Saarland University
Faculty of Natural Sciences and Technology I
Department of Computer Science

Master's Thesis

Automated Classification of Trampoline Motions Based on Inertial Sensor Input

submitted by
Heike Brock

submitted
December 20, 2010

Supervisor / Advisor
Priv.-Doz. Dr. Meinard Müller

Reviewers
Priv.-Doz. Dr. Meinard Müller
Prof. Dr. Alfred Effenberg

Eidesstattliche Erklärung

Ich erkläre hiermit an Eides Statt, dass ich die vorliegende Arbeit selbstständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel verwendet habe.

Statement under Oath

I confirm under oath that I have written this thesis on my own and that I have not used any other media or materials than the ones referred to in this thesis.

Einverständniserklärung

Ich bin damit einverstanden, dass meine (bestandene) Arbeit in beiden Versionen in die Bibliothek der Informatik aufgenommen und damit veröffentlicht wird.

Declaration of Consent

I agree to make both versions of my thesis (with a passing grade) accessible to the public by having them added to the library of the Computer Science Department.

Saarbrücken, _____ _____

(Datum / Date) (Unterschrift / Signature)

Acknowledgements

I would like to express my gratitude to my supervisor, Priv.-Doz. Dr. Meinard Müller, whose expertise and patience added considerably to my graduate experience. Furthermore, I would like to thank for giving me the opportunity to work on such an interesting thesis and the multimedia retrieval group at the Max-Planck Institute, Saarbrücken. The friendly and supportive atmosphere in the group contributed essentially to the final outcome of my studies. In this context I would like to thank particularly Thomas Helten, who always gave me advice and support during both my experiments and the writing of my thesis, and Andreas Baak. Furthermore, I would like to thank Prof. Dr. Alfred Effenberg from Hannover University for taking time to serve as my external reader.

I must also acknowledge the colleagues of my trampoline sports club Patrizia Micalizzi, Sarah Hidlmayer and Sylvia Human who supported my work as motion actors. Without their trampoline performances, no meaningful results could have been obtained. In this context, I also owe my gratitude to the Sports Institute at Saarland University who let me record the motions in their gymnasium.

I am grateful to Stefan who always tried to support and relieve me in everyday life, also in times when I did not even ask for it. I furthermore appreciate all friends and other student workers at the MPI that helped to distract me from working for some time.

Finally, I would like to thank my parents for giving my financial support during my whole studies.

Abstract

The automatic segmentation and classification of an unknown motion data stream according to given motion categories constitutes an important research problem with applications in computer animation, medicine and sports sciences. In this thesis, we consider the scenario of trampoline motions, where an athlete performs a sequence of predefined trampoline jumps. Here, each jump follows certain rules and belongs to a specific motion category such as a pike jump or a somersault. Then, the classification problem consists in automatically segmenting an unknown trampoline motion sequence into its individual jumps and to classify these jumps according to the given motion categories. Since trampoline motions are very fast and spacious while requiring special lighting conditions, it is problematic to capture trampoline motions with video and optical motion capture systems. Inertial sensors that measure accelerations and orientations are more suitable for capturing trampoline motions and therefore have been used for this thesis. However, inertial sensor output is noisy and abstract requiring suitable feature representations that display the characteristics of each motion category without being sensitive to noise and performance variations. A sensor data stream can then be transformed into a feature sequence for classification. For every motion category, a class representation (or in our case, a class motion template) is learned from a class of example motions performed by different actors. The main idea, as employed in this thesis, is to locally compare the feature sequence of the unknown trampoline motion with all given class motion templates using a variant of dynamic time warping (DTW) in the comparison. Then, the unknown motion stream is automatically segmented and locally classified by the class template that best explains the corresponding segment. Extensive experiments have been conducted on trampoline jumps from various athletes for evaluating various feature representations, segmentation and classification.

Contents

Chapter 1

Introduction

The automatic segmentation and classification of an unknown motion data stream according to given motion categories constitutes an important research problem with applications in computer animation, medicine and sports sciences. In general, classification means to assign input data into one of a given number of categories based on its contents. In relation with the general description, motion classification is the process of assigning a motion label that matches best the characteristics of a given input motion document. In this thesis, we consider the scenario of trampoline motions, where an athlete performs a sequence of predefined trampoline jumps. Here, each jump follows certain rules and belongs to a specific motion category such as a pike jump or a somersault. So stable and accurate classification results are mainly based on three fundamental methodologies: the performance of a motion, the capturing of a motion and the analysis of a motion.

Motion Performance. There are various kinds of motions that can either be simple and natural motions of everyday life or sporting motions that can have either health, enjoyment or professional competitive purposes. For this thesis, we chose a motorically demanding but also very aesthetic sports motion consisting of several different motion types that can be easily separated from each other: trampolining. Trampolining is closely related to gymnastics including similar acrobatic motions. However, due to the bounciness and elastic properties of a trampoline, even more complicated and fast motions than in gymnastics can be performed that will be challenging for classification. As every trampoline jump starts and ends with a phase where the athlete is in contact with the trampoline bed, it is however much easier to segment single jumps from a sequence of consecutive jumps than in other sports. This property is helpful for classification issues. Overall, trampolining offers a very interesting, but also stable and controlled environment for motion classification.

Motion Capturing. The idea of motion capturing has already been initiated in 1878, when Eadweard Muybridge published a photo series about a galloping horse proving that horses do not always keep contact with the ground with at least one leg. With his following publications about animal locomotion and human motion in 1887 and 1901, Muybridge was one of the pioneers for the analysis of (human) motion [37]. Since the beginning of motion analysis, motion capturing became a professional industrial branch. Nowadays, many motion capture systems for different application purposes are available that deliver three dimensional motion information. Usual motion capture systems are either marker-

(a) (b) (c) (d)

Figure 1.1. Since the beginning of motion analysis, different motion capture systems have been developed. (a) Mechanical motion capture setup with a triaxial goniometer from 1980[1]. (b) Optical marker-based motion capturing. (c) Optical markerless motion capturing. (d) Inertial motion capturing as used in this thesis.

based optical motion capture systems, mechanic or magnetic motion capture systems or markerless optical motion capture systems based on two dimensional computer vision techniques, see Figure 1.1. All those systems offer advantages, but also disadvantages regarding the size of the capture volume, the amount of mobility for the motion actor or additional overhead regarding pre- and post-processing steps like calibration and system setup. For example, optical motion capture systems offer good mobility and provide the user with high-quality data. On the other hand, a wide capture volume for spacious motions can only be obtained if combined with an appropriate indoor capture environment and a large number of optical devices available. Furthermore, optical systems come up with a highly inefficient overhead for pre-processing due to their extensive preparation phase and difficult calibration. For easy and everyday use, it would be reasonable to provide the user with some motion capture technique that does not restrict the capture volume to indoor performances of non spacious motions and the actor's mobility without complicated setup procedures. Here, innovative motion capture systems like systems based on inertial sensors can be useful, see Figure 1.1(d).

Inertial sensors, or *inertial measurement units* (IMUs), work with micro-mechanical components as accelerometers and rate gyros that supply the user with information about acceleration, rotation and orientation. Because of the mechanical components, data can be captured without extensive preparation or calibration and the sensors are of small size and weight so that they do not restrict the motion performance. Since trampoline motions are very fast and spacious and require special lighting conditions, it is problematic to capture trampoline motions with video and optical motion capture systems. Here, inertial sensors that measure accelerations and orientations are more suitable so that the trampolining motions have been captured with inertial devices in this thesis. On the other hand, the sensors do not yield positional information of a motion that can be used directly for further applications as traditional motion capture systems. In contrast to the very precise three dimensional positional data of optical marker-based motion capture data, inertial data is noisy and tends to be sparse and unintuitive as no positional data and only orientation and acceleration information can be obtained from the capture devices. Therefore, suitable and good data representations have to be found that represent well

[1] *http : //www.xsens.com/images/stories/gonio_old.jpg*

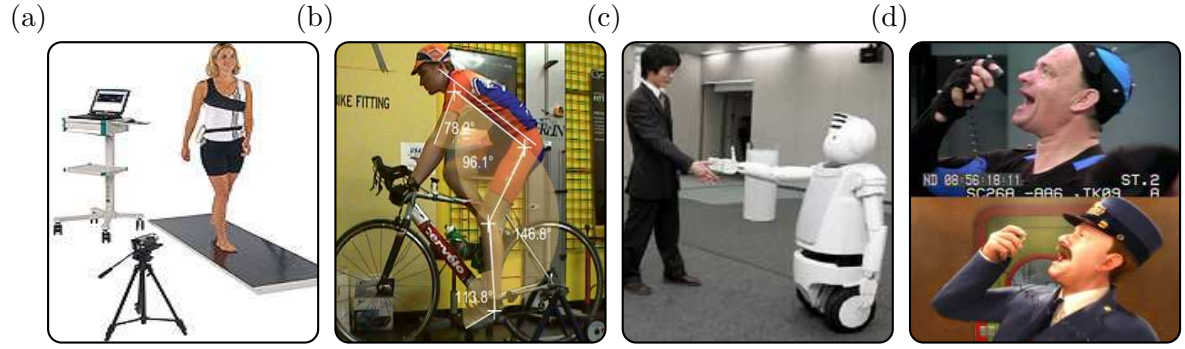

Figure 1.2. Different application fields of motion analysis[2]. (a) Gait analysis system in orthopaedics. (b) Two dimensional video analysis tool Dartfish for sport performance monitoring. (c) Robot performing a learned human motion. (d) Computer animated scene from motion capture data.

the characteristics of a given motion.

Motion analysis. Apart from the evolution of different motion capture systems, motion analysis became an interdisciplinary research field since the beginning of motion analysis in 1878 by Eadweard Muybridge. Main research fields that deal with human motion capture data are medicine, sports science, robotics and computer science. Figure 1.2 shows an overview over the application fields of motion analysis.

The most common request for motion analysis in medicine is the analysis of human gait patterns for orthopedics. Abnormalities that lead to injuries or physical infirmities can be detected from the captured motion data. In recovery applications and rehabilitation, the motion data can be used for medical survey. For example, the healing process of a patient, but also incorrect motion behavior can be determined. Survey issues that prevent fall and injury of elderly people are another application for medical motion analysis [7, 10, 21]. The main purpose of using motion capture data in sports is to supply the athlete with feedback about a previously performed motion. Using additional information from motion capture data that cannot be detected during the performance helps to identify motion phases that could be improved or to get an impression on how precise biomechanical parameters have been obeyed. Two dimensional video analysis systems like Dartfish [11] are common tools for performance monitoring, whereas three dimensional data processing tools have just started to become popular for motion analysis [6, 27]. In robotics, the main purpose of motion analysis is to find the essential parameters that characterize a real-life motion. Those characteristics can then be transfered on robots or can be automated for virtual-reality applications to produce a realistic and natural motion pattern [30, 32]. One application of motion analysis using motion capture data in computer science is computer animation. For realistic three dimensional animations, it is often required to create a motion looking in a most natural way [2, 12, 17]. Three dimensional information of a motion can be created manually by keyframe animation. The resulting motions, however, often look artificial and unnatural. To obtain more realistic motions, a motion

[2](a): $http://www.zebris.de$,
(b): $http://rosachiropracticfairfax.com/clients/2216/images/Dartfish\_Fairfax\_VA.jpg$,
(c): $http://www.weblogsinc.com/common/images/7707148614858914.JPG?0.3549340219832601$,
(d): $http://thinkdrawsell.files.wordpress.com/2008/09/polar-mocap.jpg$

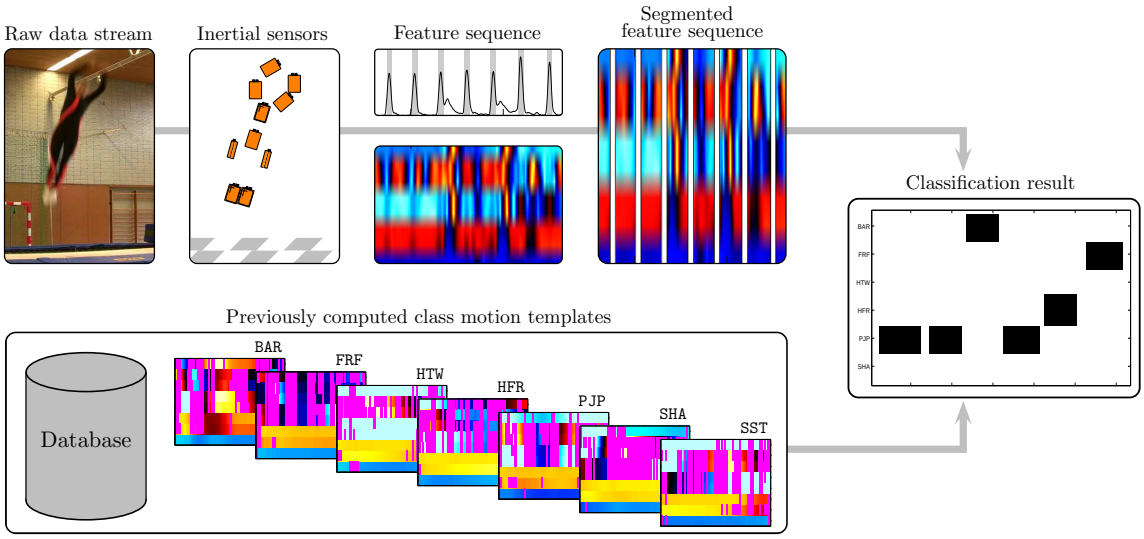

Figure 1.3. Classification method used in this thesis. Trampoline jumps from inertial sensor input is separated by an automatic segmentation method and can be labeled as motion category with previously computed motion class templates.

capture system can be used yielding positional data of the recorded motion that can easily be transferred to the motion of a three dimensional character [9]. Another application scenario in the field of computer science is the efficient retrieval of multimedia data (and herewith also of human mocap data). Here, content-based procedures for data annotation and efficient content-based retrieval methods have been developed that work on the raw data itself and do not rely on manually generated textual annotations or metadata [16, 25, 26]. Meanwhile, several techniques for multimedia retrieval and classification have been developed that make it nonessential to capture new motion data as the existing data from an unknown database can be reused instead.

All application fields of motion analysis impose different requirements on the motion capture data and the motion capture systems, but also on the usage of the acquired data. For example, in computer animation, the accuracy of the captured data is very important. Consequently, common capture system are optical marker-based capture systems. Using the precise positional data, further analysis by comparisons or similarity measures can give detailed information about the motion. In medicine and sports sciences, however, it is important to obtain the data in real-time from an easy-to-use capture system. Consequently, preferred capture systems are inertial capture systems. Here, it is often sufficient to examine statistical quantities like the correlation coefficient or the mean and median of the motion capture data or to use the Fourier Transform [5, 34]. In our scenario, as we want to analyze and understand previously performed trampoline motions on a very fine level, we introduce how to use the analysis methods for positional data on the abstract inertial data.

In this thesis, we consider the problem to automatically segment an unknown trampoline motion sequence into its individual jumps and to classify these jumps according to given motion categories. Two different processing steps are necessary for a stable and accurate classification. In the first step, the captured inertial data is processed. We will discuss how

to find meaningful representations for every motion category that display the characteristics of each motion category. As the inertial output is difficult to analyze because of it's noisy and abstract character, those feature representations have to be not too sensitive to noise, performance variations and changes in recording set-up and calibration. A sensor data stream can then be transformed into a feature sequence for classification. With an additional segmentation algorithm, the motion data stream will be separated, so that we may assume that every motion document consists of a single jump. In the second step, a class representation (or in our case, a class motion template) is learned, where each motion category is represented by example motions performed by different actors. In our scenario, a finite number of motion categories are given. Now, the idea is to locally compare the feature sequence of the unknown trampoline motion with all given class motion templates. In this thesis we use variants of dynamic time warping (DTW) for comparison. Then, an unknown motion is automatically segmented and locally classified by the class template that best explains the corresponding segment. Figure 1.3 gives a schematic overview of the classification pipeline in this thesis.

The main contributions of this thesis are the classification method based on inertial sensor input for a complex motion scenario and the design and analysis of various feature representations. For both, evaluation of feature representations as well as segmentation and classification, extensive experiments have been conducted on real trampoline movements performed by various actors.

Possible applications of the trampoline classification scenario are programs for motion understanding and motion analysis of athlete's performance that can then for example be used to supply the athlete with automatic feedback in training sessions or competitions. While a motion is performed, the input motion data can be classified. Using those classifications, deviations within the labeled class from the learned class motion template or from an optimal motion can be determined. Computer assisted training software can then use this deviation information on motion performances to give the athlete feedback about his motion technique, errors during motion performance and suggestions for improvements.

The thesis is structured as followed. In Chapter 2, elementary explanations to understand the concept of trampolining are given. The most common trampoline jumps and their technical requirements are introduced based on their different levels of difficulty. Furthermore, the elementary biomechanics of trampolining is explained. In Chapter 3, different motion capture systems and their modalities are listed and the requirements capturing trampoline motions imposes on a motion capture system are explained. The working principle of inertial sensors as well as their properties are then discussed in detail. Besides, physical quantities that occur while working with inertial sensors are recapulated. Chapter 4 gives a general introduction into the motion analysis and classification task in this thesis. All necessary methods and algorithms such as feature representations, similarity measures and motion templates are introduced and described. In Chapter 5, it is then explained how the captured trampoline motions have been built into a database. Furthermore, this chapter gives an overview of how the trampoline database has been structured and how the data has been processed for further experiments. From Chapter 6 on, all experiments that have been conducted for this thesis are described. In Chapter 6, experiments to evaluate the feature representations from Chapter 4 are discussed. Chapter 7 then evaluates the general motion templates and describes how to improve motion templates to handle

variations. Measures to quantify the evaluation results are introduced in both Chapter 6 and Chapter 7. Chapter 8 finally shows classification results for the chosen feature set and motion templates and different similarity measures. The automatic method for segmentation is explained as well. Again, all classification results for the different similarity measures are evaluated.

Chapter 2

Basic Trampolining

For the classification experiments in this thesis, we chose trampolining that is closely related to gymnastics where athletes perform acrobatic moves while bouncing on a trampoline. In 2000, trampolining became an Olympic Summer Discipline.

To better understand the specific requirements that are necessary to distinguish and classify different trampoline performances, it is helpful to understand the basic principles of trampolining. In this chapter, we will specify standard trampoline jumps for beginners as well as intermediate and advanced trampolinists and introduce the biomechanical principles and physical laws that build the base for a safe and correct jump and that allow the athlete to perform a specific trampoline jump.

In Section 2.1, basic trampoline terminology and the elementary technical description of trampolining is explained. In Section 2.2, easy trampoline jumps that can be learned with few trampolining experience are introduced. Not all of the beginner's moves listed in this section are considered in later experiments or are useful for a classification task, but as they are helpful to understand the technique and main idea of trampolining they are explained, as well. Section 2.3 describes intermediate trampoline jumps that can be learned with stable motorical knowledge of the basic moves. Many of the intermediate jumps are used for experiments in this thesis. In Section 2.4, advanced moves that require a lot of trampolining experience is listed. Those jumps have not been investigated in this thesis, but can help to improve the understanding of the principles of trampolining. Section 2.5 explains physical laws and biomechanical concepts that are the base for all trampoline moves and influence the performances of all trampolinists.

2.1 Trampoline Terminology

The trampoline bed is the elementary part of the trampoline and the part that the athlete bounces on. The bed is made up of bands of elastic material. The width of these bands contributes to the bounciness of the trampoline. The narrower the bands on the bed, the bouncier the trampoline.

Different trampoline jumps require different motorical skills, but all jumps can be split

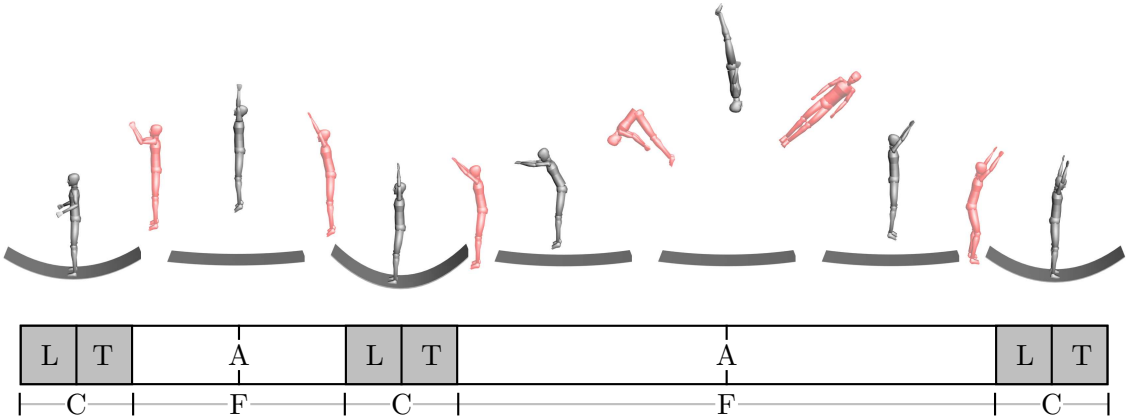

Figure 2.1. Phases of a trampoline jump: C - Contact phase divided in landing (L) and takeoff phase(T) and F - flight phase divided by the flight curve's apex (A).

into the same main phases that characterize the morphology of trampolining: the *contact phase* when the athlete is in contact with the trampoline bed and the *flight phase* when the athlete is in flight and can perform various motions. Contact and flight phases can be subdivided into further phases leading to a constant motion sequence of all trampoline jumps. In the beginning contact phase, the jump is prepared until the athlete loses contact with the trampoline bed and initiates a specific motion in the first half of the flight phase. The main part of the motion is performed at the apex of the flight curve while in the second half of the flight phase, the landing is already initiated. The jump ends with the time the athlete gets in contact with the trampoline bed again. Figure 2.1 visualizes the sequence of jump phases.

Trampoline moves can be separated easily by the contact phase and especially the moments the athlete gets in contact and loses contact with the trampoline bed. The parts of the trampolinist's body that touch the trampoline bed at the beginning and the ending of a motion further specify a trampoline jump. In general, there are four different *takeoff* and *landing* positions, see Figure 2.2, that lead to a variety of different motions:

Feet (FE): landing in a standing position

Seat (SE): landing in a sitting down position with the legs pointing straight ahead and the hands placed behind the hip

Front (FR): landing on the belly, or front, with the hands under the chin

Back (BA): landing straight on the back or on the back with arms and legs pointing upward

Landing on only one leg is not allowed as it can lead to serious injury.

The different landing and starting positions discriminate trampoline during the contact phase. During flight phase, various jumps can be performed that can be distinguished by their *motion shape*. The four different motion shapes for trampolining are:

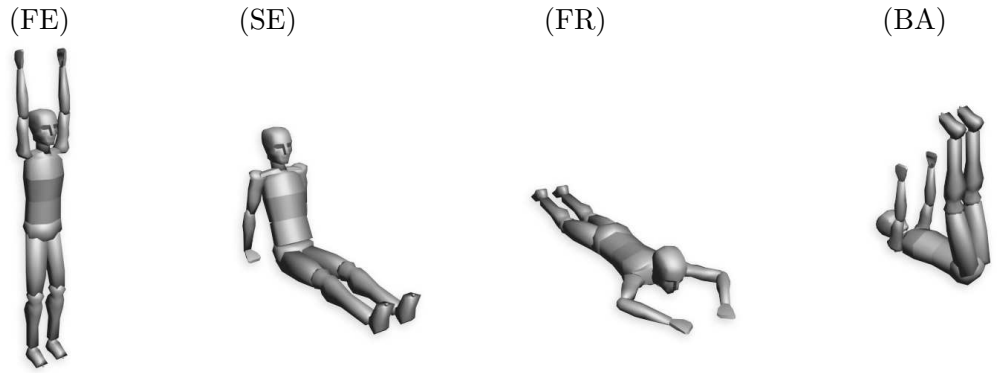

Figure 2.2. Different takeoff and landing positions: feet (FE), seat (SE), front (FR) and back (BA).

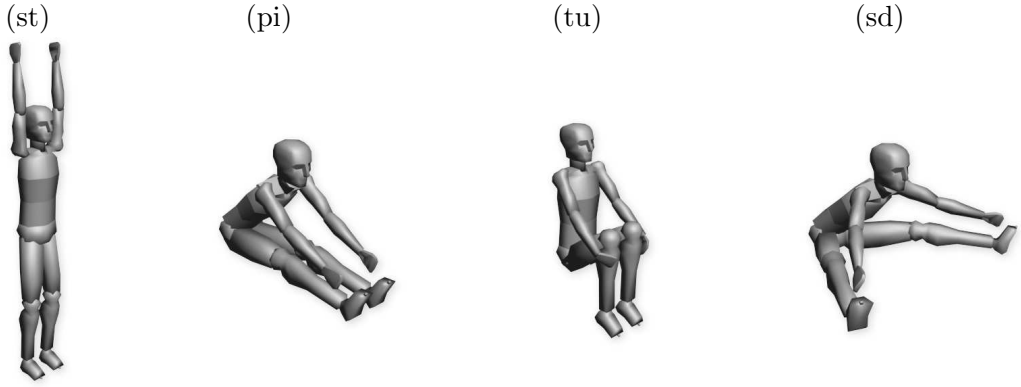

Figure 2.3. Four different possible motion shapes for trampoline moves: straight (st), piked (pi), tucked (tu) and straddle (sd).

Straight (st): upright with the body fully extended and elongated in direction of the longitudinal axis

Piked (pi): touching the toes like position with legs straight and together and the hands near the feet

Tucked (tu): curled up in a little ball with the hands on the shins and the knees together

Straddle (sd): piked shape with the feet apart

Figure 2.3 shows all different motion shapes. Those motion shapes can be executed as a single move (for example by starting and landing on both feet) as well as be contained in all other forms of somersaults and rotations. The motion shapes are an important indicator for performance differences in competitions.

During flight phase, the jumps can also be distinguished by *rotational motion*. In combination with the different starting and landing positions, the athlete has various possibilities of performing rotations around both the lateral and the longitudinal axes. Rotations around the dorsoventral axis are possible as well, but will not be discussed further in this thesis as the resulting turntables are generally not considered to be valid moves within trampolining competitions and are more difficult to learn and to perform. Figure 2.4 shows an

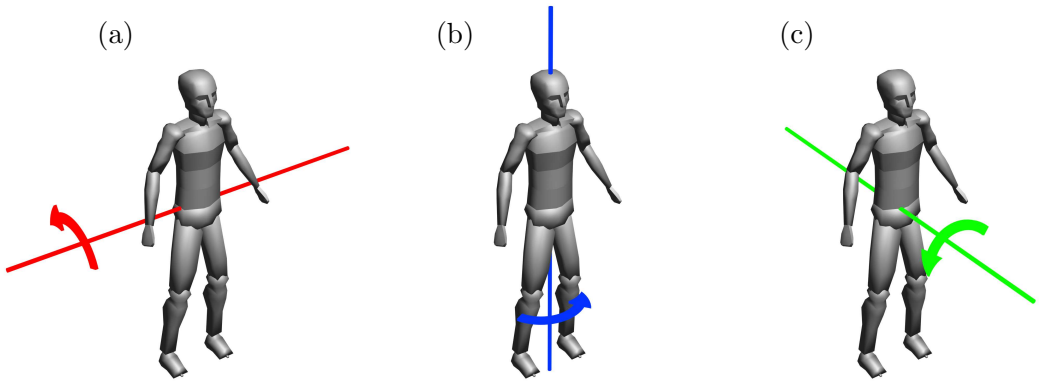

Figure 2.4. Overview over the three rotation axes: (a) lateral axis (red), (b) longitudinal axis (blue) and (c) dorsoventral axis (green).

overview of the three axes of rotation. Articles about snowboard moves sometimes include turntables moves and can offer additional information [14].

By using muscle forces of the legs and arms as well as making use of physical principles and effects like the reactive forces of the trampoline bed, the athlete can gain and control the height of the jump. For example, to take advantage of the elastic bed, the athlete can make better use of muscular actions like the stretch produced by some knee action during the landing phase.

2.2 Easy Trampoline Moves for Novices

The following jumps are basic technical skills each trampolinist has to master before learning more complicated motions. They build a base for intermediate and advanced jumping techniques and provide the athlete with knowledge about the basic shapes of a trampoline jump. None of the further listed trampoline moves contains rotations of more than 360°.

The first group of beginner's moves are moves that do not include any rotation starting and landing on both feet. The source for each trampoline motion is a simple *straight jump*. It is necessary to be able to perform this jump with motorical reliability before learning any other trampolining skills. Additionally to the straight motion style, trampoline moves can be performed in the previously describes shapes, leading to the *tuck jump*, the *pike jump* and the *straddle jump*.

The easiest rotational movement is the *twist* that can be seen as a straight jump with a rotation around the longitudinal axis by multiples of 180°. The more rotational motion the twist contains, the more difficult it becomes for the athlete.

A straight jump combined with a landing in the sitting position (or a start from the sitting position) is another beginners' move that does not require much expertise and can be achieved just by dropping on the trampoline bed. The athlete should focus on a correct landing into the *seat drop* simultaneously with the legs, the bottom and the palms of the hands. For landing in a position different to on both feet (seat, front or back landing),

Move	ID	Description
Straight Jump	STR	FE—st—FE
Tuck Jump	TJP	FE—tu—FE
Pike Jump	PJP	FE—pi—FE
Straddle Jump	SJP	FE—sd—FE
Half Twist	HTW	FE—Lo180—FE
Full Twist	FTW	FE—Lo360—FE
Seat Drop	SED	FE—SE
Seat To Stand	SST	SE—FE
Seat Half Twist To Stand	SHA	SE—Lo180—FE
Seat Half Twist To Seat Drop	SHS	SE—Lo180—SE
Half Twist Seat Drop	HTS	FE—Lo180—SE
Front Drop	FRD	FE—LaFW90—FR
Front To Feet	FRF	FR—FE
Half Twist Front Drop	HTF	FE—Lo180—LaFW90—FR
Back Drop	–	FE—LaBW90—BA
Back To Feet	–	BA—FE
Back Half Twist To Feet	–	BA—Lo180—FE
Half Twist Back Drop	–	FE—Lo180—LaBW90—BA

Table 2.1. Overview of easy trampoline jumps divided into the morphologically similar motion categories feet, twists, seat, front and back.

a correct landing technique is very important to prevent injuries like compressions of the spine at the moment of landing. In contrast to a seat drop, landing in the *back drop* requires more rotational motion that is initiated by higher pressure by the legs in forward and upward direction at the moment of leaving the trampoline bed. The *front drop* on the other hand requires higher pressure by the legs in backward and upward direction at the moment of leaving the trampoline bed. As a result, both landings contain a rotational motion of 90° around the lateral axis.

Body landings offer almost unlimited opportunities of execution for successive moves. Some examples would be the *cradle* that is a back drop followed by a half twist to a back drop, all different forms of twisting out of a seat drop and landing either on both feet or in the seat drop again, dropping to the front out of a sitting start position, a rotation of 180° around the lateral axis from the back to the front or vice versa, a rotation of 180° around the longitudinal axis from the back to the front or vice versa. The variety is limited only by the personal skills of an athlete and the physical properties that do not admit higher amounts of rotations for a height of the motion and jumping position. Rotations of 360° or more, however, cannot be considered as beginner's moves anymore.

Table 2.1 lists all easy trampoline jumps divided into groups of semantically similar motion categories determined by their main morphologies. The ID for every jump that will be used in this thesis is listed under ID, whereas the morpholic description of every motion can be found in the third column. The description of the jumps reads as followed: takeoff position—information about motion shapes and rotations—landing position. For the description of the motion shapes, the same abbreviations than in Section 2.1 with additional information on the amount of rotation in degrees and the direction of rotation (forwards or backwards).

Move	ID	Description
Somersault Backwards Tucked	BWC	FE—LaBW360 tu—FE
Somersault Backwards Piked	BWB	FE—LaBW360 pi—FE
Somersault Backwards To Seat Drop	BWS	FE—LaBW360 tu—SE
Somersault Backwards Straight	BWA	FE—LaBW360 st—FE
Somersault Forwards	–	FE—LaFW360—FE
Barani	BAR	FE—LaFW360—Lo180—FE
Three Quarter Backwards	3QB	FE—LaBW270 st—FR
Three Quarter Forwards	–	FE—LaFW270 st—BA

Table 2.2. Overview of intermediate trampoline jumps divided into the morphologically similar motion categories somersaults backwards, somersaults forwards and three quarter rotations.

2.3 Intermediate Trampoline Moves

Intermediate jumps usually include moves that contain at least one full 360° rotation or combinations between somersaults and twists. *Somersaults* can be executed in various ways either forwards or backwards in a tucked, piked or straight shape. Landing safely on both feet after performing a somersault forwards requires a good local orientation of the athlete during the rotation and is much more difficult than landing backwards with the trampoline bed as a helping orientation point. Therefore, somersaults forwards are usually combined with a half twist resulting in a *Barani* or with more twists resulting in advanced jumps.

Somersaults with a three quarter rotation around the lateral axis are called *Three quarter somersault forwards* or *backwards*. All moves containing somersaults with no full rotational motion resulting in a front or back-sided landing position, are seen as intermediate moves due to the difficulty of finding a proper landing position and the risk of injury by wrong or inexact landing positions. By an equally distributed landing on the front or the back successive rotational moves and somersaults are possible to come back to an upright standing position again.

Table 2.2 lists all intermediate trampoline jumps divided into groups of semantically similar motion categories determined by their main morphologies using the same representation than in Table 2.1. The ID for the somersaults investigated in this thesis are chosen in accordance to the official naming of trampoline jumps.

2.4 Advanced Trampoline Moves

Advanced jumps can be obtained by combining several easy and intermediate jumps in one flight phase. The complexity of trampoline moves is only restricted by physical laws as well as anatomical properties of the human body. The different landing positions can offer further possibilities of motion performance. In principle, somersaults and twists can be combined in one single move, leading to double or triple somersaults with several twists at the beginning or ending of the somersault or to full rotations with additional three quarter rotations.

Figure 2.5. 2010's world champion Dong Dong from China at the trampoline world championships in Metz, France.

Move	ID	Description
Rudolf	–	FE—LaFW360—Lo540—FE
Randolf	–	FE—LaFW360—Lo900—FE
Adolf	–	FE—LaFW360—Lo1260—FE
Cody	–	FR—LaBW450 tu—FE
Babyfliffis	–	BA—LaFW450 tu—Lo180—FE
Double Somersault Backwards	–	FE—LaBW720—FE
Double Somersault Forwards	–	FE—LaFW720—FE
Fliffis	–	BA—LaFW720—Lo180—FE

Table 2.3. Overview of advanced trampoline jumps divided into the morphologically similar motion categories somersaults forwards with twists, somersaults starting from body landings and multiple somersaults.

To be able to perform those demanding and complicated jumps, the athlete has to gain a sufficient height of flight with motorical reliability. Figure 2.5 shows the 2010 trampoline world champion in trampolining who reaches heights of almost 10 meters. Advanced jumps have not been further investigated in this thesis and are probably subject to further investigation requiring technically advanced motion actors.

Table 2.3 lists all intermediate trampoline jumps divided into groups of semantically similar motion categories determined by their main morphologies using the same representation than in Table 2.1. As for this thesis advanced jumps will not be used, we do not assign any ID to the moves.

