänen. Recognizing human motion with multiple acceleration sensors. In *2001 IEEE International Conference on Systems, Man and Cybernetics. e-Systems and e-Man for Cybernetics in Cyberspace*, pages 747–752. Ieee, 2001. ISBN 0-7803-7087-2. doi: 10.1109/ICSMC.2001.973004. URL http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=973004.

[41] M.T. Marshall, N. Peters, A.R. Jensenius, J. Boissinot, M.M. Wanderley, and J. Braasch. On the development of a system for gesture control of spatialization. In *Proceedings of the International Computer Music Conference*, number Pulkki 1997, page 260â€“266, 2006. URL http://www.idmil.org/_media/wiki/icmc2006_marshall_peters_et_al.pdf.

[42] Lisa McElligott, Michelle Dillon, Krispin Leydon, Bruce Richardson, Mikael Fernström, and Joseph A. Paradiso. 'ForSe FIElds' - Force Sensors for Interactive Environments. In Gaetano Borriello and Lars E. Holmquist, editors, *UbiComp 2002: Ubiquitous Computing*, volume 2498 of *Lecture Notes in Computer Science*, pages 168–175, Göteborg, Sweden, 2002. Springer. ISBN 978-3-540-44267-7. doi: 10.1007/3-540-45809-3_13. URL http://www.springerlink.com/index/10.1007/3-540-45809-3_13.

[43] Luiz Naveda and Marc Leman. Representation of Samba dance gestures, using a multi-modal analysis approach. In *5th International Conference on Enactive Interfaces*, pages 68–74, Pisa, Italy, 2008. Edizione ETS. doi: 1854/LU-503783. URL http://hdl.handle.net/1854/LU-503783.

[44] Kia C. Ng. Music via Motion: Transdomain Mapping of Motion and Sound for Interactive Performances. *Proceedings of the IEEE*, 92(4):645–655, April 2004. ISSN 0018-9219. doi: 10.1109/JPROC.2004.825885. URL http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=1278688.

[45] Joseph A. Paradiso and Eric Hu. Expressive footwear for computer-augmented dance performance. In *First International Symposium on Wearable Computers (ISWC '97)*, pages 165–166. IEEE Computer Society, 1997. ISBN 0-8186-8192-6. doi: 10.1109/ISWC.1997.629936. URL `http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=629936`.

[46] Joseph A. Paradiso, Craig Abler, Kai-yuh Hsiao, and Matthew Reynolds. The Magic Carpet: Physical Sensing for Immersive Environments. In *Conference on Human Factors in Computing Systems*, pages 277–278. ACM New York, NY, USA, 1997. URL `http://portal.acm.org/citation.cfm?id=1120391`.

[47] Joseph A. Paradiso, Eric Hu, and Kai-yuh Hsiao. Instrumented footwear for interactive dance. In *XII Colloquium on Musical Informatics*, volume 24, pages 89–92, Gorizia, Italy, 1998. URL `http://www.media.mit.edu/resenv/pubs/papers/98_07_CMI_Shoe.pdf`.

[48] Joseph A. Paradiso, Kai-yuh Hsiao, and Eric Hu. Interactive music for instrumented dancing shoes. In *Proceedings of the 1999 International Computer Music Conference*, pages 453–456, 1999. URL `http://gn.www.media.mit.edu/resenv/pubs/papers/99_10_ICMC_Shoe.pdf`.

[49] Bo Peng, Gang Qian, and Yunqian Ma. Recognizing body poses using multilinear analysis and semi-supervised learning. *Pattern Recognition Letters*, 30(14):1289–1294, 2009. ISSN 0167-8655. URL `http://portal.acm.org/citation.cfm?id=1595906.1596208`.

[50] Vytautas Perlibakas. Distance measures for PCA-based face recognition. *Pattern Recognition Letters*, 25(6):711–724, 2004. ISSN 01678655. doi: 10.1016/j.patrec.2004.01.011. URL `http://linkinghub.elsevier.com/retrieve/pii/S0167865504000248`.

[51] Gang Qian, Shamik Sural, Yuelong Gu, and Sakti Pramanik. Similarity between euclidean and cosine angle distance for nearest neighbor queries. In *Proceedings of the 2004 ACM symposium on Applied computing*, pages 1232–1237, New York, NY, 2004. ISBN 1581138121. doi: 10.1145/967900.968151. URL `http://portal.acm.org/citation.cfm?id=968151`.