2.5 Physical and Biomechanical Principles of Trampolining

Different trampoline beds offer different degrees of bounciness. This implies that the athlete needs to exert more or less muscular forces to gain the same height than on another trampoline bed. In dependency on the height of flight, the time to perform a move varies. For moves that are performed with similar flight heights, the time that passes while

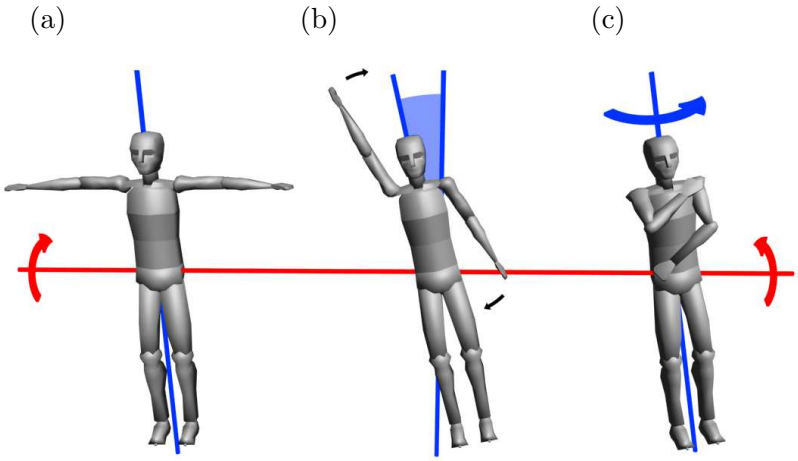

Figure 2.6. Rotational motion can be initiated during the flight phase by transfer of angular momentum. (a) Somersault rotation: the athlete has angular momentum only about his lateral axis (red line) with no twisting motion. (b) During somersault rotation, the athlete can bring his body into counterclockwise rotation (see light blue) by moving his arms in clockwise direction. The body begins a twisting motion around the longitudinal axis in order to conserve angular momentum. (c) The twisting and somersaulting motion continue without moving the arms and no external forces acting upon the athlete.

executing the motion remains stable, which will be useful for the later experiments. The higher the athlete gets thrown out of the trampoline bed, the more time he has to perform his motion, it is easier to perform advanced high rotational motions and in general, the motion technique is performed in a better way.

Different shapes of the trampoline moves as well as different starting positions require different accelerations exerted by the legs on the trampoline bed. For example, for the performance of a piked somersault compared to the performance of a tucked somersault, less rotational velocity occurs (for the same amount of force) as the athlete's moment of inertia is higher due to the legs that are stretched forward and not curled in direction of the center of gravity. Consequently, the athlete needs to transfer more reactive forces from the trampoline bed into his move to perform the same rotational motion at the same time, as a result the acceleration of the feet will be higher. The same principle is valid for the comparison of a somersault straight with a somersault piked, whereas the somersault straight is characterized by a higher moment of inertia. Differently shaped somersaults therefore can probably be distinguished by the amount of acceleration of the lower extremities. One physical idea that lies beyond this principle is the transfer of momentum, that describes the passing of the momentum from one body to another or from parts of a body to the whole body. In trampolining, the momentum can be set up in one part of the body and then be transferred to the whole body by muscular action. For example, starting in a front position and kicking with the legs transfers a momentum from the legs to the whole body when the muscles are used to lock the body straight resulting in a backwards rotation.

In general, all rotational motions are initiated by the transfer of angular momentum and depend on the amount of transferred momentum. The higher the momentum, the more rotations can be performed. Rotations around the lateral axis are usually initiated at the

(a) (b)

Figure 2.7. The physical parameters influence the quality of a motion performance. (a) Somersault opening of an advanced trampolinist in a vertical position. (b) A less experienced athlete is still in tucked shape at the same position and has to start the somersault opening later because he lacks angular momentum.

contact phase at the beginning of the move before leaving the trampoline bed and entering the *flight phase*. Rotational moves around the longitudinal axis can be initiated at the beginning of the move, but also during the aerial phase after a lateral rotation by twisting body parts. In this case, the angular momentum remains constant, but will be split over both rotational axes. Even in the absence of external torque, rotational motion can be initiated, see Figure 2.6 and [13].The biomechanics of trampolining have been investigated exactly and can be found in [4, 40].

The more experienced a trampolinist is, the better he can control and utilize a motion's physical and biomechanical parameters during flight phase. We will explain this with the motion phases for an ideal somersault technique on the trampoline (that would get the highest grades in a competition). The first phase is characterized by a fast motion initiation by the lower extremities with the upper body parts straight leading to high angular momentum. It is followed by the main part of rotation and the opening of the body by the legs to reduce the angular velocity (ideally in a vertical position with the head pointing downwards). Finally, the preparation of the landing concludes the somersault. Trampolinists on a higher level of skills usually master these phases much better than less experienced athletes that lack enough angular momentum and consequently do not open the somersault shape in a vertical position or maybe do not open their body at all to be able to perform a whole 360° rotation as it is visualized in Figure 2.7. Those differences can be quite significant and may be challenging for later retrieval results and applications including data of advanced and intermediate athletes.

Chapter 3

Inertial Motion Capturing

The trampoline jumps which are analyzed in this thesis were recorded using an inertial sensor-based motion capture system. To understand why we used such an inertial sensor-based system it is important to know the advantages and disadvantages of commonly used motion capture devices. Each of these motion capture systems has different properties and requirements concerning e.g. the recording environment, the size of the capture volume and the expressiveness of the provided data. For example, optical motion capture systems which are widely used in movie and game productions, provide very rich and easy to interpret data but they are also very restricted as far as the size of the capture volume or the lighting conditions are concerned. Unfortunately trampoline motions require a comparatively large capture volume. Furthermore, the recording of trampoline motions is restricted to locations where controlled lighting conditions are often not possible. These and other circumstances disqualify optical motion capture systems for the capturing of trampoline motions.

In recent years inertial sensors have been used in many fields such as entertainment applications, but also in medicine and sport sciences. Such inertial sensors have the advantage, that they do not impose any restrictions concerning the lighting conditions and only few constraints concerning the size of the capture volume. The drawback of such inertial sensor-based motion capture systems is the type of data they provide.

Section 3.1 gives on overview over several conventional motion capture systems. Their modalities, set up and properties will be discussed. In Section 3.2, we discuss the special requirements capturing trampoline motions imposes on a motion capture system. In Section 3.3, we show why it is reasonable to use inertial sensors for capturing trampoline motions. In Section 3.4, we discuss the capturing with inertial sensors. We introduce inertial sensors, their arrangement during the trampolining capture sessions and their working principle. Furthermore, necessary physical backgrounds and the corresponding physical quantities are recapulated.

3.1 Motion Capture Systems

There exist many different systems to capture human motion. The most common systems are: optical marker-based systems, optical markerless systems, mechanical systems, magnetic systems and inertial systems as used for this thesis. Every system is characterized by a special setup, special capture devices and special capture requirements. Due to the individual system modalities, the information content of the obtained motion capture data differs, as well. While each technology has its strengths, there is not a single motion capture technology that is perfect for every possible use. In the following, a short description of every system will be given. For further information, see [19, 38].

Optical systems. Optical marker-based systems use data from several image sensors to triangulate the three dimensional position of a subject.
Marker-based systems use retroreflective markers that are attached to an actor's body. Retroreflective markers reflect light that is produced by the optical cameras back to the cameras with a minimum scattering of light. The markers position can be tracked by the cameras and computed into positional data. Marker-based optical motion capture systems are common systems used for computer animation to obtain realistically moving objects and characters. Marker-based systems are very accurate and yield high-quality data, but are on the other hand very expensive and require an extensive setup.
Markerless systems use computer vision algorithms and methods to track motion of objects and humans with either monocular camera views or multi perspective camera views. The main contribution of markerless motion capture systems is that the motion can be captured in a natural capture environment and do not require subjects to wear special equipment for tracking.

Mechanical systems directly track body joint angles. The sensors are attached to the human body with an skeletal-like structure. While the actor moves, the articulated mechanical parts move ion the same way, measuring the performers relative motion. Because the system has an skeletal-like structure, it interferes with the the actor's performance much more than other capture systems.

Magnetical systems utilize sensors placed on the body to measure the low-frequency magnetic field generated by a transmitter source. The sensors and source are cabled to an electronic control unit that correlates their reported locations within the field. By the relative intensity of the voltage, the range of motion can be measured and tracked. The markers are not occluded by nonmetallic objects but are susceptible to magnetic and electrical interference from metal objects in the environment like wiring, which affect the magnetic field, and electrical sources such as monitors, lights, cables and computers. As the system is cabled to the electronic control unit, mobility is restricted and does not allow wide motion performances.

Inertial systems are based on miniature inertial sensors, biomechanical models and sensor fusion algorithms. Most inertial sensors include at least an accelerometer, but can consist of additional components that make the data more stable and reliable.The data is transmitted wireless via Bluetooth to a computer, where the motion is recorded or viewed. Inertial sensors are small and of low weight and do not need any external cameras, emitters or markers. However, no positional data can be obtained from the data, but only

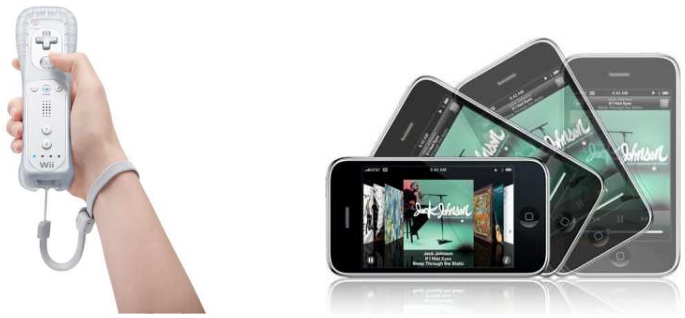

Figure 3.1. Inertial sensors are used in the Nintendo Wii Remote controller and the Apple iPhone[3].

orientational or dynamic data. Inertial systems that use additional magnetometers are sensitive to magnetic and electrical interferences in the environment, too.

Inertial sensors became quite popular during the last years. For example, inertial sensors measuring accelerations and the rate of turn are used in gaming and entertainment applications like the Nintendo Wii or the Apple iPhone [1, 28]. For example, the Nintendo Wii Remote controller with attached Wii MotionPlus includes a three dimensional accelerometer and a rate gyro. From this information, easy gestures can be recognized and displayed in the gaming application.

Table 3.1 gives an overview of the properties of all motion capture systems.

	Capture volume	System setup	Calibration	Actor mobility	Purchase costs	Data type	Data quality	Outdoor capturing	Marker occlusion	Wear comfort	Sensitivity to metal	Marker slippage
opt. marker-based	−	−−	−−	++	−−	++	++	−−	−−	+	++	++
opt. marker-less	−−	−	−−	++	−	++	+	+	−	++	++	++
mechanical	++	+	++	−−	+	++	++	++	++	−−	+	++
magnetic	−−	−	−	−	+	++	++	++	++	++	−−	−−
inertial	++	++	++	++	+	−	−−	++	++	+	−	++

Table 3.1. Overview of common motion capture systems and their advantages and disadvantages.

[3] *http://www.walyou.com/img/video−game−accessories−nintendo−wii−remote−control.jpg*, *http://i.areamobile.de/assets/handies/apple/iphone_3g/pressefotos_news/200806121426iphone_sensor_itunes_press.jpg*

(a) (b) (c) (d)

Figure 3.2. Camera setup for capturing trampoline motion with an optical motion capture system. (a) and (b) Cameras have to be mounted at the ceiling to avoid marker occlusions. (c) Capturing at broad daylight is not possible. (d) Trampoline capture session. The cameras that spot the motion from the top are clearly visible.

3.2 Capturing Trampoline Motions

Trampolining is a sport that cannot be performed as easily as many other common sports and often requires special circumstances and a special environment. Some particularities that have to be respected while capturing trampoline motion are:

Height of flight: Intermediate trampolinists can already reach heights of five meters, world elite trampolinists can even reach heights of up to ten meters. This means that the capture volume has to be large enough in vertical direction.

Motion complexity: Trampoline jumps are characterized by many rotational motions around the lateral and longitudinal axes. A motion capture system should be able to track the motions without marker occlusions and tracking errors.

Motion speed: Trampoline jumps are very fast. So the capture system should be able to capture the whole motion without latency and markers should maintain their initial position during the whole motion performance.

Mobility: As trampoline jumps are very complex, the used capture system has to provide excellent mobility for the athlete so that every motion can be performed with technical accuracy.

3.3 System Evaluation for Capturing Trampoline Motions

The specialty of trampolining in relation to other sports also influences the necessary motion capture conditions. Looking at Table 3.1, one can see that the special requirements of trampolining already exclude many capture systems. Respecting a good mobility for the actor, mechanical and magnetic motion capture systems will not be suitable. Because of the motion speed and complexity as well as the large vertical capture volume, markerless motion capture would not yield useful results, too.

In a previous work [8], trampoline motion has been recorded and captured with an optical motion capture system by Vicon [35] to use the motion data for computer animation. The

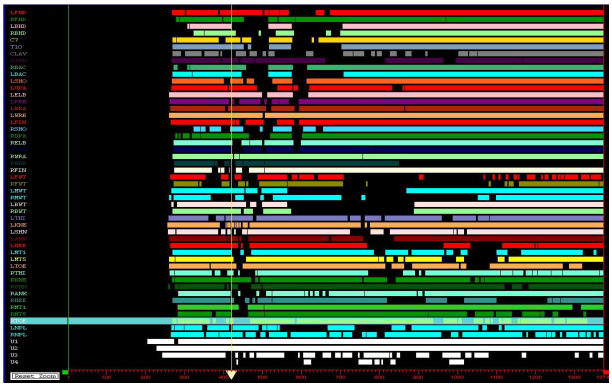

Figure 3.3. Continuity chart for all optical motion capture markers visualizes marker occlusions with optical motion capture systems. Markers are listed in vertical direction. Occlusions are indicated by gaps in the vertical timeline.

optical system offered good functionality and comfort for the actor, but the preprocessing steps, the capture process and the post-processing steps have been very inefficient. To be able to capture the whole motion and to cover the whole height of the jump with the capture volume, it is necessary to place cameras at the gymnasium's ceiling that can spot the motion from a top view. Because of this prearrangements in camera setup, spontaneous and short capture session seem to be impossible. Figure 3.2 shows images of the camera setup for optical motion capturing as conducted in [8]. Furthermore, the optical motion capture system used turned out to be very sensitive to influences like bright daylight. For example, outside capturing is only possible at night using artificial lights. Capturing at the inside in a room with huge windows can become a problem, as well. Optical motion cameras that faced directly into open windows and hereby daylight could not distinguish between the optical marker's reflection and the entering daylight.

One main problem of optical marker-based capture systems are marker occlusion that occur while capturing. Figure 3.3 shows a continuity chart from the ViconIQ software for optical motion capture data. In the continuity chart, in vertical direction all markers are listed (indicated by different colors) while the time in frames progresses horizontally. Marker occlusions are clearly visible as continuity gaps in the timeline. There are algorithms that estimate missing markers from the captured data to become invariant against the missing marker problem of optical motion capture data [3, 29]. But in general, it is not possible to use optical motion capture for all motions and capture scenarios and, which can be problematic, not under all circumstances and in all locations. Marker occlusions can sometimes be prevented by using either a small capture volume or using many capturing cameras, but often they cannot be avoided as they are caused by the way a motion is executed. For example, by bending the upper body to the ground markers at the upper body may be occluded by the legs. For trampolining, those unavoidable marker occlusions occur for example during a pike jump or a piked somersault, where marker from the front hip or the thorax will be occluded by the bend legs and arms.

Overall, one can say that optical motion capture systems as provided by Vicon or Motion Analysis [22] suffer from three main disadvantages that make it difficult to capture motion with a such a vertically huge capture volume: the setup of the cameras to reach a huge

capture volume, the influences by daylight and unavoidable marker occlusions. This leaves space for the last possible capture system that has been chosen for this thesis: inertial sensors. Inertial sensors are worn directly on the actor's body and data is transferred from a digital bus (worn on the actor's back) via Bluetooth to the capturing computer. As the sensors obtain all data by the mechanical elements that are embedded in the sensors, it is not necessary to capture the motion from outside. This means, no marker occlusions as for the optical motion capture system can occur. Additionally, the sensors are insensitive against daylight and light differences. Due to the fact that principally no cameras have to be used to capture the motion and the data transfer via Bluetooth to the computer, the process of calibration and camera setup reduces to a short-time setup. Spontaneous capturing becomes much easier and all above described problems that can occur while capturing trampoline moves do not need to be taken into account. The working principle of inertial sensors is explained in detail in the next section.

However, as the transmitting bus has been attached to the athlete's back, the motion performance has been slightly restricted, as well. Jumps that end in a back landing have not been captured for the experiments. Additionally, inertial sensors are responsive to ferromagnetic materials nearby. They will be described in Section 5.1.

3.4 Inertial Sensors

For this thesis, an inertial motion capture system provided by Xsens Dynamics Technologies [39] has been chosen to capture trampolining. Inertial sensors offer several advantages that enable an easy and short time set up of the system as well as easy data acquisition. We have seen that optical motion capture systems undergo a lot of disadvantages while capturing trampoline motion and that those disadvantages can be avoided with an inertial capture system.

The XSens inertial motion capture system consists of inertial MTx motion trackers. The motion trackers are *inertial measurement units* (IMUs) that do not deliver positional information as optical motion capture systems, but inertial data that can be obtained by mechanical components. This data is abstract and does not give information about a motion's morphologic properties as it can be visualized with optical motion capture systems.

3.4.1 Inertial capture setup

Only ten inertial MTx motion trackers s_1,. . .,s_{10} have been used to capture the trampoline motion.

The sensors have been placed at the lower and upper extremities, where each extremity has been equipped with two sensors at the outer and inner part of the extremity resulting in four sensors at the lower extremity (left s_2 and s_7, right s_3 and s_8) and four at the upper extremity (left s_4 and s_9, right s_5 and s_{10}), at the lower spine (s_1) and next to the actor's clavicula (s_6). Figure 3.4(a) shows the location of the sensors subject to an actor's body. The sensors have been aligned to the growth direction of the bones of the actor so

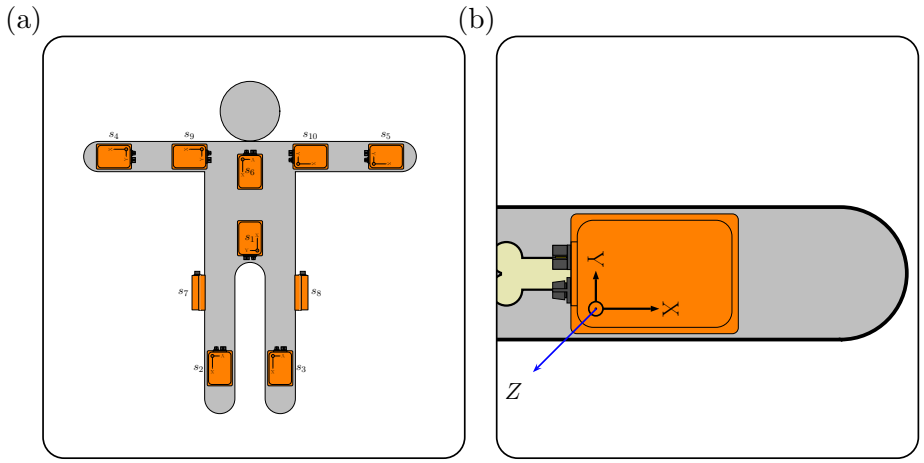

Figure 3.4. (a) Locations of the ten motion sensors attached to the human body. (b) Inertial sensors are attached in direction of the body's limb and can measure the limb's orientation.

that the orientation of the equipped limb segments could have been calculated as shown in Figure 3.4(b).

3.4.2 Physical Background

As inertial sensors make use of physical principles and mechanical engineering, the most important terms that occur while working with inertial sensors will shortly be recapulated.

Inertial sensors usually are made up of at least one *accelerometer* detecting the current rate of acceleration, and a *rate gyro* detecting changes in rotational attributes. The term IMU is usually referred to a box containing three accelerometers and three gyroscopes for the local x-, y- and z-axis.

The accelerometer detects the local *acceleration* $\boldsymbol{a}$ of the sensor with respect to the axis the accelerometer is aligned to in the local coordinate system. Using a test mass at one end of a cantilever beam, the force that acts on this test mass can be determined. The acceleration is then defined as the ratio between acting force and mass of the test element (Newton's second law). Besides, the acceleration designates the change of velocity over time or the second derivative of the position over time. With the sensor in rest or a constant motion of the sensor with no change in velocity, the acceleration caused by gravity $\boldsymbol{g}$ and it's direction can be measured. For the IMU, acceleration $\boldsymbol{a}$ is usually measured in three orthogonal planes as th x-, y- and z-axis and can be thought of a superposition consisting of the gravity $\boldsymbol{g}$ and the actual acceleration $\boldsymbol{m}$ of the motion in direction of each axis.

To determine rotation and angular velocity, a rate gyro is used. Rotation may occur without a change in acceleration, think of rotating the sensor in the x-y plane parallel to the earths surface. To determine the orientation of the sensor or the so-called *rate of turn*, a rate gyro uses a vibrating structure measuring displacement caused by Coriolis force that can be explained as followed: an object on a rotating disc is subject to tangential acceleration due to the rotation. This acceleration is proportional to the object's distance from the center of the disk, hence the object experiences a greater acceleration near the outside than near the center. A rotational motion about the sensor's longitudinal axis

produces this Coriolis acceleration proportional to the rate of rotation which can be measured and calculated to an externally applied torque and the angular velocity. In usual *Micro-Electro-Mechanical Systems* (MEMS), the rate gyro consists of three small vibrating tuning forks as vibrating structure driven with different external signals for the three axes. The tuning forks normal vibration mode is in one plane and the Coriolis induced displacements are in another orthogonal plane.

Table 3.2 lists the physical quantities for translational and rotational motion with their common notation. The rotational quantities cohere in a similar way than the translational quantities.

Translational	Rotational
Translation	Angle
Velocity—derivative of translation	Angular velocity—derivative of angle
Acceleration—derivative of velocity	Angular acceleration—derivative of angular velocity
Mass	Moment of inertia
Linear momentum—overall system state	Angular momentum—overall system state
Force—change of linear momentum	Torque—change of angular momentum

Table 3.2. Overview of physical quantities for translational and rotational movement.

3.4.3 Sensor Properties

The XSens MTx motion tracker that is used for this thesis consists of six individual MEMS sensors which are three rate gyroscopes and three accelerometers and of three additional magnetometers.

As previously described, the three accelerometers and the rate gyros measure the local acceleration and rate of turn for all three axes and the direction of gravity with the sensor in rest. Using the fact that the sensor only measures gravity in case that it is in rest or subject to an unaccelerated motion, 2 degree of freedom (DOF) orientation of the sensor with respect to the canonical direction of gravity can already be calculated with the accelerometer [18]. For devices such as the Apple iPhone, this information is sufficient enough. To determine the orientation of the sensors around the global vertical axis (which means to determine the direction of the magnetic north pole and the direction each inertial sensor is pointing to), additional information from the magnetometers will be used. Those magnetic field sensors can measure the direction of the earth's magnetic field similar to a compass and enable to calculate a full three DOF of the sensors with respect to a global coordinate system defined by the North and the vertical axis defined by the measurements of the inertial sensors. The third axis is defined by calculating the cross product of the two previous axes yielding in the wanted global coordinate system. Figure 3.5 shows the working principle for the sensor in rest or in unaccelerated motion.

To predict the orientation of the sensor during movement phases, information about the angular velocity out of the rate gyro will be taken into calculation. A more robust estimation of the sensor's orientation will be obtained combining the prediction from the rate

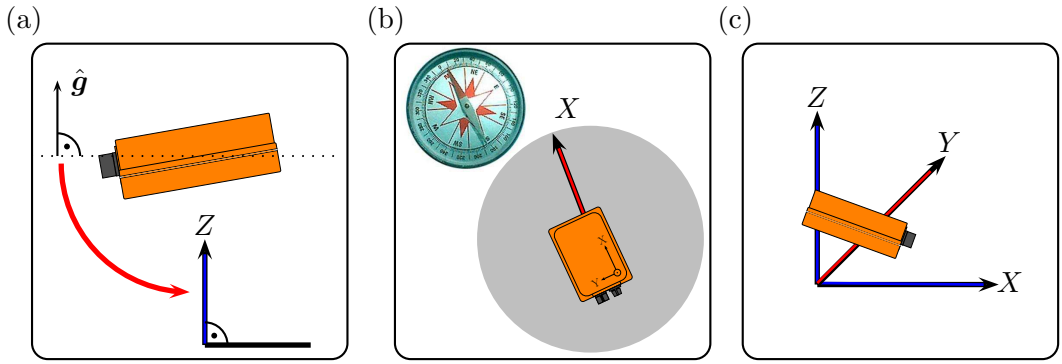

Figure 3.5. Working principle for the MTx motion tracker in rest. A global coordinate system is set up to determine global orientations. (a) By measured acceleration with the sensor in rest, 2DOF can already be determined. If no acceleration occurs, only the gravity $\hat{\boldsymbol{g}}$ will be measured that determines the vertical Z-axis of the global coordinate system. (b) The measurement of the magnetic north pole is projected onto the plane perpendicular to Z (gray) and defines the global X-axis. (c) For full 3 DOF, the Y-axis can be defined from $X \times Z$.

gyros and the measurement from the accelerometer. Stable information about the sensor's orientation with respect to a direction defined by the gravity vector $\hat{\boldsymbol{g}}$ can be calculated using a Kalman filter, see [20]. This stable orientation information can then be used to describe a motion in a sufficiently precise way, see Chapter 4.

All information, that means the sensor's three dimensional acceleration data, the sensor's rate of turn data, the sensor's magnetic data and the sensor's estimated orientation data is made available from the captured data and can be used for further motion analysis. Local orientation, rate of turn and acceleration of each limb segment of interest (as long as equipped with a sensor) can be tracked.

Working with inertial data meansto work with abstract data that tends to be sparse. The data can only deliver information about local accelerations and rotations within the sensor's coordinate system. The information can be transformed into orientation data, but cannot deliver reliable positional information, so that captured motion cannot be characterized by positional descriptions.

Positional data could be obtained by double integrating the measured accelerations, but as inertial data is prone to noise and due to the integration, the computed position will not be equal to the exact position. This means, inertial sensors cannot deliver reliable positional data unaffected by drift, noise and inaccuracy. In navigation, one has to handle a similar problem to estimate one's current position by known or estimated speeds over elapsed time and course out of a previous position. Here, this process of position estimating is known as dead reckoning [36].

Because of this unreliability, only the stable acceleration, gyroscopic and orientation data will be used and processed in this thesis. The three-dimensional orientation information of the ten used MTx sensors has been used for classification, while the acceleration data will be used for segmentation.

Chapter 4

Motion Classification

The classification of unknown motion data streams according to given motion categories requires three methodologies: motion performance (that have been predefined trampoline jumps in our scenario), motion capturing (in our case, by inertial sensors) and motion analysis. An extensive description on trampolining and inertial motion capturing has been given in the latter chapters. The last main step in the classification pipeline then is to analyze the sensor output motion data, to automatically segment an unknown trampoline motion sequence into its individual jumps, and to classify these jumps according to given motion categories.

The idea for motion classification is to locally compare the feature sequence of the unknown trampoline motion with previously computed class representations in form of motion templates. Comparing a motion sequence with a motion template, we want to get information on how similar the motion is to the template. Here, it is necessary to find a suitable definition of similarity. To this end, one has to decide how to deal with different kinds of variation in the data, as for example temporal or actor-specific variations. The unknown motion sequence is then locally classified by the class template that best explains the corresponding segment. In this chapter, the general and principal idea of how to use inertial sensor data for classification is given. Further information on methods and algorithms will be given in the following sections that explain the experiments conducted for this thesis.

Analyzing the inertial output may be difficult as the data is noisy and abstract. Understanding the essential characteristics of a motion class, we can design various feature representations that capture the characteristics of a motion category. In Section 4.1, we explain how to use the inertial output data to transform it into feature representations. In Section 4.2, we explain the classification process and introduce methods that define similarity. In Section 4.3, it is described how to build the motion templates for every motion category.

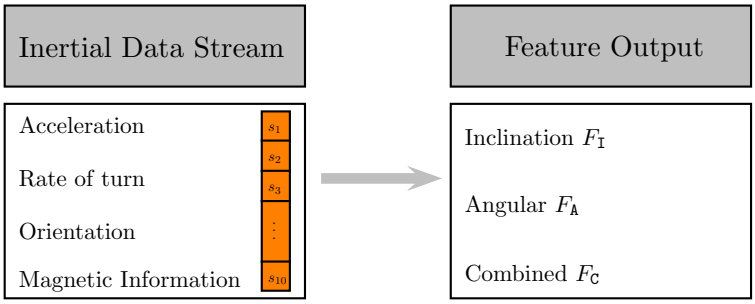

Figure 4.1. A feature representation can be defined by a feature function F that maps the inertial sensor data I on a feature output $O = \mathbb{R}^f$ for some $f \geq 1$.

4.1 Feature Representations

To use the abstract inertial output data, the motion information has to be transformed into semantically meaningful and suitable feature representations. In this context, we have to respect both motion semantics and motion performance. To maintain all motion semantics, feature representations should grasp the essential characteristics of every motion class and enable to distinguish all motion classes from each other. In this regard, we often use the term *discriminative power* that gives information on how unique a motion category can be represented by the chosen feature representation and how easily it can be separated from the other moves. The better the discriminative power of a feature set, the better the motion classes can be distinguished from each other. However, with respect to motion performance, we also expect the feature representations to be less sensitive to performance variations and changes in recording setup and calibration. Actor-dependent performance variations might be interesting for other applications, but as they are of no interest for our trampoline classification scenario, they are left unconsidered.

4.1.1 General Feature Notations

Each trampoline jump follows certain technical aspects and rules that are determined by biomechanical parameters, see Section 2.5. Those rules constitute the essential characteristics of a motion class and hence assign a specific motion category such as a pike jump or a somersault to every performed jump. To display those characteristics in the feature representations, we have to transform the inertial data into suitable information, see Figure 4.1.

For the following, we need the definition of a feature representation. For this end, we define a *feature function* F that maps the inertial sensor data I on a feature output $O = \mathbb{R}^f$ for some $f \geq 1$:

$$F : I \to O. \tag{4.1}$$

One element $v \in O$ is then referred to as *feature vector*, one property of the inertial data within the feature function as *feature* like for example the acceleration of the left arm's sensor. For every feature function, the output O is also referred to as *feature space* $\mathcal{F} = \mathbb{R}^f$.

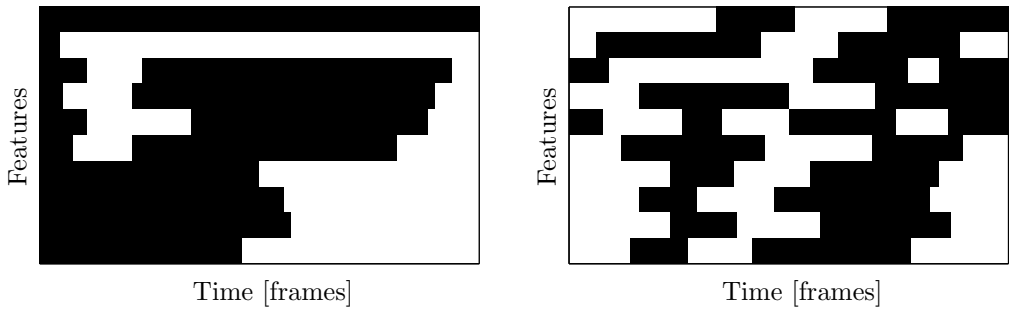

Figure 4.2. Binary feature matrices for two different motion classes evaluated in the same way. Semantically differences are clearly visible. Every row represents one feature vector while time runs along the horizontal axis.

With a feature function, the sensor input stream can then be transformed into a *feature sequence* as for example $X = (x_1, x_2, \ldots, x_N)$ with $x_n \in \mathcal{F}$ for $n \in [1 : N] := \{1, 2, \ldots, N\}$. N denotes the length of the feature sequence.

To obtain direct semantic information on specific motion classes and their characteristics, we display the feature sequence as *feature matrix* M. For a motion data input stream D of length N, the feature matrix is defined as $M \in \mathbb{R}^{f \times N}$. Then, the nth column of M, denoted by $M(n)$, equals the feature vector $F(D(n))$, $n \in [1 : N]$.

Two example feature matrices can be found in Figure 4.2. For demonstration, we assume that every feature only contains binary information, such as whether the sensor's x-axis is pointing downwards. In the feature matrices, every row denotes a feature (and here, indicates the ten sensors used), whereas the temporal information is contained in the feature vectors (which means framewise in the columns). Differences are clearly visible in the feature matrices indicating semantical differences and therefore feature representations of two different motion classes.

4.1.2 Feature Description

Previous experiments showed the results that feature representations based on pure acceleration data do not yield powerful and robust discriminations between different motion classes. So for the following experiments, the feature functions will be mappings of the sensor input containing the orientations of the ten sensors relative to the common global coordinate system. In order to express the orientation of the inertial unit with respect to the global coordinate system we use rotations expressed as unit quaternions (see [33]). Each such quaternion defines a 3D rotation $\mathbb{R}^3 \to \mathbb{R}^3$, which we also refer to as q. Let $\mathsf{q}[\boldsymbol{x}]$ denote the rotated vector for a vector $\boldsymbol{x} \in \mathbb{R}^3$. The orientations of the ten sensors are referred to as as $I = (\mathsf{q}_1, \ldots, \mathsf{q}_{10})$ with q_s, $s \in [1 : 10]$.

In the following, we will introduce features of two different types: inclination and angular-based features.

Inclination features: The inclination features are very closely related to the sensor's orientation in the global coordinate system. For every sensor, the *pitch* ϕ_s is determined. ϕ_s describes the inclination of the sensor's X-axis and therefore the bone's inclination in

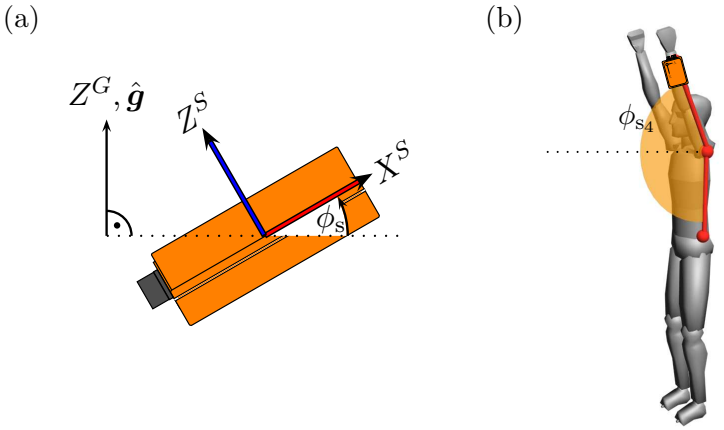

Figure 4.3. (a) Pitch ϕ_s of a sensor with respect to the plane defined by $\hat{\boldsymbol{g}}$. (b) Inclination angle $F_{s_4} := \phi_{s_4}$

relation to the plane defined by the direction of the up direction $\hat{\boldsymbol{g}}$, see Figure 4.3.

The global up direction $\hat{\boldsymbol{g}}$, represented by the vector $(0,0,1)^T$ is transformed to the sensor local coordinate system using $\overline{\mathsf{q}_s}[\cdot]$. Using this, the pitch is defined as the angle between a sensor's $\hat{\boldsymbol{g}}_{\boldsymbol{s}}$ and the local X-axis:

$$\hat{\boldsymbol{g}}_s = \overline{\mathsf{q}_s}\left[(0,0,1)^T\right], \tag{4.2}$$

$$\phi_s = \frac{\pi}{2} - \arccos\left\langle \hat{\boldsymbol{g}}_s, (1,0,0)^T \right\rangle, \tag{4.3}$$

For the ten sensors, we then obtain the inclination feature functions $F_1 := \phi_{s_1} \ldots F_{10} := \phi_{s_{10}}$. Because of the definition of ϕ_s that determines the amount of inclination of the sensor's X-axis in relation to the up direction, those features are called *inclination features*.

Angular-based features: Based on the sensors' orientation within the global coordinate system, the angle between two sensors respectively between the two bones the sensors are attached to can be calculated. This angle represents the difference of orientation between the two bones. For example, an angle of zero between the sensors s_5 and s_{10} means that both sensors are pointing towards the same direction and consequently, that the right arm has to be stretched.

Information whether the extremities are straight or bend will be obtained by the angle θ_{st} determined by orientation differences along the vertical direction, information whether the legs are spread will be obtained by the angle ϑ_{st} determined by orientation differences along the horizontal axis between the limbs of the upper legs. Here, θ_{st} represents the relative position of elbows and knees to the sensor's bones while ϑ_{st} represents the angle between both legs, see Figure 4.4.

The angles θ_{st} and ϑ_{st} between two different sensors can be computed in a quite easy way as all sensors used for this calculation were attached at the actor's body facing in the same direction. To obtain the corresponding angle between both orientation vectors, one can use the scalar product. Additionally, for the angle at the outer extremity we define an absolutely straight extremity as 180° or π and the biologically not possible total bend as zero. For the angle between the legs, we define a total spread of the legs as 180° or π and

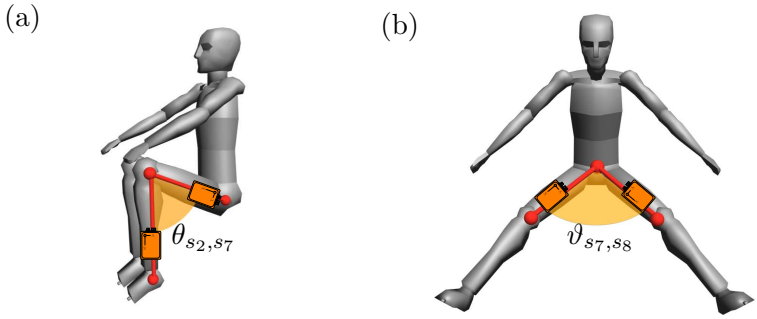

Figure 4.4. (a) The angle $F_{13} := \theta_{s_2,s_7}$ between two bones of the same extremity, respectively the left leg. (b) The angle $F_{15} := \vartheta_{s_7,s_8}$ between the bones of different extremities, respectively the upper legs.

no spread as zero, see [15]. As before, we assume q_s,q_t, $s, t \in [1:10]$ to be the orientation of the sensors. This results in following formulas

$$\theta_{st} = \pi - \arccos\left\langle \mathsf{q}_\mathrm{s}, \mathsf{q}_\mathrm{t} \right\rangle, \tag{4.4}$$

$$\vartheta_{st} = \arccos\left\langle \mathsf{q}_\mathrm{s}, \mathsf{q}_\mathrm{t} \right\rangle. \tag{4.5}$$

Those angular information results in five additional feature functions, the feature functions $F_{11} \dots F_{14}$ for the angle between bones of the same extremity and F_15 for the angle between bones of different extremities. Because of the use of angles between two sensors or respectively bones, the features are called *angular-based features*.

Table 4.1 lists a description of all used features, their type and their ID as they have been used to compose different features.

ID	Type	Description
F_1	inclination	Inclination of lower spine ϕ_{s_1}
F_2	inclination	Inclination of left lower leg ϕ_{s_2}
F_3	inclination	Inclination of right lower leg ϕ_{s_3}
F_4	inclination	Inclination of left lower arm ϕ_{s_4}
F_5	inclination	Inclination of right lower arm ϕ_{s_5}
F_6	inclination	Inclination of belly neck ϕ_{s_6}
F_7	inclination	Inclination of left upper leg ϕ_{s_7}
F_8	inclination	Inclination of right upper leg ϕ_{s_8}
F_9	inclination	Inclination of left upper arm ϕ_{s_9}
F_{10}	inclination	Inclination of right upper arm $\phi_{\mathrm{s}_{10}}$
F_{11}	angular1	Angle between left lower and upper arm θ_{s_4,s_9}
F_{12}	angular1	Angle between right lower and upper arm $\theta_{s_5,s_{10}}$
F_{13}	angular1	Angle between left lower and upper leg θ_{s_2,s_7}
F_{14}	angular1	Angle between right lower and upper leg θ_{s_3,s_8}
F_{15}	angular2	Angle between left upper leg and right upper leg ϑ_{s_7,s_8}

Table 4.1. Description of the used feature functions with feature ID and feature type.

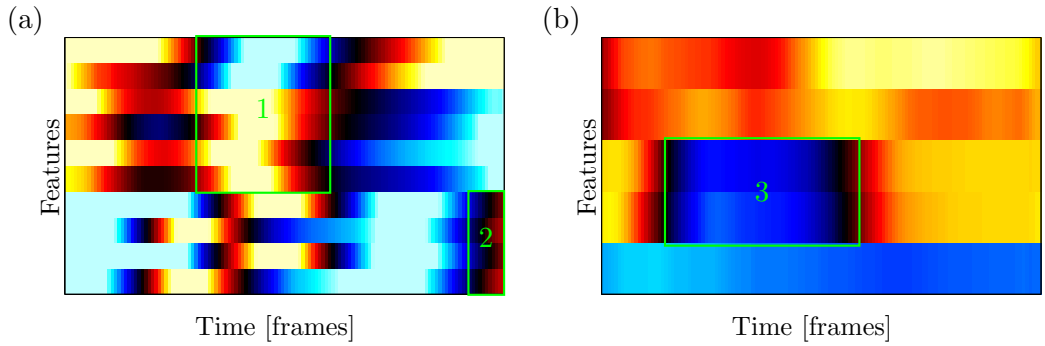

Figure 4.5. Feature matrices for the same database move represented by different feature types. Semantically meaningful regions are marked. (a) Feature matrix for the inclination feature function $F_{\texttt{I10}}$. (b) Feature matrix for the angular feature function $F_{\texttt{A5}}$.

4.1.3 Composed Feature Sets

From the different feature types, we built several feature sets that can either be pure inclination feature sets, pure angular feature sets or combined feature sets. All feature sets and their compositions are displayed in Table 4.2.

ID	Contained feature functions
$F_{\texttt{I5}}$	$(F_1, \ldots, F_5)^T$
$F_{\texttt{I10}}$	$(F_1, \ldots, F_{10})^T$
$F_{\texttt{A5}}$	$(F_{11}, \ldots, F_{15})^T$
$F_{\texttt{A3}}$	$(F_{13}, \ldots, F_{15})^T$
$F_{\texttt{I5A5}}$	$(F_{\texttt{I5}}^T, F_{\texttt{A5}}^T)^T$
$F_{\texttt{I10A5}}$	$(F_{\texttt{I10}}^T, F_{\texttt{A5}}^T)^T$
$F_{\texttt{I5A3}}$	$(F_{\texttt{I5}}^T, F_{\texttt{A3}}^T)^T$
$F_{\texttt{I10A3}}$	$(F_{\texttt{I10}}^T, F_{\texttt{A3}}^T)^T$

Table 4.2. Description of the composed feature sets listing their contained feature functions.

To combine the different feature types and to make the feature sets comparable among each other, we have to normalize every feature vector within a feature set. For this, we map every feature vector to the range of $[-1, 1]$. In the following, the normalized feature functions will be used.

In Figure 4.5, two sample feature matrices of inclination and angular feature type for the same database move are visualized. Figure 4.5(a) shows the feature matrix computed using the feature set $F_{\texttt{I10}}$, while Figure 4.5(b) shows the feature matrix of the same jump computed using the feature set $F_{\texttt{A5}}$. One can see that the feature set $F_{\texttt{I10}}$ consists of ten dimensional feature vectors whereas $F_{\texttt{A5}}$ only consists of five dimensional feature vectors. Semantically meaningful regions have been marked in the feature matrices. In region (1), we can see that the pitch of all arms' sensors and of the root reach the absolute maximum value. This indicates that the sensor's X-axis is pointing downwards and we can assume that the jump contains rotational motion around the lateral axis as for a somersault motion class. Region (2) indicates that the landing will be no feet landing as the pitch is in the range of 0. In fact, the represented motion class is the motion class `BWS` Somersault Backwards To Seat Drop. In case of the angular feature set $F_{\texttt{A5}}$, we can mainly see from

region (3) that the jump has been performed in a tucked shape.

4.2 Classification by Similarity Measures

In this thesis, we will locally compare the unknown feature sequence with all given class motion templates. Herewith, we determine it's similarity to the already known motion templates. Then, the unknown feature sequence can be classified by the class template that best explains the corresponding segment. From a technical point of view, the classification problem can be modeled as follows. Assume that we are given a finite set Λ referred to as *label set*. An element $\lambda \in \Lambda$ is referred to as *label*. The label set refers to the different motion classes to be considered in the classification, i. e.,

$$\Lambda = \{\texttt{BAR}, \texttt{FRF}, \texttt{HTW}, \ldots\}.$$

Now, let $\mathcal{D} := \{D_1, D_2, \ldots, D_K\}$ be a motion database annotated with respect to a given label set Λ. In other words, for each document D_k, we are given a label $\lambda_{D_k} \in \Lambda$. Then two documents are said to belong to the same motion class, if they have the same label.

To find similarities between motions, it is crucial to define the term of similarity. We will then introduce distance measures for comparison, that are variants of DTW.

4.2.1 Motion Similarity

Motions can undergo various variations that can make it difficult to identify motions that belong to the same motion class as similar. Especially, if performed by different actors oder under different capture conditions, semantically similar motions do not need to be numerical similar [23]. Similarity measures for comparison of two motions should be invariant against those differences and variations. Variations that can occur during motion performance are:

Spatial variations: those variations can occur while several motion performances have been captured under varying capture circumstances or under different system setup and calibration. For example, differences in the trampoline's position and alignment should not influence the motion similarity measure.

Temporal variations: those variations comprise all differences in timing and the overall speed a motion is performed with. For example, the time of a jump depending on the flight's height should not influence the similarity measure.

Style variations: those variations mean differences that occur within different motion performances. Style variations are mostly actor dependent, but can also occur within the motions of one actor. For example, a technically excellent somersault performance should still be similar to a technically less accurate performance.

In the following, we assume motions to be semantically similar if they represent the same action including spatial and temporal variations.

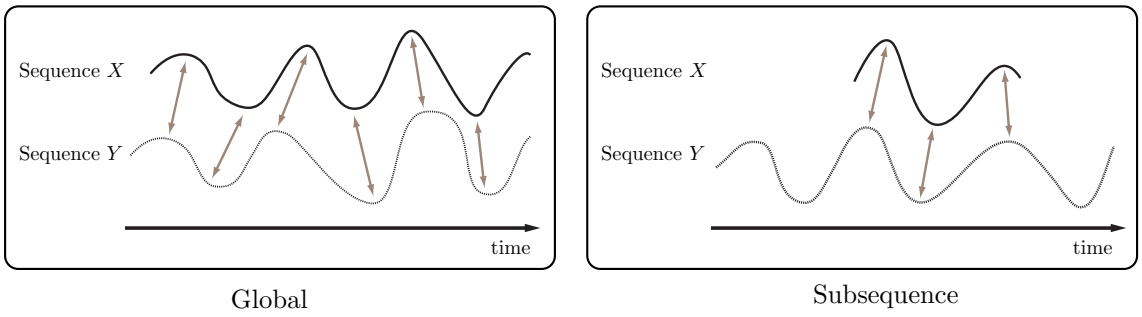

Figure 4.6. DTW variants used in this thesis. (a) Global DTW aligning two time-dependent sequences X and Y. (b) Subsequence DTW aligning sequence X with a subsequence of Y.

By suitable feature representations, spatial variations should already be taken out of account for the similarity measure. In terms of the style variations, the feature representations have also been requested to be as invariant as possible. However, style variations still occur. For the trampoline motions, the most style variations are mainly expected to occur at the upper extremities as individual technical executions, but can also be found at the lower extremities and the limb. With the similarity measure in the local comparison, we cannot eliminate those variations. In the further experiments, we will discuss how to deal with style variations. Here, we focus on eliminating temporal variations.

4.2.2 Classification strategies

For this thesis, we used two different scenarios for classification, where the similarity measure is based on DTW: document-based comparison with global DTW distance measures and subsequence-based comparison with subsequence DTW and a distance function. Originally, DTW has been used in automatic speech recognition and data mining to cope with time deformations of time-dependent data, but can meanwhile also be used to deal with temporal variations in motion retrieval and classification tasks [31]. While global DTW methods compares whole feature sequences (of similar length), subsequence DTW methods compares feature sequences of different length, see Figure 4.6.

Document-based: In a document-based classification scenario, the similarity between two time-dependent motions respectively two feature sequences $X = (x_1, x_2, \ldots, x_N)$ of length N and $Y = (y_1, y_2, \ldots, y_M)$ of length M has to be determined. The goal is to find the optimal alignment between the given sequences. We associate X to a feature matrix $M_X := [x_1 x_2 \ldots x_n] \in \mathbb{R}^{f \times N}$ and Y to a feature matrix $M_Y := [y_1 y_2 \ldots y_m] \in \mathbb{R}^{f \times M}$. As described in Section 4.1, the columns of M_X represent the feature vectors of X. For the given sequences, the feature space $\mathcal{F}$ is determined by $x_n, y_m \in \mathcal{F}$ for $n \in [1 : N]$ and $m \in [1 : M]$. To quantify the differences between two feature vectors $x, y \in \mathcal{F}$, we need a *local cost measure* which is also referred to as *local distance measure* and is defined as a function

$$c : \mathcal{F} \times \mathcal{F} \to \mathbb{R}_{\geq 0}. \tag{4.6}$$

If the cost $c(x, y)$ is small, x and y are called similar to each other, if the cost $c(x, y)$ is large, x and y are different to each other. With computing the local cost measure for each pair of elements of X and Y, one then obtains a *cost matrix* $C \in \mathbb{R}^{N \times M}$ with $C(n, m) :=$

$c(x_n, y_m)$. Depending on the local cost measure used, the resulting cost matrices will differ. Different local distance measures for motion data like the quaternion-based pose distance can be found in [23] and will not be discussed further in detail. For this thesis, only the Euclidean L^2 norm has been used that describes the length of a vector or ordinary distance from the origin to a point x. With the L^2 norm as distance measure, the cost is defined by

$$c(x, y) := \|x - y\| . \tag{4.7}$$

Using the cost matrix, we want to find an alignment between X and Y having minimal overall cost.Such an alignment is represented by a *warping path* $p_\ell = (n_\ell, m_\ell) \in [1 : N] \times [1 : M]$ with $\ell \in [1 : L]$ where L is the length of the warping path. Furthermore, the warping path has to satisfy three conditions — the boundary condition, the monotonicity condition and the stepsize condition. The boundary condition forces the warping path to start at position $p_1 = (1, 1)$ and to end at position $p_L = (N, M)$ (which means the entire sequences X and Y have to be aligned). The monotonicity condition assures the warping path to respect issues of timing and sequential time procession. The step size condition ensures as kind of continuity condition that no element will be omitted and that there are no replications in the alignment (the classical step sizes are (0,1), (1,0) and (1,1)). Monotonicity condition and step size condition are closely related to each other. For more information about warping paths and their conditions, see [23].

The cost of a warping path p between X and Y with respect to the local cost measure c is defined as

$$c_p(X, Y) := \sum_{\ell=1}^{L} c(x_{n_\ell}, y_{n_\ell}). \tag{4.8}$$

The optimal warping path is then defined as warping path with minimal overall cost and can be used as quantity to measure similarity of two feature sequences under the given cost measure c.

The optimal warping path can be computed using dynamic programming techniques. To this end we define the accumulated cost matrix $D \in \mathbb{R}^{N \times M}$. D hast to satisfy the following requirements:

$$D(n, 1) = \sum_{k=1}^{n} c(x_k, y_1) \tag{4.9}$$

for $n \in [1 : N]$

$$D(1, m) = \sum_{k=1}^{m} c(x_1, y_k) \tag{4.10}$$

for $n \in [1 : M]$ and

$$D(n, m) = \min \{D(n - 1, m - 1), D(n - 1, m), D(n, m - 1)\} + C(n, m) \tag{4.11}$$

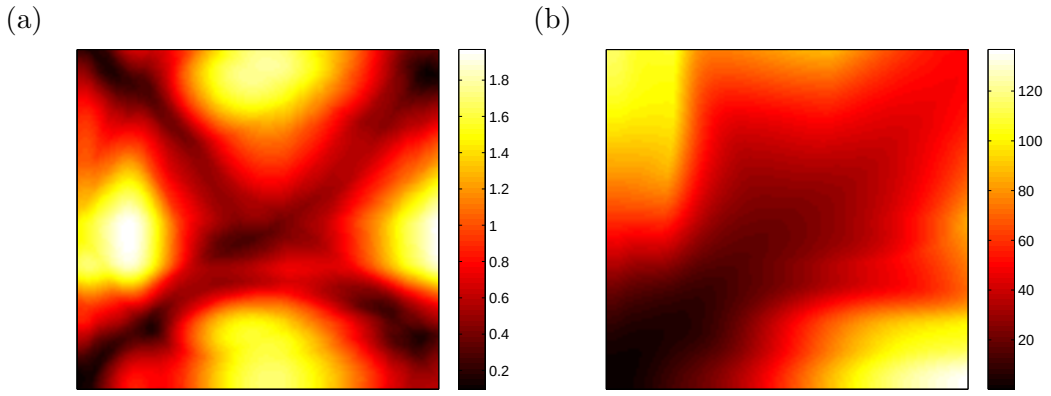

Figure 4.7. Cost matrix (a) and accumulated cost matrix (b) for feature sequences of the same motion class. High costs are represented by white color, low cost by dark color.

for $1 < n \leq N$ and $1 < m \leq M$.

The accumulated cost $D(N, M)$ also referred to as DTW distance $DTW(X, Y)$ represents the overall cost for the similarity measure between two sequences X and Y (and herewith the cost of the optimal alignment of X and Y). In the following, we use $DTW(X, Y)$ to measure the similarity between two feature sequences.

Figure 4.7 shows one example cost matrix and accumulated cost matrix comparing two trampoline jumps of the same motion class. Similar parts with low cost are characterized by dark color, whereas high costs representing huge differences in the compared feature columns are indicated by light color. In the cost matrix C, the ending of one motion appears similar to the beginning of the other motion and vice versa. This is not surprising, as for trampoline moves that are landing and starting on both feet, the beginning and ending phase of each move should be similar which means moving in a straight and upright position with the feet closed. For the accumulated cost matrix D, the optimal warping path goes almost straight from the bottom left to the top right implying that the duration of both jumps and the motion performance of both jumps is similar. Medium cost at the top right end of the accumulated matrix where the warping path ends, on the other hand, indicates that both motions are not absolutely similar accounting for actor-individual motion performances.

Subsequence-based: In a subsequence-based classification scenario, similarities between a feature sequence X with a much longer sequence Y like for example a trampoline routine have to be found. Here, the goal is to identify the subsequence within Y which is most similar to X.

Again, we use the feature sequences X and Y and the L^2 distance as defined in the previous paragraph as cost measure to calculate the cost matrix C. The main differences to the global DTW are that boundary condition and stepsize condition will be modified. As we compare a short query motion to a longer motion sequence, the boundary condition as defined for the global DTW does no longer hold. The stepsize for the Subsequence DTW will be modified to the step sizes (2,1), (1,2) and (1,1) accounting for the different lengths of the two sequences and to avoid degenerations in the alignment.

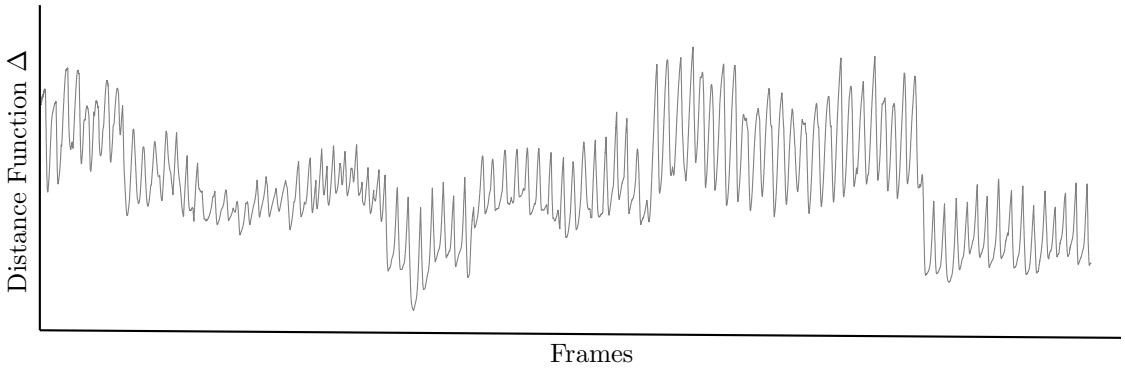

Figure 4.8. Sample distance function between motion sequence X and a longer motion sequence Y. X is contained in Y at the position of zero cost. Passages around the position of zero cost are indicated by low cost, as well and comprise semantically similar motions.

The most similar passages within Y are then retrieved using the following assumption: First, we want to find a subsequence $Y(a* : b*) := (y_{a*}, y_{a*+1}, \ldots, y_{b*})$ with $1 \leq a* \leq b* \leq M$ that minimizes the DTW distance to X over all possible subsequences of Y for a fixed cost function c. The optimal warping path can then be computed in a similar way than for the global DTW to determine the accumulated cost matrix D. All subsequences of Y that are close to X with respect to the DTW distance is then defined by a distance function

$$\Delta : [1 : M] \rightarrow \mathbb{R}. \tag{4.12}$$

The distance function assigns each index $b \in [1 : M]$ the minimal distance $\Delta(b)$ between X and a subsequence $Y(a : b)$ of Y ending in y_b. The DTW-minimizing a can be determined by computing the optimal warping path again and is defined as $DTW\,(X, Y\,(a : b))$.

The distance function gives a natural overview over the data within the database and ranks the retrieved matches according to the cost represented by the Δ-values. This means, the best match between X and Y can be determined by the index minimizing Δ. Figure 4.8 shows a sample distance function. Document endings within Y can be identified easily by positions of lower cost. Furthermore, similar cost levels indicate similarities between the passage of Y and X.

4.3 Motion Templates

To assign a motion category to every trampoline jump, we need references that contain information about the motion characteristics of every motion class. For this, we learn a class representation for each category, that we call *Motion Template* (MT) according to [25], while a finite number of motion categories are known in our scenario. Each motion category is represented by a class of example motions performed by different athletes, so that the motion template can be computed as average feature representation by all class example motions. By the averaging process, those feature matrices should still represent the most significant information of each motion class but discard regions that are variant

among the example motions. This means, that variations from the example motions will not influence the standardized motion template as classifier.

4.3.1 Computation

In general, we only compute the average over all learning motions that comprise the same motion class to obtain the desired class representations for every motion class. However, as the original motions are varying in length and temporal structure, the feature matrices differ, as well, so that we cannot just compute the average over all feature matrices, see Figure 4.9. So the class MTs have been computed with the following algorithm that uses temporal alignment.

First, one feature matrix is chosen as reference. With DTW, we can then compute the optimal alignment of the remaining motions and determine the optimal warping path for each alignment resulting in eight feature matrices of the same length than the reference motion. The values of all feature matrices of same length can then be averaged to one combined input representation of all input motions. Additionally, the standard deviation for each matrix entry can be computed to identify those regions within the feature matrix that undergo the most variation. Nevertheless, the averaged matrix does not represent the real average of all motions yet as all remaining motions are aligned to the one reference motion.

As the current feature matrix is biased by the influence of the reference motion, the averaging step is repeated using all other motions as reference motion. As a result, we get eight averaged matrices of different length. Each feature matrix has it's individual length then, whereas the semantic information is already very similar and the essential information is maintained.

To improve the result and the average value of the MT, we iterate the averaging process for each averaged feature matrix until the standard deviation undergoes a certain threshold or after a predefined number of iterations. For the trampoline experiments, already ten iterations turned out to deliver already similar results for each averaged matrix. To obtain a MT that is almost average in all matrix entries, 20 iteration steps have been used and, as all of the eight matrices contain very similar matrix entries, one of the averaged matrices then be chosen as MT. The small number of iterations necessary to already get a low standard deviation for each matrix entry can be explained by the fixed length of each trampoline move in the range between 120 and 140 frames (captured at 100 Hz). The length of the moves could only differ to a large amount by differences in the jump height of several decimeters and meters, but as all actors have been on a similar intermediate level of skills, each move comprises of similar length. As the skills of all trampolinists that performed for the used database have been on a similar level, the height only differs to a small amount and each trampoline move lasts approximately the same, so that the temporal differences between each feature matrix have been already quite small in the beginning of the iteration process. Further details on the MT computation process can be found in [25].

The differences between learned motions and the iterated MT is illustrated in Figure 4.9. The essence of the class could been grasped out of all learning motions, whereas variations

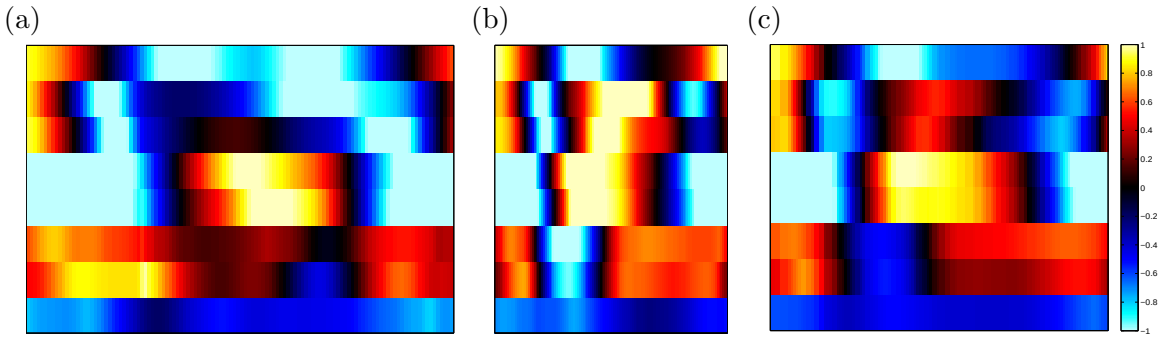

Figure 4.9. Variant feature matrices (a),(b) of a barani jump and the iterated MT (c). The semantic information of the motion class is maintained while individual variations and both local and global temporal variations are averaged.

that do not occur in all feature matrices have been weakened.

Chapter 5

Database Description

As all of the following experiments have been conducted on a self-built trampoline database, this chapter describes how the database has been acquired and how it has been organized for all further experiments.

In Section 5.1, the recording setup and data acquisition is described. Furthermore, the full information on the captured data and special particularities that occurred during the capture process is listed in a table. Section 5.2 describes the next step of the database construction, the post-processing of the inertial data. It is described how the database has been organized, ordered and splitted for special experiments as well as training and testing issues. For this, we also introduce the notation of dissimilarity matrices based on DTW distance measures.

5.1 Data Acquisition and Processing

5.1.1 Recording Setup

The data has been captured in a standard gymnasium using one new trampoline with a high quality and bouncy trampoline bed. As it is described in Section 3.3, inertial sensors do not require intensive setup or calibration, so the data could be recorded with little overhead and a short preparation phase using the Xsens Moven motion capture suit equipped with 10 inertial MTx motion trackers as positioned in Figure 3.4. Two Xbus master have been used that were connected to five sensors each to transfer the data to a computer via Bluetooth.

5.1.2 Capture Process

In two capture sessions, motions four intermediate trampolinists have been captured. For the following discussion, the actors will be named `hb`, `pm`, `sh` and `sm` according to their initials.

Each actor had to perform ten different exercises in at least two takes with three repetitions

per take resulting in the database $\mathcal{D}^{\mathrm{T}}$. The exercises were either single moves, short combined sequences of up to four moves or routines of ten consecutive different moves as they have to be executed in trampoline competitions. The range of the moves' technical requirements varied from easy moves like a tuck jump to different landing positions like the front drop and advanced moves like differently shaped somersaults. In the following table one finds the list of all takes of $\mathcal{D}^{\mathrm{T}}$ (sorted first by actor and then by scene), the respective lengths in frames, the corresponding XSens Take Number, a contents description, and, possibly, a comment. Scene 08 has been an optional scene that has only been performed by one actor while the routines have been captured at the end of the capture session in the scenes 09, 10 and 11.

FileNamePrefix	#(fr.)	Take	Description	Comments
TR2_hb_01_01	3522	085	1: Basic Jumps (PJP, TJP, SJP)	
TR2_hb_01_02	3814	086	·	
TR2_hb_02_01	3791	087	2: Front combination (HFR, FRF)	
TR2_hb_02_02	4566	088	·	
TR2_hb_03_01	6451	089	3: Somersault seat combination (TJP, BWS, SHA, HTW)	second combination: full twist
TR2_hb_03_02	6063	090	·	
TR2_hb_04_01	4707	091	4: Seat combination, Somersault Tucked (SED, SST, SJP, BWC)	
TR2_hb_04_02	5840	092	·	
TR2_hb_05_01	6351	093	5: Somersault piked combination (PJP, BWB)	
TR2_hb_05_02	6942	094	·	
TR2_hb_06_01	6715	095	6: Barani (BAR)	tucked
TR2_hb_06_02	6624	096	·	tucked
TR2_hb_07_01	7503	097	7: Somersault straight, full twist (BWA, FTW)	
TR2_hb_07_02	6321	098	·	
TR2_hb_08_01	8885	099	8: Three quarter backwards (3QB)	optionally
TR2_hb_08_02	9242	100	·	·
TR2_hb_09_01	3388	101	9: L8	only 9 jumps
TR2_hb_09_02	3581	102	·	only 9 jumps
TR2_hb_09_03	3490	103	·	
TR2_hb_09_04	3199	104	·	
TR2_hb_10_01	3580	105	10: Modified L7	
TR2_hb_10_02	3574	106	·	
TR2_hb_10_03	3402	107	·	
TR2_hb_11_01	4010	108	11: Freestyle	
TR2_hb_11_02	4065	109	·	
TR2_pm_01_01	3627	000	1: Basic Jumps (PJP, TJP, SJP)	6/4 Sensor Setup
TR2_pm_01_02	3290	001	·	·
TR2_pm_02_01	2940	002	2: Front combination (HFR, FRF)	·
TR2_pm_02_02	3852	003	·	·
TR2_pm_03_01	4266	004	3: Somersault seat combination (TJP, BWS, SHA, HTW)	·
TR2_pm_03_02	4363	005	·	Changed to 5/5 Sensor Setup
TR2_pm_04_01	4492	006	4: Seat combination, Somersault Tucked (SED, SST, SJP, BWC)	
TR2_pm_04_02	4255	007	·	
TR2_pm_05_01	578	008	5: Somersault piked combination (PJP, BWB)	Time outs
TR2_pm_05_02	1791	009	·	Time Outs
TR2_pm_05_03	3886	010	·	
TR2_pm_05_04	4218	011	·	
TR2_pm_06_01	3315	013	6: Barani (BAR)	tucked
TR2_pm_06_02	4233	014	·	tucked
TR2_pm_06_03	1003	015	·	Time outs, data loss
TR2_pm_06_04	—	016	·	data loss
TR2_pm_06_05	3875	019	·	straight
TR2_pm_07_01	—	017	7: Somersault straight, full twist (BWA, FTW)	data loss
TR2_pm_07_02	—	018	·	data loss

FileNamePrefix	#(fr.)	Take	Description	Comments
TR2_pm_07_03	4289	020	·	
TR2_pm_07_04	3904	021	·	
TR2_pm_09_01	3382	022	9: L8	
TR2_pm_09_02	3323	023	·	
TR2_pm_10_01	3495	024	10: Modified L7	
TR2_pm_10_02	3164	025	·	
TR2_pm_11_01	3757	027	11: Freestyle	Variation of M7
TR2_pm_11_02	3475	028	·	Variation of M7
TR2_sh_01_01	4235	029	1: Basic Jumps (PJP, TJP, SJP)	
TR2_sh_01_02	3928	030	·	
TR2_sh_02_01	4388	031	2: Front combination (HFR, FRF)	
TR2_sh_02_02	3913	032	·	
TR2_sh_03_01	4445	033	3: Somersault seat combination (TJP, BWS, SHA, HTW)	
TR2_sh_03_02	4152	034	·	
TR2_sh_04_01	3796	035	4: Seat combination, Somersault Tucked (SED, SST, SJP, BWC)	
TR2_sh_04_02	3873	036	·	
TR2_sh_05_01	3416	037	5: Somersault piked combination (PJP, BWB)	
TR2_sh_05_02	3208	038	·	data unuseful (wrong orientation)
TR2_sh_05_03	3044	039	·	data unuseful (wrong orientation)
TR2_sh_06_01	5743	040	6: Barani (BAR)	tucked, only 2 barani
TR2_sh_06_02	4186	041	·	tucked, only 1 barani
TR2_sh_06_03	4471	042	·	tucked, only 1 barani
TR2_sh_06_04	2494	043	·	tucked, only 1 barani
TR2_sh_06_05	3302	044	·	tucked, only 1 barani
TR2_sh_07_01	6098	045	7: Somersault straight, full twist (BWA, FTW)	
TR2_sh_07_02	5864	046	·	
TR2_sh_09_01	4301	047	9: L8	
TR2_sh_09_02	6071	048	·	
TR2_sh_10_01	4109	049	10: Modified L7	only 9 jumps
TR2_sh_10_02	3419	050	·	only 9 jumps
TR2_sh_10_03	3014	051	·	
TR2_sh_10_04	3143	052	·	
TR2_sh_11_01	3646	053	11: Freestyle	L3
TR2_sh_11_02	3176	054	·	L3
TR2_sm_01_01	3639	055	1: Basic Jumps (PJP, TJP, SJP)	
TR2_sm_01_02	4323	056	·	
TR2_sm_02_01	3206	057	2: Front combination (HFR, FRF)	Time outs at the end
TR2_sm_02_02	4283	060	·	
TR2_sm_03_01	4196	061	3: Somersault seat combination (TJP, BWS, SHA, HTW)	
TR2_sm_03_02	3703	062	·	
TR2_sm_04_01	5041	063	4: Seat combination, Somersault Tucked (SED, SST, SJP, BWC)	
TR2_sm_04_02	4151	064	·	
TR2_sm_05_01	—	066	5: Somersault piked combination (PJP, BWB)	
TR2_sm_05_02	1160	067	·	Time outs, data loss
TR2_sm_05_03	1542	068	·	Time outs
TR2_sm_05_04	3408	069	·	
TR2_sm_06_01	4535	070	6: Barani (BAR)	tucked, after somersault bw
TR2_sm_06_02	6930	071	·	tucked, after somersault bw
TR2_sm_06_03	3024	072	·	tucked
TR2_sm_07_01	4016	073	7: Somersault straight, full twist (BWA, FTW)	
TR2_sm_07_02	3786	074	·	
TR2_sm_09_01	3262	075	9: L8	
TR2_sm_09_02	3128	076	·	only 8 jumps
TR2_sm_09_03	2995	077	·	
TR2_sm_10_01	2886	078	10: Modified L7	
TR2_sm_10_02	2872	079	·	
TR2_sm_11_01	2980	080	11: Freestyle	M5
TR2_sm_11_02	2755	082	·	M5
TR2_sm_11_03	2731	083	·	L3
TR2_sm_11_04	2809	084	·	L3

Inertial sensors include some specific properties that have to be kept in mind during the capture process as they can lead to noisy data and can deteriorate the quality of the information captured. For example, ferromagnetic materials influence the sensor's measurement of the magnetic field and can therefore result in wrong orientation data. To circumvent these inaccuracies, the sensors should be running for some time after the system start or after changing the location and herewith changing the magnetic properties of the surrounding before the capture process. During the capture process made for this thesis, for example, no attention was paid to the magnetic property of the trampoline skid leading to few unusable takes that were captured after the athlete took a rest near the trampoline skid. Furthermore, one always has to expect some takes to fail because of timeouts in the data transfer that occur due to problems with the wireless Bluetooth data transmission, see Table 5.1.2.