[52] Chotirat Ann Ratanamahatana and Eamonn Keogh. Three myths about dynamic time warping data mining. In *Proceedings of SIAM International Conference on Data Mining*, volume 05, 2005. URL `http://siam.org/proceedings/datamining/2005/dm05_50ratanamahatanac.pdf`.

[53] Nishkam Ravi, Nikhil Dandekar, Preetham Mysore, and Michael L. Littman. Activity Recognition from Accelerometer Data. *Proceedings of the National Conference on Artificial Intelligence*, 20(3):1541–1546, 2005. URL `http://www.aaai.org/Papers/IAAI/2005/IAAI05-013.pdf`.

[54] Hiroaki Sakoe and Seibi Chiba. Dynamic Programming Algorithm Optimization for Spoken Word Recognition. *IEEE Transactions on Acoustics, Speech and Signal Processing*, 26(1):43–49, 1978. doi: 10.1.1.114.3782.

[55] Stan Salvador and Philip Chan. FastDTW: Toward Accurate Dynamic Time Warping in Linear Time and Space. In *KDD Workshop on Mining Temporal and Sequential Data*, pages 70–80, 2004. URL `http://cs.fit.edu/~pkc/papers/tdm04.pdf`.

[56] Steven Salzberg. Distance metrics for instance-based learning. In *Proceedings of the 6th International Symposium on Methodologies for Intelligent Systems*, pages 399–408. Springer, 1991. URL `http://www.springerlink.com/index/70132j9318g70474.pdf`.

[57] William A. Sethares and Thomas W. Staley. Periodicity transforms. *IEEE transactions on Signal Processing*, 47(11):2953–2964, 1999. URL `http://minds.wisconsin.edu/bitstream/1793/10052/1/file_1.pdf`.

[58] Wayne Siegel. Two compositions for interactive dance. In *Proc. of the Int. Computer Music Conf.(ICMC99)*, pages 56–59, 1999. URL `http://mtg.upf.edu/mosart/papers/p04.pdf`.

[59] Wayne Siegel and Jens Jacobsen. The challenges of interactive dance: An overview and case study. *Computer Music Journal*, 22(4):29â€“43, 1998. URL `http://www.jstor.org/stable/3680892`.

[60] Jacob Smith. I Can See Tomorrow In Your Dance: A Study of Dance Dance Revolution and Music Video Games. *Journal of Popular Music Studies*, 16(1):58–84, 2004. ISSN 1524-2226. doi: 10.1111/j.0022-4146.2004.00011.x. URL `http://www.blackwell-synergy.com/links/doi/10.1111%2Fj.0022-4146.2004.00011.x`.

[61] Prashant Srinivasan, David Birchfield, Gang Qian, and Assegid Kidané. A pressure sensing floor for interactive media applications. *ACM International Conference Proceeding Series; Vol. 265*, 2005. URL `http://portal.acm.org/citation.cfm?id=1178526`.

[62] D. Andrew Stewart. SonicJumper composer. In *International Conference on New Interfaces for Musical Expression (NIME06)*, pages 103–105, Paris, France, 2006. URL `http://www.nime.org/2006/proc/nime2006_103.pdf`.

[63] Kai-Tai Tang, Howard Leung, Taku Komura, and Hubert P. H. Shum. Finding repetitive patterns in 3D human motion captured data. *Conference On Ubiquitous Information Management And Communication*, page 7, 2008. URL `http://portal.acm.org/citation.cfm?id=1352876`.

[64] Ryan Ulyate and David Bianciardi. The Interactive Dance Club: Avoiding Chaos in a Multi-Participant Environment. *Computer Music Journal*, 26(3):40–49, 2002. ISSN 0148-9267. doi: 10.1162/014892602320582963. URL `http://www.mitpressjournals.org/doi/abs/10.1162/014892602320582963`.

[65] Rudolf von Laban. *Principles of dance and movement notation*. Macdonald and Evans, London, 1956.

[66] Todd Winkler. Making Motion Musical: Gesture Mapping Strategies for Interactive Computer Music. In *Proceedings for the 1995 International Computer Music Conference*, San Francisco, CA, 1995.

[67] Todd Winkler. Creating Interactive Dance with the Very Nervous System. In *Proceedings of Connecticut College Symposium on Arts and Technology*, 1997. URL `http://www.brown.edu/Departments/Music/sites/winkler//papers/Interactive_Dance_1997.pd.pdf`.

[68] Todd Winkler. Audience Participation and Response in Movement-Sensing Installations. In *Proceedings of the 2000 International Computer Music Conference*, 2000.