5.2 Post Processing

As counter draw to the easy use and short set up of the capture system, more post processing steps and several steps of file format transfer are required until the data recorded could be used. In detail, the raw data for every take consisted of two motion streams (containing the data of five sensors) that have been transferred by the two Xbus master. Those takes then had to be synchronized and combined into one signal. Additionally, the data from one digital video camera that captured the athlete's motion has been used to annotate the motion streams. By this video annotations, the motion stream for each take could then be segmented and cut into separate trampoline jumps.

As the other takes, the routine exercises have been annotated manually, but have been left as one consecutive motion stream containing several trampoline jumps. The consecutive routines and their annotations will be used for classification later in Chapter 8. With the four actors and the single and combined motion exercises, a cut database $\mathcal{D}^{\mathrm{C}}$ consisting of more than 750 trampoline moves and 19 different moves could be acquired.

Out of this huge dataset, consistent data has then been determined and selected into a selected cut database $\mathcal{D}^{\mathrm{CS}} \subset \mathcal{D}^{\mathrm{C}}$. In this context, consistent means that the motion has been performed in a constant motion style without any technical or morphological outliers that would make similarity measures between motions of the same motion class more difficult.

In a first selection step, all data deviating clearly from the general style of a motion class have been discarded for $\mathcal{D}^{\mathrm{CS}}$. For this, the feature representation of every jump in $\mathcal{D}^{\mathrm{C}}$ in form of the feature matrix has been examined. By this, wrongly cut jumps based on erroneous annotations (that would for example differ significantly in length) and differences in motion style or failed jumps could be identified. Figure 5.1 shows sample feature matrices for the same motion class. The outlier in motion style is clearly visible by differing feature vectors 3,4,5 and 6. The variation can be traced back to the motion performance, where the actor variated the position and orientation of the arms at the ending of the jump depending on whether the move has been performed separately or within a routine.

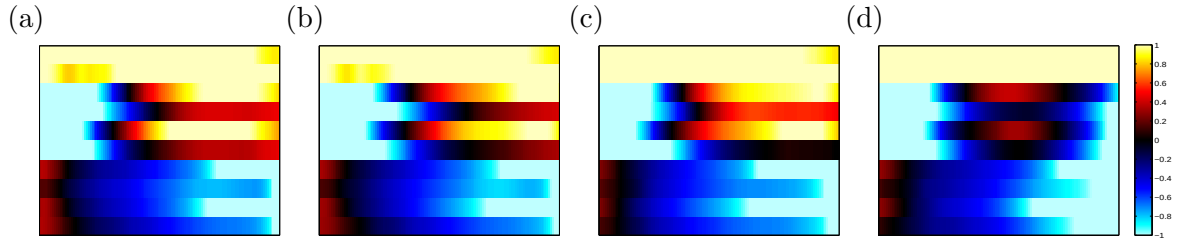

Figure 5.1. Sample feature representations for the same motion class with feature set $F_{\mathtt{I10}}$. All jumps have been performed by the same motion actor. While the arms are pointing upwards at the end of each jump in (a), (b) and (c), for the outlier motion (d), they have been moved downwards (see feature vectors 3, 4, 5 and 6).

In a second selection step, we discard all jumps of a motion class that are characterized by high costs. Here, we compute a *dissimilarity matrix* that contains the cost for the comparison of each pair of documents within all jumps in $\mathcal{D}^{\mathrm{C}}$ that comprise the same motion class.

5.2.1 Dissimilarity Matrix

Let $\mathcal{D} = (D_1, ..., D_K)$ be a database consisting of documents D_k, $k \in [1 : K]$. F is defined to be a feature function and $D \in \mathcal{D}$ to be a document. Then, $F(D)$ denotes the resulting feature sequence. For two documents D_1 and D_2, we define the *document-based distance* (or global DTW distance) between D_1 and D_2 with respect to F to be

$$\mathrm{DTW}_F(D_1, D_2) := \mathrm{DTW}(F(D_1), F(D_2)).$$

Using this definition, we can compute a *document-based dissimilarity matrix* $M_{\mathcal{D},F} \in \mathbb{R}^{K \times K}$ for a given database $\mathcal{D}$ and a feature function F. Then, the matrix entry for the i-th and j-th document in $\mathcal{D}$ is

$$M_{\mathcal{D},F}(i, j) := \mathrm{DTW}_F(D_i, D_j).$$

This similarity score will then be normalized over both document lengths into

$$m_{ij} = M_{\mathcal{D},F}(i, j) / \left(|D_i| + |D_j|\right).$$

The dissimilarity matrix is characterized as followed. Each document's cost with itself can be found at the diagonal axis and equals zero. Semantically different jumps should have a higher cost than semantically similar jumps. Jumps of one motion class performed by the same actor are supposed to have a lower cost in the comparison than the jumps performed by different actors as each actor has it's individual motion style which is almost constant over several motion performances.

Figure 5.2 shows the sample distance matrices for all motions within $\mathcal{D}^{\mathrm{C}}$ for two different motion classes. For both matrices, motions that are performed by the same actor than the query motion are represented by lower costs than the remaining motion performances. In Figure 5.2(a), some significant variation is visible in the middle of the dissimilarity matrix

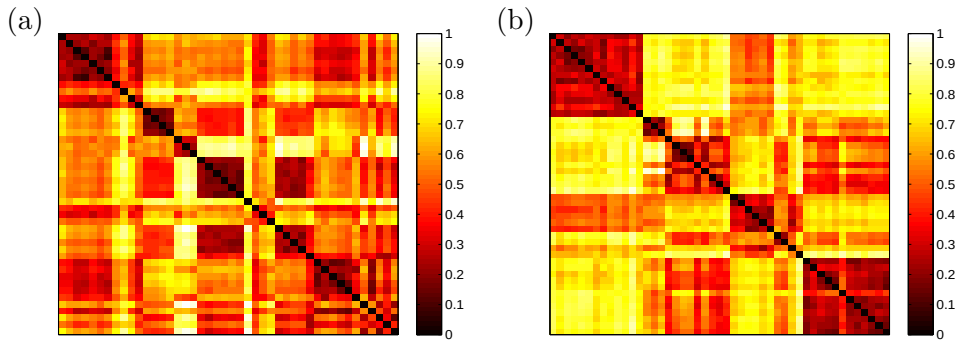

Figure 5.2. Dissimilarity matrices for two different motion classes within $\mathcal{D}^{\mathrm{C}}$. Motions of the same actor are more similar to each other than other motions indicated by lower costs c around $c = 0.5$. Outlier in (a) are clearly visible by higher costs.

indicated by light matrix entries. Those jumps have been discarded as outliers for the selection of $\mathcal{D}^{\mathrm{CS}}$. In Figure 5.2(b), four patterns indicating four different motion styles (which means the individual style of every actor) are visible. Additionally, the motions of actor `hb` and `sh` and the motions of actor `pm` and `sm` seem to imply higher numeric similarities indicated by lower costs c around $c = 0.5$.

Finally, four moves per actor have been selected out of all captured trampoline jumps for $\mathcal{D}^{\mathrm{CS}}$. Twelve motion classes have been chosen leading to a total number of 192 trampoline jumps.

Chapter 6

Feature Set Evaluation

The following chapters document the extensive experiments that have been conducted on the inertial trampoline database.

In Chapter 4, we have introduced several (composed) feature sets that transform the inertial motion data stream into meaningful feature representations. Then, the similarity of the feature representations to the previously computed MTs can be determined and the passages within the feature sequences can be labeled as specific motion classes. To enable stable and precise labelings, feature representations that show differences of dissimilar motion classes are helpful. In the next experiments, we hence measure the discriminative power of the given feature sets and determine the most discriminative one. Furthermore, we explain the results of the experiments by interpretation of the semantic characteristics of different motion categories as it is also done for motion understanding and motion analysis in the fields of sports science. In our scenario, information on the morphological or semantic aspects expresses how difficult it is to distinguish dissimilar motion classes.

We evaluate the feature sets on documents containing a single motion each using global DTW in Section 6.1, and on one single document containing a sequence of different motions using subsequence DTW in Section 6.2. Both sections evaluate the feature sets qualitatively and quantitatively. Finally, in Section 6.3, we conclude the chapter by choosing the best and most discriminative feature set for all further experiments.

6.1 Document-based Evaluation

In a first experiment, we test the significance and power of the composed feature sets from Chapter 4 using the dissimilarity matrix in the same way as in Chapter 5. In this context, the significance of a feature set expresses to which extent the numeric feature values can represent the morphological and semantical properties of each motion class. We evaluate how similar the feature representations for all motion classes are and how good motion classes can be discriminated from each other with a given feature set. As we evaluate the feature sets document-wise, we will use the cut trampoline jumps from the selected cut database $\mathcal{D}^{\mathrm{CS}}$.

For the following experiments, $\mathcal{D}^{\mathrm{CS}}$ has been split into a test database $\mathcal{D}^{\mathrm{C1}}$ with $\mathcal{D} = (D_1, D_2, \ldots, D_K)$ that contains the documents with an even index and one training database $\mathcal{D}^{\mathrm{C2}}$ with $\mathcal{D} = (T_1, T_2, \ldots, T_N)$ that contains the documents with an odd index. In this chapter, we will only use the data from $\mathcal{D}^{\mathrm{C1}}$. $\mathcal{D}^{\mathrm{C2}}$ is chosen to compute the MTs and will be used in later experiments, see Chapter 7.

Before we start, we fix the IDs and abbreviations for every motion class that will be used during the following experiments, see Table 6.1. A precise overview of the different motion categories and their semantics can be found in Chapter 2. The abbreviations for the somersault jumps (`BWC` and `BWB`) are taken from common trampoline naming that designates a tucked somersault as *c*- and a piked somersault as *b*-shaped.

ID (long)	ID (short)	Description
`Bar`	`BAR`	Barani
`FrFeet`	`FRF`	Front to feet
`HaTw`	`HTW`	Half twist
`HaFr`	`HFR`	Half twist front
`PiJump`	`PJP`	Pike jump
`SeHaTw`	`SHA`	Seat half twist stand
`SeStd`	`SST`	Seat stand
`SomPi`	`BWB`	Somersault piked
`SomSe`	`BWS`	Somersault to seat
`SomTu`	`BWC`	Somersault tucked
`SdJump`	`SJP`	Straddle jump
`TuJump`	`TJP`	Tuck jump

Table 6.1. Description and IDs (long ID and 3-digit short ID) for the motion classes used in our experiments.

6.1.1 Dissimilarity Matrix

For the feature evaluation, we use a dissimilarity matrix (see Chapter 5) $M \in \mathbb{R}^{K \times K}$ by computing the normalized similarity scores m_{ij} for each pair of documents in $\mathcal{D}^{\mathrm{C1}}$. Comparing the similarity scores among different feature sets, we obtain information on how good every feature set discriminates all motion classes. In general, one can state the following. If the feature set is highly discriminative, the similarity scores for documents from different motion classes are high and every motion class is separable from all other motion classes. If the feature set is less discriminative, different motion classes will have similar similarity scores and cannot be separated accurately. If the similarity scores between two motion classes are similar among several feature functions so that the motion classes cannot be distinguished from each other, we expect the motion classes to be semantically similar.

For all different feature sets, similarity matrices are computed to compare the feature sets used. A qualitative evaluation of the dissimilarity matrix for a feature set can be made as followed. In case of a discriminative feature design, dark blocks within every motion class should be visible in the distance matrix around the diagonal. They indicate that m_{ij} for two documents within one motion class is smaller than m_{ij} for two documents from different motion classes.

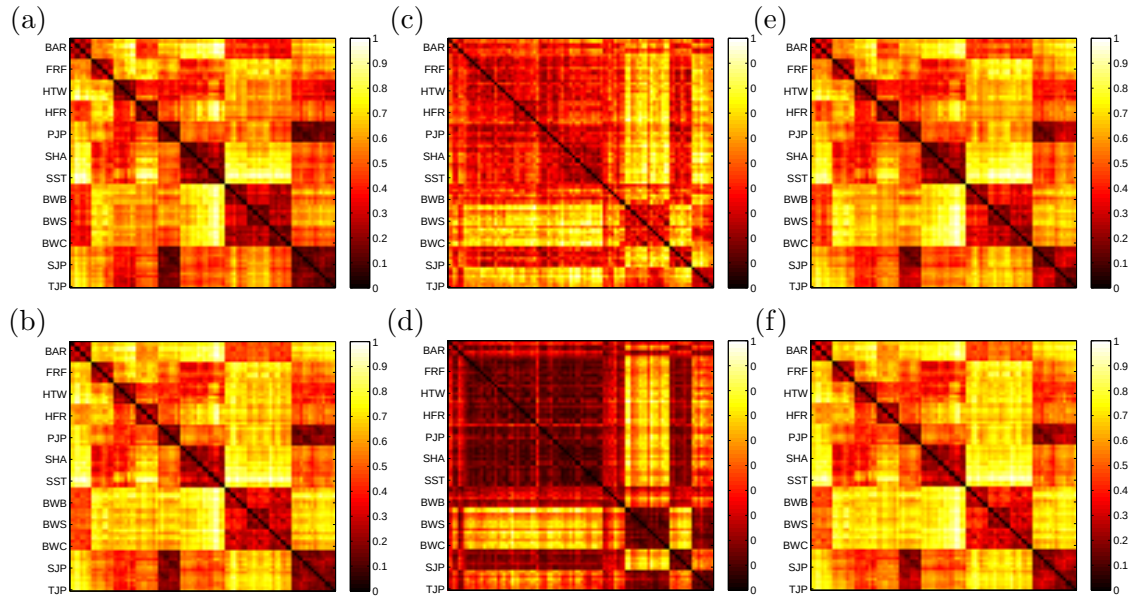

Figure 6.1. Dissimilarity matrices for document-based retrieval and selected feature sets. Left: Inclination feature sets $F_{\texttt{I5}}$ (a) and $F_{\texttt{I10}}$ (b). Middle: Angular-based feature sets $F_{\texttt{A5}}$ (c) and $F_{\texttt{A3}}$ (d). Right: Combined feature sets $F_{\texttt{I5A3}}$ (e) and $F_{\texttt{I10A3}}$ (f).

6.1.2 Results and Discussion

The dissimilarity matrices for all given composed feature sets visualize how precise every feature set or feature type can discriminate between all motion classes, see Figure 6.1. In the following, we analyze all feature sets.

Inclination features: $F_{\texttt{I5}}$ appears to be a good discriminator for almost all motion classes and does not seem to deliver worse results than the $F_{\texttt{I10}}$ feature set. This seems to be surprising at first glance, as $F_{\texttt{I10}}$ offers more numerical information about the motions (and especially about the motions of the extremities). However, data of the inner sensors, as taken into account in $F_{\texttt{I10}}$ does not seem to offer more and better information for a feature description. Information on the pitch ϕ_s of the upper arms and upper legs seems to be redundant as it does not contain other information than the pitch of the lower arms and lower legs in most jumps. As it can be detected from Figure 6.1, only between the semantically very similar motions `PJP`, `TJP` and `SJP`, $F_{\texttt{I5}}$ does not yield any discrimination, as the information that denotes differences between those jumps cannot be expressed by the feature set. In particular, from the pitch ϕ_s of both lower legs only, one cannot tell whether the legs are spread or not. Figure 6.2 shows the feature matrices for one example actor for all three jumps.

Angular-based features: The angular based feature $F_{\texttt{A5}}$ does not obtain good discrimination results for most motion classes. Only for the discrimination between motion classes that are characterized by angular differences at extremities like the `TJP` and `SJP`, the angular-based feature types yield a good discrimination of all motion classes. The result, however, is natural, as for most other motion classes, no angular information and changes occur in the extremities. As a result, the comparison between all motion classes

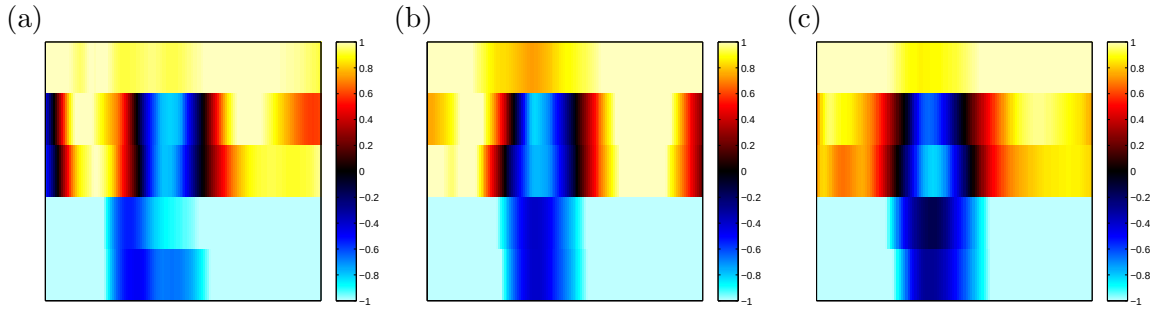

Figure 6.2. Differences between the feature matrices of TJP (a), PJP (b) and SJP (c) can hardly be discovered with the feature set $F_{\texttt{I5}}$.

will result in equal similarity scores. Figure 6.3 shows the feature matrices for one example actor for three different jumps of the motion classes SST, HTW and HFR. Hardly any differences in the feature matrices can be found with an angular feature set for those motion classes similar than for the $F_{\texttt{I5}}$ feature set with the basic shaped motion classes PJP,TJP and SJP. However, SST, HTW and HFR could have been discriminated well with $F_{\texttt{I5}}$. In cases where $F_{\texttt{I5}}$ fails, $F_{\texttt{A5}}$ discriminates well and vice versa, so that we interpret that the $F_{\texttt{A5}}$ feature set behaves complementary to the $F_{\texttt{I5}}$ feature set.

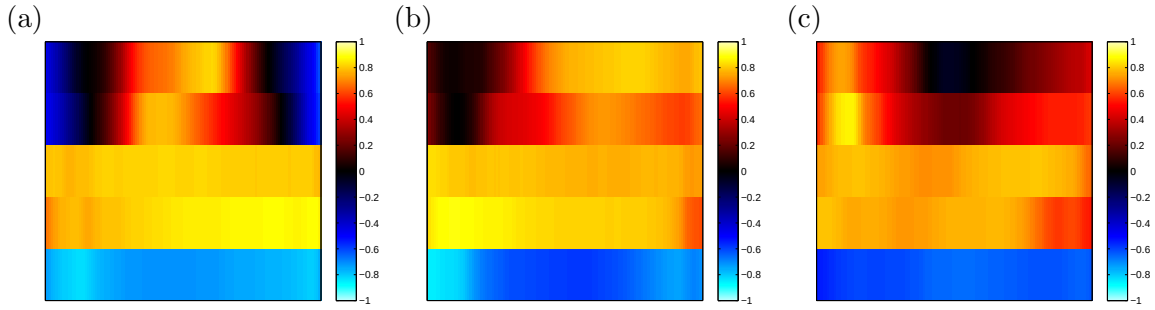

Figure 6.3. Differences between the feature matrices of SST (a), HTW (b) and HFR (c) can hardly be discovered with the feature set $F_{\texttt{A5}}$.

During our experiments, we found out that variations in the execution style are very frequent in the angle of the arms among all actors. To facilitate invariance under these special variations, we created $F_{\texttt{A3}}$ which only represents angular characteristics of the legs. One can see out of Figure 6.1, that the angular-based feature types become more discriminative without the information of the arms.

Combined features: As both feature types offer strengths and weaknesses in discriminating several motion classes, a combined feature set should be ideal to discriminate all motion classes. The combined feature types use features from both $F_{\texttt{I5}}$ and $F_{\texttt{A5}}$ leading to a better discrimination between PJP, TJP and SJP than the pure $F_{\texttt{I5}}$ feature set and a good discrimination among all remaining motion classes, see the differently colored blocks in Figure 6.1 and the high costs for similarity scores of documents of different motion classes. By visible comparison, the feature set $F_{\texttt{I5A3}}$ seems to deliver the best results. All motion classes show certain differences in the feature matrices, see Figure 6.4, so that each motion class can be distinguished from the other motion classes. In particular, this feature set is less sensitive to variations in the arm movement among all actors. Moreover, even

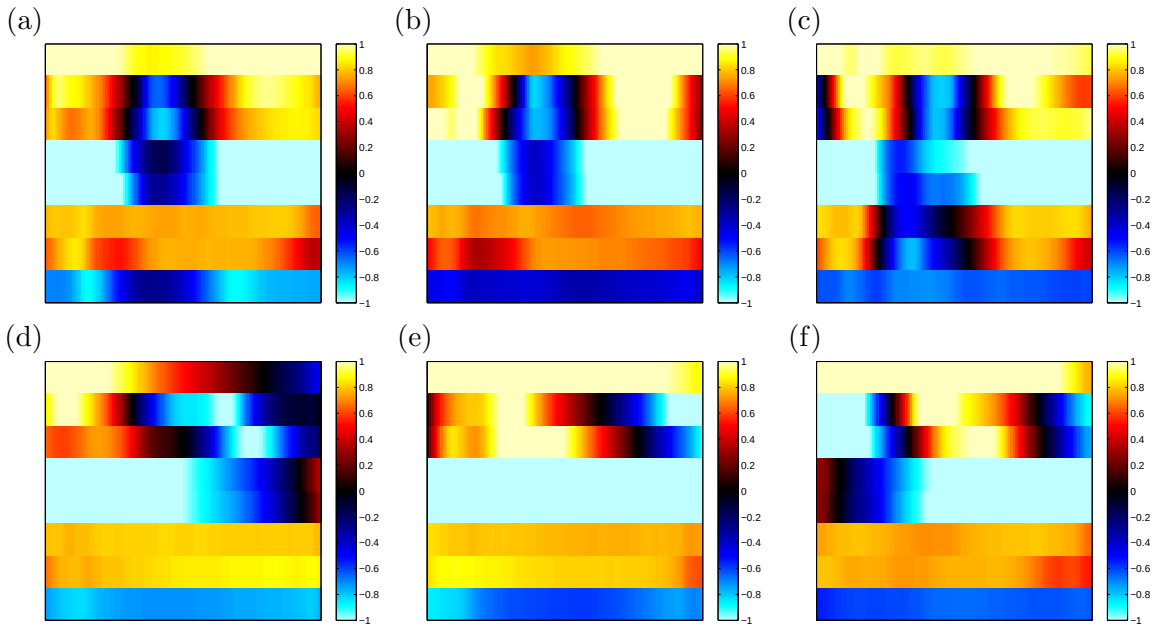

Figure 6.4. Feature matrices of example motions computed with the combined feature set $F_{\texttt{I5A3}}$. Differences between `PJP`, `TJP` and `SJP` (upper row (a) to (c)) and `SST`, `HTW` and `BWC` (bottom row (d) to (f)) are visible.

the motion classes `PJP`, `TJP` and `SJP` have differing feature matrices, which was impossible with $F_{\texttt{I5}}$.

After interpreting all distance matrices, one can conclude the following:

- Moves that are semantically similar, for example differently shaped somersaults `BWB` and `BWC` or differently shaped basic moves like `SJP` and `PJP` jump, are hard to distinguish.
- Variations within the arm movements leads to differences that can worsen the discriminative power of selected feature set.

6.1.3 Evaluation Measures

In addition to visual evaluations on dissimilarity and feature matrices, feature sets can also be evaluated numerically. Now, any two motion documents within the same motion class are considered as similar, whereas two motion documents from different classes are considered as dissimilar. To measure the degree of performance invariance of a given feature set F, we compute the distances between any two motion documents that belong to the same motion class.

Let $\max_{\mathrm{I}}$ be the maximum value of the resulting distances within one motion class (here called as inner motions). Note that $\max_{\mathrm{I}}$ should be as small as possible for a good feature set. Similarly, let $\min_{\mathrm{O}}$ be the minimum value of the distances of all remaining motion classes (here called as other motions). Note that $\min_{\mathrm{I}}$ should be large to indicate a high

Feature	ϵ	υ	δ
$F_{\texttt{I5}}$	0.267	0.507	1.878
$F_{\texttt{I10}}$	0.304	0.555	1.849
$F_{\texttt{A5}}$	0.518	0.827	2.937
$F_{\texttt{A3}}$	0.381	0.902	3.785
$F_{\texttt{I5A5}}$	0.341	0.531	1.645
$F_{\texttt{I5A3}}$	0.288	0.484	1.657
$F_{\texttt{I10A5}}$	0.343	0.556	1.735
$F_{\texttt{I10A3}}$	0.313	0.535	1.759

Table 6.2. Mean distance measures over all motion classes for all selected feature sets.

discriminative power of a feature set. Finally, we form the quotient $\delta := \max_{\mathrm{I}} / \min_{\mathrm{O}}$. δ should be less than 1 in case of a strong discriminative feature set. However, δ is very prone to outliers within the computed distances. One high cost value within the inner motion class documents or one low cost within all other motion class documents can deteriorate the measure. Therefore, we introduce further measures that are less influenced by outliers, namely ϵ and υ.

Let μ_{I} be the mean and σ_{I} the standard deviation over the computed distances. In the case that the feature set has a high degree of performance invariance, μ_{I} should be small. Similarly, let μ_{O} be the mean and σ_{O} the standard deviation over the distances of any two motion documents from different motion classes. If the feature sets are of high discriminative power, μ_{O} should be large. Then $\epsilon := \mu_{\mathrm{I}}/\mu_{\mathrm{O}}$ expresses the within-class distance μ_{I} relative to the across-class distance μ_{O}. Note that a small value of ϵ is a desirable property of a given feature function F.

For an even more meaningful evaluation, we then define $\mu_{p\mathrm{O}}$ to be the mean over the smallest $p\%$ of all distances of different motion classes. Finally, we get $\upsilon := \mu_{\mathrm{I}}/\mu_{p\mathrm{O}}$ as further distance measure. Again, υ should be as small as possible.

6.1.4 Results and Discussion

To evaluate all feature sets quantitatively, we examine the previously described measures δ, ϵ and υ. The mean δ-, ϵ- and υ-values over all motion classes affirm the results that have been visualized and determined qualitatively, see Table 6.2. The feature sets $F_{\texttt{I5}}$ and $F_{\texttt{I5A3}}$ result in the best ϵ- and υ-values. As it was mentioned before, the δ-value is highly affected by outliers, so the δ-values are less significant. For every motion class, there is at least one computed distance for jumps of different motion classes of lower cost than the distances within one motion class. Nevertheless, the δ-values are best for the combined-feature types, as well. For the angular-based feature they are much worse than for the inclination feature types indicating that those feature sets have lowest discriminative power.

We now want to have a closer look at the discriminative power of all feature sets for different motion classes. For this, we use the υ measure that comprises the most meaningful information about the distribution of document distances. Table 6.3 shows the υ-values for all motion classes and the mean υ-value over all feature sets. As already indicated by the qualitative analysis, the angular-based feature sets cannot distinguish between motions

Feature	BAR	FRF	HTW	HFR	PJP	SHT	SST	BWB	BWS	BWC	SJP	TJP	Mean
$F_{\texttt{I5}}$	0.558	0.478	0.453	0.343	0.696	0.433	0.459	0.520	0.416	0.509	0.603	0.614	0.507
$F_{\texttt{I10}}$	0.478	0.500	0.461	0.409	0.702	0.521	0.615	0.603	0.437	0.609	0.680	0.641	0.555
$F_{\texttt{A5}}$	0.900	0.835	0.980	0.672	0.914	0.819	0.770	0.983	0.661	0.850	0.932	0.601	0.827
$F_{\texttt{A3}}$	1.103	0.891	1.198	0.904	1.204	0.960	0.907	0.839	0.631	0.784	0.780	0.621	0.902
$F_{\texttt{I5A5}}$	0.563	0.516	0.518	0.388	0.609	0.516	0.544	0.574	0.480	0.605	0.587	0.472	0.531
$F_{\texttt{I5A3}}$	0.565	0.497	0.466	0.354	0.587	0.458	0.497	0.491	0.421	0.517	0.513	0.444	0.484
$F_{\texttt{I10A5}}$	0.505	0.523	0.502	0.432	0.643	0.552	0.634	0.611	0.473	0.635	0.640	0.527	0.556
$F_{\texttt{I10A3}}$	0.494	0.512	0.472	0.412	0.639	0.529	0.627	0.568	0.442	0.593	0.611	0.523	0.535

Table 6.3. υ values for p = 20% among all different motion classes for all feature sets.

that are not characterized by their motion shape, see Figure 6.3, but can discriminate TJP and SJP. As visualized in Figure 6.2, inclination feature sets cannot discriminate documents of motion classes PJP, TJP and SJP. For example, the υ-values of the PJP for $F_{\texttt{I5}}$ and $F_{\texttt{I10}}$ are much higher than the υ-values for HTW, HFR or BWS.

6.2 Subsequence-based Evaluation

For a second feature evaluation experiment, we choose a much more realistic scenario since a given data stream will not be segmented beforehand. Here, we do not evaluate the distance between two jumps in $\mathcal{D}^{\text{C1}}$, but compare one feature sequence X with a long data stream using subsequence DTW.

For the experiments in this section, we will simulate the long data stream by concatenating the test database $\mathcal{D}^{\text{C1}}$ to the consecutive database $\mathcal{D}^{\text{Seq}}$ without any separation data in between each jump. We will store the concatenation information in an additional data structure for later evaluation of the subsequence evaluation results.

We recapulate that in the subsequence DTW algorithm, a short feature sequence X is compared to a longer feature sequence Y. Low costs in the computed distance curve indicate passages in Y that are similar to the query. In the ideal case, all locations of lowest cost indicate jumps of the same motion class as X, yielding a correct labeling of Y with X. In practice, however, positions of low cost in the distance curve will also relate to jumps of different motion classes, yielding to false label. For a good feature set, the number of correct labels should be much higher than the number of false labels. Note that morphological uniqueness of a motion class should lead to a very good separation from all other motion classes and thus labels for these motion classes should all be correct.

6.2.1 Retrieval

We know from $\mathcal{D}^{\text{C1}}$ that every motion class is represented by eight jumps in $\mathcal{D}^{\text{Seq}}$. In our experiments, we use this knowledge to retrieve the end positions of the eight most similar jumps out of the distance curve that can be determined by using the ranked Δ-values of the distance curve. The resulting positions are then compared with the concatenation information from the database concatenation.

As feature sequence X, we use every jump contained in $\mathcal{D}^{\text{C1}}$. This means, that for every

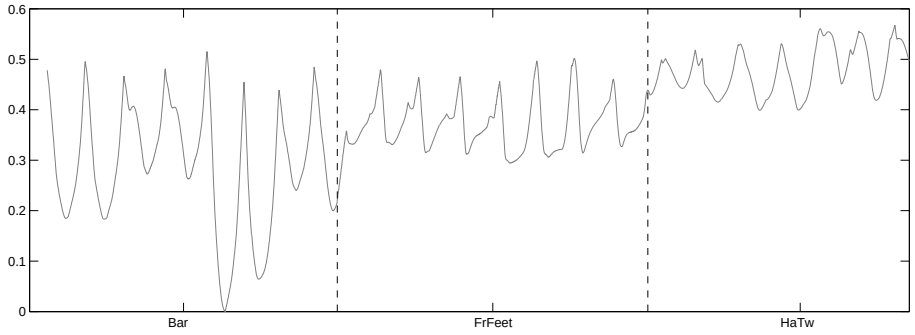

Figure 6.5. Distance curve for motion class BAR. The motion stream Y has been build so that for every motion class, every second jump is performed by a different actor. The 5th BAR jump within Y equals the query motion X. The cost for all jumps within the motion class varies among different actors representing different motion styles.

query sequence X, the minimum cost of the distance function, also called *cost of the match*, will be zero since X is also compared to itself. The position b_0 of the cost of the best match (*rank 1*) can be found at the end position of X within $\mathcal{D}^{\text{Seq}}$. Note that the cost of positions in the neighborhood of b_0 are also low as they are influenced by the low cost of b_0. Thus, we exclude the neighborhood around b_0 for the following processing steps. We call this neighborhood *false alarm region* and use half the query length to the right and left of the index b_0. We continue in this manner to to find *rank of the match* $2 \ldots K$. Retrieval results that belong to the same motion class as X are called *true hits*, retrieved results that belong to different motion classes as X are called *false hits*.

The distance curve can give information on how much variation all jumps from one motion class undergo. Figure 6.5 shows an example of how the distances for the BAR jumps within $\mathcal{D}^{\text{C1}}$ vary among all four actors (representing four different styles of motion performance). BAR jumps that are performed by the same actor than the one in X have lowest costs. Jumps from different actors even may have higher costs than jumps from different motion classes due to high variation among different actors.

The number of true and implicitly the number of false hits among the results from rank 1 to 8 can deliver much information about the the quality of the feature set and it's discriminative power or the morphological aspects of the motion in comparison to all other motions. The discriminative power of a feature set has already been described in section 6.1 and will be evaluated in a similar way for the subsequence retrieval. With respect to information on the morphological aspects, we expect the following. If it is easy to recognize the characteristics of a motion class that occur in a real-life trampoline performance for non-specialized spectators, the motion classes are also easier to characterize numerically. Consequently, more true hits for the retrieval should occur. For the distance measure, this means that the cost between jumps of different motion classes is high.

6.2.2 Distance Measures

To quantify the differences and the discriminative power of each feature set, we define similar distance measures than in Section 6.1 based on the α, β and γ measures from [24].

Using the concatenation information, we can retrieve the costs for all jumps within $\mathcal{D}^{\text{Seq}}$ that belong to the same motion class than X which are the cost for all true hits. The maximum of those costs is referred to as $\max_T^X$. Among all jumps within $\mathcal{D}^{\text{Seq}}$ that belong to different motion classes than X (which are all possible false hits), we define the minimal cost as $\min_F^X$. Furthermore, let μ_T^X be the mean over the cost values for all true hits of the same motion class and μ_F^X the mean value over the cost values for all false hits of different motion classes.

The γ measure then is defined as

$$\gamma^X := \max_T^X / \min_F^X. \tag{6.1}$$

Ideally, the γ-value should be smaller than one indicating that all jumps that belong to one motion class have lower cost than all jumps from different motion classes. Unfortunately, the γ-value is prone to outliers and can not always deliver significant information.

Distance measures that yield more meaningful results are the α and the β measures. The α measure is defined as

$$\alpha^X := \mu_T^X / \mu_F^X. \tag{6.2}$$

The β measure states an even more meaningful distance measure. Here, we only respect the mean value of the lowest p% cost values within all false hits $\mu_F^{p,X}$. β is then defined as

$$\beta^X := \mu_T^X / \mu_F^{p,X}. \tag{6.3}$$

If β is small, we know that the difference between the mean cost of all true hits and the false hits with the lowest costs is very high. So the β measure can display information on the discriminative power of a feature set in a very exact way. An appropriate feature set should yield small α-, β- and γ-values.

6.2.3 Results and Discussion

From the first eight hits for a specific query motion class in the corresponding distance function we draw conclusions regarding morphological issues and feature evaluation.

Regarding the morphologies of a motion class, it becomes clear that there are motion classes that we can transform better into significant feature representations than other motion classes. Figure 6.6 shows a distance curve for one sample actor for the morphologically significant motion class `BWB` and a distance curve for the semantically insignificant motion class `PJP` that is semantically similar to the motion class `SJP`. For semantically similar motion classes, the number of false hits raises and the α-, β- and γ-values deteriorate. The distance between the maximum cost within the correct motion class and the minimum cost within the remaining motion classes decreases.

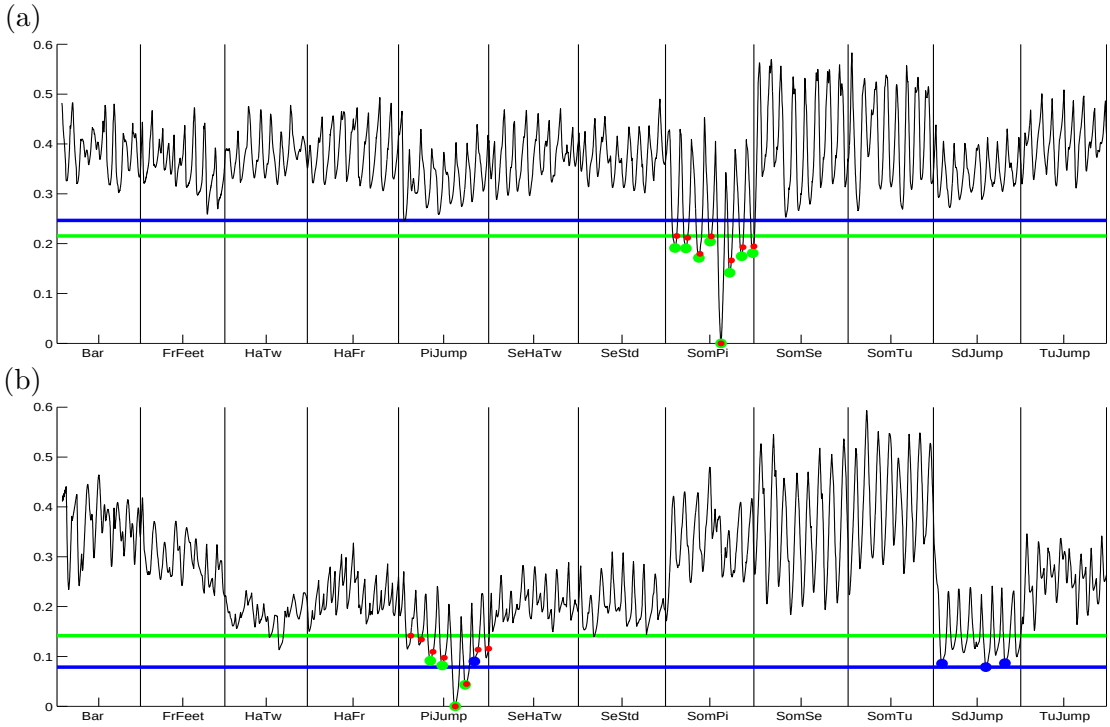

Figure 6.6. Distance curves for the motion classes `BWB` (a) and `PJP` (b) for Actor `hb`. In (a), all true hits are retrieved within the first eight hits (indicated by green dots) whereas in (b), false hits (indicated by blue dots) appear. Note that the position of the document endings (indicated by red dots) cannot always be found at the same position than the retrieved hits (blue dot for the retrieval of the ninth pike jump in the database).

As one can see in Figure 6.6, the `BWB` jumps can be discriminated quite well from all other motion classes. The rotational motion around the lateral axis leads to a very significant pitch of all sensors and thus the somersault jumps stand out from all other motion classes.

On the other hand, motion classes that semantically do not differ to a large amount are hard to distinguish. Those are, for example, all basic and easy jumps such as `HTW`, `PJP`, `TJP` and `SJP`. All of those motion classes contain less discriminative characteristics and are therefore hard to distinguish numerically. It was already mentioned that those moves cannot be discriminated by all feature designs and can even hardly be described differently in a numerical way irrespective to the chosen feature set as they are morpholocially similar. Figure 6.7 illustrates morphological significance and shows a feature matrix of the significant motion class `BWB` and a feature matrix of the less significant motion class `PJP`. Both motion classes only differ by the rotational motion around the lateral axis. This difference makes one motion class easy to retrieve within the concatenated database, while the other one is hard to distinguish from other motion classes. In principal, it seems that the easier one trampoline jump is, the less discriminative information it contains and the harder it is to transform it's morphologic properties into unique numeric descriptions.

It becomes obvious from Figure 6.6 that for the `PJP`, the positions of lowest cost cannot always be found at the document's end within $\mathcal{D}^{\text{Seq}}$. Even if we define a tolerance of several frames as natural deviations caused by the hand-made annotations, not all local cost minima can be related to the document's end position from the annotations. As

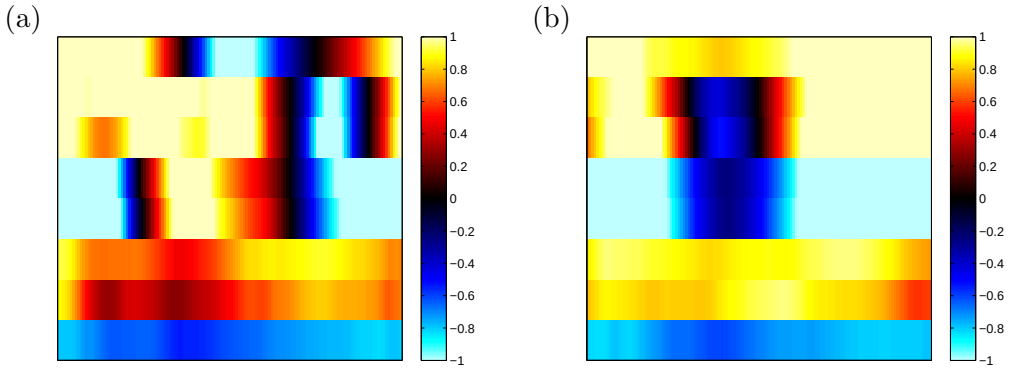

Figure 6.7. Comparison between numerical representation of the significant motion class BWB (a) and the insignificant PJP motion class (b). In (a), all features of $F_{\texttt{I5}}$ (rows within the feature matrix) contain characteristic information whereas in (b) whole features do not contain enough discriminative information to distinguish the motion class from similar motion classes or a standing pose.

those shifts mainly occur for the basic jumps, we expect those variations to be a result of different motion style and especially speed differences during the flight phase. This means, that for some PJP jumps, the elementary motion technique (moving the legs and arms into a piked motion shape followed by a motion release phase into the straight motion shape) has been performed earlier or faster during flight phase. Then, for the rest of the flight phase, nothing happens any more, so that the ending of a jump may be found before the real end of the motion document within $\mathcal{D}^{\text{Seq}}$. This information could later be useful for performance monitoring, but for the current experiments, it will be necessary to become invariant against those temporal shifts.

For the document based retrieval, the $F_{\texttt{I5A3}}$ feature set turned out to be the best overall feature representation. In concordance to those results, graphical visualizations with the distance curves yield similar results for the subsequence-based experiments. With the $F_{\texttt{I5A3}}$ feature set, most of the true hits could be retrieved and the distance to the remaining motion classes is relatively high. In a quantitative evaluation with the introduced distance measures, however, the α- and β-values are better for the $F_{\texttt{I5}}$ feature set. Based on the results from the document based evaluation, only the three best feature sets have been further evaluated in the subsequence evaluation process. Table 6.4 shows an overview over the distance measures (averaged over all query motions) for different feature sets. However, we can see that the difference between the average β-values for $F_{\texttt{I5A3}}$ and the average β-values for $F_{\texttt{I5}}$ is much smaller than the difference of the α-values. As the β-value is more meaningful than the α-value and as we already have seen that the $F_{\texttt{I5}}$ feature set cannot discriminate all motion classes in a satisfying way, we nevertheless choose $F_{\texttt{I5A3}}$.

Feature	μ_T	μ_F	α	μ_T	$\mu_F^{20\%}$	β	max_T	min_F	γ
$F_{\texttt{I5}}$	0.116	0.360	0.319	0.116	0.224	0.530	0.182	0.118	1.698
$F_{\texttt{I5A5}}$	0.154	0.344	0.449	0.154	0.253	0.621	0.223	0.154	1.583
$F_{\texttt{I5A3}}$	0.117	0.326	0.361	0.117	0.224	0.541	0.176	0.128	1.532

Table 6.4. Averaged distance measures over all query motions for feature sets $F_{\texttt{I5}}$,$F_{\texttt{I5A5}}$ and $F_{\texttt{I5A3}}$.

Selected distance curves including values for distance measure and the true and false hits indicated by green dots (true hits) and blue dots (false hits) are listed in Figure 6.8 at the end of this chapter. For all distance curves, the feature set $F_{\texttt{I5A3}}$ has been used. The ending of each motion document within $\mathcal{D}^{\text{Seq}}$ is indicated by small red dots. One can see that the retrieved positions do not always coincide exactly with the positions that are given in the concatenation information. Within some tolerance region around the document ending, hits are recognized as true hits.

6.3 Conclusion

With the feature evaluation experiments, we gained insight into the characteristics and discriminative quality of every motion class. Furthermore, the discriminative power of all given feature representations has been determined. The main results are as followed.

A combined feature set is the best to discriminate all motion classes from each other as by the combination of the inclination features with angular-based feature sets, motion classes that differ in angles between the up vector and the sensor's X-axis and motion classes that differ in angles between the bones of extremities can be distinguished.

By suitable feature representations, dissimilar motion classes can be distinguished better. We can fix suitable feature representations in the feature evaluation step, but cannot diminish the similarity costs within all motion classes.

For many motion classes, individual differences within the motion style of each actor become visible. In general, each actor is recognizable by it's individual technique that slightly differs to the motion style of other actors. For example, the position of the arms can vary at the beginning and the ending of each move. The athlete can start with the hands pointing upwards or help to increase it's inertial moment by moving the arms from upwards from the side of the body. The amount of momentum transfer from the trampoline bed to the body can vary as well as the timing of moves and flight phases itself. Due to natural restrictions and individual proportions of the athlete's body, the shape of the moves will always slightly differ among different actors resulting in different motion styles and shapes. How to cope with those variations will be discussed in the next chapter.

As the $F_{\texttt{I5A3}}$ feature set turned out to be the best discriminative feature set of all tested feature sets, it will be used for all following experiments.

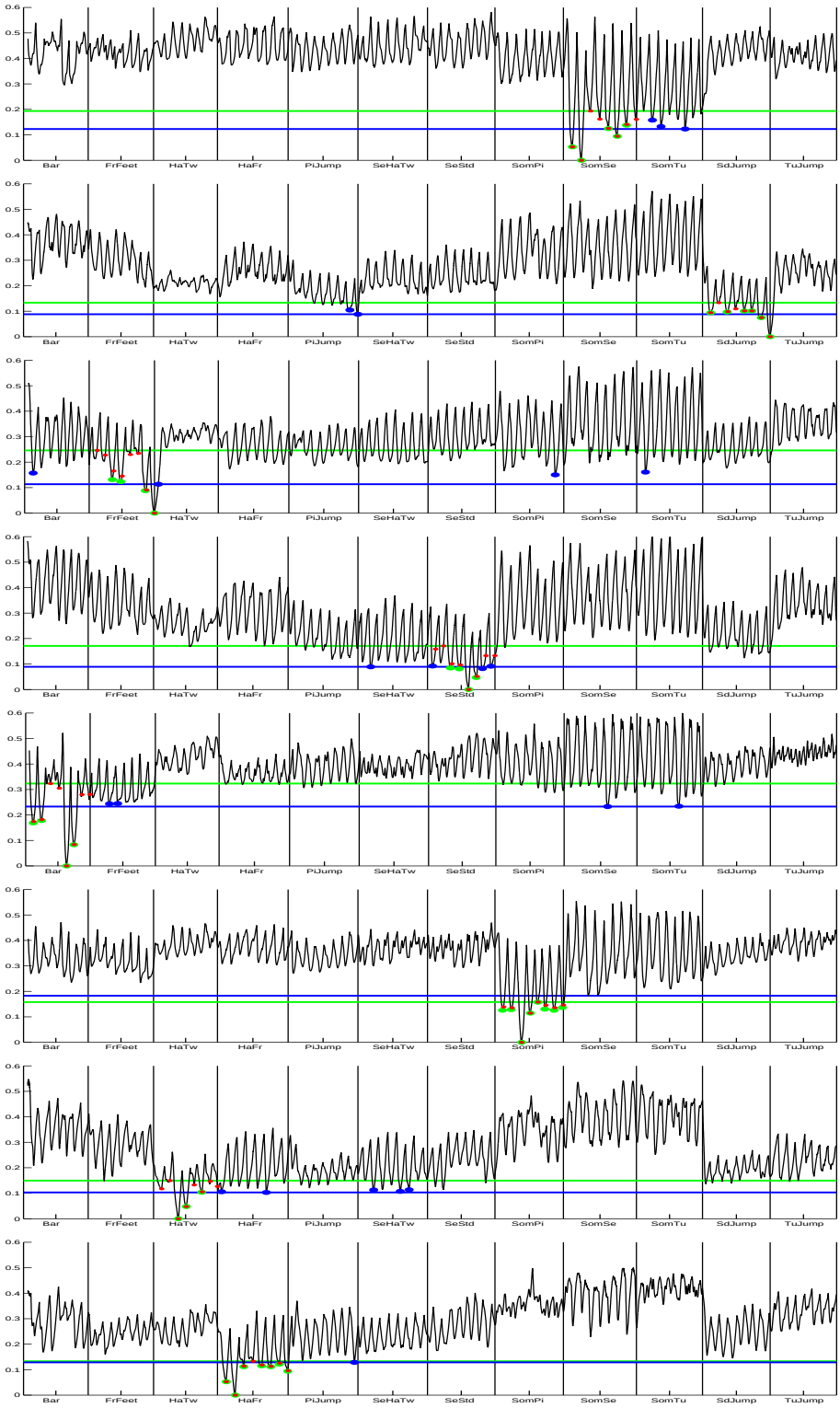

Figure 6.8. Distance curves for selected motion classes and actors. The true hits are indicated by green dots, false hits by blue dots and annotated document ends for the queried motion class by red dots. The green line represents the maximal cost value $\max_T^X$ within all true hits, the blue line represents the minimum cost value $\min_F^X$ within all false hits.

Chapter 7

Motion Template Evaluation

To classify a feature sequence, we compare it to the previously computed class MTs that are averaged feature matrices of sample jumps from one motion class. The MTs represent the main characteristics of a motion class and lead to low costs in the distance measure if compared to a jump of the same motion class so that the jump can be assigned the correct motion class label. However, MTs can still not disregard actor-individual style variations within the feature sequences leading to high costs in the distance measures. We have seen that the trampoline jumps within one motion class undergo style variations that cannot be addressed by the methods we used so far and that huge cost differences between jumps from different actors occur. Especially the motion of the upper extremities turned out to be very variant among all actors and even within the same actor. To make the classification result invariant against style variations, we have to use information from the MT computation step. By this information, we can mask out all those regions within the feature sequence that are expected to be subject to highest variation. Then, the cost of all true hits will be as close as possible, ignoring outliers in motion style so that the classification should become more stable and precise.

In Section 7.1, we first give an introduction into the use and purpose of motion templates as averaged feature matrices. We will see why MTs are suitable as class representations. In Section 7.2, we then introduce evaluation measures that enable to quantify the results. In Section 7.3, the general method of masking MTs to become invariant against actor-specific variations is described. A further method of intelligent MT masking to handle performance variations is then introduced in Section 7.4. Finally, Section 7.5 concludes the chapter.

7.1 Usage of Motion Templates

As a first introductive experiment into the MT concept, we want to show the effect of MTs. For this, we use subsequence DTW with the same retrieval algorithm than in Section 6.2. We use $\mathcal{D}^{\mathrm{Seq}}$ as long feature sequence Y, while all class MT feature matrices are used as query X. To compute the MTs, $\mathcal{D}^{\mathrm{C2}}$ is used that contains feature representations of two sample jumps per Actor and motion class. As for the computation of every motion class

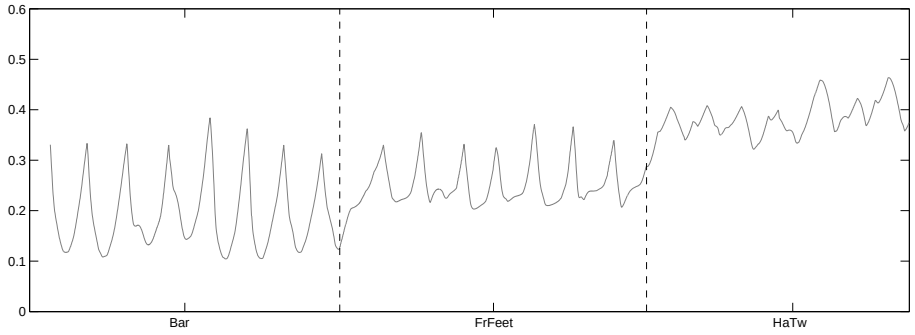

Figure 7.1. Distance curve for the query MT of motion class `BAR`. The cost differences between all true hits is reduced in comparison to Figure 6.5.

template jumps of all four actors have been used, actor-specific variations are averaged and hence their influence on the distance measure is reduced. As a result, the subsequence retrieval results will improve. To be more precise, the distances for a comparison between X and Y among all jumps of the same motion class will approach approximately the same cost level. Then, also the number of false hits among the eight first retrieval hits can be decreased. For a quantitative evaluation, the previously described distance measures α, β and γ will be used again, see Section 6.2.

7.1.1 Results and Discussion

Using the MT as query feature sequence X, the costs of all true hits approach the same cost. We can see in Figure 7.1 that the distance curve for all jumps of the queried motion class `BAR` is on nearly the same level (and so of similar cost).

One can see that in comparison to the subsequence retrieval distance curve from Figure 6.5, the cost differences between all true reduced. As the query motion is standardized and not subject to variations in motion style, only the essence of the motion description has been grasped. Those semantic characteristics will be found in all feature representations of the same motion class and therefore the retrieval cost for all jumps is at similar cost level. On the other hand, as the query feature sequence is not contained in $\mathcal{D}^{\mathrm{C1}}$, the ideal cost of the match will not occur any more. In the next section, we will introduce evaluation measures and quantify the results.

Note that the retrieval results became better, but are still subject to variations that occur in motion documents contained in $\mathcal{D}^{\mathrm{Seq}}$. Furthermore, the retrieval cost for true hits is high for some motion classes.

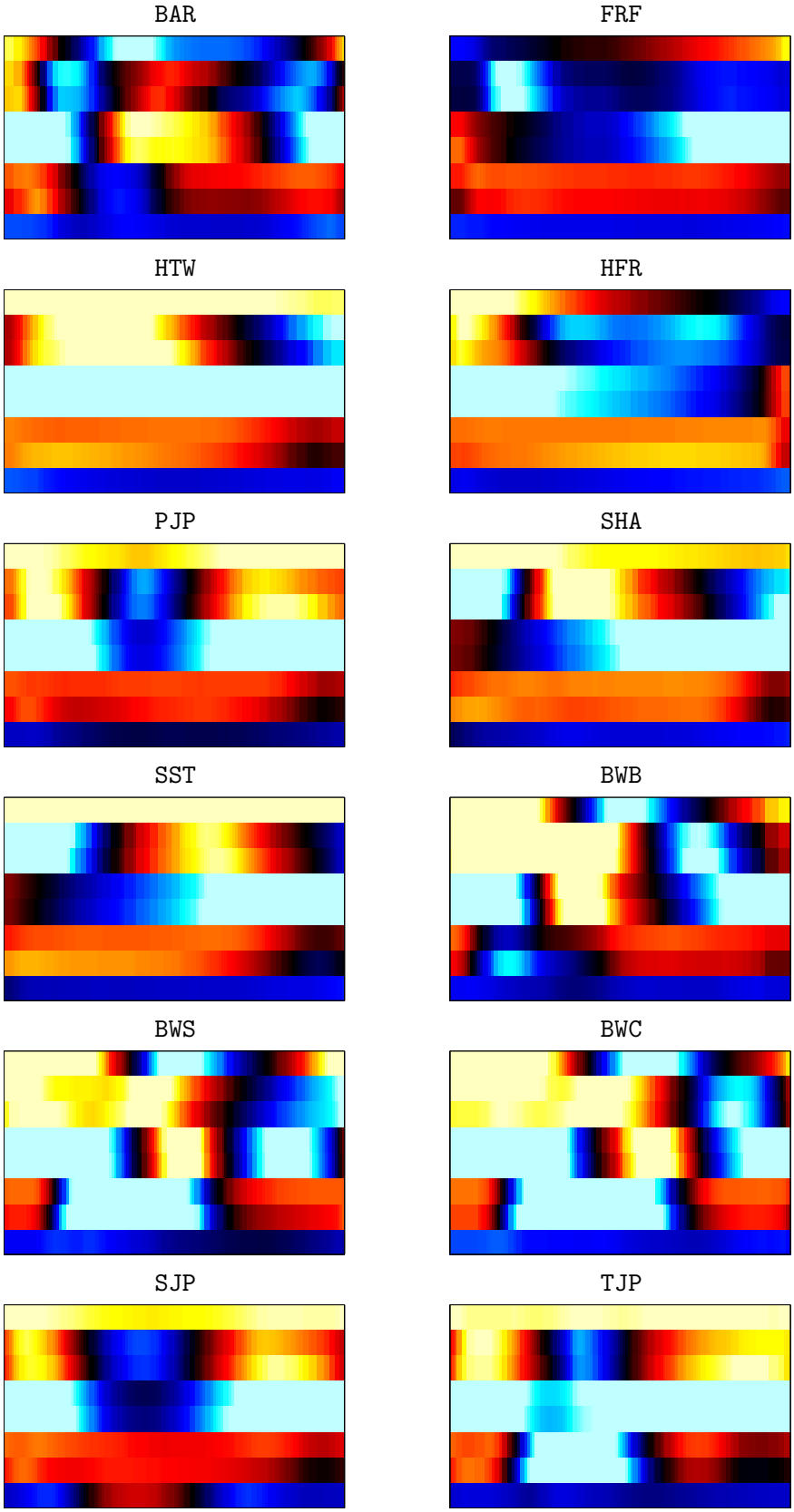

Figure 7.2. Normalized motion templates of all motion classes as given for the similarity measure in alphabetical order from left to right and top to bottom.

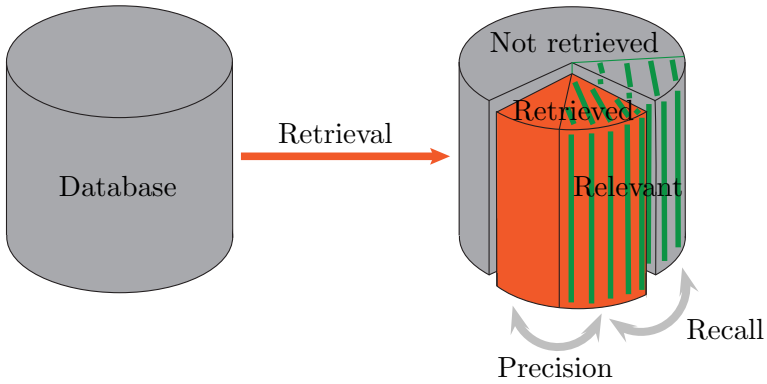

Figure 7.3. Schematic view of precision $\mathcal{P}$ and recall $\mathcal{R}$ for information retrieval. $\mathcal{P}$ is determined by the number of relevant documents within the retrieval results whereas $\mathcal{R}$ is determined by the number of retrieved relevant documents within the number of all relevant documents inside the database.

7.2 Precision and Recall as Evaluation Measures

We have seen that we can improve invariance against spatial variations by better suited feature sets. We now want to introduce evaluation metrics that help to quantify improvements in invariance against actor-individual variations on base of subsequence retrieval methods.

7.2.1 Precision and Recall

Precision $\mathcal{P}$ and *Recall* $\mathcal{R}$ are measures to evaluate pattern recognition algorithms and information retrieval methods. They are defined in terms of a set of retrieved documents with relevant and irrelevant documents as schematically shown in Figure 7.3. In our subsequence retrieval, as described in Section 6.2, the set of relevant documents contained in $\mathcal{D}^{\text{Seq}}$ has a size of eight for every queried motion class. Here, a true hit means that a retrieved document is relevant. Analogously, irrelevant retrieved documents are called false hits.

Let k, $k \in [1:K]$ be the rank of a given match, where K is the maximum rank (in our case $K = 96$). Now, for every k, Precision $\mathcal{P}$ is defined as the number of true hits over the number of all retrieved documents by $\mathcal{P}_k := |T \cap M_k|/|M_k|$ and Recall $\mathcal{R}$ is defined as the number of true hits over the number of all relevant documents that are contained in the database by $\mathcal{R}_k := |T \cap M_k|/|T|$. Here, M_k is the set of all matches up to rank k and T the set of all relevant documents (in our case $|T| = 8$).

In an excellent retrieval scenario, $\mathcal{P}$ will be one (which means 100%) and $\mathcal{R}$ will be one after $k = |T|$ retrieval steps.

7.2.2 Evaluation with Precision and Recall

Combining Precision and Recall by the harmonic mean yields another metric to measure the accuracy of the retrieval, the (standard) *F-measure* that is defined as $F_k := 2 \cdot P_k \cdot$

$R_k/(P_k+R_k)$. From the F-measure, we can determine the maximum F-measure that is defined as $F := \max F_k, k \in [1:K]$.

With the F-measure, we can quantify the accuracy of retrieval results. Table 7.1 shows the average maximum F-measure for subsequence retrieval without MT as query motion X and with MT as query motion X for selected feature sets that confirm the feature evaluation results from Chapter 6. To guarantee that the results will be comparable, we use all motions within $\mathcal{D}^{\mathrm{C2}}$ that are also contained in the averaged MT as query motions for the subsequence retrieval from now on.

Feature Set	F-measure	F-measure with MT
$F_{\mathtt{I5}}$	0.617	0.854
$F_{\mathtt{I5A5}}$	0.651	0.927
$F_{\mathtt{I5A3}}$	0.680	0.906

Table 7.1. Averaged maximum F-measure for selected feature sets for subsequence retrieval with and without MT as query motion X.

One can see that using MTs as query improves the average F-measure as we have already illustrated. One can also see that the $F_{\mathtt{I5A3}}$ feature set that has been chosen as best feature set has a good averaged F-measure.

We will now have a closer look on how to evaluate retrieval results by Precision and Recall. We therefore store the values for precision and recall for the first $K = 96$ hits (which is the length of the database) in an additional data structure. Then, we can plot both values for every $k \in K$ in a *Precision-Recall-diagram* (PR-diagram). Figure 7.4 shows the PR-diagram for selected motion classes where precision is signed on the y-axis and recall is signed on the x-axis. In the upper row, averaged PR-diagrams over all precision and recall values for the retrieval process with all feature representations within $\mathcal{D}^{\mathrm{C2}}$ is presented. In the bottom row, PR-diagrams for a retrieval with MTs is plotted.

In Figure 7.4, the PR-diagrams for selected motion classes are shown. Those motion classes represent the results significantly. The PR-diagrams for all other motion classes appear in a similar way. For all motion classes, one can see that the retrieval results improve by using MTs.

In the ideal retrieval scenario, the number of retrieval steps to obtain the maximum F-measure will be the number of similar motions contained in the database $\mathcal{D}^{\mathrm{Seq}}$ (and the maximal F-measure can easily be detected in the PR-diagrams as the edge at the plot's orthogonal shape then).

7.2.3 Confusion Matrices

With the definition of $\mathcal{P}$ and $\mathcal{R}$, we now want to examine how motion classes are confused with each other with regard to a given query feature matrix. Here, we count the number of true hits n_{MC} until the position k that yields the maximum F-measure (which can vary for different motion classes). Then, we normalize n_{MC} by the number of retrieval steps k and store the values in a *confusion matrix* M_C for every motion class that is visualized

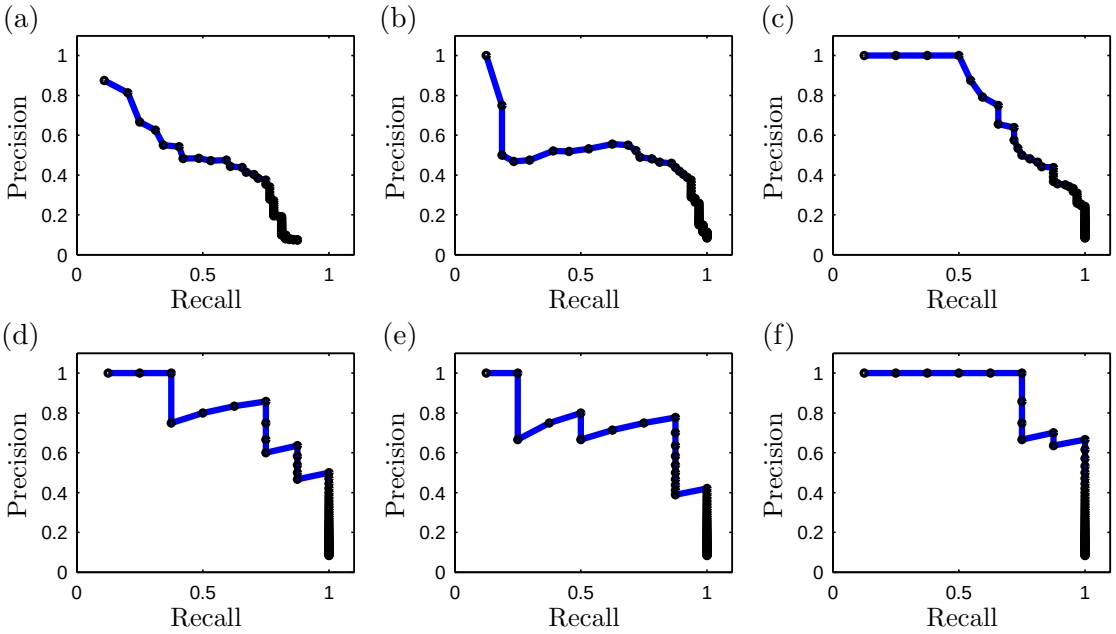

Figure 7.4. PR-diagrams for selected motion classes `SST` (left), `PJP` (middle) and `BAR`(right). In the upper row, averaged PR-diagrams for subsequence retrieval can be found, in the bottom row PR-diagrams for subsequence retrieval with a query MT can be found.

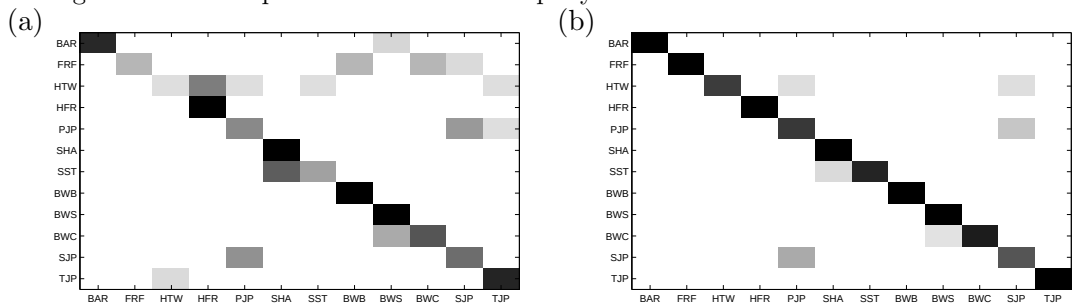

Figure 7.5. (a) Confusion Matrix for subsequence retrieval without MT. (b) Confusion Matrix for subsequence retrieval with MT.

in Figure 7.5. The rows of a confusion matrix represent the motion classes of the query feature sequence X, whereas the columns represent the motion classes of the match in the feature sequence Y. Dark entries indicate a high percentage of the retrieved motion class among all hits, whereas light colors indicate a low percentage.

The confusion matrix yields a good impression on how the motion classes are mixed up under different query matrices. For example, one can see that for the subsequence retrieval with MT query feature sequence, less motion classes are confused than for the subsequence retrieval without MT.

All confusion matrices show that there are several motion classes that can be hardly be discriminated and will be confused to other motions with a high probability. Especially all those motion classes that have a high semantic similarity such as `PJP` and `SJP` will be confused with each other as it has already been shown in Chapter 6. Motion classes that only differ by their landing positions like `BWC` and `BWS` may be confused as well as motion

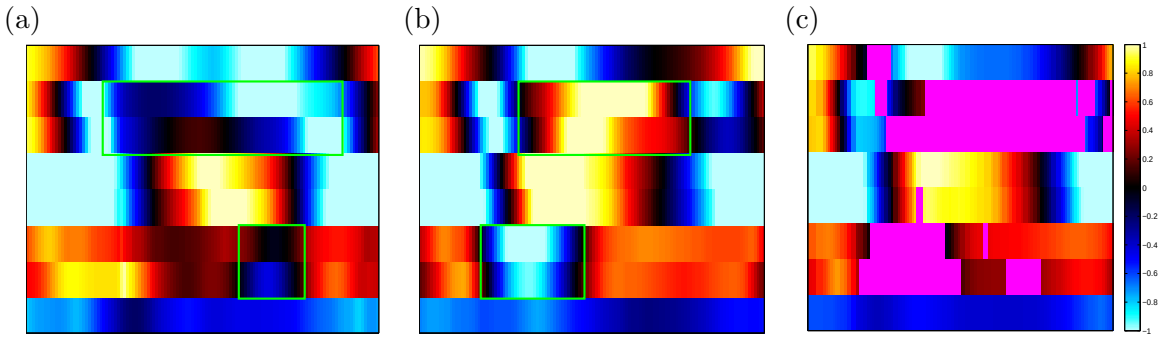

Figure 7.6. Variant feature matrices (a),(b) of a barani jump and the AQM (c) of the MT shown in Figure 4.9. Most variant regions indicated by green boxes are marked out.

classes that only differ in twist motions like `SST` and `SHA`. However, for all confused motion classes, the results are not surprising as the semantical differences between those motion classes are very small.

7.3 Dealing with Style Variations

We have seen that the results of the subsequence retrieval experiments got better by using MTs, but are still affected by actor-individual variations from the motion documents within $\mathcal{D}^{C2}$ so that the cost for all true hits have been relatively high. Variations of jumps within $\mathcal{D}^{C2}$ can raise the cost of a comparison between the averaged MT and a passage within Y of the same motion class. As actor-individual style variations are also expected to occur during motion classification in the unknown data stream, we now want to diminish the influence of those variations.

The idea is that for all trampoline moves, variations usually occur in special temporal regions. For trampolining, those regions are usually at points in time where each actor is free to move without restrictions by biomechanical and physical properties. For example, we expect those regions to be at the beginning and the ending of the motion performance as each trampolinist has it's own individual style of starting a motion and moving the arms during the contact phase with the trampoline bed. Further possible regions for large variations can also be during the flight phase like such as for the `BAR`, where every actor is free to chose the jump's motion shape. Those temporal regions can then be left unconsidered in the similarity measure while comparing the MT to an unknown motion document.

For this, we use the information we got out of the MT computation. Remember that we stored the information on the standard deviation in the feature matrices of all MTs for every frame in an additional data structure. Now, we can identify the most variant regions and mark them as irrelevant for the distance measures. In our case, we define the most variant 25% of all single values within every MT to be the *automated quartile mask* (AQM) for the corresponding MT.

Figure 7.6 shows the masking of the most variant 25% of all feature values of the `BAR` template that has already been shown in Figure 4.9. The masked regions are positioned at the significantly different feature matrix entries indicated by the green boxes.

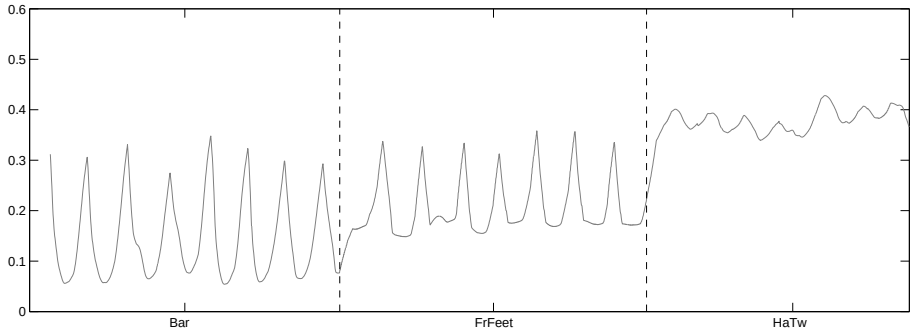

Figure 7.7. Distance curve for motion class `BAR` with AQM query MT. The cost differences among all true hits is reduced in comparison to Figure 6.5 and Figure 7.1.

7.3.1 Results and Discussion

Using AQM MTs, we do only consider those regions within the query MT that are almost constant within all training jumps. In other words, we only consider those parts that are the same for all jumps and should therefore probably contain relevant information that describes the motion semantically. We will now evaluate the improvement on a subsequence retrieval with AQM MTs.

We will now see in Figure 7.7 how the distances for motions of the same motion class improved by the same `BAR` distance curve as in Figure 6.5 and Figure 7.1. One can see that we succeeded to equalize the cost of all true hits but also to reduce the average cost to retrieve the relevant motions.

In a quantitative evaluation with the quality measures α, β and γ it also becomes clear that using a AQM MT as query feature sequence X improves the retrieval results. A comparison of subsequence retrieval and subsequence retrieval with a AQM MT can be found in Table 7.2.

μ_T	μ_F	α	μ_T	$\mu_F^{20\%}$	β	max_T	min_F	γ
0.049	0.240	0.204	0.049	0.143	0.357	0.068	0.072	1.049

Table 7.2. Averaged distance measures over all AQM query MTs for the $F_{\texttt{I5A5}}$. The measures got better than for the subequence retrieval without MT masking and confirm the graphical results.

In the evaluation of Section 6.2, we saw that for some of the basic motion classes, the local cost minimum is not found at the annotated document ends. By using a MT with an AQM as query motion, the problem of spatial shift of the local minimum is minimized and remains within the defined tolerance region so that all relevant motions can be retrieved as true hits first. Figure 7.8 shows the distance curve for the query motion class `PJP`. Other than for the subsequence retrieval curve, the shift of the local minimum is so small that it remains within a defined tolerance region for the AQM MT query sequence. We suppose this is due to the fact that the masking of the `PJP` MT is mainly located in the landing and takeoff phases close to the contact phase that are subject to many variations and depending extremely on the accuracy and quality of the performed jump.

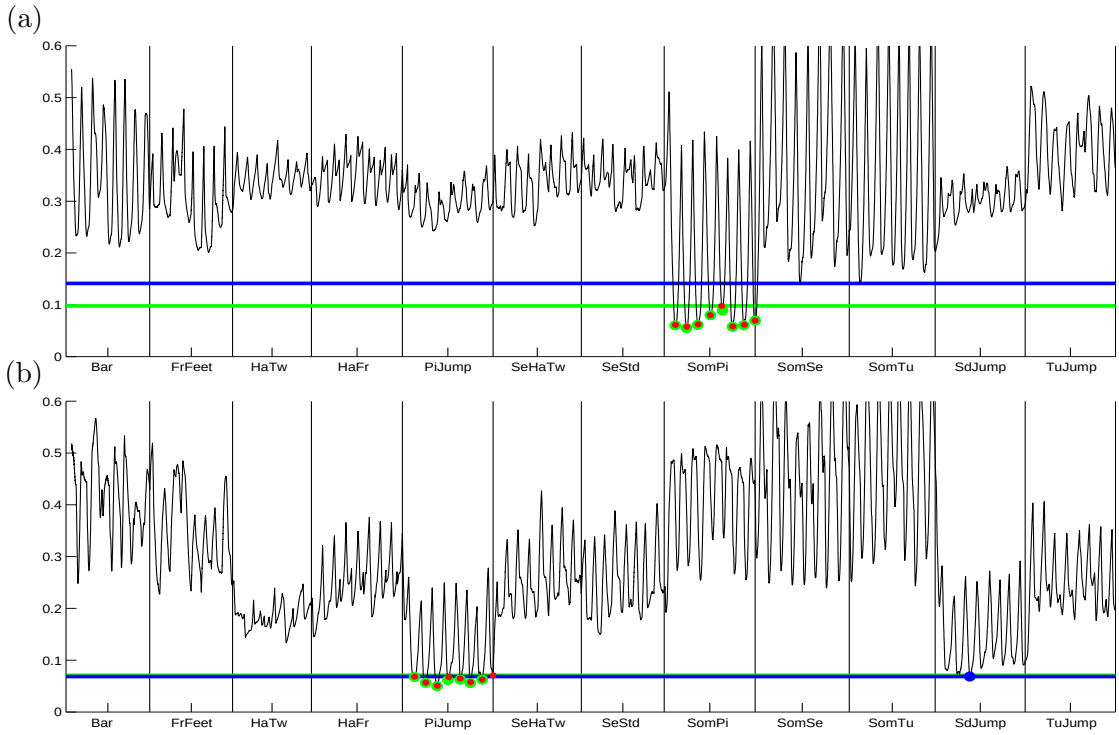

Figure 7.8. Distance curves using AQM MTs for subsequence retrieval. (a) In subfigure (a), the masked MT of motion class `BWB` was used as query, whereas in subfigure(b) the masked MT of motion class `PJP` was used. The number of true hits raises while the number of false hits diminishes. Better distance measures are displayed by higher cost differences between jumps of the same motion class than the query MT and motions from other motion classes, lower costs for $\max_T^X$ and higher costs for $\min_F^X$. In (b), The positions of a local minimum are located closer to the annotated document endings.

The distance curve in Figure 7.8 shows that by masking a MT, also differences in motion styles can be left out of consideration in the retrieval process. Then, the same holds for classification and the identification of similar motions is improved.

Figure 7.8 shows that not only the number of true hits raises while the number of false hits diminishes, but also that the distance between all motions from one motion class to all other motion classes becomes larger so that the discrimination between different motion classes becomes more accurate. One can also see that the relation between $\max_T^X$ and $\min_F^X$ gets better so that both values approaches or so that $\max_T^X$ even falls below $\min_F^X$ as it is expected to be for the ideal case.

Again, selected distance curves for each MT query including values for distance measure and the true and false hits indicated by green (true hits) and blue dots (false hits) are listed in Figure 7.12. The hits within each motion class are of more similar cost than the curves in Figure 6.8 and the evaluation measure are of better value.

For further evaluation, we will now use the newly introduced measures $\mathcal{P}$ and $\mathcal{R}$. In the PR-diagram from Figure 7.9, one can see that with using a masked MT, the values for Precision and Recall further improve. Again, only some selected motion classes will be plotted that stand for the results for all motion classes.

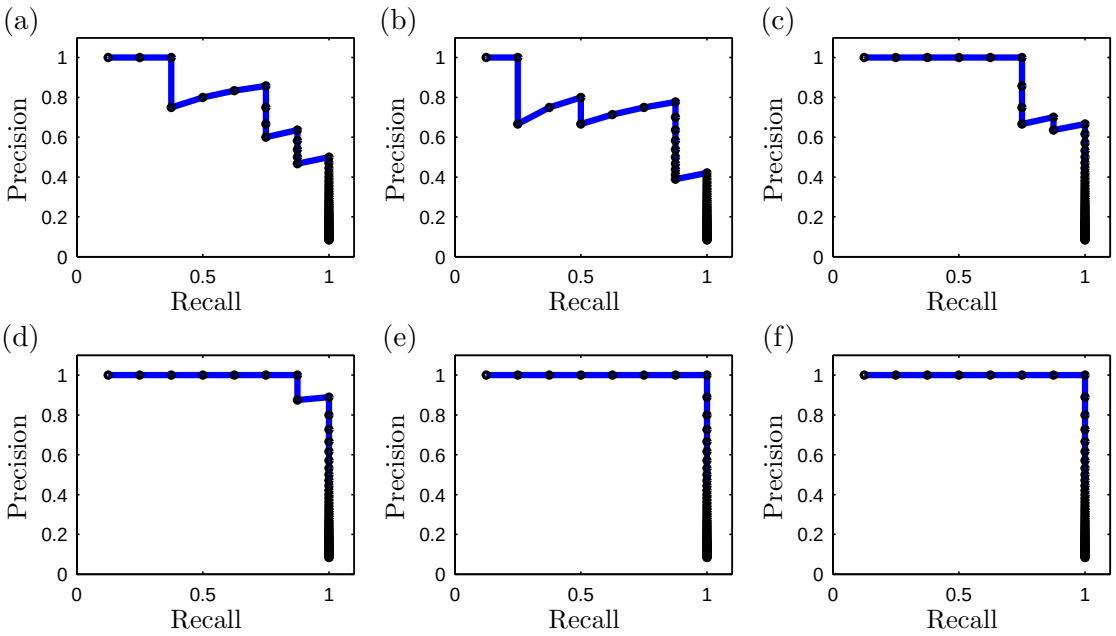

Figure 7.9. PR-diagrams for selected motion classes `SST` (left), `PJP` (middle) and `BAR`(right). In the upper row, PR-diagrams for subsequence retrieval with a query MT can be found, in the bottom row PR-diagrams for subsequence retrieval with a AQM query MT can be found.

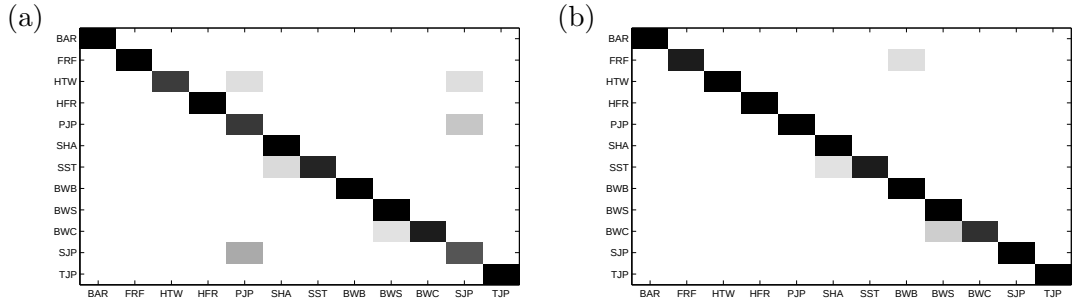

Figure 7.10. (a) Confusion Matrix for subsequence retrieval with MT. (b) Confusion Matrix for subsequence retrieval with AQM MT. Less motion classes are confused with the queried motion class and the number of false hits diminished.

If we visualize the retrieval results in a confusion matrix, the improvement of the retrieval results can be affirmed as in Figure 7.10. Nevertheless, for the confusion matrices, one can see that the `PJP`, `TJP` and `SJP` jumps are still sensitive to confusion as they do not contain enough semantically characteristic information.

As a result, we can confirm that for most motion classes, the used masking method offers satisfying and stable retrieval results invariant to style variations with the $F_{\texttt{I5A3}}$ feature set. Semantically insignificant motion classes cannot be addressed by the automatic masking method so that we need another masking method for those motion classes.

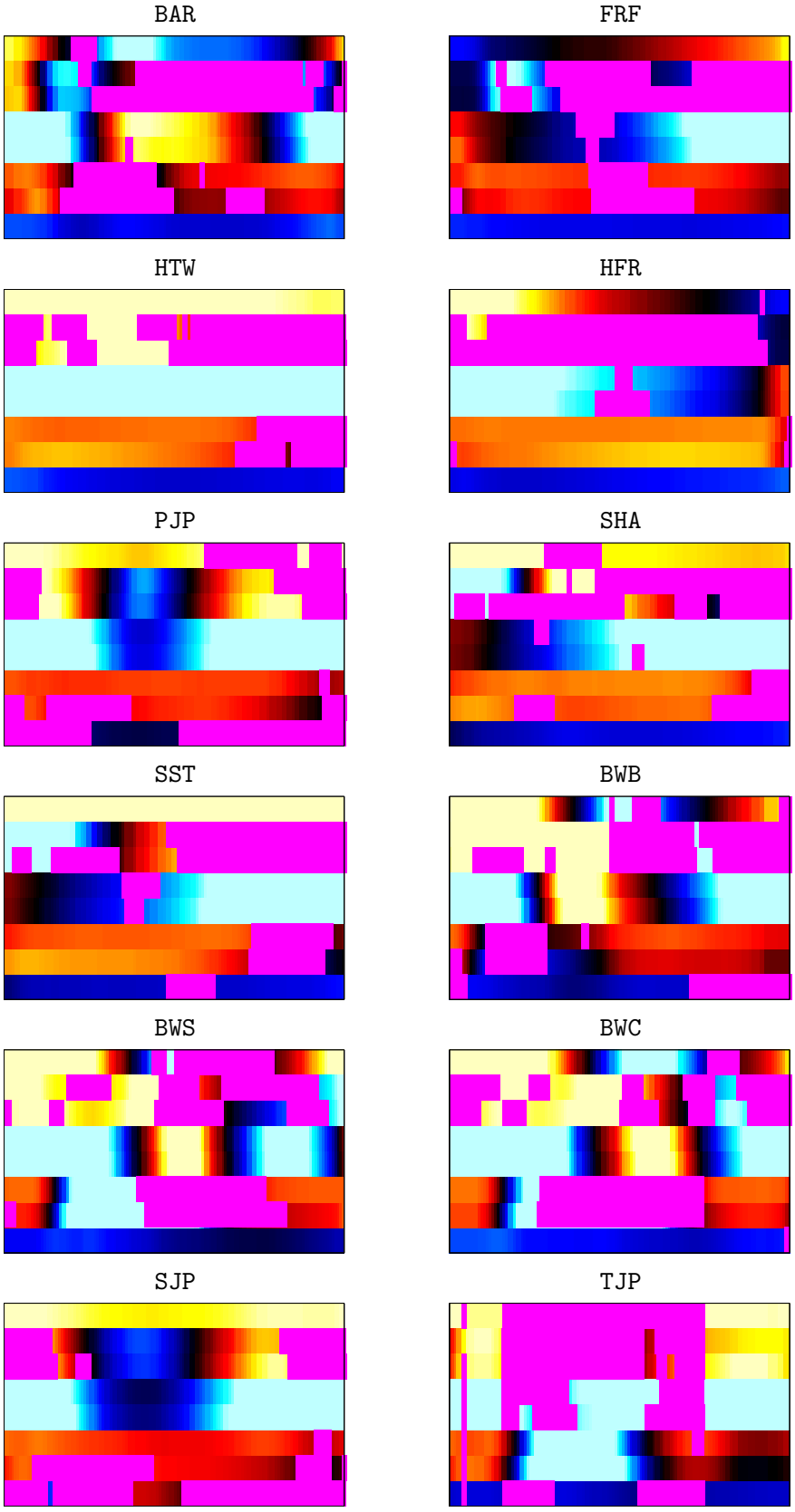

Figure 7.11. Normalized motion templates with AQM for all motion classes in alphabetical order from left to right and top to bottom. Masked regions are positioned at either feature vectors that lie at the beginning and ending of a jump or at the beginning and ending of the flight phase.

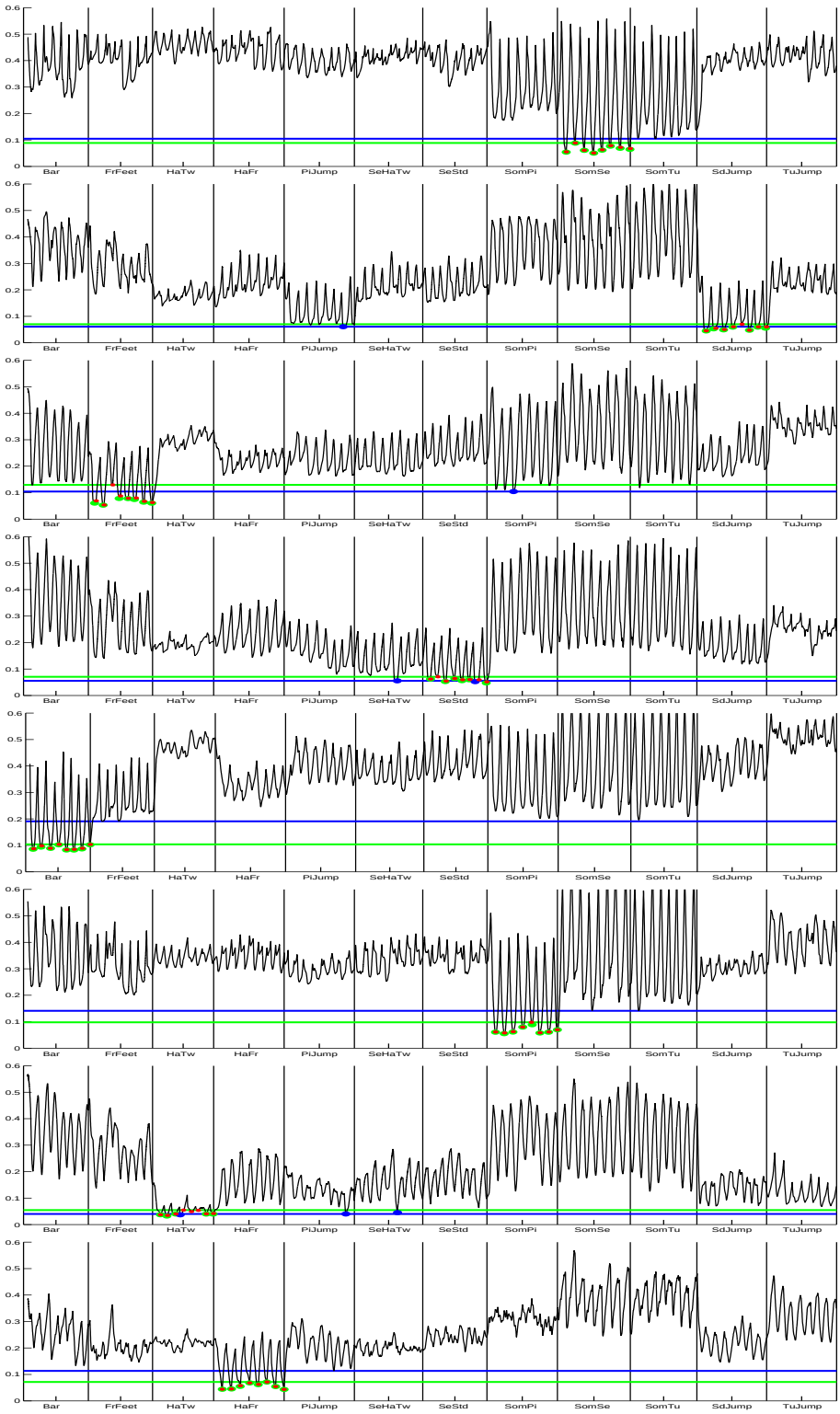

Figure 7.12. Distance curves for AQM MT queries of selected motion classes. The true hits are indicated by green dots, false hits by blue dots and annotated document ends for the queried motion class by red dots. The green line represents the maximal cost value $\max_T^X$ within all true hits, the blue line represents the minimum cost value $\min_F^X$ within all false hits.

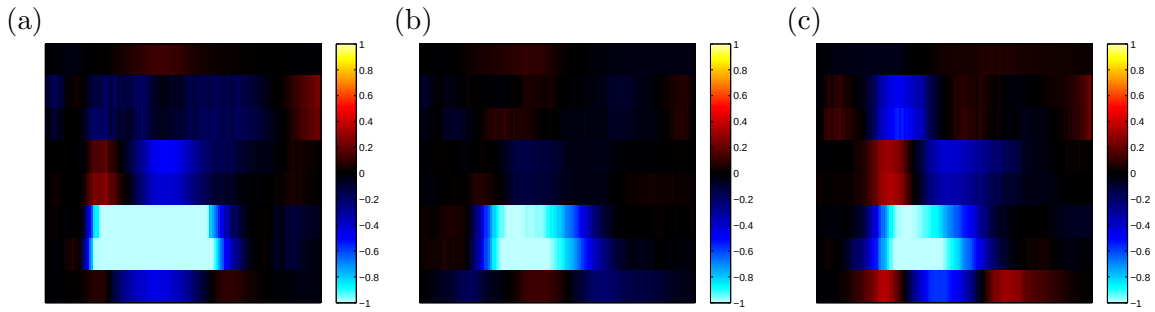

Figure 7.13. Matrices containing information on the differences between semantically similar motion classes for intelligent masking. (a) Difference matrix for `TJP` jumps. (b) Difference matrix for `PJP` jumps. (c) Difference matrix for `SJP` jumps.

7.4 Intelligent Masking for Morphologically Similar Moves

We have seen that it is especially difficult to discriminate semantically less significant and similar motion classes such as `PJP`, `TJP` and `SJP`. In this section, we investigate how to alter the masking algorithm for a better discrimination among those motion classes. Possible methods point out the potential evolution towards generic algorithms.

Here, the main idea is to use expert knowledge about every motion class to grasp the relevant information contained in the feature representations in a more efficient way. In particular, the influence of all unique motion class characteristics on the similarity measure is amplified. This means that for every frame which contains characteristic information in at least one feature vector entry, we mask all feature vector entries that do not contain discriminating information. Consequently, only the unmasked feature entries containing discriminative information are respected for the similarity measure.

From morphologic considerations, we know that the motion classes `PJP`, `TJP` and `SJP` only differ by angular changes in the legs during the flight phase's apex representing the different standard motion shapes. We now want to learn this knowledge for a better motion class discrimination.

To transfer the expert knowledge into numerical knowledge that can be used automatically, we compute an averaged feature matrix over all three MTs. By subtracting the matrices of the MTs from the averaged feature matrix, we get three so called difference matrices for every motion class. To this end, we first have to warp all templates to the same length using DTW. The difference matrices contain information about all discriminative and unique regions within the feature representation for every motion class. For the difference matrices, we then define a threshold t that marks the $t\%$ most differing matrix entries of highest absolute value. Figure 7.13 shows the difference matrices for all three motion classes.

Using the thresholded difference information, we can create a new MT mask for the three affected motion classes. Here, for all most variant frames within the class MT we exclude all thresholded relevant feature vectors and mask all remaining feature vector entries. This means for example that for the `TJP` jump, only the two rows representing F_{13} and F_{14} within the feature matrix remain unmasked in all frames where the angle between

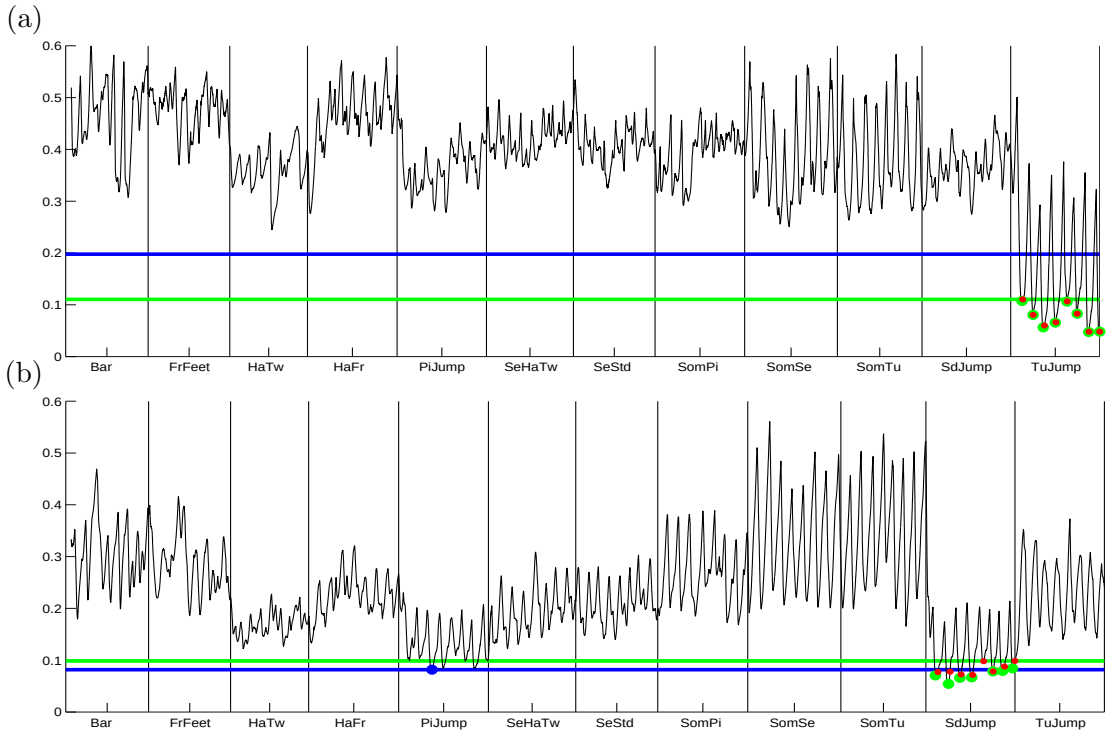

Figure 7.14. Retrieval results with IM using expert knowledge. (a) The masking works fine to discriminate the `TJP` motion class. (b) The masking did not yield significant improvements for the `PJP` and `SJP` motion class so that the moves are still confused with each other.

upper and lower leg is significant. As this mask uses some trained expert knowledge about the morphologic differences between the affected motion classes, we call it *intelligent mask* IM.

For evaluation of the effects of the new MT masks, we will repeat the subsequence retrieval scenario with the IM. Figure 7.14 shows the results for `TJP` and `SJP` motion class query MT (the results for the `PJP` class MT are very similar to the `SJP` class MT). One can see that for `TJP` jump, the masking method works very well as the distance for all true hits differs significantly to the costs for all other motion classes. For the `SJP` and `PJP` motion class, on the other hand, no definite improvement could be achieved. A probable explanation for this could be that for the two motion classes, only one feature (F_{15}) differs over relatively few frames (and hence over a short motion time). Those tiny differences will not influence the similarity measurements sufficiently.

Finding the right threshold that defines the most variant regions within every difference matrix is a difficult problem. If too many parts of the MT will be masked, even other motion classes that have not been investigated for the intelligent masking and are semantically unique can no longer be discriminated from different motion classes (like for example from the `HTW` motion class) as too many parts of the feature matrix that characterize the motion class will be masked out. On the other hand, if we mark too less regions, differences cannot be identified and the motion classes will be mixed up. In Figure 7.15, we can see the problem of 'overmasking' where the `HTW` motion class became similar to the masked query `PJP` class MT leading to false hits. For our experiments, we use $t = 25\%$ as

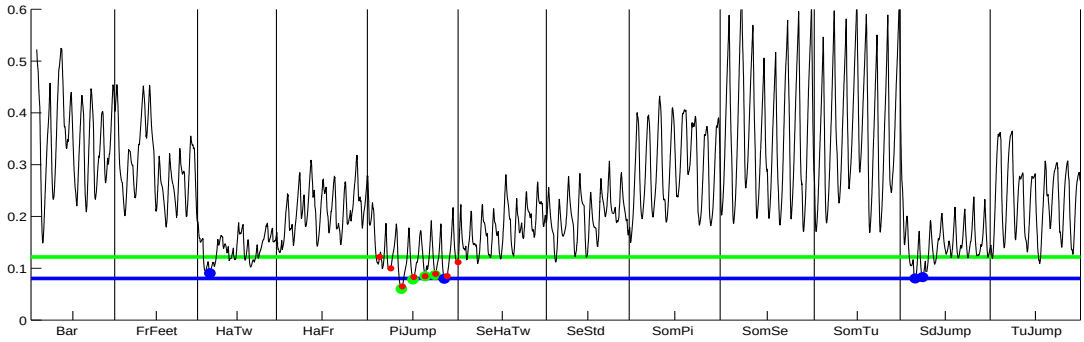

Figure 7.15. Retrieval result for a masked PJP class MT. As the threshold for masking has been inappropriate, the HTW can hardly be distinguished from the query motion class leading to false hits.

treshold to select regions of the most variant values which turned out to work well.

For the most precise masking, it would be necessary to use two levels of computation of different granularities: the first one which retrieves semantically different motion classes by the AQM MT and the second more precise one that retrieves all those motions that are semantically similar and re-masks them with the IM algorithm then.

Another method to calculate a discriminative mask would be to use 'negative' information as discriminator, whereas, negative information means information from other motion classes. Those motion classes do not contain the wanted discriminating characteristics. If we then subtract the negative information from the averaged feature matrix, one can again identify all regions of highest difference for masking the feature vector entries that are not part of those regions. In our experiments, the usage of negative information did not yield better results, so that this masking method will not been used for the consecutive classification experiments. To get useful results, further experiments that define threshold and other parameters will be necessary.

As it was already mentioned, the methods introduced are first algorithmic steps to control and influence the similarity measures between semantical similar motion classes by expert knowledge. For further exact and fully-automatic masking of the class MT, generic algorithms would be necessary. They have not been investigated for this thesis and can be examined in future work.

7.5 Conclusion

We have seen that masking all regions of high variance as irrelevant for the similarity measure improves the invariance against actor-specific variations. We have also seen that for special semantically insignificant motion classes, the AQM method where we exclude the most variant $P\%$ of every motion class MT does not suffice. Here, we need more sophisticated masking methods that tranform morphological expert knowledge into numerical information. However, it is very difficult to transfer this knowledge in a significant way. This means that those algorithms have to be further investigated and evolved in future work.

Nevertheless, during the experiments that were described in the last two chapters, we have fixed very discriminative and significant methods that enable a stable and satisfying classification.

Chapter 8

Classification Evaluation

Using the best suited feature set as it has been determined in previous experiments and the masked motion templates that are invariant to motion variations, we will now test the accuracy of the developed classification method in various experiments. Here, we use the DTW-based similarity measures introduced in Chapter 4 and evaluate their accuracy with respect to the jump labeling.

In Section 8.1, the jumps in the routines were manually segmented using the existing annotations and the classification was conducted using global DTW. A more practical scenario is described in Section 8.2 where we use DTW on the unsegmented routines. As further step, an automatic segmentation algorithm is introduced that automatically segments the routines into single jumps in Section 8.3. These jumps are then again classified using global DTW. In Section 8.4, the previously described classification scenarios are compared. Finally, Section 8.5 ends the chapter with a conclusion.

8.1 Document-based Classification

As in the experiments from Chapter 6, we start with some classification tests that use global DTW as similarity measure for classification in an controlled environment using segmented jumps.

During the capture process, different routines have been recorded that will be used as unknown data streams for classification now, see Section 5.1. The jumps from the routines are not included in the test and training databases $\mathcal{D}^{\mathrm{C1}}$ and $\mathcal{D}^{\mathrm{C2}}$. One different routine per actor has been chosen out of the captured database, so that overall 13 routines (three routines from actor `hb`, three routines from actor `pm`, three routines from actor `sh` and four routines from actor `sm`) have been available for classification, see Table 8.1. Two routines have been performed by all actors in the same order of jumps (those routines are predefined mandatory routines for intermediate trampoline competitions), whereas (at least) one routine has been a free style routine that contained jumps individually chosen by every actor. So the free style routines also contain motion classes that have not been learned as motion templates. In the following, we will see as which motion class those

ID	Name	Actor	Description
$R1$	L8	hb	HFR, FRF, TJP, BWS, SHA, PJP, BWB, SJP, BWC, BAR
$R2$	Modified L7	hb	BWC, SJP, BWS, SHS, SST, FRD, FRF, HTW, TJP, BAR
$R3$	Freestyle	hb	BWA, HTW, SED, SHS, SHA, PJP, BWB, TJP, BWC, FTW
$R4$	L8	pm	HFR, FRF, TJP, BWS, SHA, PJP, BWB, SJP, BWC, BAR
$R5$	Modified L7	pm	BWC, SJP, BWS, SHS, SST, FRD, FRF, HTW, TJP, BAR
$R6$	Freestyle	pm	3QB, FRF, SJP, BWB, BAR, BWA, TJP, BWS, SST, BWC
$R7$	L8	sh	HFR, FRF, TJP, BWS, SHA, PJP, BWB, SJP, BWC, BAR
$R8$	Modified L7	sh	BWC, SJP, BWS, SHS, SST, FRD, FRF, HTW, TJP, BAR
$R9$	Freestyle	sh	SED, SST, SJP, HTS, SHA, TJP, SED, SHS, SST, HTW
$R10$	L8	sm	HFR, FRF, TJP, BWS, SHA, PJP, BWB, SJP, BWC, BAR
$R11$	Modified L7	sm	BWC, SJP, BWS, SHS, SST, FRD, FRF, HTW, TJP, BAR
$R12$	Freestyle	sm	3QB, FRF, SJP, BWC, BAR, PJP, BWB, TJP, BWS, SHA
$R13$	Freestyle	sm	SED, SST, SJP, HTS, SHA, TJP, SED, SHS, SST, HTW

Table 8.1. Description of the routines used for classification experiments.

unknown motion classes will be labeled.

As a first experiment, we classify each document within the test database $\mathcal{D}^{\mathrm{C1}}$ to can get an overview of how exact each motion class will be classified with the methods chosen to handle variation.

First, we compare each document with the given masked MTs (for the TJP, we used the IM, whereas for all other motion classes, we used the AQM). Remember, that the MTs have been build from the training database $\mathcal{D}^{\mathrm{C2}}$ which is disjoint to $\mathcal{D}^{\mathrm{C1}}$. As a result, we obtain a matrix M_{Cl} that contains the distances for the similarity measures of each pair of document and MT. We can then classify every database document as a specific motion class by assigning a label to it. For this, we have to find the motion class that yields the minimal distance in the similarity measure for each document within M_{Cl} and constitutes the document's label.

For a second experiment, we will classify the additionally captured routine moves in the same way than the test database $\mathcal{D}^{\mathrm{C1}}$. In this scenario, we use the manual annotations of each routine to cut single jumps that can be compared to the MTs. As some of the routines include motion classes that have not been learned as MTs, we furthermore introduce an additional motion class *unknown* UNK for the classification experiments. If the minimal cost for a match overgoes a certain threshold, the document will then be classified as unknown.

8.1.1 Results

Figure 8.1 shows the classification results for the test database $\mathcal{D}^{\mathrm{C1}}$. As the database has been concatenated in alphabetical order, in the ideal case, the classification matrix would only consist of correctly classified motion class blocks. One can see that almost every jump within $\mathcal{D}^{\mathrm{C1}}$ has been classified correctly. Only for the semantically similar motions PJP and SJP that are already known as problematic from the previous experiments, a false classification result occurs. Overall, the results are very good as we can see that in most

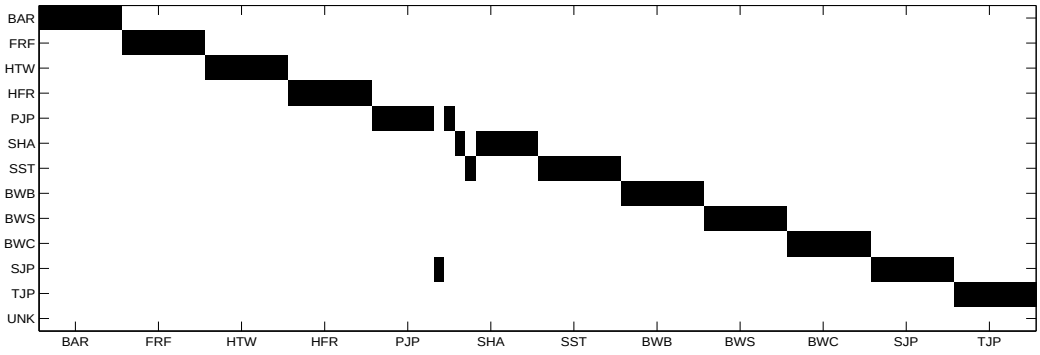

Figure 8.1. Classification results for $\mathcal{D}^{C1}$ with document based classification using global DTW methods. Single jumps have been cut by manual annotations. The classified motion class is plotted on the vertical axis while the annotated documents are depicted along the horizontal axis.

cases, the correct motion class if of lowest cost.

For the routine classifications, the results are of similar quality. Most of the jumps could have been classified correctly with the exception of some jumps from the `PJP` and `SJP` motion class and some jumps inaccurately performed jumps. With respect to the classification of the unknown jumps, we can see that they could not always be identified as unknown. Figure 8.2 shows an example of the Routines $R4$ and $R11$. For $R4$, all jumps that belong to the routine have been classified correctly, whereas for $R11$, the unknown motion classes could not been identified as unknown. The red boxes in the classification plot show the positions and motion classes from the video annotations as ground truth.

As every routine starts and (usually also) ends with straight jumps, we will learn an additional MT for the `STR` motion class in the same way than the other MTs from Section 4.3. Moreover, an additional automatic quartile mask for the `STR` MT is defined to circumvent influences of style variation that will be used for all following classification experiments. For all routines, the classification of the `STR` jumps yields most wrongly labeled motion documents compared to all motion classes, see Figure 8.2. Mislabeling of `STR` jumps especially occur for the motion classes `HTW` and `UNK`, as `STR` jumps do not contain any unique characteristics and many variations. Consequently, the motion class cannot be distinguished from other motion classes as stable as all other motion classes. Especially the `HTW` motion class, that only differs by rotational motion around the longitudinal axis, is semantically very similar. As none of our given feature sets does use rotational information from the rate gyros, both motion classes cannot be discriminated.

Again, finding a correct threshold to label jumps as `UNK` is a difficult design issue that can degrade or improve the classification results considerably. In case of a threshold which is chosen too high, unknown jumps will not be identified as `UNK`. In case of a threshold which is set too low, even known motions (that would probably have been classified correctly) will be classified as `UNK`. For our experiments, we defined a threshold t_{class} in accordance to the minimal cost values that usually appear for a true classification of value $t_{class} = 0.18$.

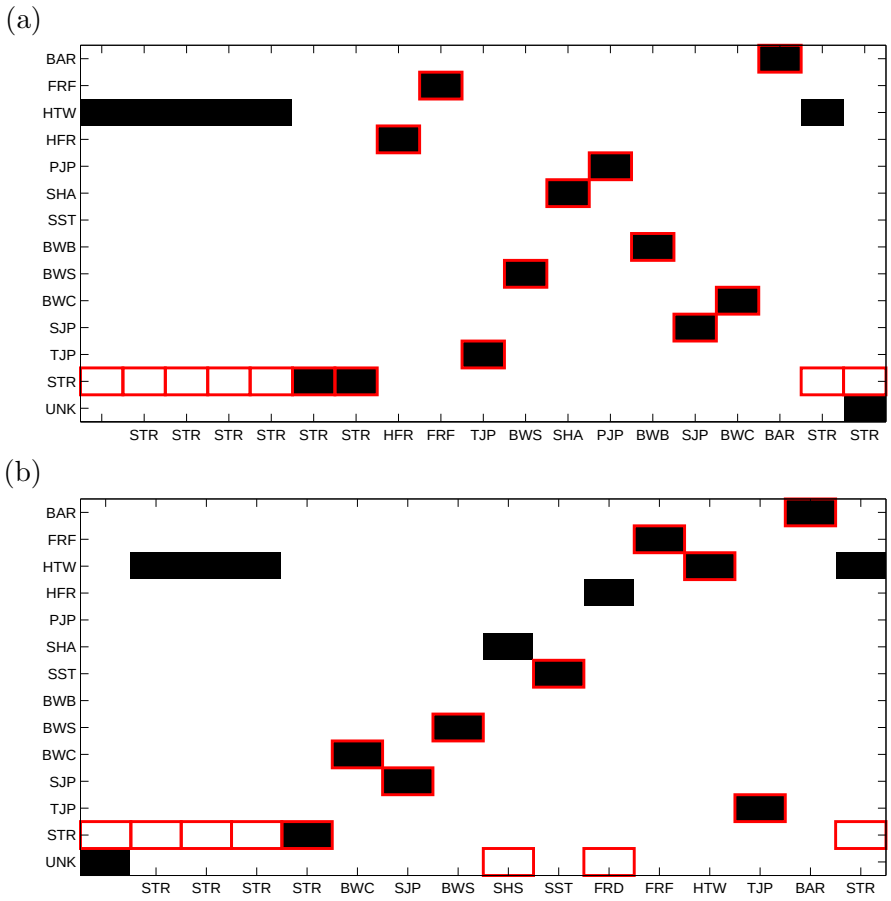

Figure 8.2. Classification results for the Routines $R4$ and $R11$ with document based classification using global DTW. Every jump has been cut using the manual annotations that also work as ground truth (red boxes). (a) All motions from the routine are classified correctly. (b) Unknown motion classes are not labeled as `UNK`, but as different motion class. `STR` jumps are often mislabeled as `HTW` which can hardly be discriminated.

8.2 Subsequence-based Classification

Classifying documents that have already been segmented manually into single jumps is not a very realistic scenario, as usually unknown data streams that occur in a classification scenario will contain more than one motion. We therefore repeat all classification experiments from Section 8.1 with subsequence DTW and the concatenated test database $\mathcal{D}^{\text{Seq}}$ respectively the consecutive routine streams.

Other than for the global classification, we do not obtain cost matrices, but distance curves. Additionally, we store the warping paths that will be obtained from the similarity measure in an own data structure. To classify the motions within the motion stream, we calculate a minimal distance curve DC_{min} over all distance curves that are obtained by subsequence DTW using the various MTs as queries.

We then find the position of absolute minimal cost within DC_{min} and identify the motion class respectively the MT this minimal cost belongs to. With the computed warping paths, we can define all frames within the motion stream that will be labeled by the same

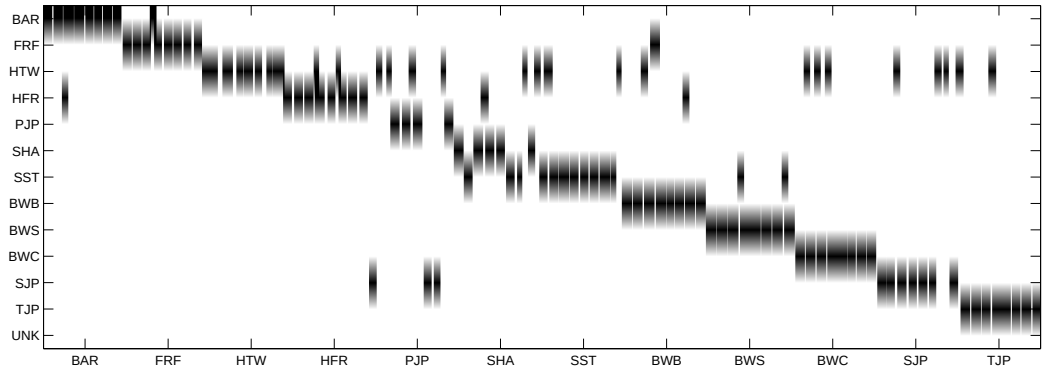

Figure 8.3. Classification results for the concatenated test database $\mathcal{D}^{\text{Seq}}$ using MTs either masked with quartile mask or with an intelligent automatic mask (for the `TJP` category).

motion class than the one at position of minimal cost. Remember, that the minimal cost is supposed to be found at the end of a document so that the warping path labels all frames that come before the position of minimal cost. Within a region of half the template length of the respective class MT of the minimal cost, we define a *label stop region* which acts similar than the false alarm for the subsequence retrieval. This means that those frames cannot be retrieved as minimal cost any more. The classification procedure will then be repeated until every frame of the minimal distance curve has been classified.

Note that, as the label stop region excludes all frames that are near the local minimum within half the class MT length, frames outside of this region can be labeled a second time as different motion class. So, one time frame can be labeled multiple times.

8.2.1 Results

The results for the classification of all documents contained in the concatenated test database $\mathcal{D}^{\text{Seq}}$ contain more mislabeled passages than mislabeled documents in the document based classification, compare Figure 8.3 and Figure 8.2. This is not surprising, as in the subsequence classification scenario, all motions are concatenated to one big data stream without any segmentation. As a result, it may be that not all document endings will be found at their exact (annotated) positions. Then, the similarity distance raises so that the chance of mislabeling and confusing motion classes raises. Another problematic result can be that not the whole jump can be classified within one classification step due to inaccurately computed warping paths. As some frames may be missing in the warping path computations, not all frames that belong to one jump will be labeled within one computation step. This means, small regions can be left unclassified first and may be classified by another motion label later. In Figure 8.3, one can see those subsequence classification problems as for all classes, small passages between the documents have not been classified or have been classified as wrong motion classes in a further computation step.

In particular, most mislabeled motion classes are assigned to be `HTW` jumps. We have already seen that the `HTW` jumps can not be characterized in a semantically meaningful

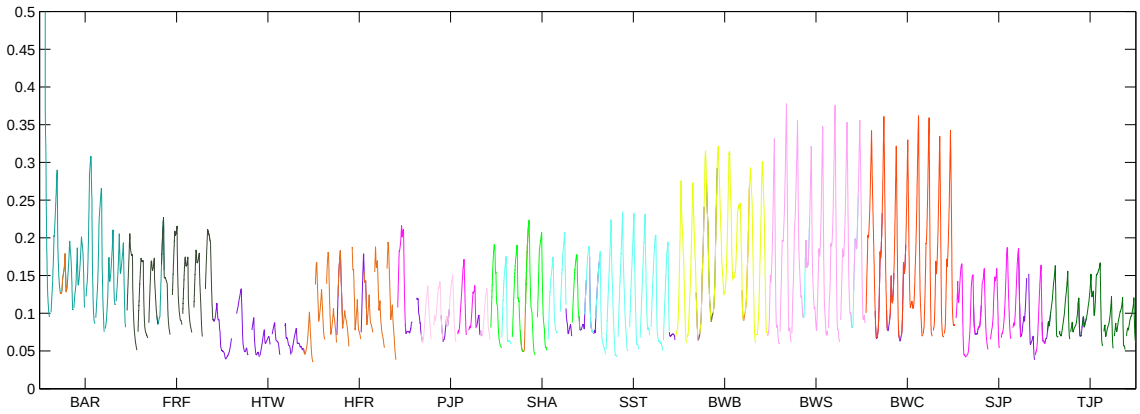

Figure 8.4. Distance curve $DC_{\min}$ of overall minimal costs for all distance curves. Every MT is represented by it's individual color. All mislabeled frames can be identified by wrong label color.

way with the chosen feature set. Framewise parts of other jumps that do not undergo any characteristic information can then be similar to the `HTW` jump frames and consequently be classified by the `HTW` label. Furthermore, again `PJP` jumps are confused with `SJP` jumps indicating again that the distance of comparison between both motion classes is very small and that the motion classes can hardly be discriminated.

For graphical visualization of the results, we chose another method. In Figure 8.4 we plotted the minimal distance curve over all distance curves $DC_{\min}$ that has then been used for the frame-wise classification. For all frames, the corresponding MT of lowest cost has been computed. For every motion class, we define an individual color. We then plot every frame of $DC_{\min}$ by the color of the motion class of lowest cost. As D_C has been concatenated alphabetically, every eight jumps, $DC_{\min}$ should be classified by a new motion class label. For motion classes that can be discriminated well like the somersault jumps, eight positions of minimal cost can be identified in the distance curve plot. With respect to mislabeling, mostly only frames in between the positions of local minimal cost have been classified incorrectly. This displays the same problems as in Figure 8.3, where small passages in between the document endings have been labeled with a wrong motion class. Again, one can see that in between the significant positions of local minimal cost, incorrect classified frames will be most likely classified as `HTW` (colored purple) even if they do not comprise of `HTW` jumps.

The problems of unclassifying and mislabeling small passages within the subsequence classification will even become more clearly in the second experiment, the routine classification. We recap that the routine moves have not been cut and do not only contain concatenated jumps, but also every other information that has been obtained during the capture process (like the contact phases with the trampoline bed or the phases of beginning and ending when the athlete is standing still on the trampoline). So more information that has not been learned in MTs has to be handled during the classified process leading to worse results.

Since the region around retrieved positions of local cost is defined by half the length of the corresponding MT to the left and to the right, but the number of frames that will be classified is defined by information from the warping paths to the left of the retrieved position, overlaps where a single frame has been classified as two motion classes can arise

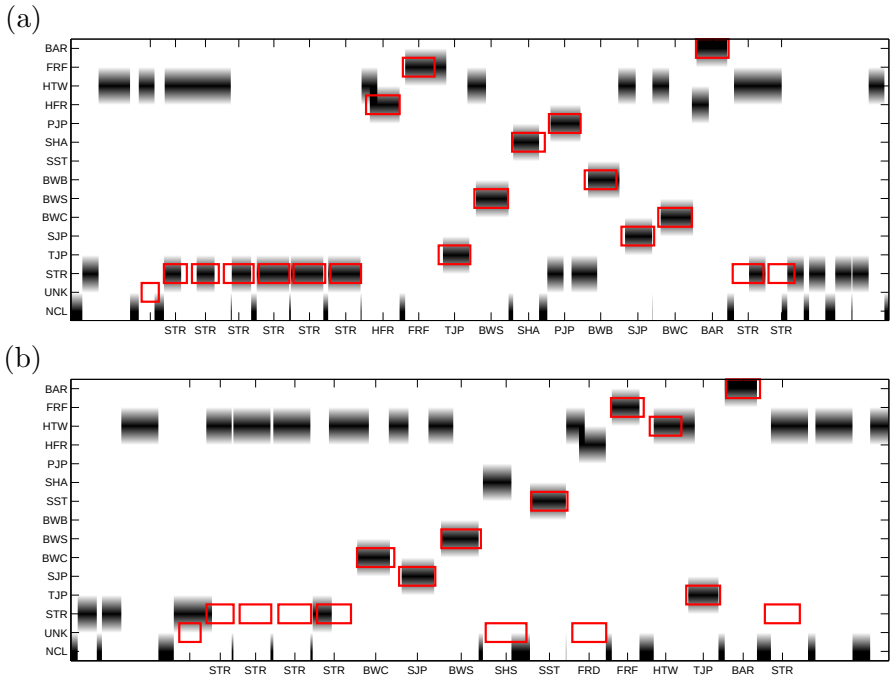

Figure 8.5. Classification results for the Routines $R4$ and $R11$ with subsequence-based classification on the consecutive routine streams. Clutter that cannot be classified occurs regularly and the classifications deviate from the frame-wise used ground truth.

as for the concatenated database $\mathcal{D}^{\text{Seq}}$. Due to the unknown frames that do not belong to any learned motion but belong to some other event in the motion stream, furthermore clutter (which means short shreddings of unclassified frames) occurs. This clutter will mainly not be classified as full motion and is likely to be excluded from classification by the classification stop regions, but can also be misclassified as part of a routine jump. Those unclassified parts are similar to the incorrect classifications from Figure 8.4, but confuse the classification results even more.

Because of those in-between parts, we will introduce another additional label *not classified* `NCL` that will be assigned to all frames that cannot be identified as a given motion class and can even not be identified as unknown motion class. For some cases, those unclassified regions comprise the contact phases with the trampoline bed or to the beginning and ending of the routine, but sometimes only consist of several small frames that do not obtain any information. Figure 8.5 shows the retrieval results for the same Routines $R4$ and $R11$ than for the document-based classification with the additional classes `NCL`, `UNK` and `STR`. Again, the red boxes indicate the positions of the motion classes from the annotation as ground truth, whereas in the subsequence classification, the information from the annotations will be used frame-wise and not document-wise as in Section 8.1.

Deviations from the frame-wise ground truth often occur at the beginning of every jump, see $R4$, but can also occur at other positions like the motion ending as in $R11$. Furthermore, one motion can also be split into two different motion class labels. For those deviations from the ground truth that mainly occur because of the clutter, we have to eliminate them. The same holds for the overlaps in classification and the frames that have been classified as any motion class without containing any motion information. For

example, misclassified shreddings occurs to a large extent at the beginning of the move where the actor is just standing on the trampoline bed. This phase cannot be identified as unclassified waiting and standing motion so that consequently, parts of the phase will be classified as jumps (mainly as an insignificant jump such as `STR` of `HTW`). The remaining parts will then be left unclassified due to the classification stop regions, so that many short classified regions occur that should not be classified at all. The same principle holds for the end of the routine, when the athlete decelerates his body to return to a standing position. In the following section, we will address this problem of missing segmentation within the data stream.

8.3 Classification with Automatic Segmentation

We have seen in Section 8.1 that the classification results will be very good while comparing segmented motion documents with the MTs. In Section 8.2 we have set up the more realistic scenario that does not classify single documents, but passages within an unknown motion stream. The classification results, however, degrade significantly in the more realistic scenario. The classification becomes prone to clutter and additional information that has not been learned as MT or that cannot be identified as unknown frames. To establish a classification method that uses the advantages of document based classification in a realistic scenario we therefore implement an automatic segmentation algorithm that splits the trampoline motion stream into it's single jumps. As a result, labeling problems of unlearned events that occur in a trampoline routine will be diminished.

Remember that trampoline data can easily be segmented by the moments the athlete is in contact with the trampoline bed. For the athletes that acted in $\mathcal{D}^{\mathrm{T}}$, all trampoline jumps are designated by approximately the same jump lengths and similar motion heights. Furthermore, the contact phases with the trampoline bed have characteristic kinematic properties that can be identified quite easily and facilitate motion segmentation. We will use those properties of the trampoline data now to enable more stable classification results.

8.3.1 Segmentation Algorithm

To retrieve every single jump within one trampoline routine, we use the following blanket contact detection algorithm based on elementary physics. As we know from physics the athlete will undergo a free fall motion during the flight phase of the jump (so any locally measured acceleration has to be close to zero and can only be influenced by moving parts of the body during motion performance). On the other hand, the acceleration will be very high at the moments the athlete's flight phase is interrupted (and the athlete gets in contact with the trampoline bed). We can then use those simple assumptions to define a natural separation between flight phase and trampoline contact phase: all frames that contain accelerations higher than a certain threshold t_{seg} will be identified as contact phase frames. In general, we will mark flight phase intervals from the moment the acceleration undergoes the threshold until the acceleration reaches the threshold again.

For a stable acceleration that is irrespective to different landing positions and hence different sensor orientations, we will compute the absolute acceleration over all three accel-

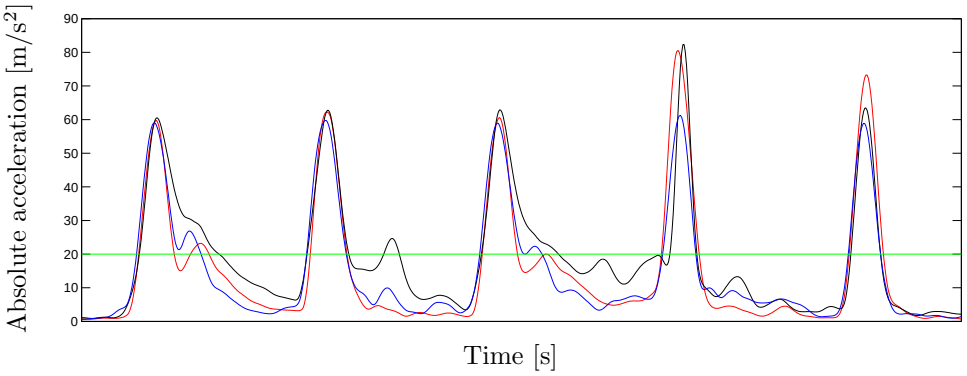

Figure 8.6. Acceleration Plot for root sensor (red) and the sensors attached to the left leg: left ankle (black) and left hip (blue). The green line shows the threshold $t_{\text{seg}} = \boldsymbol{a} \geq 20\,\text{m/s}^2$. The accelerations for the leg sensors are higher, whereas the acceleration peaks are much sharper for the root sensor and do not merge into accelerations from the flight phase (as the acceleration from the left ankle's sensors).

eration axes. This means that the acceleration will always be positive and acceleration information of all axes will be taken into account. As noise overlies the captured data, we will additionally filter the absolute acceleration data with a low pass filter to obtain a smooth acceleration curve.

While comparing the acceleration curves of every sensor, we find differences in the acceleration depending on where the sensors are attached to the actor's body. For example, the highest acceleration occurs at the (lower) legs' sensors, but those sensors are also subject to highest accelerations during flight phases and do not offer as sharp acceleration peaks as other sensors, though. The sensors of the arms yield sharp peaks of acceleration, but are subject to several variations during the landing and takeoff phase, so that they do not seem to deliver reliable results. Consequently, we decided to use the acceleration information of the sensor s_1 that has been attached at the lower spine. As threshold, we choose $t_{\text{seg}} = \boldsymbol{a} \geq 20\,\text{m/s}^2$ which appeared to be a separative value between flight and contact phase from the acceleration plots. Figure 8.6 illustrates different acceleration plots for the root sensor near the spine and the acceleration of the sensors attached to the legs. One can see that the acceleration peaks of the leg's sensors are much broader than the peaks for the root sensor. For some of the leg's peaks, the contact phase directly goes over into the flight phase. Contact phase intervals could then not always bee identified correctly or would be segmented over a too long period of time.

Despite of the sharper peaks in the root sensor's acceleration during the contact phase, segmentation errors may occur when the acceleration overgoes the defined threshold during the flight phase. Those accelerations will mainly occur during high rotational motion classes like for example during `BAR` jumps that are subject to dynamic motions and hence accelerations during the flight phase. In this case, additional contact phases will be found that do not exist in the routine. There are several ways to handle those segmentation errors. One would be to raise the threshold t_{seg} for the contact phase identification so that the acceleration that occurs during flight phase will always lie beyond the threshold. Then, it is however necessary to countervail the fact that the retrieved contact phases will be too short compared to the real contact phases. One possibility would be to add some frames

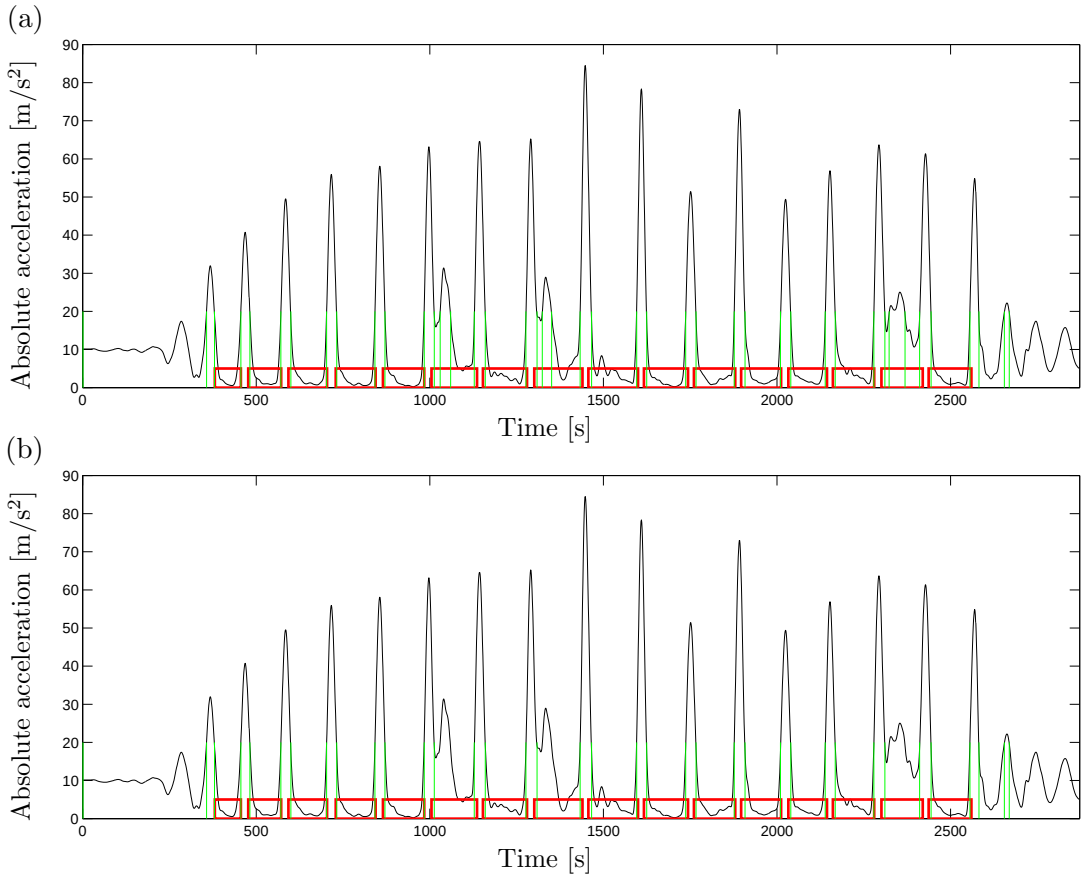

Figure 8.7. Automated Segmentations for the introduced segmentation algorithm. The automatic segmentations are indicated by green lines, the red boxes represent the flight phases as from the video annotations. (a) For high accelerations that occur during the flight phase, the routine will be segmented during flight phase, as well. (b) With the additional segmentation condition of minimal length, the move will not be separated and the routine will be classified correctly.

at the ending and beginning of the contact phase interval so that the determined contact interval will be prolongated. This method requires many trials to find the best suited threshold and the ideal number of additional frames for the ending and beginning of the contact interval so that this method will not been used. For the following classification, we will use another method that is based on essential technical rules of trampolining. Knowing that the flight phase of every trampoline jump has to consist of a minimal number of frames, we add a heuristic to the segmentation algorithm which causes that a jump will not be segmented if it consists of less frames than the defined minimal duration. This means that, if the acceleration during flight phase will be high shortly after the beginning of the jump's flight phase, this high acceleration will not affect the segmentation. As all trampoline moves from the trampoline database are in between 120 and 140 frames of length, we will assume the half minimal length of 60 frames to be the minimal possible length of a trampoline jump. Even for the first preparing jumps where the athlete gains height at the beginning of the routine, the duration is longer than 60 frames. High accelerations during the flight phase moreover only occur at the first half of the move (for all trampoline jumps within $\mathcal{D}^{\mathrm{T}}$ that are used for this thesis). Consequently, only the real blanket contacts will be segmented, see Figure 8.7.

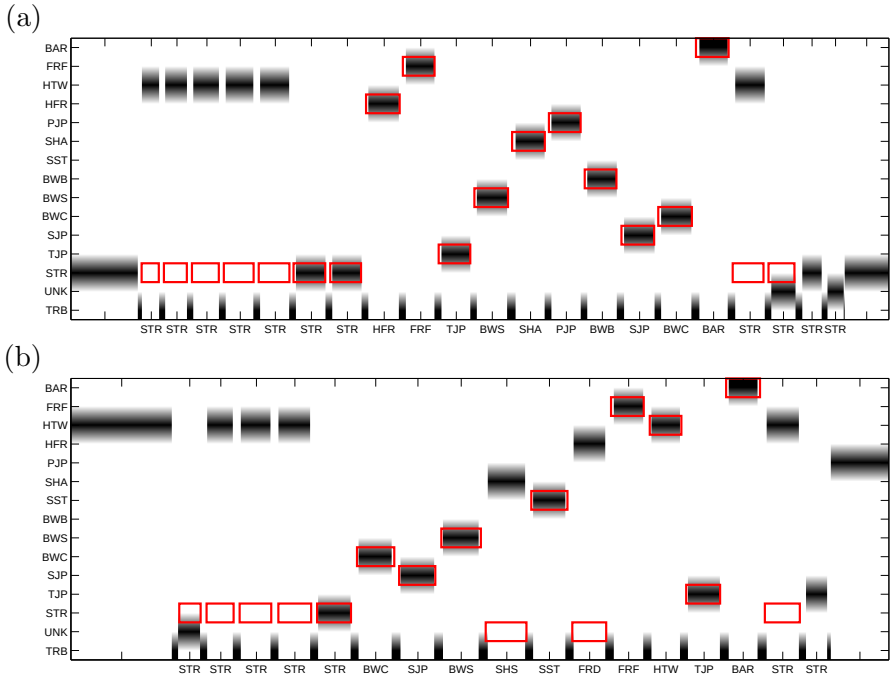

Figure 8.8. Classification results for the Routines $R4$ and $R11$ with document-based classification for automatically segmented jumps from the routine streams. Classification results improve and most deviations from the frame-wise used ground truth annotations only occur at the beginning of the jumps.

Using again the video annotations for the jumps within the routines as ground truth (indicated by the red boxes), we can see that the automatic segmented jumps deviate only within few frames from the annotated segmentation. Overall, the segmentation algorithm works very well and separates the jumps within the routine in motion documents of similar lengths and positions than the handmade annotations.

8.3.2 Results

After segmenting all trampoline jumps within a routine into single documents by the automatic segmentation algorithm, we can conduct document-based classification with global DTW as in Section 8.1. As the subsequence-based classification results suffered a lot from the problem of missing segmentation and unlearned events that occurred within the motion stream, we expect the automatic segmentation to yield more stable classification results.

We will visualize the classification results again with the Routines $R4$ and $R11$ to enable a comparison between all classification methods in Figure 8.8. The automatic segmented contact phases will be labeled as additional motion type *trampoline bed* `TRB`.

At first glance, it becomes clear that the classification results are much better than for the subsequence classification. The classification and labeling information is much more regular and more consistent to the video annotations. Due to the document-based clas-

sification, every frame within the routine is assigned just one label different than for the unsegmented classification. Moreover, as the blanket contact phases are determined automatically by inversion of the segmented jumps, they can be left out of consideration for a distance comparison with the MTs. As a result, no clutter in form of short unclassified passages or short incorrectly classified shreddings will occur as it has been especially at the beginning and the ending of the routine for the subsequence-based classification.

With the automatic segmentation algorithm, the first and last blanket contacts in the routine define beginning and ending of the relevant routine information. The two intervals before and after this relevant phase can then also be treated as single motion documents and be compared to the MTs like every other jump within the routine. In most cases, they will be classified as `UNK`, as the beginning and ending of a routine is very different to every regular trampoline jump (and hence to every learned template). Consequently, all shreddings of incorrectly classified motions and tiny unclassified passages that have affected the subsequence-based classification disappear.

The moves that are parts of the routines (excluding the preparing `STR` jumps) can be classified correctly in most cases. However, the straight jumps will still be confused with the motion classes `HTW` and `UNK`. Remember those motion classes have already been confused in the the document-based classification of the annotation-based segmented routines and can be discriminated from each other less accurate than the other motion classes.

The ending phase often consists of a final deceleration jump, so that the ending phase can sometimes also be confused with insignificant motion classes like the `HTW` or `STR` jump, see Figure 8.8 (b). However, as those phases do not comprise any meaningful information, the assigned label of this jump is of less significance for the overall classification than the assigned labels for mid-routine jumps.

Deviations from the ground truth mainly appear at the beginning of a jump. We will ignore them as natural deviations that occur because of slight differences between the manually annotated contact intervals (that are defined by the annotated flight phases) and the automatically segmented jump intervals. All in all, the results are much better than for the subsequence-based classification and yield a stable classification for the trampoline routines.

Figure 8.10 at the end of this chapter shows the classification results for all 13 routines, whereas the same Routines $R1$, $R4$, $R7$ and $R10$ as well as the same Routines $R2$, $R5$, $R8$ and $R11$ are plotted in a coherent way followed by the freestyle routines ordered by $R3$, $R6$, $R12$, $R9$ and $R13$.

8.4 Evaluation Metrics for Classification Results

To quantify the classification result we already discussed during the last sections, we will introduce an evaluation metric that works similar than the Confusion Matrices from Chapter 7 and visualizes all motion classes a motion class is mixed up with. To evaluate all classification methods, we will store the information on how often every document or frame will be classified correctly or incorrectly for all classifications methods. For the incorrect classifications, we additionally store the label of the motion class that has been

Documentwise AutoSegmentation
$Mean = 84.65\%$

BAR	FRF	HTW	HFR	PJP
100%	80.00%	85.71%	100%	100%
SHA	SST	BWB	BWS	BWC
87.50%	88.89%	100%	100%	100%
SJP	TJP	STR	UNK	
100%	100%	33.93%	9.09%	

Documentwise Annotations
$Mean = 83.99\%$

BAR	FRF	HTW	HFR	PJP
90.00%	70.00%	85.71%	75.00%	100%
SHA	SST	BWB	BWS	BWC
87.50%	100%	100%	100%	100%
SJP	TJP	STR	UNK	
91.67%	92.31%	40.18%	43.48%	

Framewise AutoSegmentation
$Mean = 77.37\%$

BAR	FRF	HTW	HFR	PJP
84.77%	72.19%	77.54%	91.35%	90.56%
SHA	SST	BWB	BWS	BWC
78.59%	80.30%	89.48%	88.84%	88.57%
SJP	TJP	STR	UNK	TRB
91.16%	91.46%	32.74%	8.20%	94.81%

Framewise Subsequence
$Mean = 57.05\%$

BAR	FRF	HTW	HFR	PJP
70.28%	80.05%	58.00%	64.46%	52.28%
SHA	SST	BWB	BWS	BWC
57.07%	74.68%	67.58%	64.94%	74.25%
SJP	TJP	STR	UNK	NCL
50.58%	51.16%	53.59%	0.0%	36.18%

Table 8.2. Evaluation of Classification results. Percentage of correctly classified documents respectively frames for the total occurrence of a motion class.

classified instead. We can then count the overall quantity of all appeared jumps for every motion class. By normalizing the number of correctly and incorrectly classified jumps by the total number of appearances for every motion class, we can obtain information on the accuracy of every classification method.

Table 8.2 expresses those normalized quantities as absolute percentage of correct labeling. For a comparison between the subsequence-based classification and the automatic segmentation algorithm, we will evaluate the classification frame-wise and mark for every frame whether it is classified correctly or not. To compare the document-based classification of manually annotated documents with the automatic segmentation algorithm, we will evaluate the classification document-wise and mark for every document whether it is classified correctly or not.

We can see from Table 8.2, that the classifications that are based on global DTW achieve higher accuracies in the labeling than the subsequence-based classifications. As for the subsequence-based classification results, we compare every frame of all routines with the annotated ground truth, the overall accuracy of the classification is much more affected by noise in form of mislabeled frames in the frame-wise evaluation. Furthermore, we have already mentioned in Section 8.2, that in a frame-wise subsequence classification, clutter and mislabelings occur more frequently than for the classification on a segmented data stream. Hence, the mean value for the classification accuracy is lowest in this scenario and for all motion classes, frames are mixed up with labels of dissimilar motion classes. For the document-wise evaluation, we can see that significant motion classes as somersaults

that have already been discussed in Chapter 6 are labeled correctly every time they occur in a routine and hence have an accuracy of 100%. Less significant motion classes that only differ by rotational motion around the longitudinal axis like SHA and SST cannot be labeled correctly in all cases. Here, we expect the motion classes to be mixed up with each other.

The graphical visualization of the data can be found in Figure 8.9. As in Table 8.2, we compare the results either document-wise or frame-wise in dependence of the used classification method. In Figure 8.9 (a) and (b), we compare the classification results of the document-based classification in a document-wise way with the automatic segmentation. In Figure 8.9 (c) and (d), we compare the classification results of the subsequence-based classification in a frame-wise way with the automatic segmentation.

The evaluation matrices can be interpreted as followed: on the vertical axis, the data from the ground truth annotation is plotted. On the horizontal axis, the data from the classification is plotted. For every row and it's corresponding motion class, one can then identify all motion classes that have been confused with the respective true motion class. This also means that for every row, we can then easily see all motion classes the motion class is mislabeled as. For example, the BAR has been labeled as BAR for mostly all frames that belong to the BAR label according to the annotations, but also labeled as TRB for several frames with the automatically segmented classification, see Figure 8.9 (a). The darker the box on the diagonal, the more correctly labeled jumps have been classified. In the case of an ideal classification of 100% accuracy, the box would be of pure black color.

In the frame-wise classification evaluation of the automatically segmented jumps, we can see that for every motion class, several frames are misclassified as TRB contacts according to the information from the annotations. As we have already mentioned, the automatic segmentations are some frames shorter than the annotations at the beginning, so those frames will be marked as incorrectly labeled. On the other hand, the TRB contact phases are also sometimes assigned different motion class labels, which is also due to the not absolutely identical contact phase segmentations. All in all, one can see that the most confusions occur for the motion classes STR, UNK, HTW and TRB which cannot always be distinguished, as we already discussed in the last sections.

In the frame-wise evaluation of the subsequence-based classification, we can see that every motion will be mislabeled as STR for some frames. This is mainly due to the problems of clutter, overlaps and short misclassified shreddings we introduced in Section 8.2.

In the document-wise evaluation we can see that for both matrices, most documents are classified correctly and that the classifications do only undergo few labeling errors. So we can state that both classification methods are very stable. For the classification of UNK jumps, the automatic segmentation even performs better. This may probably be subject to the different document lengths as the segmented jumps are all shorter than the annotated jumps that eliminates similarities to learned motion classes at the beginning of the class feature representation. We could assume hence, that, with a shorter jump length, UNK jumps can be classified more precisely as UNK. Furthermore, we can see that jumps from the motion classes SHA and SST are really mixed up with each other as it was supposed while evaluating Table 8.2.

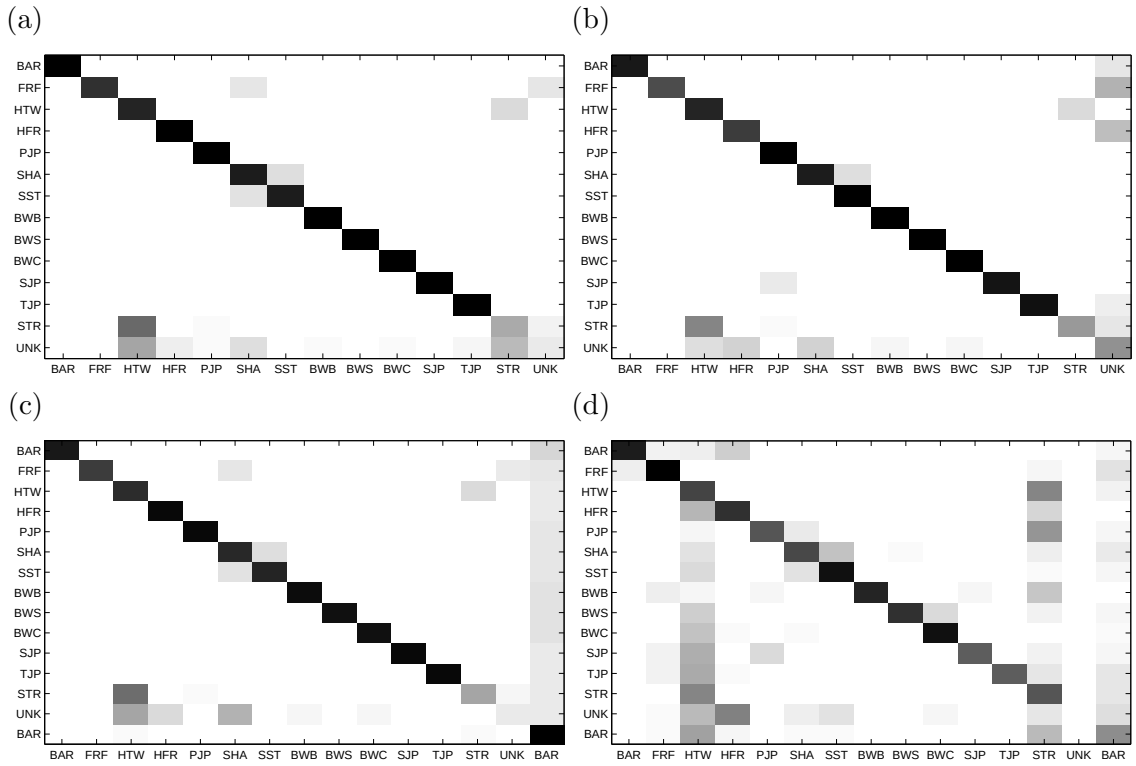

Figure 8.9. Evaluation of the classification results: document-wise (top row) for the automatic segmentation (a) and annotation based segmentation (b) and frame-wise (bottom row) for the automatic segmentation (c) and subsequence-based classification (d).

The evaluation measures confirm our results that the automatically segmented jumps offer a stable and good possibility to classify a consecutive data stream without being prone to clutter and noise. As we can see, the quality of the classification with automatic segmentation lies in between the classification quality of the two other methods, but does not offer much worse results than the artificial and unnatural document-based classification.

8.5 Conclusion

We have seen that document-based classification with global DTW computations yields very good results to classify the trampoline data, but does not offer any realistic classification scenario. The realistic scenario with subsequence DTW computations however, does not yield as excellent results as the document-based classification.

We therefore introduced a way to combine the more accurate document-based classification with the classification of a long data stream. By an automatic segmentation algorithm based on the detection of all contact phases with the trampoline bed, we obtained a stable method to segment a long motion stream into (semantically correct) motion documents that can then be classified document-wise. This segmentation is quite simple and additionally offers satisfying classification results, so that it can also be used for further classification scenarios in real-time and quasi real-time.

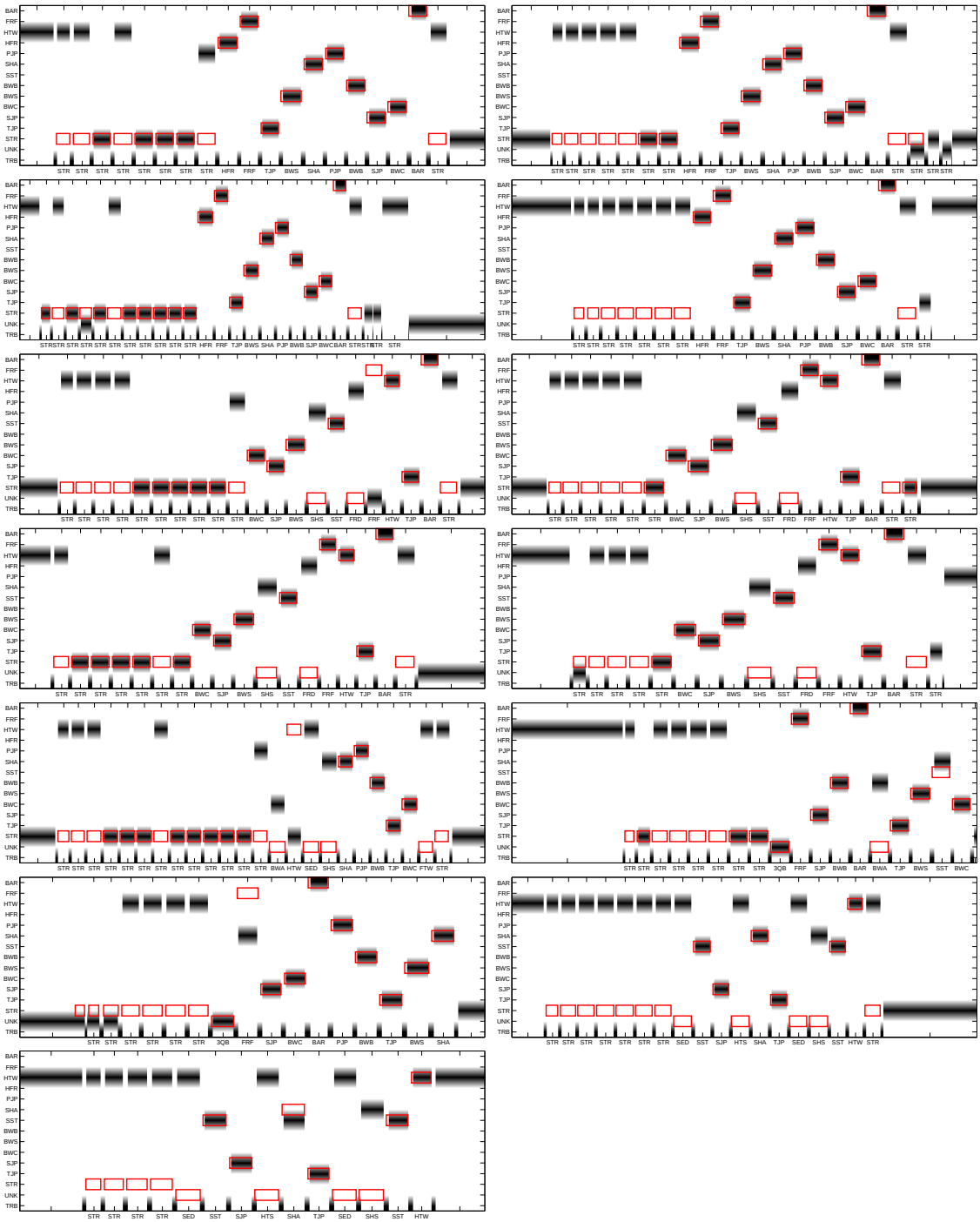

Figure 8.10. Classification results for all routines with document-based classification of automatic segmented jumps from the routines. First, the Routines $R1$, $R4$, $R7$ and $R10$ (first two rows), then the Routines $R2$, $R5$, $R8$ and $R11$ (row 3 and 4) and finally the freestyle routines $R3$, $R6$, $R12$, $R9$ and $R13$ are displayed (bottom rows).

Chapter 9

Conclusion

In this thesis, we introduced a method for automatically classifying unknown trampoline routines that were captured using inertial sensors. To this end, we transformed the inertial sensor data to a feature representation which reflects the specific characteristics of different trampoline jumps. Then, the unknown feature sequence was automatically segmented and locally classified using DTW as similarity measure. Here, the individual segments were compared to previously learned MTs representing specific motion categories.

Our main contributions can be summarized as follows. Firstly, we recorded, annotated and composed the databases used for the experiments. Secondly, we developed the algorithm for the automatic segmentation of the feature sequences. Thirdly, we systematically analyzed various feature representations. Here, one goal was to find a feature representation that shows high discriminative power. Another goal was to minimize the influence of style variations within one motion class. We addressed this problem by introducing different masking techniques for a given motion template. Here, our algorithm marks regions within the feature sequence that are expected to be subject to high style variation.

By using the techniques presented in this thesis we obtained accurate classification results for most motion classes tested. As the experiments showed, some motion classes which include rotations about the longitudinal axis were often mixed up with semantically similar motion classes that do not include such rotations. This behavior is due to the fact that the examined feature representations do not capture rotations about this body axis. This drawback could be solved by adding additional feature functions that indicate such rotations.

As future work, we will extend the feature representations in a way that also rotations about the longitudinal body axis are incorporated. Furthermore, the classification scenario has only been tested on previously recorded trampoline motions and in a pure offline scenario. Here, we want to create an application that can be used for sports science and computer-assisted training. To this end, it is necessary to implement the classification system within an online framework. In a real-time application, classification results can then for example be used to generate direct feedback to the athlete about the technical precision of his performance in an acoustical or visual way.

Appendix A

Source Code

In this chapter, the headers of selected MATLABfunctions created during the writing of this thesis are reproduced. The headers contain information about the name of the described function and its input/output behavior.

Feature Evaluation

The `getDistanceMatrix` function computes and returns a distance matrix for all motion feature representations within a selected database.

Sample usage:

```
distMatrix = getDistanceMatrix(IncliFeatures,Parameter);
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%Name: getDistanceMatrix
%Author: Heike Brock
%Date: 2010/07/21
%
%DESCRIPTION:
%This function computes the distance matrix for given motion feature
%representations by computing the DTW distance d_ij for every feature
%representation with every feature representation. The distance matrix can
%then be plotted and saved.
%
%INPUT:
%   The first argument must be a Kx1 cell array containing feature matrices
%   of equal feature set.
%
%OPTIONAL PARAMETERS:
%   The second argument can be a struct defining additional arguments:
%       FeatureName : string defining the name of the chosen feature set
%       FeatureSet : struct defining the chosen feature set
%       Plot : boolean value defining whether the distance matrix will be
%       plotted (false=default).
%       FigDir : string defining the direcory the plotted figure will be
%       saved in.
%
%OUTPUT:
%   The returned argument is the resulting distance matrix.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

The `getDistanceMeasures` function computes and returns the distance measures ϵ, υ and δ for all motion feature representations within a selected database.

Sample usage:

```
[DocAlpha, DocBeta, DocGamma] = getDistanceMeasures(distMatrix,FeatEval_Parameter);
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%Name: getDistanceMeasures
%Author: Heike Brock
%Date: 2010/07/21
%
%DESCRIPTION:
%This function computes the mean distance measures alpha, beta and gamma for
%document based feature evaluation over all motion classes within the
%database. The function uses a previously computed distance matrix.
%
%INPUT:
%   The first argument must be a KxK distance matrix containing the DTW
%   distance d_ij for all motions.
%
%OPTIONAL PARAMETERS:
%   The second argument can be a struct defining additional arguments:
%       DBLength : integer defining the number of moves that are contained
%       in the database
%       Percentile : double defining the percentage for the beta measure
%       between 0 and 1
%       NumMotClasses : integer defining the number of different motion
%       classes contained in the database
%
%OUTPUT:
%   The first returned argument is the mean document based alpha value.
%   The second returned argument is the mean document based beta value.
%   The third returned argument is the mean document based gamma value.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

Subsequence Evaluation

The `getDistCurves` function concatenates the feature representations within a database to one consecutive motion stream and computes the distance curves that will be obtained with subsequence DTW. The function can either be used with the feature representations from one database or from two databases. Using only the feature representations from one database, the query motion will be included in the database. Using feature representations from two different databases, one database will function as training database.

Sample usage:

```
[DistCurves, info] = getDistCurves(IncliFeatures,IncliFeatures,Subseq_Parameter);
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%Name: getDistCurves
%Author: Heike Brock
%Date: 2010/08/01
%
%DESCRIPTION:
%This function concatenates the feature representations of a database into
%one consecutive motion stream and computes the distance curve for every
%feature representation as query motion with subsequence DTW.
%
%INPUT:
%   The first argument must be a Kx1 cell array containing feature matrices
```

```
%   of equal feature type.
%   The second argument must be a Mx1 cell array containing feature matrices
%   of equal feature type.
%
%OPTIONAL PARAMETERS:
%   The third argument can be a struct defining additional arguments:
%       FeatureName : string defining the name of the chosen feature set
%       FeatureSet : struct defining the chosen feature set
%       ConcatInfoStyle : string defining the type of concat info returned.
%                   'old' or 'struct' produces a struct with several
%                   fields.
%                   'new' or 'interval' produces a Dx2 matrix with the
%                   start and end indices of all D documents inside the
%                   concatenated data (default).
%       SubSequence : a boolean value defining wether this function shall
%            do a subsequence match (true) or a global match
%            (false=default).
%       dn : 1xS integer array defining valid steps (n direction of C).
%            Default is [1 1 0].
%       dm : 1xS integer array defining valid steps (m direction of C).
%            Default is [1 0 1].
%       dw : 1xS double array defining the weight of each step.
%            Default is [1 1 1].
%
%OUTPUT:
%   The first returned argument is a KxL array containing the distance curves
%   for every query motion.
%   The second returned argument is a Kx2 array containing information of
%   the position of the original documents inside the concatenated data
%   stream.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

The `subsequenceEvaluation` function performs subsequence retrieval for a chosen query motion on base of previously computed distance curves. The first eight hits will be marked and the distance measures will be defined. The results can be plotted in a graph.

Sample usage:

```
[RetrInfoVals, retrHits_Actor] = subsequenceEvaluation(IncliFeatures,DistCurves, info, SubseqRetr_Parameter);
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%Name: subsequenceEvaluation
%Author: Heike Brock
%Date: 2010/08/03
%
%DESCRIPTION:
%This function evaluates the feature sets with subsequence DTW using
%previously defined distance curves and computes the distance measures for
%the retrieval results.
%
%INPUT:
%   The first argument must be a Kx1 cell array containing feature matrices
%   of equal feature type.
%   The second argument must be a KxL array containing the distance curves for
%   every query motion.
%   The third argument must be a Kx2 array containing information of the
%   position of the original documents inside the concatenated data stream.
%
%OPTIONAL PARAMETERS:
%   The fourth argument can be a struct defining additional arguments:
%       FeatureName : string defining the name of the chosen feature set
%       FeatureSet : struct defining the chosen feature set.
%       Percentile : integer defining the percentage for the beta measure
%       in percent.
%       varTolerance : double defining a tolerance region around the
```

```
%       document end positions indicating true hits.
%       moveIndx : integer defining move acting as query motion. The range
%       goes from 1 to 8.
%       Plot : boolean value defining whether the retrieval results will be
%       plotted (false=default).
%       FigDir : string defining the direcory the plotted figure will be
%       saved in.
%       NumMoves : integer defining the number of moves each motion class
%       consists of.
%
%OUTPUT:
%   The first returned argument is a struct containing the distance
%   measures for the chosen query.
%   The second returned argument is a Kx1 array containing all retrieved
%   hits of minimal cost.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

Motion Templates

The `computeMotionTemplate` function computes first an averaged matrix over all feature representations from one motion class and iterates the process of calculation for a specific number of iteration steps.

Sample usage:

```
[MotTempl, MotTemplStd] = computeMotionTemplate(IncliFeaturesTraining,numClassMotions,MT_Parameter);
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%Name: computeMotionTemplate
%Author: Heike Brock
%Date: 2010/08/21
%
%DESCRIPTION:
%This function computes the Motion Templates for every motion class
%contained in a database and the standard Deviations for every motion
%template.
%
%INPUT:
%   The first argument must be a Kx1 cell array containing feature matrices
%   of equal feature type.
%   The second argument must be an integer defining the number of moves
%   every motion class is characterised by.
%
%OPTIONAL PARAMETERS:
%   The third argument can be a struct defining additional arguments:
%       FeatureName : string defining the name of the chosen feature set
%       FeatureSet : struct defining the chosen feature set.
%       Plot : boolean value defining whether the motion templates will be
%       plotted (false=default).
%       Iterations : integer defining the number of iteration steps for
%       calculating the MTs.
%
%OUTPUT:
%   The first returned argument is a cell array containing the Motion
%   Templates for all motion classes.
%   The second returned argument is a cell array containing the matrices
%   defining std for every MT.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

The `getMTMask` function computes a mask for every MT according to a defined threshold value and the standard deviations of every MT.

Sample usage:
```
MTMask = getMTMask(MotTempl,MotTemplStd,MT_Parameter);
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%Name: getMTMask
%Author: Heike Brock
%Date: 2010/08/28
%
%DESCRIPTION:
%This function creates a Mask for every passed MT in dependance of the Mts
%standard deviation and a defined threshold.
%
%INPUT:
%   The first argument must be a Nx1 cell array containing MTs of equal
%   feature type.
%   The second argument must be a Nx1 cell array containing the std of
%   every MT.
%
%OPTIONAL PARAMETERS:
%   The third argument can be a struct defining additional arguments:
%       FeatureName : string defining the name of the chosen feature set
%       FeatureSet : struct defining the chosen feature set.
%       Plot : boolean value defining whether the masked MTs will be
%       plotted (false=default).
%       Threshold : double defining the threshold to mask the MTs.
%
%OUTPUT:
%   The returned argument is a 12x8 cell array containing the masks for all
% MTs.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

The `getMTMaskFromKnowledge` function computes an intelligent mask for the MTs of the motion classes tuck jump, pike jump and straddle jump. Additional knowledge about the moves'shapes will be taken into account.

Sample usage:
```
Int_MTMask = getMTMaskFromKnowledge(MotionTemplate, MT_Parameter.Threshold);
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%Name: getMTMaskFromKnowledge
%Author: Heike Brock
%Date: 2010/09/15
%
%DESCRIPTION:
%This function creates an intelligent Mask using expert knowledge for the
%moves pike jump, tuck jump and straddle jump.
%
%INPUT:
%   The first argument must be a Nx1 cell array containing MTs of equal
%   feature type.
%   The second argument must be a double defining the threshold that
%   determines the most variant data.
%
%OUTPUT:
%   The returned argument is a 3x1 cell array containing the intelligent masks
%   for all three motion classes tuck jump, pike jump and straddle jump.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

The `determinePrecisionRecall` function yields values for precision and recall for a defined number of retrieval steps.

Sample usage:
```
[precision, recall] = determinePrecisionRecall(retrHits, res{m}, i, numClassMoves, varTolerance);
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%Name: determinePrecisionRecall
%Author: Heike Brock
%Date: 2010/10/12
%
%DESCRIPTION:
%This function determines Precision and Recall. It uses information about
%the retrieved hits and information about the true hits.
%
%INPUT:
%   The first argument must be a Kx1 array containing the positions of
%   retrieved hits.
%   The second argument must be a 1x12 cell containing the intervall
%   positions for every motion class.
%   The third argument must be an integer defining the current number of
%   retrieval step.
%   The fourth argument must be an integer defining the number of trampoline
%   jumps within one motion class.
%   The fifth argument must be a double defining the variance region within
%   a hit can be determined as true hit.
%
%OUTPUT:
%   The first returned argument is a double defining the value for
%   precision.
%   The second returned argument is a double defining the value for recall.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

Classification

The `classifyRoutineMovesDocwise` function classifies motion data streams documentwise with global DTW with segmentation information (either annotations or automatic segmentations). The label number of every classified document will be returned.

Sample usage:

```
ClassResDocWise = classifyRoutineMovesDocwise(Annotations, ClassifMotions, ClassifFeatureStreams, Class_Parameter);
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%Name: classifyRoutineMovesDocwise
%Author: Heike Brock
%Date: 2010/10/12
%
%DESCRIPTION:
%This function classifies a motion data stream based on segmentation
%information documentwise using global DTW.
%
%INPUT:
%   The first argument must be a Rx1 cell array containing structs with the
%   segmentation information for the motion streams to be classified.
%   The second argument must be a Rx1 cell array containing structs with the
%   motion data streams to be classified.
%   The third argument must be a Rx1 cell array containing the feature
%   representation for the motion streams.
%
%OPTIONAL PARAMETERS:
%   The fourth argument can be a struct defining additional arguments:
%       FeatureName : string defining the name of the chosen feature set
%       FeatureSet : struct defining the chosen feature set.
%       MT: Nx1 cell array containing the MTs for every motion class.
%
%OUTPUT:
%   The returned argument is a Rx1 cell array containing the documentwise
```

```
%   classification results.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

The `classifyRoutineMovesFramewise` function classifies motion data streams framewise with subsequence DTW with segmentation information (either annotations or automatic segmentations). A matrix containing the label number of every classified frame will be returned.

Sample usage:

```
ClassResFrmWise = classifyRoutineMovesFramewise(Annotations, ClassifMotions, ClassifFeatureStreams, Class_Parameter);
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%Name: classifyRoutineMovesFramewise
%Author: Heike Brock
%Date: 2010/10/17
%
%DESCRIPTION:
%This function classifies a motion data stream based on segmentation
%information framewise using global DTW.
%
%INPUT:
%   The first argument must be a Rx1 cell array containing structs with the
%   segmentation information for the motion streams to be classified.
%   The second argument must be a Rx1 cell array containing structs with the
%   motion data streams to be classified.
%   The third argument must be a Rx1 cell array containing the feature
%   representation for the motion streams.
%
%OPTIONAL PARAMETERS:
%   The fourth argument can be a struct defining additional arguments:
%       FeatureName : string defining the name of the chosen feature set
%       FeatureSet : struct defining the chosen feature set.
%       MT: Nx1 cell array containing the MTs for every motion class.
%       Mask: Nx1 cell array containing the masks for all motion classes.
%
%OUTPUT:
%   The returned argument is a Rx1 cell array containing the framewise
%   classification results.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

The `autoSegmentJumps` function segments motion data streams automatically with a threshold using the normed filtered acceleration data.

Sample usage:

```
[docIntv, blanketContacts] = autoSegmentJumps(ClassifMotions,Segment_Parameter);
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%Name: autoSegmentJumps
%Author: Heike Brock
%Date: 2010/10/17
%
%DESCRIPTION:
%This function segments the motions within a motion stream automatically on
%base of a filtered and normalised acceleration data.
%
%INPUT:
%   The first argument must be a Rx1 cell array containing structs with the
%   segmentation information for the motion streams to be classified.
%
%OPTIONAL PARAMETERS:
%   The second argument can be a struct defining additional arguments:
```

```
%       Threshold : integer defining the threshold of acceleration value
%       for segmentation.
%       Plot : defines whether the segmentation curves will be plotted
%       Annotations : needed for the plot function to visualise the
%       annotation based document endings.
%
%OUTPUT:
%   The first returned argument is a Rx1 cell array containing the
%   intervalls for every segmented document of every routine.
%   The second returned argument is a Rx1 cell array containing the blanket
%   contact information of every routine.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

Bibliography

[1] Apple, *Apple homepage*. http://www.apple.com, Accessed March 10th, 2010, 2010.

[2] O. Arikan, D. A. Forsyth, and J. F. O'Brien, *Motion synthesis from annotations*, ACM Transactions on Graphics (TOG), 22 (2003), pp. 402–408.

[3] A. Aristidou, J. Cameron, and J. Lasenby, *Real-time estimation of missing markers in human motion capture*, in Bioinformatics and Biomedical Engineering, 2008. ICBBE 2008. The 2nd International Conference on, May 2008, pp. 1343 –1346.

[4] A. Armenti, *The Physics of sports*, vol. 1 of The Physics of Sports, American Institute of Physics, 1992.

[5] B. Auvinet, G. Berrut, C. Touzard, L. Moutel, N. Collet, D. Chaleil, and E. Barrey, *Reference data for normal subjects obtained with an accelerometric device*, Gait & Posture, 16 (2002), pp. 124–134.

[6] W. Blajer and A. Czaplicki, *Contact modeling and identification of planar somersaults on the trampoline*, Multibody System Dynamics, 10 (2003), pp. 289–312.

[7] P. Boissy, S. Choquette, M. Hamel, and N. Noury, *User-based motion sensing and fuzzy logic for automated fall detection in older adults*, Telemedicine Journal and eHealth, 13 (2007), pp. 683–694.

[8] H. Brock, *SpoCap - Motion Capture im Sport anhand der Beispiele Trampolin und Stabhochsprung*. Hochschule der Medien Stuttgart, 2008.

[9] J. Chai and J. K. Hodgins, *Performance animation from low-dimensional control signals*, ACM Transactions on Graphics (TOG), 24 (2005), pp. 686–696.

[10] K. Culhane, M. OConnor, D. Lyons, and G. Lyons, *Accelerometers in rehabilitation medicine for older adults*, Oxford Journal Age and Ageing, (2005), pp. 556–560.

[11] Dartfish, *Video analysis software*. http://www.dartfish.com/en/coaching_software/sports-science.htm, Accessed October 14th, 2010, 2010.

[12] M. Dontcheva, G. Yngve, and Z. Popović, *Layered acting for character animation*, ACM Transactions on Graphics, 22 (2003), pp. 409–416.

[13] C. Frohlich, *Do springboard divers violate angular momentum conservation?*, American Journal of Physics, 47 (1979), pp. 583–592.

[14] J. Harding, C. G. Mackintosh, A. G. Hahn, and D. A. James, *Classification of aerial acrobatics in elite half-pipe snowboarding using body mounted inertial sensors*, The Engineering of Sport 7, 2 (2008), pp. 447–456.

[15] D. Q. Huynh, *Metrics for 3D rotations: Comparison and analysis*, Journal of Mathematical Imaging and Vision, 35 (2009), pp. 155–164.

[16] L. Kovar and M. Gleicher, *Automated extraction and parameterization of motions in large data sets*, ACM Trans. Graph., 23 (2004), pp. 559–568.

[17] J. Lee, J. Chai, P. S. A. Reitsma, J. K. Hodgins, and N. S. Pollard, *Interactive control of avatars animated with human motion data*, ACM Transactions on Graphics (TOG), 21 (2002), pp. 491–500.

[18] J. Lee and I. Ha, *Real-time motion capture for a human body using accelerometers*, Robotica, 19 (2001), pp. 601–610.

[19] M. Liverman, *The animator's motion capture guide: organizing, managing, and editing*, Charles River Media Game Development, Charles River Media, 2004.

[20] H. J. Luinge and P. H. Veltink, *Measuring orientation of human body segments using miniature gyroscopes and accelerometers*, Medical and Biological Engineering and Computing, 43 (2005), pp. 273–282.

[21] S. T. Moore, H. G. MacDougall, J.-M. Gracies, H. S. Cohen, and W. G. Ondo, *Long-term monitoring of gait in parkinson's disease*, Gait & Posture, 26 (2007), pp. 200–207.

[22] MotionAnalysis, *Motion capture systems from motion analysis.* `http://www.motionanalysis.com/`, Accessed March 10th, 2010, 2010.

[23] M. Müller, *Information Retrieval for Music and Motion*, Springer, 2007.

[24] M. Müller and S. Ewert, *Towards timbre-invariant audio features for harmony-based music*, IEEE Transactions on Audio, Speech, and Language Processing (TASLP), 18 (2010), pp. 649–662.

[25] M. Müller and T. Röder, *Motion templates for automatic classification and retrieval of motion capture data*, in Proceedings of the 2006 ACM SIGGRAPH/Eurographics Symposium on Computer Animation, ACM Press, 2006, pp. 137–146.

[26] M. Müller, T. Röder, and M. Clausen, *Efficient content-based retrieval of motion capture data*, ACM Transactions on Graphics (TOG), 24 (2005), pp. 677–685.

[27] F. Multon, L. Hoyet, T. Komura, and R. Kulpa, *Interactive control of physically-valid aerial motion: application to vr training system for gymnasts*, in VRST '07: Proceedings of the 2007 ACM symposium on Virtual reality software and technology, New York, NY, USA, 2007, ACM, pp. 77–80.

[28] Nintendo, *Nintendo homepage.* `http://www.nintendo.com`, Accessed March 10th, 2010, 2010.

[29] T. Piazza, J. Lundström, A. Kunz, and M. Fjeld, *Predicting missing markers in real-time optical motion capture*, in Modelling the Physiological Human, N. Magnenat-Thalmann, ed., vol. 5903 of Lecture Notes in Computer Science, Springer Berlin / Heidelberg, 2009, pp. 125–136.

[30] N. Pollard, J. Hodgins, M. Riley, and C. Atkeson, *Adapting human motion for the control of a humanoid robot*, in Proceedings ICRA '02. IEEE International Conference on Robotics and Automation, 2002., vol. 2, 2002, pp. 1390 – 1397 vol.2.

[31] L. Rabiner and B. H. Juang, *Fundamentals of Speech Recognition*, Signal processing, Prentice Hall, 1993.

[32] Y. Saitou, T. Nagano, T. Seki, M. Ishikawa, and S. Hara, *Optimal high-jump control of linear 1-d.o.f trampoline robot*, in SICE 2002. Proceedings of the 41st SICE Annual Conference, vol. 4, Aug. 2002, pp. 2527 – 2530 vol.4.

[33] K. Shoemake, *Animating rotation with quaternion curves*, ACM SIGGRAPH Computer Graphics, 19 (1985), pp. 245–254.

[34] K. Tong and M. H. Granat, *A practical gait analysis system using gyroscopes*, Medical Engineering & Physics, 21 (1999), pp. 87–94.

[35] Vicon, *Motion capture systems from vicon*. `http://www.vicon.com`, Accessed March 10th, 2010, 2010.

[36] Wikipedia, *Dead reckoning*. `http://en.wikipedia.org/wiki/Dead_reckoning`, Accessed October 6th, 2010, 2010.

[37] ——, *Eadweard muybridge*. `http://de.wikipedia.org/wiki/Eadweard_Muybridge`, Accessed July 23rd, 2010, 2010.

[38] ——, *Motion capture*. `http://en.wikipedia.org/wiki/Motion_capture`, Accessed June 10th, 2010, 2010.

[39] Xsens, *3D motion tracking*. `http://www.xsens.com`, Accessed March 10th, 2010, 2010.

[40] M. R. Yeadon, *The biomechanics of twisting somersaults. part* 1: Rigid body motions, Journal of Sports Sciences, 11 (1993), pp. 187–198.

List of Figures

List of Tables